UNIVERSIDADE DE BRASÍLIA – UnB
INSTITUTO DE GEOCIÊNCIAS - IG
PÓS-GRADUAÇÃO EM GEOCIÊNCIAS APLICADAS E GEODINÂMICA

CORREÇÃO DA DISTORÇÃO GEOMÉTRICA EM ORTOMOSAICOS DE IMAGENS ÓPTICAS OBTIDAS POR RPA, CAUSADA POR VARIAÇÕES NO ÂNGULO DE GUINADA, POR MEIO DO MÉTODO PARAMÉTRICO DE TRANSFORMAÇÃO PROJETIVA, COM DERIVAÇÃO DOS TERMOS DA MATRIZ DE ROTAÇÃO R_ψ

SÉRGIO ROBERTO HORST GAMBA

Tese de Doutorado Nº 59
Programa de Pós-Graduação em Geociências Aplicadas e Geodinâmica

BRASÍLIA - 2020

UNIVERSIDADE DE BRASÍLIA – UnB
INSTITUTO DE GEOCIÊNCIAS - IG
PÓS-GRADUAÇÃO EM GEOCIÊNCIAS APLICADAS E GEODINÂMICA

CORREÇÃO DA DISTORÇÃO GEOMÉTRICA EM ORTOMOSAICOS DE IMAGENS ÓPTICAS OBTIDAS POR RPA, CAUSADA POR VARIAÇÕES NO ÂNGULO DE GUINADA, POR MEIO DO MÉTODO PARAMÉTRICO DE TRANSFORMAÇÃO PROJETIVA, COM DERIVAÇÃO DOS TERMOS DA MATRIZ DE ROTAÇÃO R_ψ

SÉRGIO ROBERTO HORST GAMBA

Tese de doutorado apresentada junto ao Programa de Pós-Graduação em Geociências Aplicadas e Geodinâmica, área de concentração Geoprocessamento e Análise Ambiental, para obtenção do título de doutor.

Orientador: Prof. Dr. Edson Eyji Sano

BRASÍLIA - 2020

UNIVERSIDADE DE BRASÍLIA – UnB
INSTITUTO DE GEOCIÊNCIAS - IG
PÓS-GRADUAÇÃO EM GEOCIÊNCIAS APLICADAS E GEODINÂMICA

CORREÇÃO DA DISTORÇÃO GEOMÉTRICA EM ORTOMOSAICOS DE IMAGENS ÓPTICAS OBTIDAS POR RPA, CAUSADA POR VARIAÇÕES NO ÂNGULO DE GUINADA, POR MEIO DO MÉTODO PARAMÉTRICO DE TRANSFORMAÇÃO PROJETIVA, COM DERIVAÇÃO DOS TERMOS DA MATRIZ DE ROTAÇÃO R_ψ

SÉRGIO ROBERTO HORST GAMBA

Tese de Doutorado
Programa de Pós-Graduação em Geociências Aplicadas e Geodinâmica

Orientador:
Prof. Dr. Edson Eyji Sano

Banca Examinadora:
Prof. Dr. Manuel Eduardo Ferreira (Universidade Federal de Goiás)
Prof. Dr. Ricardo Ruviaro (Universidade de Brasília)
Prof. Dr. Hermann Johann Heinrich Kux (Instituto Nacional de Pesquisas Espaciais)

BRASÍLIA – 2020

FICHA CATALOGRÁFICA

GAMBA, Sérgio Roberto Horst

Correção da distorção geométrica em ortomosaicos de imagens ópticas obtidas por RPA, causada por variações no ângulo de guinada, por meio do método paramétrico de transformação projetiva, com derivação dos termos da matriz de rotação R_{ψ}. Sérgio Roberto Horst Gamba. Orientador Edson Eyji Sano. Brasília, 2020.

266p.

Tese (Doutorado em Geociências Aplicadas e Geodinâmica), nº 59 - Universidade de Brasília/Instituto de Geociências, 2020.

1. Aerolevantamento; 2. UAV; 3. Estatística Espacial; 4. Correção Geométrica.

REFERÊNCIA BIBLIOGRÁFICA

GAMBA, S. R. H. Correção da distorção geométrica em ortomosaicos de imagens ópticas obtidas por RPA, causada por variações no ângulo de guinada, por meio do método paramétrico de transformação projetiva, com derivação dos termos da matriz de rotação R_{ψ}. Brasília, Instituto de Geociências, Universidade de Brasília, 2020, 266p. Tese de Doutorado.

DEDICATÓRIA

À minha família,
Estela, esposa, Alison e Ariel, filhos, e neta Maria Luísa,
pelo apoio e companhia ao longo da caminhada.

AGRADECIMENTOS

A Deus, por ter me concedido, através de sua bondade infinita, o potencial de concretizar mais uma conquista em minha vida.

Ao meu orientador e professor, Edson Eyji Sano, pelos seus conhecimentos e auxílio que me fizeram crescer tanto na minha vida acadêmica quanto na profissional.

Aos colegas do mestrado e do doutorado, pelos esclarecimentos e contribuições de informações, sem os quais a realização deste se tornaria mais árdua.

À empresa Esteio Engenharia e Aerolevantamentos S.A. e, em especial, aos profissionais Valther e Amauri, pelo pleno apoio no planejamento e execução dos voos, bem como no trabalho de campo realizado.

A CAPES, pelo apoio prestado ao desenvolvimento da minha tese.

EPÍGRAFE

“Mede o que é mensurável e torna mensurável o que não o é.”

(Galileu Galilei)

RESUMO

Aeronaves pilotadas remotamente (RPA) têm sido utilizadas como plataformas de baixo custo nas atividades de aerolevantamentos, principalmente para imageamentos ópticos. O objetivo deste trabalho é corrigir, por meio do método matemático paramétrico de transformação projetiva com derivação dos termos da matriz de rotação R_ψ (guinada), erros geométricos causados pelas variações nos ângulos de guinada em ortomosaicos não corrigidos de imagens ópticas obtidas por RPA, utilizando algoritmo de rotação de imagem em linguagem de programação Python. Foram selecionadas duas áreas-teste localizadas nos estados do Rio de Janeiro (área 1, relevo plano) e Minas Gerais (área 2, relevo ondulado). O RPA de asa-fixa de classe 3 utilizado neste projeto foi o PT-UAV, pertencente à empresa Esteio Engenharia e Aerolevantamentos S.A. O sensor empregado foi a câmera Alfa Nex 3 Sony, com resolução máxima de 14 MPixels e distâncias focais de 16 mm e 35 mm. Para o registro dos pontos de apoio e de verificação, foi utilizado o receptor GPS Hiper da Topcon e a antena Hiper GD. A metodologia foi dividida em sete fases: a) planejamento e execução dos voos; b) determinação dos pontos de apoio e de verificação; c) geração dos ortomosaicos iniciais nos programas Agisoft Photoscan e E-Foto; d) avaliação do comportamento da distribuição espacial, normalidade e acurácia posicional das amostras, com análise de tendência e de precisão dos ortomosaicos iniciais com a aplicação do padrão de exatidão cartográfica (PEC) analógica e digital, que trata das especificações técnicas de aquisição de dados geoespaciais e vetoriais, utilizando o método disponível no aplicativo GeoPEC; e) criação matemática do método paramétrico de transformação projetiva com derivação dos termos da matriz de rotação R_ψ e criação do algoritmo em linguagem de programação Python, capazes de minimizar erros geométricos; f) aplicação dos algoritmos de correção nos mosaicos iniciais e geração de novos ortomosaicos e modelos digitais de elevação no Agisoft Photoscan. Nesta fase, o E-Foto não foi utilizado, pois seu desempenho foi inferior ao Agisoft Photoscan; e g) nova avaliação no aplicativo GeoPEC. O método SfM do Agisoft Photoscan apresentou maior eficiência que a correlação de Pearson e o método dos mínimos quadrados do E-Foto na geração de modelos digitais de superfície. Os resultados finais revelaram melhores acurácias nas planimetrias das áreas 1 e 2. A área 1 corrigida apresentou menor erro médio quadrático (RMS), com uma escala maior de 1:1.000; também foi possível obter PEC A, analógico e digital. Apesar do relevo ondulado da área 2 corrigida, mantendo-se a mesma escala de 1:2.000, o RMS foi menor, com melhoria no PEC B para o analógico e PEC C para o digital; também obtiveram-se as melhores acurácias nas altimetrias das áreas 1 e 2. A área 1 corrigida apresentou menor

RMS, com uma equidistância menor de 60 metros; também foi possível obter PEC A, analógico e digital. Para a área 2 corrigida, mantendo-se a mesma equidistância de 50 metros e os PECs, o RMS foi menor. As avaliações foram com padrão disperso e distribuição normal, obtendo-se redução no resultado tendencioso ou resultado não tendencioso. Como conclusão, o modelo matemático proposto, por meio de um algoritmo específico na linguagem de programação Python, apresentou resultados significativos na redução do RMS e melhores PEC, analógico e digital.

Palavras-chave: Aerolevantamento. UAV. Estatística Espacial. Correção Geométrica.

ABSTRACT

Remotely piloted aircrafts (RPA) have been used as low-cost platforms in aerial survey activities, mainly for optical imaging. The objective of this work is to correct, by means of the parametric mathematical method of projective transformation with derivation of the terms of the rotation matrix Rψ (yaw), geometric errors caused by the variations in yaw angles in orthomosaic, uncorrected optical images obtained by RPA, using image rotation algorithm in Python programming language. We selected two test sites located in the states of Rio de Janeiro (area 1, flat relief) and Minas Gerais (area 2, wavy relief). The class 3, fixed-wing RPA used in this project was the PT-UAV belonging to the Esteio Engenharia e Aerolevantamentos S.A. The sensor employed was the Alfa Nex 3 Sony camera, with a maximum resolution of 14 MPixels, focal lengths of 16 mm and 35 mm. For registration of support and verification points, the Topcon Hiper GPS receiver and the Hiper GD antenna were used. The methodology was divided in seven phases: a) planning and execution of flights; b) determination of support and verification points; c) generation of the initial orthomosaics in the Agisoft Photoscan and E-Foto software; d) evaluation of the initial orthomosaics in terms of the behavior of the spatial distribution, normality, and positional accuracy of the samples, with trend and precision analyzes, with the application of the standard analogical and digital cartographic accuracy, which deals with the technical specifications of acquisition of geospatial and vector data, using the method available in the GeoPEC application; e) mathematical creation of the parametric method of projective transformation with derivation of the terms of the Rψ rotation matrix and creation of the algorithm in Python programming language, capable of minimizing geometric errors; f) application of correction algorithms in the initial mosaics and generation of new orthomosaics and digital elevation models in Agisoft Photoscan; At this stage, E-Foto was not used, as its performance was inferior to Agisoft Photoscan; and g) new evaluation in the GeoPEC application. The SfM method of Agisoft Photoscan showed greater efficiency than Pearson's correlation and the least squares method of E-Foto in the generation of digital surface models. The final results revealed better accuracy in the planimetry of areas 1 and 2. Corrected area 1 had a lower RMS, with a larger scale of 1: 1,000. It was also possible to obtain PEC A, analog and digital. Despite the corrugated relief of corrected area 2, maintaining the same scale of 1:2,000, the RMS was lower, with improvement in PEC B for analogic and PEC C for digital. A better accuracy in the altimetry of areas 1 and 2 was also obtained. The corrected area 1 had a lower RMS, with a shorter equidistance of 60 meters. It was also possible to obtain PEC A, analogic and digital. For the corrected area 2, keeping

the same 50 meters equidistance and the PECs, the RMS was lower. The evaluations were with a dispersed pattern and normal distribution, obtaining a reduction in the biased or non-biased results. In conclusion, the proposed mathematical model, using a specific algorithm in the Python programming language, showed significant results in reducing RMS and better PEC, analogic and digital.

Keywords: Aerial Survey. UAV Spatial Statistics. Geometric Correction.

SUMÁRIO

1-INTRODUÇÃO.....23
1.1 JUSTIFICATIVAS.....23
1.2 PROBLEMA.....24
1.3 OBJETIVOS.....24
1.4 HIPÓTESE.....25
1.5 ÁREAS DE ESTUDO.....25
1.6 MATERIAIS.....29
1.7 MÉTODOS.....31
2- FUNDAMENTAÇÃO TEÓRICA.....36
2.1 AERONAVES REMOTAMENTE PILOTAS (RPA).....36
2.1.1 História.....36
2.1.2 Conceito e Nomes.....37
2.1.3 Aplicações da RPA......38
2.1.4 Legislação sobre a RPA.....39
2.1.5 Utilização da RPA no Aerolevantamento Óptico.....40
2.2 FOTOGRAMETRIA DIGITAL.....41
2.2.1 Generalidades.....41
2.2.2 Produção do ortomosaico.....44
2.2.2.1 Orientação interior.....47
2.2.2.2 Orientação exterior.....52
2.2.2.3 Modelo digital de elevação (MDE).....64
2.3 CORREÇÃO GEOMÉTRICA.....71
2.3.1 Ângulo *yaw* (de deriva, de guinada, azimute, *ψ)*.....72
2.3.2 Transformações geométricas.....75
2.3.3 Correção geométrica no ângulo de guinada.....92
2.4 MÉTODO PARA AVALIAÇÃO DOS ORTOMOSAICOS.....98
2.5 APLICATIVO PYTHON E BIBLIOTECA DIGITAL OPENCV.....103
3-EXPERIMENTO E RESULTADOS PRELIMINARES......105
3.1 PLANEJAMENTO E EXECUÇÃO DOS VOOS.....105
3.2 DETERMINAÇÃO DOS PONTOS DE APOIO E DE VERIFICAÇÃO.....109

3.3 GERAÇÃO DE ORTOMOSAICOS INICIAIS.....112
3.4 AVALIAÇÃO DOS ORTOMOSAICOS INICIAIS.....118
4-ALGORITMO DE CORREÇÃO GEOMÉTRICA NO ÂNGULO DE GUINADA....125
4.1 EXPERIMENTO INICIAL COM ALGORITMO NO EXCEL.....125
4.2 ALGORITMO DE CORREÇÃO GEOMÉTRICA PARA O ÂNGULO DE GUINADA NA LINGUAGEM DE PROGRAMAÇÃO PYTHON.....147
5-EXPERIMENTO E RESULTADOS FINAIS.....149
5.1 GERAÇÃO DE ORTOMOSAICOS CORRIGIDOS.....158
5.2 AVALIAÇÃO DOS ORTOMOSAICOS CORRIGIDOS.....160
6-CONCLUSÕES.....165
7-REFERÊNCIAS 167
APÊNDICE A.....176

LISTA DE FIGURAS

Figura 1. Mapa do distrito de Itaipuaçu, município de Maricá, RJ.....26
Figura 2. Mapa da região de garimpo de Capoeirana, município da Nova Era, MG...28
Figura 3. RPA da Esteio.....29
Figura 4. Câmera Alfa Next 3 da Esteio.....30
Figura 5. Receptor GPS Hiper da Topcon e a antena Hiper GD da Esteio.....30
Figura 6. 1ª a 4ª fase do método.....34
Figura 7. 5ª a 7ª fase do método.....35
Figura 8. Imagem da RPA da Força Aérea Brasileira.....38
Figura 9. Reconstrução de um espaço tridimensional (espaço-objeto), a partir de um conjunto não vazio de imagens bidimensionais digitais (espaço-imagem).....41
Figura 10. Exemplo de recobrimento aéreo longitudinal e lateral de uma sequência de fotos obtidas por duas linhas de voo adjacentes.....43
Figura 11. Processo de ortorretificação, que transforma uma imagem em perspectiva central para perspectiva ortogonal.....44
Figura 12. Fluxograma de processos do E-Foto.....46
Figura 13. Representação do corpo da câmera, coordenadas da câmera (x,y) e coordenadas da imagem (η,ξ).....48
Figura 14. Transformação afim.....50
Figura 15. Dados do sensor para orientação interior no E-Foto.....53
Figura 16. Elementos de orientação exterior.....54
Figura 17. Parâmetros de atitude de um sensor colocado em plataforma aérea.....55
Figura 18. Efeitos das atitudes *roll* (rolamento), *pitch* (arfagem) *e yaw* (guinada) da aeronave na geometria da imagem.....56
Figura 19. Sistema aeronáutico e fotogramétrico em relação ao ENU.....57
Figura 20. Parâmetros geométricos da câmera.....59
Figura 21. Condição de colinearidade.....61
Figura 22. Dados da imagem para orientação exterior no programa E-Foto.....63
Figura 23. Diferenças entre MDS e MDT ou MDE.....65
Figura 24. Mecanismo de busca por pontos homólogos.....66
Figura 25. Determinação do ponto homólogo por meio do método de correspondência

por mínimos quadrados.....67
Figura 26. Dados para a extração do MDE no E-Foto.....69
Figura 27. Tela de ortorretificação no ambiente E-Foto.....70
Figura 28. Ângulo de deriva.....72
Figura 29. Distorção devido ao *yaw*.....73
Figura 30. Transformações de um objeto no espaço E para o espaço E': 1- translação; 2- transformação ortogonal; 3- transformação isogonal; 4- transformação afim; 5- transformação projetiva; 6- transformação polinomial.....76
Figura 31. Transformação ortogonal. (a) rotação entre os sistemas bidimensionais; (b) rotação e translação entre os sistemas bidimensionais.....77
Figura 32. Transformação isogonal. (a) rotação entre os sistemas bidimensionais; (b) rotação, translação e fator de escala entre os sistemas bidimensionais.....78
Figura 33. Transformação afim.....79
Figura 34. Geometria da transformação projetiva no plano.....81
Figura 35. Sistema referencial tridimensional em três eixos em três níveis e as matrizes de rotação.....83
Figura 36. Efeito da rotação em ω sobre a projeção dos pontos e movimento total das componentes Y e Z no plano de projeção.....84
Figura 37. Efeito da rotação em φ sobre a projeção dos pontos e movimento total das componentes X e Z no plano de projeção.....85
Figura 38. Movimento total das componentes X e Y no plano de projeção.....86
Figura 39. Geometria do modelo de colinearidade.....89
Figura 40. Tipos de visada com o RPA.....92
Figura 41. Mosaico da primeira área com três faixas de voo do RPA.....93
Figura 42. Ciclo trigonométrico com os quatro quadrantes.....94
Figura 43. Ciclo trigonométrico com a redução de quadrantes.....95
Figura 44. Imageamento da primeira área, região de Itaipuaçu, Maricá, RJ, envolvendo três faixas de voo (amarelo, verde e azul).....105
Figura 45. Imageamento da segunda área, região de garimpo de Capoeirana, Nova Era, MG, envolvendo uma faixa de voo (amarelo).....106
Figura 46. MDE da área 1 com 30 cm de resolução.....114
Figura 47. MDE da área 2 com 30 cm de resolução.....115

Figura 48. Ortomosaico da subárea noroeste da área 1 com 50 cm de resolução....115
Figura 49. Ortomosaico da área 2 com 50 cm de resolução.....116
Figura 50. MDE da área 1 com 12 cm de resolução.....116
Figura 51. MDE da área 2 com 7 cm de resolução.....117
Figura 52. Ortomosaico da área 1 com 12 cm de resolução.....117
Figura 53. Ortomosaico da área 2 com 7 cm de resolução.....118
Figura 54. Imagem RPA teste com ângulo de guinada 0° no aplicativo Excel.....127
Figura 55. Imagem RPA teste com ângulo de guinada 13,19° no aplicativo Excel....128
Figura 56. Imagem RPA teste *Rd11*$_{\psi}$ com ângulo de guinada 13,19° corrigida no aplicativo Excel.....130
Figura 57. Imagem RPA teste *Rd21*$_{\psi}$ com ângulo de guinada 13,19° corrigida no aplicativo Excel.....131
Figura 58. Imagem RPA teste com ângulo de guinada 56,75° no aplicativo Excel....133
Figura 59. Imagem RPA teste *Rd11*$_{\psi}$ com ângulo de guinada 56,75° corrigida no aplicativo Excel.....134
Figura 60. Imagem RPA teste *Rd21*$_{\psi}$ com ângulo de guinada 56,75° corrigida no aplicativo Excel.....136
Figura 61. Imagem RPA teste com ângulo de guinada 90° no aplicativo Excel....137
Figura 62. Imagem RPA teste *Rd11*$_{\psi}$ com ângulo de guinada 90° corrigida no aplicativo Excel.....139
Figura 63. Imagem RPA teste *Rd21*$_{\psi}$ com ângulo de guinada 90° corrigida no aplicativo Excel.....141
Figura 64. Imagem RPA teste com ângulo de guinada 168,70° no aplicativo Excel..142
Figura 65. Imagem RPA teste *Rd11*$_{\psi}$ com ângulo de guinada 168,70° corrigida no aplicativo Excel.....144
Figura 66. Imagem RPA teste *Rd21*$_{\psi}$ com ângulo de guinada 168,70° corrigida no aplicativo Excel.....146
Figura 67. Três faixas de voo da primeira área com suas imagens ópticas originais, sendo 5 imagens na faixa 1 (P4 a P8), coluna da esquerda, 6 imagens na faixa 2 (P11 a P16), coluna central, e 6 imagens na faixa 3 (P4 a P9), coluna da direita.....149
Figura 68. Três faixas de voo da primeira área com suas imagens ópticas corrigidas

nas faixas 1 e 3 e imagens originais na faixa 2, sendo 5 imagens na faixa 1 (P4 a P8), coluna da esquerda, 6 imagens na faixa 2 (P11 a P16), coluna central, e 6 imagens na faixa 3 (P4 a P9), coluna da direita.....151
Figura 69. Faixa de voo da segunda área com suas imagens ópticas originais, sem correção, num total de 28 imagens (P1 a P28).....153
Figura 70. Faixa de voo da segunda área com suas imagens ópticas corrigidas num total de 28 imagens (P1 a P28).....155
Figura 71. MDE corrigido da área 1 com 12,6 cm de resolução.....158
Figura 72. MDE corrigido da área 2 com 6 cm de resolução.....159
Figura 73. Ortomosaico corrigido da área 1 com 12 cm de resolução.....159
Figura 74. Ortomosaico corrigido da área 2 com 6 cm de resolução.....159

LISTA DE TABELAS

Tabela 1. Valores de Padrão de Exatidão Cartográfica (PEC) e de Erro-Padrão (EP), conforme Decreto-Lei no. 89.817/84 e Especificações Técnicas de Aquisição de Dados Geoespaciais Vetoriais (ET-ADGV). PCD = Produtos Cartográficos Digitais.32
Tabela 2. Ângulos *roll, pitch* e *yaw* no RPA Echar 20B.....73
Tabela 3. Ângulos *roll, pitch* e *yaw* no RPA 1.....73
Tabela 4. Ângulos *roll, pitch* e *yaw* no RPA MD4-1000.....74
Tabela 5. Ângulos *roll, pitch* e *yaw* no RPA MLB BAT3.....74
Tabela 6 Ângulos *roll, pitch* e *yaw* no RPA 2.....74
Tabela 7. Dados relativos a centros perspectivos, altura, *pitch*, *roll* e *yaw* do aerolevantamento fotogramétrico efetuado na primeira área.....106
Tabela 8. Dados relativos a centros perspectivos, altura, *pitch*, *roll* e *yaw* do aerolevantamento fotogramétrico efetuado na segunda área (Faixa 1).....108
Tabela 9. Coordenadas dos pontos de apoio e de verificação no campo e elevações da primeira área.....110
Tabela 10. Coordenadas dos pontos de apoio e de verificação no campo e elevações na segunda área.....111
Tabela 11. Dados estatísticos planimétricos da área 1 (processamento no Agisoft PhotoScan) do GeoPEC.....118
Tabela 12. Dados estatísticos altimétricos da área 1 (processamento no Agisoft PhotoScan) do GeoPEC.....119
Tabela 13. Dados estatísticos planimétricos da área 2 (processamento no Agisoft PhotoScan) do GeoPEC.....119
Tabela 14. Dados estatísticos altimétricos da área 2 (processamento no Agisoft PhotoScan) do GeoPEC.....120
Tabela 15. Dados estatísticos planimétricos da área 1 (processamento no E-Foto) do GeoPEC.....121
Tabela 16. Dados estatísticos altimétricos da área 1 (processamento no E-Foto) do GeoPEC.....121
Tabela 17. Dados estatísticos planimétricos da área 2 (processamento no E-Foto) do GeoPEC.....122

Tabela 18. Dados estatísticos altimétricos da área 2 (processamento no E-Foto) do GeoPEC.....123
Tabela 19 – Dados estatísticos (erro médio quadrático) das Áreas 1 e 2. PEC = Padrão de Exatidão Cartográfica; PCD = Produtos Cartográficos Digitais; Ep = Equidistância; e Es = Escala.....123
Tabela 20. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 0°.....127
Tabela 21. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 13,19°128
Tabela 22. Matriz *Rd11*$_{\psi}$ com ângulo de guinada de 13,19°.....129
Tabela 23. Matriz de coordenadas (x, y, z) 3x18 *Rd11*$_{\psi}$ para o ângulo de guinada 13,19° corrigida.....130
Tabela 24. Matriz *Rd21*$_{\psi}$ com ângulo de guinada de 13,19°.....131
Tabela 25. Matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ para o ângulo de guinada 13,19° corrigida.....132
Tabela 26. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 56,75°133
Tabela 27. Matriz *Rd11*$_{\psi}$ com ângulo de guinada de 56,75°.....134
Tabela 28. Matriz de coordenadas (x, y, z) 3x18 *Rd11*$_{\psi}$ para o ângulo de guinada 56,75° corrigida.....135
Tabela 29. Matriz *Rd21*$_{\psi}$ com ângulo de guinada de 56,75°.....135
Tabela 30. Matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ para o ângulo de guinada 56,75° corrigida.....136
Tabela 31. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 90°.....138
Tabela 32. Matriz *Rd11*$_{\psi}$ com ângulo de guinada de 90°.....138
Tabela 33. Matriz de coordenadas (x, y, z) 3x18 *Rd11*$_{\psi}$ para o ângulo de guinada 90° corrigida.....139
Tabela 34. Matriz *Rd21*$_{\psi}$ com ângulo de guinada de 90°.....140
Tabela 35. Matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ para o ângulo de guinada 90° corrigida.....141
Tabela 36. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 168,70°.....142
Tabela 37. Matriz *Rd11*$_{\psi}$ com ângulo de guinada de 168,70°.....143
Tabela 38. Matriz de coordenadas (x, y, z) 3x18 *Rd11*$_{\psi}$ para o ângulo de guinada 168,70° corrigida.....144

Tabela 39. Matriz *Rd21$_\psi$* com ângulo de guinada de 168,70°.....145
Tabela 40. Matriz de coordenadas (x, y, z) 3x18 *Rd21$_\psi$* para o ângulo de guinada 168,70° corrigida.....146
Tabela 41. Dados estatísticos planimétricos da área 1 corrigida do GeoPEC.....160
Tabela 42. Dados estatísticos altimétricos da área 1 corrigida do GeoPEC.....160
Tabela 43. Dados estatísticos planimétricos da área 2 corrigida do GeoPEC.....161
Tabela 44. Dados estatísticos altimétricos da área 2 corrigida do GeoPEC.....162
Tabela 45 – Dados estatísticos erro médio quadrático da planimetria das Áreas 1 e 2 não corrigidos e corrigidos no Agisoft Photoscan. PEC = Padrão de Exatidão Cartográfica; PCD = Produtos Cartográficos Digitais; e Es = Escala.....162
Tabela 46 – Dados estatísticos erro médio quadrático da altimetria das Áreas 1 e 2 não corrigidos e corrigidos no Agisoft Photoscan. PEC = Padrão de Exatidão Cartográfica; PCD = Produtos Cartográficos Digitais; e Eq = Equidistância.....163

LISTA DE ABREVIATURAS E SIGLAS

ANAC	Agência Nacional de Aviação Civil
ANOVA	Analysis of Variance
ARP	Aeronave Remotamente Pilotada
ASCII	American Standard Code for Information Interchange
CA	Certificado de Aeronavegabilidade
CCP	Coordenadas dos Centros de Perspectiva
CM	Certificado de Matrícula
CMOS	Complementary Metal-Oxide-Semiconductor
CTM	Cadastro Territorial Multifinalitário
DECEA	Departamento de Controle do Espaço Aéreo
DEM	Digital Elevation Model
DSG	Diretoria do Serviço Geográfico do Exército Brasileiro
DTM	Digital Terrain Model
EQM	Erro Quadrático Médio
EP	Erro-Padrão
ET-ADGV	Especificações Técnicas de Aquisição de Dados Geoespaciais Vetoriais
EUA	Estados Unidos da América
FAB	Força Aérea Brasileira
FOV	Field of View
GCP	Ground Control Points
GNSS	Global Navigation Satellite System
GSD	Ground Sample Distance
IBGE	Instituto Brasileiro de Geografia e Estatística
ICA	Instrução do Comando da Aeronáutica
IFOV	Instantaneous Field of View
IMU	Inertial Measurement Unit
INCRA	Instituto Nacional de Colonização e Reforma Agrária
INDE	Infraestrutura Nacional de Dados Espaciais
INPE	Instituto Nacional de Pesquisas Espaciais
IPTU	Imposto Predial e Territorial Urbano

Lat	Latitude
Long	Longitude
MDE	Modelo Digital de Elevação
MDT	Modelo Digital do Terreno
MDS	Modelo Digital de Superfície
MMQ	Método dos Mínimos Quadrados
MNE	Modelo Numéricos de Elevação
NOTAM	Notice To Airmen
OACI	Organização de Aviação Civil Internacional
OpenCV	Open Source Computer Vision Library
PEC	Padrão de Exatidão Cartográfica
PEC-PCD	Padrão de Exatidão Cartográfica de Produtos Cartográficos Digitais
RAB	Registro Aeronáutico Brasileiro
RBAC	Regulamento Brasileiro da Aviação Civil Especial
RMS	Root Mean Square
RPA	Remotely Piloted Aircraft
RPAS	Remotely Piloted Aircraft System
RPC	Rational Polinomial Coefficients
S*f*M	Structure from Motion
SIG	Sistema de Informações Geográficas
SIRGAS	Sistema de Referência Geocêntrico para as Américas
SISCEAB	Sistema de Controle do Espaço Aéreo Brasileiro
SPSS	Statistical Package for Social Sciences
TAG	Transformação Afim Geral
TIF	Tagged Image File
UAS	Unmanned Aircraft Systems
UAV	Unmanned Aerial Vehicle
UERJ	Universidade Estadual do Rio de Janeiro
UTM	Universal Transversa de Mercator
VANT	Veículos Aéreos Não Tripulados

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

Os Sistemas de Aeronaves Remotamente Pilotadas (RPA), termo adotado tecnicamente pela Organização de Aviação Civil Internacional, conhecido em inglês como *Remotely Piloted Aircraft*, conforme Instrução do Comando da Aeronáutica 100-40/2018 (ICA 100-40/2016), têm sido utilizados como plataformas eficazes e de baixo custo em imageamentos ópticos (EISENBEIß, 2009; AKAR, 2017; LEMES et al., 2017; QAYYUM et al., 2017). As RPA podem auxiliar na implantação de Cadastros Territoriais Multifinalitários (CTM) (OLIVEIRA e BRITO, 2019), visto pelos gestores como uma forma de melhorar a arrecadação do Imposto Predial e Territorial Urbano (IPTU) (LEMES et al., 2017). A coleta de dados para implantação do CTM deve ser feita de maneira ágil e eficaz para que não haja oclusão de dados.

Entretanto, a qualidade dos produtos gerados pelos imageamentos ópticos com RPA, como ortomosaicos, necessita ser analisada. A validação da qualidade das informações cartográficas tem sido um tema cada vez mais evidente e de extrema importância (FONSECA NETO et al., 2017). Para se obter o padrão de exatidão cartográfica de produtos cartográficos digitais (PEC-PCD) de classes A ou B, em ortomosaicos obtidos por meio de sensores ópticos embarcados em RPA, é necessário corrigir distorções geométricas nas imagens relacionadas com variações nos ângulos de guinada, devido à instabilidade da plataforma RPA provocada pela deriva no voo (JAIMES, 2016). A avaliação do PEC-PCD em ortomosaicos obtidos por RPA é fundamental na determinação da qualidade das cartas cartográficas, principalmente quando se trata de escalas de representação relativamente grandes. A ausência de cartas atualizadas em grandes escalas tem impulsionado a utilização de RPA (ALVES JÚNIOR et al., 2018).

Os aplicativos E-Foto e Agisoft Photoscan não corrigem de maneira efetiva as distorções geométricas nas imagens provocadas pelo ângulo de guinada, sendo necessário um modelo matemático de correção, aplicado por um algoritmo em linguagem de programação Python.

1.2 PROBLEMA

Esta tese de doutorado procura abordar o seguinte problema técnico: Como corrigir erros geométricos causados pelas variações nos ângulos de guinada em ortomosaicos obtidos por RPA de asa-fixa de classe 3, que não utilizam plataformas "*gimbal*" giro estabilizadas?

1.3 OBJETIVOS

O objetivo geral deste trabalho é corrigir, por meio do método matemático paramétrico de transformação projetiva com derivação dos termos da matriz de rotação R_ψ (guinada), erros geométricos causados pelas variações nos ângulos de guinada em ortomosaicos não corrigidos de imagens ópticas obtidas por RPA, utilizando algoritmo de rotação de imagem em linguagem de programação Python.

Os objetivos específicos são:

a) Propor um método matemático paramétrico de transformação projetiva com derivação dos termos da matriz de rotação $R_{\psi,}$ que realiza a correção do ângulo de guinada, modificando a fórmula da transformação projetiva clássica (ortogonal) e aumentando a rotação para aumentar o grau de correção;

b) Desenvolver um algoritmo de rotação de imagem em linguagem de programação Python, com a nova fórmula da transformação projetiva, capaz de minimizar erros geométricos em ortomosaicos de imagens ópticas obtidas por RPA; e

c) Avaliar o PEC analógico e digital por meio do aplicativo GeoPEC dos ortomosaicos, antes e após correção pelo algoritmo de rotação de imagem em linguagem de programação *Python*.

1.4 HIPÓTESE

O método matemático paramétrico de transformação projetiva com derivação dos termos da matriz de rotação $R_{\psi,}$ que modifica a fórmula da transformação projetiva clássica, aumenta a rotação da imagem, seguida por aumento significativo do grau de correção do ortomosaico.

1.5 ÁREAS DE ESTUDO

Este trabalho considerou duas áreas distintas de aerolevantamento com sensor de imagens ópticas em plataforma RPA de asa-fixa classe 3. A primeira área foi escolhida devido à presença de relevo plano, declividade de 0,8%, com valores de elevação máxima de 12,959 m, mínima de 1,567 m, média de 6,938 m e desvio-padrão de 3,483 m.

A área de estudo está localizada na região de Itaipuaçu, distrito do município de Maricá, na região dos Lagos, estado do Rio de Janeiro, próxima à praia de Itaipuaçu, a 20 km da Baia de Guanabara (**Figura 1**), carta SF-23-Z-B-V-3-SO. As coordenadas da área imageada pelo sistema de coordenadas geográficas são:

22° 58’ 27” S / 42° 57’ 03” W;
22° 57’ 43” S / 42° 57’ 06” W;
22° 57’ 41” S / 42° 56’ 31” W;
22° 58’ 25” S / 42° 56’ 33” W.

As coordenadas da área imageada pelo sistema de projeção Universal Transversa de Mercator (UTM) são:

7457873,29 S / 710073,03 E;
7459228,08 N / 710006,46 E;
7459275,66 S / 711004,40 E;
7457922,86 S / 710928,45 E.

A área de análise foi de 0,42 km^2, considerando a coleta dos pontos de apoio e de verificação, bem como a retirada da faixa marítima. O município de Maricá é

rodeado por maciços costeiros que formam um arco. As serras principais são: Calaboca, Mato, Lagarto, Silvado, Espraiado e Tiririca. Outra formação importante é a vasta planície costeira, entre as bases dos maciços e a linha da costa, onde está localizado o distrito Itaipuaçu (ALIPRANDI et al., 2014).

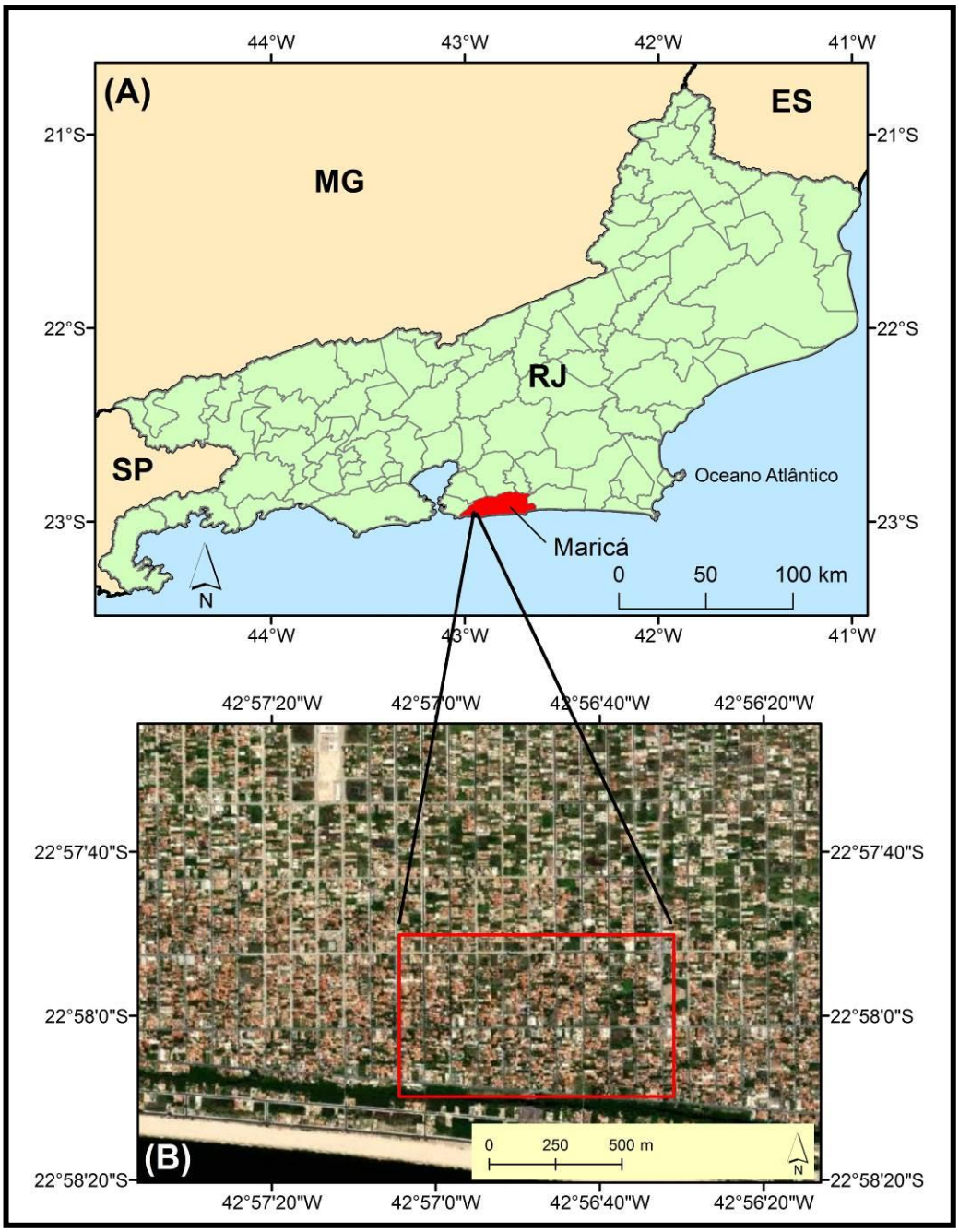


Figura 1. Mapa de localização da Área 1 no município de Maricá, RJ (A) e na área urbana do distrito de Itaipuaçu, município de Maricá (B). Imagem de satélite corresponde à imagem de alta resolução disponível na plataforma Google Earth™.

O município apresenta um grande complexo lagunar que contempla as lagoas de Maricá, Barra de Maricá, do Padre, Guarapina e Jaconé, além dos canais de Ponta Negra e de Itaipuaçu que ligam as lagoas ao mar. A localidade também é conhecida por suas praias oceânicas, dentre as quais se destacam as de Jaconé, Ponta Negra, Barra de Maricá, Zacarias, do Francês e Itaipuaçu (ALIPRANDI et al., 2014).

A segunda área foi escolhida devido à presença de relevo ondulado, declividade de 15%, com valores de elevação máxima de 684,40 m, mínima de 603,43 m, média de 634,70 m e desvio-padrão de 23,2 m. A área de estudo está localizada na região de garimpo em Capoeirana, área rural do município da Nova Era, estado de Minas Gerias (**Figura 2**), carta SE 23-Z-D-IV-2-SE. As coordenadas da área imageada pelo sistema de coordenadas geográficas são:

19° 42' 11” S / 43° 04’ 02” W;
19° 42’ 14” S / 43° 04’ 02” W;
19° 42’ 12” S / 43° 03’ 13” W;
19° 42’ 16” S / 43° 03’ 13” W.

As coordenadas UTM da área imageada são:
7820238,00 N / 702581,00 E;
7820140,00 S / 702593,00 E;
7820194,00 S / 704021,00 E;
7820069,00 S / 704010,00 E.

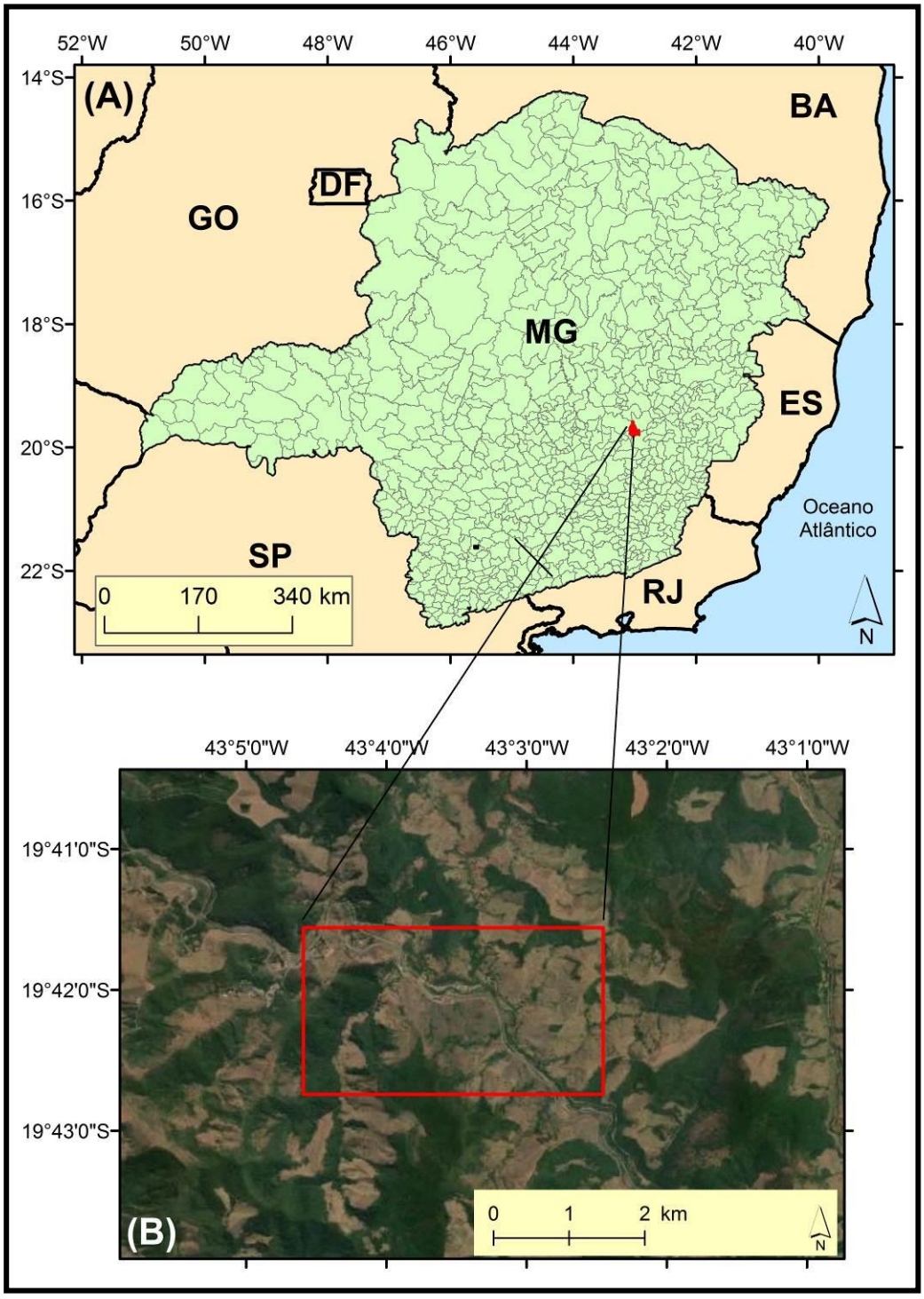


Figura 2. Mapa de localização da Área 2 no município de Nova Era, MG (A) e na área rural em uma região de garimpo no mesmo município (B). Imagem de satélite corresponde à imagem de alta resolução disponível na plataforma Google Earth™.

A área de análise foi de 0,07 km^2, considerando a coleta dos pontos de apoio e de verificação. Aproximadamente 70% do município de Nova Era é montanhoso, com 25% de áreas onduladas e ocupadas por mares de morros, sendo os 5% restantes predominantemente planos. A altitude máxima de 1.222 metros encontra-se no Alto dos Passos, enquanto a altitude mínima, de 790 metros, localiza-se na foz do córrego Barbosa. O ponto central da cidade está a 524,46 m (IBGE, 2009).

1.6 MATERIAIS

O RPA de asa fixa de classe 3 utilizado neste projeto foi o PT-UAV, pertencente à empresa Esteio Engenharia e Aerolevantamentos S.A. (**Figura 3**), com 1,47 m de comprimento, 2,00 m de envergadura, peso de 6 kg, carga útil de 2 kg, autonomia de 90 minutos, velocidade de cruzeiro de 60 a 100 km/h, motor de 2 HP e teto operacional de 2000 m. O sensor empregado (**Figura 4**) foi a câmera Alfa Nex 3 Sony, com resolução máxima de 14 MPixels, distâncias focais de 16 e 35 mm, velocidade de obturação de 30 a 1/4000 segundos e abertura do diafragma entre f/2.8 a f/22. Para o registro dos pontos de apoio e de verificação, foi utilizado o receptor GNSS Hiper da Topcon e a antena Hiper GD, série 256-0690 (**Figura 5**).

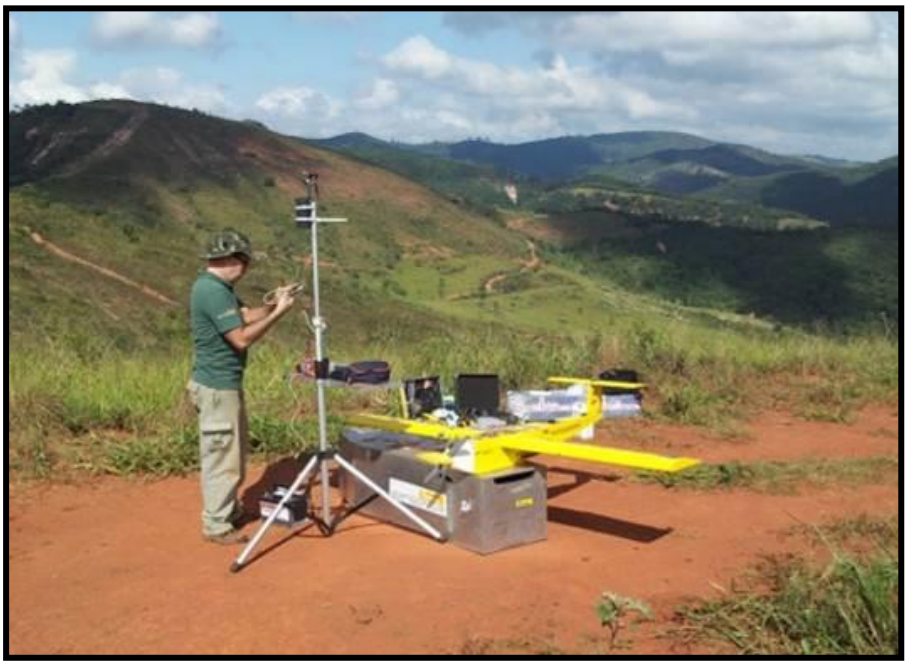


Figura 3. RPA PT-UAV da Esteio. Fonte: http://www.esteio.com.br

Figura 4. Câmera Alfa Next 3 da Esteio. Fonte: http://www.esteio.com.br

Figura 5. Receptor GNSS Hiper da Topcon e antena Hiper GD da Esteio.
Fonte: http://www.esteio.com.br

1.7 MÉTODOS

O método empregado neste trabalho foi dividido em sete fases. A primeira fase foi o planejamento e a execução dos voos, que incluiu a delimitação das áreas de imageamento, a definição do número de faixas de recobrimento aéreo em área urbana de Itaipuaçu, RJ, e região de garimpo em Capoeirana, MG, a verificação das condições meteorológicas para o melhor dia e hora para a realização dos voos, a utilização do piloto automático modelo MP2128g2® (GNSS + Inercial), a aplicação do sistema de disparo automático da câmera Alfa Nex 3 Sony, a utilização das altitudes de voo de 300 m e de 1080 m, conforme o relevo das áreas, a regulagem da abertura de diafragma em f/8 e de velocidade de obturação em 1/2000 segundos para a câmera Alfa Nex 3 Sony, a solicitação de um aviso aos aeronavegantes (NOTAM) ao Departamento de Controle do Espaço Aéreo para garantir a segurança do voo e a elaboração e execução dos planos de voos (BRASIL, 2018).

A segunda fase foi determinar os pontos de apoio e de verificação com o equipamento Topcon Hiper, com antena Hiper GD, número de série 256-0690, logo após a execução dos voos, utilizando o sistema de projeção cartográfica Universal Transversa de Mercator (UTM) e o Sistema de Referência Geocêntrico para as Américas (SIRGAS 2000), a fim de obter dados altimétricos e planimétricos (FITZ, 2009).

Na terceira fase, foi a aplicação da orientação interior e exterior, com a geração de modelo digital de superfície (MDS) e de ortomosaicos nos aplicativos E-foto e Agisoft Photoscan (COELHO e BRITO, 2016). O aplicativo livre E-Foto dispõe de algoritmos específicos para a orientação interior e exterior e geração de MDS e de ortomosaicos e encontra-se disponível para *download* no endereço eletrônico <http://www.efoto.eng.uerj.br/>. O aplicativo comercial russo Agisoft Photoscan permite o processamento das imagens aéreas captadas por RPA e geração da base cartográfica do terreno. No Agisoft Photoscan, tem-se o método *Structure from Motion* (SfM), técnica que permite a extração de informações tridimensionais a partir de imagens estáticas em 2D, e consiste na tomada de diversas imagens de uma cena a partir de pontos de vista diferentes.

A quarta fase foi de avaliação do ortomosaico por meio do Padrão de Exatidão Cartográfica (PEC) analógico e digital, consulta às Especificações Técnicas de Aquisição de Dados Geoespaciais Vetoriais (ET-ADGV) da Infraestrutura Nacional de Dados Espaciais (INDE) (IBGE, 2017), além da inspeção topográfica presente na norma brasileira NBR 13.133 e da avaliação de ortofotos de aerolevantamento e bases cartográficas utilizadas no processo de georreferenciamento de imóveis rurais, segundo Norma de Execução 02 de 2018 do Instituto Nacional de Colonização e Reforma Agrária (INCRA), no aplicativo GeoPEC. Nessa avaliação, é verificado o comportamento da distribuição espacial, a normalidade e a acurácia posicional das amostras, além das análises de tendências e de precisão (SANTOS et al., 2016).

O programa livre GeoPEC fornece ao usuário uma ferramenta amigável e de fácil utilização, permitindo uma avaliação da acurácia posicional de seus produtos, aliado à ET-ADGV (IBGE, 2017) da INDE, além da inspeção topográfica presente na NBR 13.133 e da avaliação de ortofotos de aerolevantamento e bases cartográficas utilizadas no processo de georreferenciamento de imóveis rurais, segundo a Norma de Execução 02 de 2018 do INCRA (Tabela 1) (SANTOS, 2018). O aplicativo GeoPEC é de domínio público e encontra-se disponível para *download* na página eletrônica do curso de Engenharia de Agrimensura da Universidade Federal de Viçosa (http://www.eam.ufv.br). Nesta fase, foi verificada, ainda, a existência de erros geométricos (erros sistemáticos) por meio das análises de tendências e de precisão (D'ALGE, 2007).

Tabela 1. Valores do Padrão de Exatidão Cartográfica (PEC) e do Erro-Padrão (EP), conforme Especificações Técnicas de Aquisição de Dados Geoespaciais Vetoriais (ET-ADGV). PCD = Produtos Cartográficos Digitais; Eq = Equidistância; Es = Escala.

Classe PEC	Classe PEC-PCD	PEC-Planimetria	EP-Planimetria	PEC-Altimetria	EP-Altimetria
-	A	0,28 mm x Es	0,17 mm x Es	0,27 Eq	1/6 Eq

A	B	0,5 mm x Es	0,3 mm x Es	1/2 Eq	1/3 Eq
B	C	0,8 mm x Es	0,5 mm x Es	3/5 Eq	2/5 Eq
C	D	1,0 mm x Es	0,6 mm x Es	3/4 Eq	1/2 Eq

Fonte: Elaborada pelo autor.

A quinta fase foi a criação de algoritmos de correção geométrica. Nessa fase, foi desenvolvido matematicamente o método paramétrico de transformação projetiva com derivação dos termos da matriz de rotação R_ψ (guinada) para corrigir os erros geométricos gerados pelo ângulo de guinada (ANTON e RORRES, 2012; POOLE, 2004). Na sequência, foram criados algoritmos em linguagem de programação Python capazes de minimizar erros geométricos (SEVERANCE, 2009; MATTHES, 2017). O aplicativo para aplicar a linguagem de programação Python é de domínio público e encontra-se disponível para *download* na página eletrônica https://www.Python.org/

Na sexta fase, foram aplicados os algoritmos de correção em linguagem de programação Python nos mosaicos iniciais, a fim de obter novos ortomosaicos e MDS corrigidos no Agisoft Photoscan. O Agisoft Photoscan apresentou menores RMS que o E-Foto, sendo o aplicativo utilizado nos resultados finais (COELHO e BRITO, 2016).

O método empregado com suas quatro fases iniciais está representado de forma resumida na **Figura 6**.

A sétima fase consistiu em novamente avaliar os ortomosaicos já corrigidos, por meio do PEC analógico e digital, segundo a ET-ADGV da INDE (IBGE, 2017), além da inspeção topográfica presente na norma brasileira NBR 13.133 e da avaliação de ortofotos de aerolevantamento e bases cartográficas utilizadas no processo de georreferenciamento de imóveis rurais, segundo Norma de Execução 02 de 2018 do INCRA, no aplicativo GeoPEC. Nessa avaliação, foi verificado o comportamento da distribuição espacial, a normalidade e a acurácia posicional das amostras, além das análises de tendências e de precisão (SANTOS et al., 2016).

O método empregado com suas quatro fases iniciais está representado de forma resumida na **Figura 7**.

Após a definição da metodologia, segue-se com a fundamentação teórica, considerando todo o processo de geração do ortomosaico.

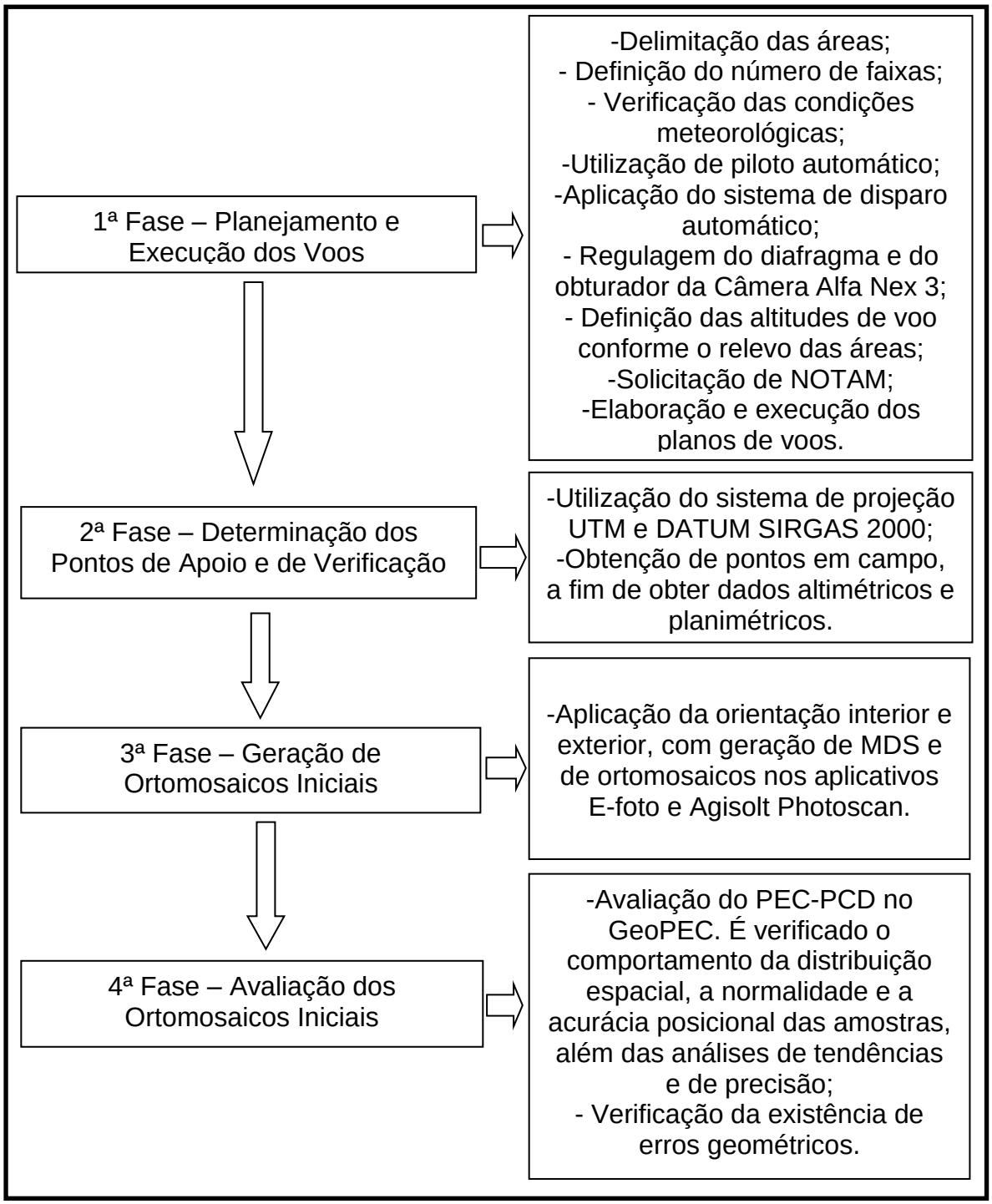


Figura 6. 1ª a 4ª fase do método. Fonte: Elaborada pelo autor.

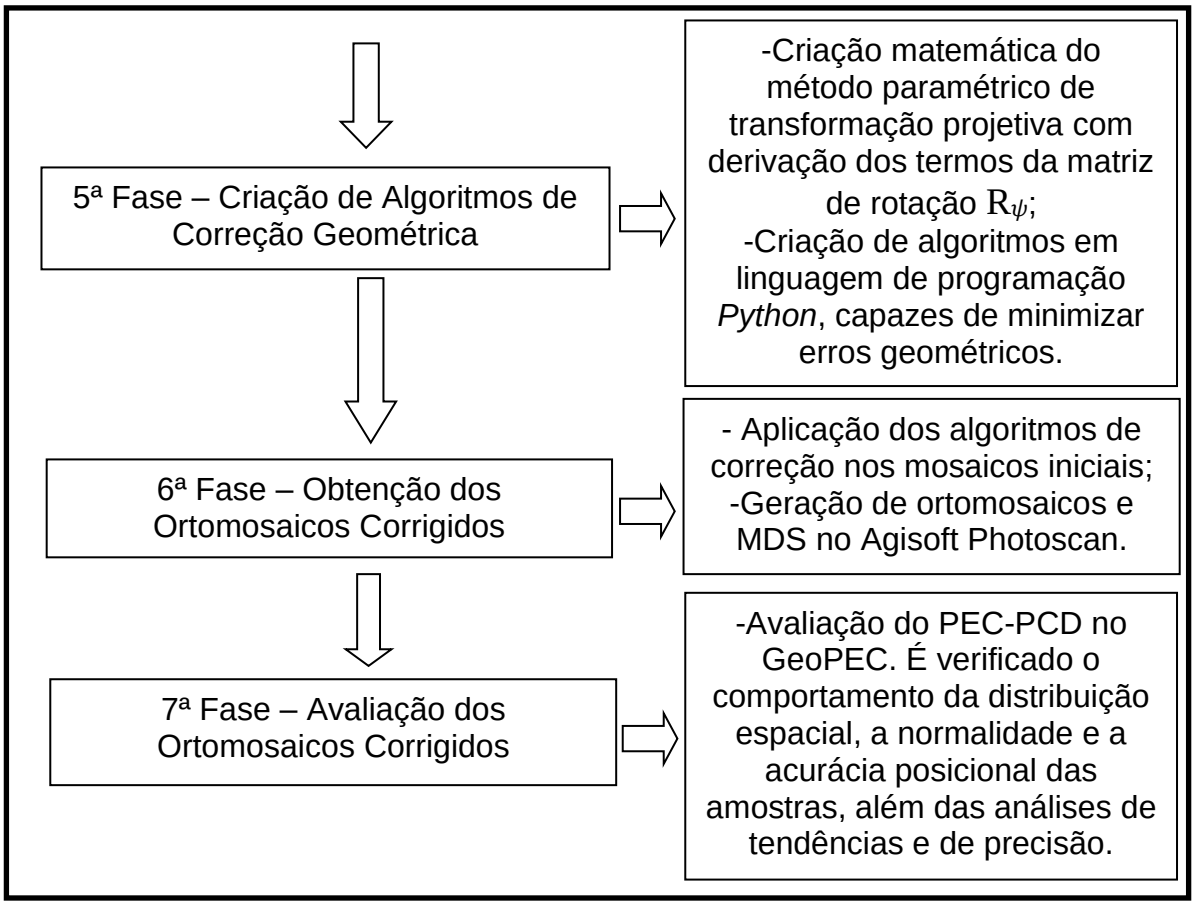


Figura 7. 5ª a 7ª fase do método. Fonte: Elaborada pelo autor.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 AERONAVES REMOTAMENTE PILOTAS (RPAs)

2.1.1 História

Durante a Primeira Guerra de Independência Italiana, em 1848-1849, os austríacos, que controlavam a maior parte da Itália, produziam balões autônomos equipados com bombas para atacar Veneza (NARDINI, 2016). Na guerra hispano-americana, em 1898, os americanos utilizaram uma câmera em uma pipa para observação da área inimiga (MUNARETTO, 2017). Porém, como origem de fato das RPA, tem-se o Curtiss/Sperry *Flying Bomb* da Marinha Americana. Tratava-se de um míssil primitivo que foi construído em 1918. No entanto, foi a Inglaterra, a partir de um navio, que controlou pela primeira vez remotamente três aeronaves Fairey Queen (MUNARETTO, 2017).

A partir de 1935, surgiram os *target drones,* modelos RP-1 a RP-4, como ferramentas de treinamento para a artilharia antiaérea do Exército Americano (MUNARETTO, 2017). Da Segunda Guerra Mundial, em que as bombas alemãs V1 e V2 e as aeronaves americanas B17, B24 e PB4Y-1 eram controladas remotamente por rádio, até as guerras da Coréia e do Vietnã, houve um grande avanço no que diz respeito ao controle de sistemas não-tripulados (ALMEIDA, 2012). Israel desenvolveu as RPAs Scout e Pioneer como plataformas leves e de tamanho reduzido. O Scout transmitia vídeos em tempo real em 360 graus da área de interesse (MUNARETTO, 2017). Desde 1982, as RPAs têm sido utilizadas nos conflitos militares e na guerra global contra o terrorismo, promovida principalmente pelos EUA (MUNARETTO, 2017).

A evolução das RPAs ocorreu principalmente por causa do potencial de uso em operações militares (NOVAIS, 2011; OLIVEIRA & BRITO, 2019). Atualmente, na área comercial, as RPAs permitem voos regulares, permitindo que produtores possam tomar decisões confiáveis, economizando tempo e dinheiro em comparação ao uso de dados de satélites e de aeronaves comerciais (JORGE et al., 2019).

2.1.2 Conceito e nomes

RPA é uma sigla em inglês de *Remotely Piloted Aircraft* (aeronave remotamente pilotada). Trata-se de aeronave não tripulada, pilotada a partir de uma estação de pilotagem remota e que é utilizada para propósitos que não são recreativos (ZANETTI et al., 2017). Com a evolução das plataformas de sistemas sensores, sejam elas orbitais ou aerotransportadas, surgiu a necessidade da criação de uma plataforma que reduzisse os custos de aerolevantamento para imageamento óptico, de radar, a laser, multiespectral e geofísico (JENSEN, 2009). Desta forma, os Sistemas de Aeronaves Não Tripuladas (*Unmanned Aircraft Systems* – UAS, em inglês) (BRASIL, 2018) têm sido utilizados como plataforma de baixo custo na atividade de aerolevantamento, principalmente no imageamento óptico (ZANETTI, 2017).

No Brasil, as aeronaves não tripuladas são conhecidas como drones, do inglês Zangão, Veículos Aéreos Não Tripulados (VANT) (*Unmanned Aerial Vehicle* – UAV, em inglês) e, ainda, Aeronave Remotamente Pilotada (ARP). O termo adotado tecnicamente pela Organização de Aviação Civil Internacional, com abrangência internacional, para esse tipo de aeronave, é o *Remotely Piloted Aircraft System* (RPAS) (BRASIL, 2018) (**Figura 8**).

O controle do acesso das RPAs no espaço aéreo brasileiro cabe ao Departamento de Controle do Espaço Aéreo (DECEA), órgão central do Sistema de Controle do Espaço Aéreo Brasileiro (BRASIL, 2018). Compete à ANAC administrar o Registro Aeronáutico Brasileiro, com as funções de efetuar o registro das RPAs, bem como de emitir certificados de matrícula e de aeronavegabilidade das RPAs (BRASIL, 2018).

Figura 8. Imagem da RPA da Força Aérea Brasileira.
Fonte: http://www.defesaaereanaval.com.br/esquadrao-horus-12gav.

2.1.3 Aplicações da RPA

As aplicações do uso da RPA parecem ser, a princípio, infinitas. Dentre as principais aplicações, podem ser destacadas (BOERY, 2016):

- Controle de lavouras;
- Aplicação de inseticidas;
- Controle de tráfego;
- Exploração mineral;
- Filmagens cinematográficas;
- Levantamento topográfico;
- Monitoramento de desmatamento;
- Investigação de acidentes, avaliação e gestão de desastres;
- Medição de imóveis para fins de cobrança do IPTU;
- Medição de radioatividade;
- Monitoramento de linhas de transmissão de energia, de distribuição de gás, de óleo e de água;
- Patrulhamento de fronteiras;
- Pesquisa e levantamento de dados;
- Proteção da fauna e flora;

- Reflorestamento;
- Serviços meteorológicos;
- Transporte de cargas;
- Vigilância, monitoramento e controle de incêndios florestais; e
- Vigilância territorial e marítima.

2.1.4 Legislação sobre a RPA

Considerando várias características das RPAs como asas fixas, asas rotativas, dirigíveis e ornitópteros, tamanhos, desempenhos e aplicações, a regulamentação para o emprego de RPAs tem-se mostrado complexa e desafiadora em todas as partes do mundo, principalmente pelo fato de não haver piloto a bordo (BRASIL, 2018). Segundo o Ministério da Defesa, as legislações atuais que tratam de forma detalhada sobre homologação e autorização de voo da RPA são (BRASIL, 2018):

- RBAC E nº 94 de 02 de maio de 2017: requisitos gerais para aeronaves não tripuladas de uso civil;
- Resolução ANAC n° 419 de 02 de maio de 2017: aprova o Regulamento Brasileiro de Aviação Civil Especial n° 94;
- ICA 100-40 de 22 de dezembro de 2016: Sistemas de Aeronaves Remotamente Pilotadas e o acesso ao espaço aéreo brasileiro;
- Resolução ANAC nº 377 de 15 de março de 2016: regulamenta a outorga de serviços aéreos públicos para empresas brasileiras e dá outras providências;
- Portaria Normativa nº n° 101/GM-MD, de 26 de dezembro de 2018: dispõe sobre adoção de procedimentos para a atividade de aerolevantamento no território nacional;
- ICA 63-13 de 11 de novembro de 2013: procedimentos dos órgãos do SISCEAB relacionados com AVOEM, AVANAC e AVOMD;
- Decreto nº 7.845 de 14 de novembro de 2012: regulamenta procedimentos para credenciamento de segurança e tratamento de informação classificada em qualquer grau de sigilo e dispõe sobre o Núcleo de Segurança e Credenciamento;

- Lei nº 11.182 de 27 de setembro de 2005: cria a Agência Nacional de Aviação Civil (ANAC) e dá outras providências;
- Decreto nº 2.278 de 17 de julho de 1997: regulamenta as atividades de aerolevantamento no território nacional;
- Lei nº 7.565 de 19 de dezembro de 1986: Código Brasileiro de Aeronáutica; e
- Decreto-Lei nº 1.177 de 21 de junho de 1971: dispõe sobre aerolevantamentos no território nacional e dá outras providências.

2.1.5 Utilização da RPA no aerolevantamento óptico

Conforme o Decreto-Lei n° 1.177 de 21 de junho de 1971, artigo 3°, o aerolevantamento consiste em um conjunto de operações aéreas e/ou espaciais de medição, computação e registro de dados do terreno a partir do emprego de sensores e/ou equipamentos adequados (fase aeroespacial), bem como a interpretação dos dados levantados ou a sua tradução sob qualquer forma (fase decorrente), gerando mapas e cartas cartográficas.

Segundo a Portaria n° 101/GM-MD, de 26 de dezembro de 2018, artigo 10°, a organização que realiza todas as fases do aerolevantamento, aeroespacial e decorrente, será classificada como categoria A e deve estar inscrita no Ministério da Defesa. As empresas que oficialmente estão trabalhando com RPA para fins de aerolevantamento fotogramétrico digital realizam inscrição no Ministério da Defesa, disponível em: <http://www.defesa.gov.br/cartografia-e-aerolevantamento-claten>.

A fotogrametria digital é uma ciência que tem como objetivo a reconstrução de um espaço tridimensional (3D) a partir de imagens bidimensionais. O ortomosaico é um importante produto da fotogrametria digital (COELHO e BRITO, 2016).

A RPA é uma tecnologia atual que permite um aerolevantamento com baixo custo. A alta qualidade de imagem obtida é um fator significativo para a eficiência e a qualidade dos produtos de mapeamento tais como MDE, que representam o modelo digital de superfície (MDS) exprimindo altitudes, necessários para geração de ortomosaicos. A qualidade de um MDE e de um ortomosaico depende principalmente da resolução da câmera, da altura de voo e da precisão dos pontos de controle do terreno (BERTESKA e RUZGIENÈ, 2013).

Dessa forma, é necessário avaliar os produtos obtidos por RPAs. O padrão brasileiro de acurácia posicional para dados espaciais é definido pelo Decreto-Lei n° 89.817 de 1984, de acordo com as tolerâncias definidas no PEC e EP (SANTOS et al., 2016). Tais tolerâncias têm seus valores definidos em função da escala de avaliação dos dados espaciais e das classes A, B ou C, definidas por esse Decreto-Lei. Em 2010, a Diretoria do Serviço Geográfico do Exército Brasileiro publicou a ET-ADGV, documento este ligado ao INDE, criado em 2008 pelo Decreto-Lei nº 6.666. Em um de seus itens, a ET-ADGV explica como deve ser a aplicação do Decreto-Lei nº. 89.817 e cria uma classe mais restritiva destinada para o PEC-PCD (OLIVEIRA & BRITO, 2019).

2.2 FOTOGRAMETRIA DIGITAL

2.2.1 Generalidades

A fotogrametria digital pode ser definida como a reconstrução de um espaço tridimensional, chamado de espaço-objeto, a partir de um conjunto não vazio de imagens bidimensionais digitais, chamado de espaço-imagem (**Figura 9**) (COELHO e BRITO, 2016).

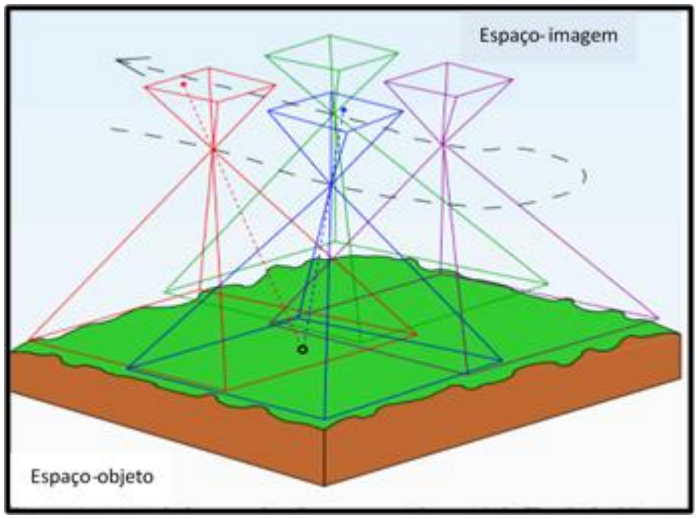


Figura 9. Reconstrução de um espaço tridimensional (espaço-objeto), a partir de um conjunto não vazio de imagens bidimensionais digitais (espaço-imagem). Fonte: Vermeer e Ayehu (2017).

Há dois sistemas relacionados: um sistema bidimensional próprio de cada câmera, com origem aproximadamente no centro de seu quadro, e de coordenadas determinadas por calibração da câmera em laboratório; e outro sistema tridimensional, mais comum e que representa o sistema de coordenadas do terreno sobre o qual as imagens são obtidas, terreno que pode estar sendo representado em coordenadas geodésicas (latitude: λ; longitude: φ; e altura: *II* ou altitude: *h*), planialtimétricas do tipo transversa de Mercator (leste: *E*, norte *N* e altura: *H* ou altitude: *h*) ou cartesianas (*X*, *Y* e *Z*). A diferença entre altura e altitude reside no fato de a primeira estar referenciada a uma figura geométrica (elipsóide de revolução) e a última, a uma figura geofísica (geóide) (ANDRADE, 1998).

Também é necessário um conjunto de GCP, que são expressos no espaço-objeto. Uma vez locados no espaço-imagem, têm-se os parâmetros de entrada para a dedução da função que mapeia um sistema no outro (VERMEER e AYEHU, 2017). Os GCPs são a materialização do referencial com que se deseja trabalhar (ANDRADE, 1998). Os GCPs devem ser feições bem definidas no terreno e na imagem, geralmente de grande contraste espectral em relação aos seus arredores na imagem (ARAÚJO et al., 2007). Cerca de seis a nove pontos são necessários, dependendo dos dados disponíveis, da qualidade dos GCPs e do tamanho da área estudada (JACOBSEN, 2003). Com a utilização das RPAs, recomenda-se o uso de 4 a 5 pontos de controle por km^2 (COVENEY e ROBERTS, 2017; SOUZA, 2018). Geralmente quanto maior número de GCPs, melhores são os resultados finais (WOLF et al., 2014).

A fototriangulação ou aerotriangulação permite a determinação de coordenadas de pontos num referencial específico (ANDRADE, 1998). Por meio desta técnica, a partir de apenas alguns pontos de apoio, é possível gerar uma infinidade de outros, com precisões aceitáveis para que sejam utilizados como se fossem de apoio (SOUZA, 2018). A aerotriangulação permite a economia de tempo na produção de dados cartográficos digitais, além de redução dos custos de produção das cartas topográficas. Para a execução da aerotriangulação, são necessários o conhecimento prévio das coordenadas dos centros de perspectiva (CCP) no espaço-objeto e os respectivos ângulos de Euler, ou ângulos de atitude, da câmara aérea para cada imagem adquirida. A aerotriangulação utiliza o método dos feixes perspectivos,

como é o caso do método dos mínimos quadrados (MMQ), que minimiza a função que quantifica os resíduos do ajustamento (COELHO e BRITO, 2016).

Para que a fotogrametria digital atinja resultados satisfatórios, deve-se dispor de dados de boa qualidade. Esses dados são necessários na determinação dos pontos de controle, bem como na obtenção de pontos de teste ou de verificação da qualidade do mapeamento produzido (COELHO e BRITO, 2016). Os pontos de verificação são utilizados no processo de avaliação da qualidade posicional dos produtos gerados e são, na maioria das vezes, coletados em campo (OLIVEIRA et al., 2018).

É relevante que o recobrimento aéreo por meio de imagens ópticas no sentido longitudinal tenha pelo menos duas tomadas de ângulos diferentes, com área de superposição, a fim de viabilizar a visão estereoscópica, tridimensional, que possibilita acurácias e precisões (MONICO et al., 2009) na restituição tridimensional do espaço-objeto. Para a fotogrametria digital tradicional, os recobrimentos longitudinais e laterais devem estar na ordem dos 60% e dos 30%, respectivamente, (**Figura 10**) (MOHLER, 1968). O recobrimento longitudinal e lateral para as RPAs devem estar na ordem dos 80% e dos 60%, respectivamente (MUNARETTO, 2017). Uma sobreposição longitudinal maior que 50% permite a configuração de von Gruber para a marcação de pontos de ligação (SOUZA, 2018). Entretanto, o relevo da superfície terrestre apresenta descontinuidades e há movimentos irregulares na plataforma de voo, o que torna praticamente impossível o mapeamento automático.

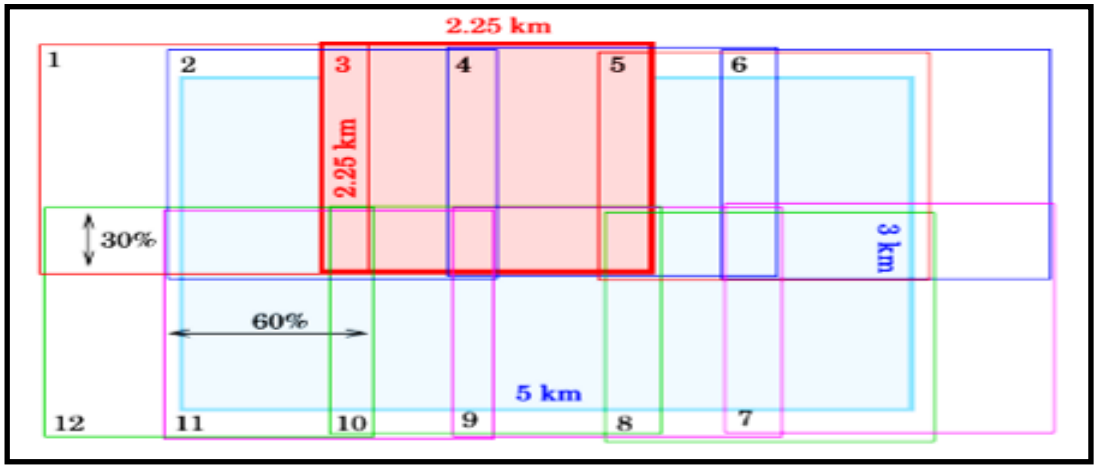


Figura 10. Exemplo de recobrimento aéreo longitudinal e lateral de uma sequência de fotos obtidas por duas linhas de voo adjacentes. Fonte: Vermeer e Ayehu (2017).

2.2.2. Produção do ortomosaico

O processo de ortorretificação, que transforma imagens obtidas com perspectiva central para perspectiva ortogonal (**Figura 11**), minimiza as distorções provocadas pelas irregularidades da superfície terrestre e geradas pela instabilidade da plataforma de voo (ZANETTI et al., 2017). As imagens obtidas por câmeras convencionais encontram-se em perspectiva central, com os inúmeros raios de luz advindos de diferentes pontos imageados, passando por um centro de perspectiva, localizado no sistema óptico da câmera. O conjunto desses raios é chamado feixe perspectivo (ROBERTO et al., 2017). Uma imagem em perspectiva central não pode ser tomada como fonte de informação métrica segura, uma vez que possui erros devido à rotação do sensor e devido ao relevo, inerentes à perspectiva cônica (COELHO e BRITO, 2016).

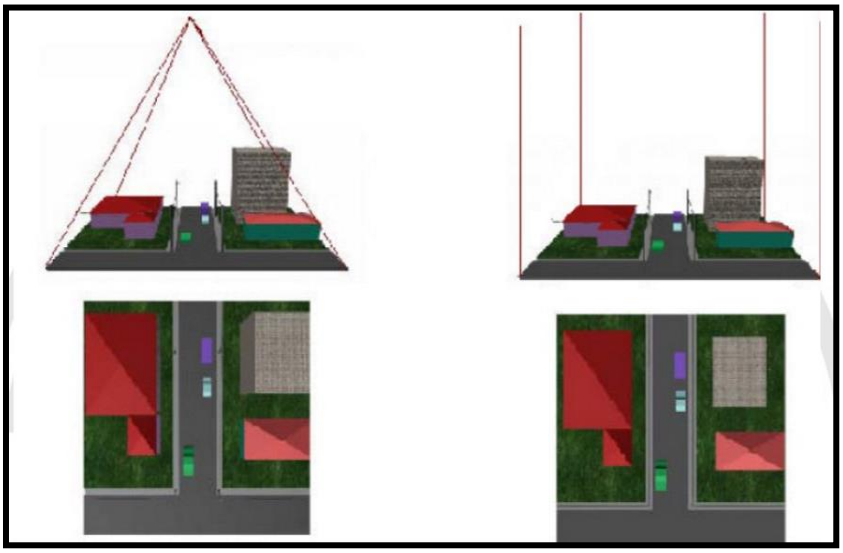


Figura 11. Processo de ortorretificação, que transforma uma imagem em perspectiva central para perspectiva ortogonal. Fonte: Coelho e Brito (2016).

O termo ortoimagem é uma imagem digital baseada em uma ortorretificação. O ortomosaico é uma reunião de ortoimagens (JENSEN, 2009). Outro sinônimo antigo para ortoimagem é o ortofoto digital e, para ortomosaico, é a ortofotocarta. Cabe frisar que o termo imagem, fotografia ou foto se refere a um material analógico e o

termo imagem digital ou simplesmente imagem se refere a um material digital (ANDRADE, 1998).

Em uma projeção ortogonal, raios ortogonais são projetados a partir da região imageada. Os raios nunca se encontram e a imagem final não possui desvios nem distorções relativos ao relevo (JENSEN, 2009). A perspectiva ortogonal é um fenômeno artificial. Para obter ortomosaicos, é necessário realizar uma transformação sobre as imagens já existentes, chamada ortorretificação. A retificação é um processo que retira as distorções relativas à rotação da câmera (ANDRADE, 1998), porém, a ortorretificação, além de realizar a retificação, elimina a distorção relativa ao relevo. A imagem, em projeção ortogonal, ao contrário da projeção central, pode ser tomada como um documento cartográfico (COELHO e BRITO, 2016). A qualidade da ortorretificação depende diretamente da quantidade e distribuição dos GCPs, além do modelo matemático escolhido (ZANETTI et al., 2017).

O processo de geração de ortomosaicos requer os seguintes passos relativos à correção da imagem: orientação interior, orientação exterior e MDE (KOEVA et al., 2016). O aplicativo livre E-Foto dispõe de algoritmos específicos para a orientação interior e exterior, geração de MDE e de ortomosaicos, além de permitir o processo da aerotriangulação. O aplicativo E-Foto pode ser executado tanto em ambiente Linux quanto em Windows, com um mínimo de requisitos de hardware, e encontra-se disponível para *download* no endereço eletrônico http://www.efoto.eng.uerj.br/. O E-Foto é uma iniciativa acadêmica de desenvolvimento de um aplicativo livre para fotogrametria digital. O projeto E-Foto está sendo desenvolvido no laboratório de fotogrametria da Faculdade de Engenharia da Universidade Estadual do Rio de Janeiro (UERJ) desde 2004 (E-FOTO, 2019). O projeto é interdisciplinar, reunindo áreas do conhecimento acadêmico, como a modelagem matemática, Geodésia, princípios da fotogrametria e engenharia de software. É possível modificar seu código fonte e desenvolver novos módulos sobre ele e acrescentar suas próprias contribuições ao projeto (TRAMONTINA et al., 2017).

No aplicativo, deve-se criar o projeto e configurar todas as informações necessárias ao mesmo. Em seguida, é necessário realizar a orientação interior e exterior ou aerotriangulação. Nesta primeira fase de inicialização do projeto, obtêm-

se as imagens orientadas. Na segunda fase de geração de produtos fotogramétricos, pode-se criar um mapa vetorizado a partir de medidas estereoscópicas, um MDE e uma ortoimagem ou um ortomosaico. Por fim, o projeto como um todo permite a integração com um sistema de informações geográficas (SIG). O fluxograma do processamento fotogramétrico no aplicativo E-Foto obedece a uma hierarquia de execução (**Figura 12**). De forma resumida, o aplicativo é composto pelos seguintes módulos: criação do projeto fotogramétrico, orientação interior, orientação exterior, aerotriangulação, *stereoplotter*, MDS ou MDE, ortoretificação e integração com o aplicativo SIG (NUNES e BRITO, 2017).

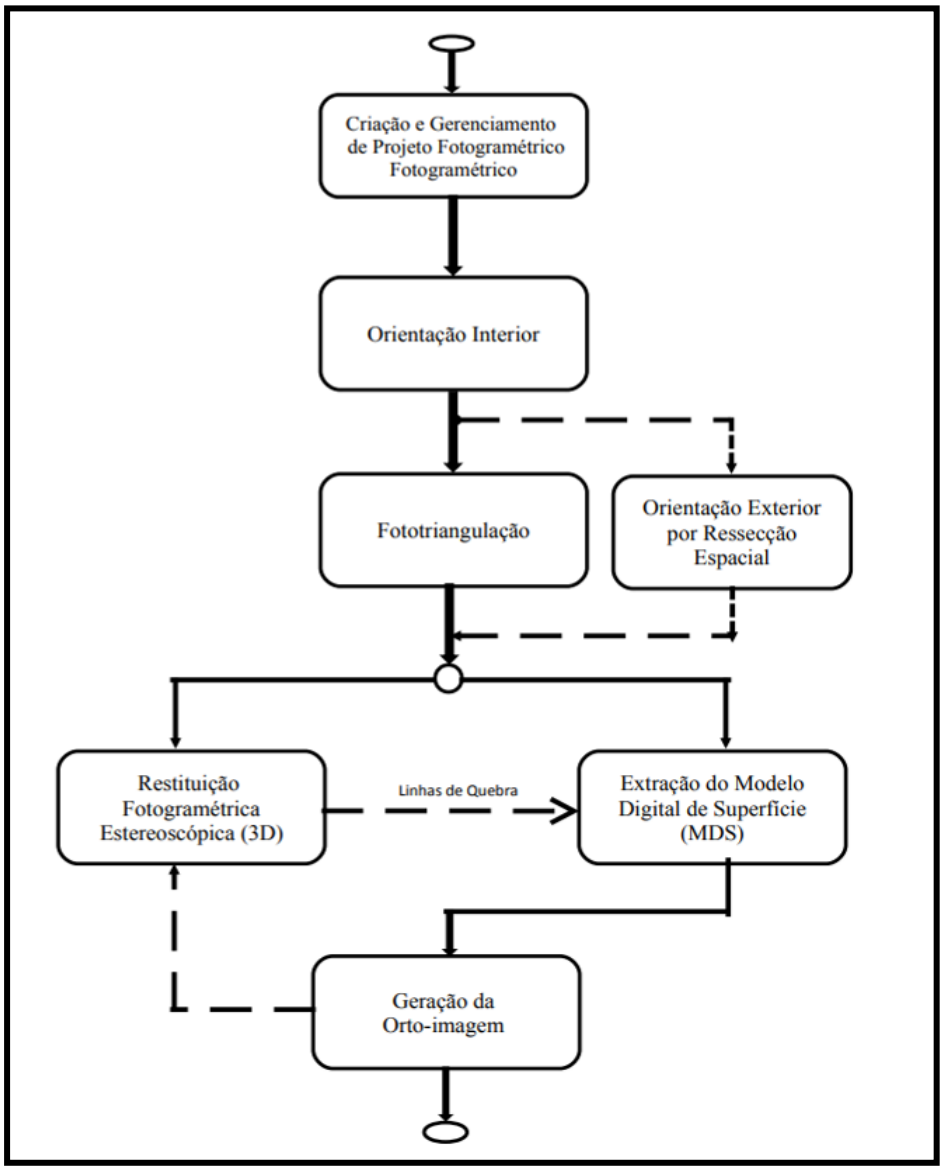


Figura 12. Fluxograma de processos do E-Foto. Fonte: Nunes e Brito (2017).

O aplicativo comercial russo Agisoft Photoscan é utilizado para o processamento das imagens aéreas captadas por RPAs e geração da base cartográfica do terreno. O Agisoft Photoscan realiza a fotogrametria digital e a produção de dados espaciais em 3D, podendo ser utilizado em aplicações SIG. Sua tecnologia usa a técnica de fotogrametria digital, com métodos de visão por computador e isso resulta em um sistema de processamento automatizado e inteligente (AGISOFT, 2019).

O processamento das imagens e a construção do modelo 3D no Agisoft PhotoScan compreendem quatro fases (AGISOFT, 2019):

- A primeira fase é o alinhamento da câmara. Procura-se por pontos em comum em imagens e combina-os. Nesta fase, forma-se uma nuvem de pontos.
- A segunda fase é a construção da nuvem de pontos densa. Com base nas posições da câmera estimadas e das próprias imagens, uma nuvem de pontos densa é definida.
- A terceira fase é a construção de uma malha. É construída uma malha poligonal 3D que representa a superfície de objeto com base na nuvem de pontos densa.
- Por fim, na quarta fase, a malha pode ser texturizada e utilizada para a geração de ortomosaicos.

O fluxo de trabalho inclui as seguintes etapas no Agisoft PhotoScan: carregar imagens; inspecionar imagens carregadas; remover imagens desnecessárias; alinhar imagens; formar nuvens de pontos densa; gerar malha, modelo poligonal 3D; criar textura; construir modelos em mosaico; produzir MDE; originar ortomosaico; e exportar resultados (AGISOFT, 2019).

2.2.2.1 Orientação interior

A orientação interior corresponde à reconstrução do feixe perspectivo, ou seja, o referenciamento da imagem em relação à câmera (ANDRADE, 1998). O sistema câmera-imagem é composto basicamente de uma perpendicular ao centro da imagem e mede um comprimento igual à distância focal calibrada. A orientação interior consiste apenas em colocar as imagens, uma a uma, em posição

semelhante à que exerciam dentro da câmera, no momento em que foram obtidas (VERMEER e AYEHU, 2017) (**Figura 13**).

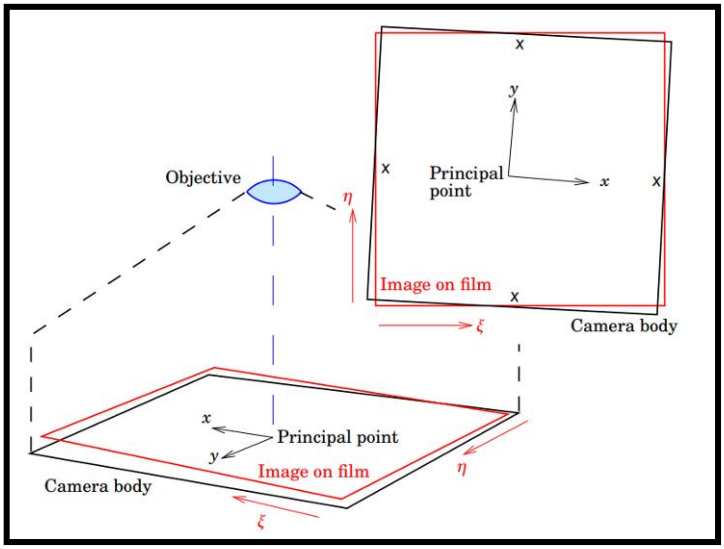


Figura 13. Representação do corpo da câmera, coordenadas da câmera (x,y) e coordenadas da imagem (η, ξ). Fonte: Vermeer e Ayehu (2017).

A orientação interior é o processo pelo qual se pode recuperar a referência do sistema de coordenadas da imagem de volta para o sistema de coordenadas métricas da câmera fotogramétrica. Primeiramente, isso é possível por meio da medição das marcas fiduciais da imagem, em milímetros, obtidas nas câmeras analógicas aéreas verticais em aeronaves. Entretanto, nas imagens provenientes de câmeras digitais aéreas verticais transportadas por RPAs, por meio do sensor *Complementary Metal-Oxide-Semiconductor* (CMOS), não métricas, normalmente não são encontradas as marcas fiduciais. Nesses casos, basta que se trabalhe com as dimensões do sensor (em mm) e do respectivo tamanho do seu *pixel* (em μm) (TOMMASELLI et al., 2010). A transformação entre *pixels* e milímetros serve para corrigir vários erros de aquisição das imagens, tais como efeitos atmosféricos não-modelados durante a aquisição da imagem digital via câmera, que causam deformações na imagem (rotações, translações e fatores de escala)(COELHO e BRITO, 2016).

Além das dimensões do sensor e do tamanho do *pixel*, os parâmetros de orientação interior para imagens ópticas obtidas da RPA são basicamente: a

distância focal (em milímetros); a distância entre o ponto de convergência da luz até o ponto onde a imagem será projetada; as coordenadas do ponto principal (x, y) (em mm); os coeficientes de distorção radial simétrica (k_1, k_2 e k_3), a distorção gerada pelo deslocamento radial de um ponto na imagem de sua posição correta, ou seja, uma mudança no ângulo entre o raio de luz e o eixo óptico, causado pela refração sofrida pelo raio de luz ao atravessar o sistema óptico; e os coeficientes da distorção descentrada (P_1, P_2), distorção criada pela impossibilidade de alinhamento entre os eixos ópticos das lentes que compõem o sistema de lentes, causando um deslocamento na posição de um ponto na imagem. Todos os parâmetros possuem unidade em milímetros. Esses dados são geralmente obtidos no certificado de calibração da câmera (CAMPOS et al., 2015).

Um modelo matemático simples de correção seria uma transformação de similaridade (ANDRADE, 1998) (Eq. 1):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + k \cdot \begin{bmatrix} cos\theta & sen\theta \\ -sen\theta & cos\theta \end{bmatrix} \cdot \begin{bmatrix} \xi \\ \eta \end{bmatrix} \quad (1)$$

Onde: x e y representam as coordenadas da câmera (ponto principal); x_0 e y_0 na matriz representam o vetor de transformação (descreve o deslocamento entre as coordenadas da imagem e as coordenadas da câmera); η e ξ representam as coordenadas da imagem; K é o fator de escala (lado da imagem em milímetros dividido pelo lado da imagem em *pixels*); e Θ é o ângulo de rotação.

Outra maneira de realizar a orientação interior é considerar os fatores de escala Cx e Cy (COELHO e BRITO, 2016) (Eqs. 2 e 3):

$$Cx = \frac{lado\ da\ foto\ em\ milimetros}{lado\ da\ foto\ em\ pixels} \frac{mm}{px} \quad (2)$$

$$Cy = -\frac{lado\ da\ foto\ em\ milimetros}{lado\ da\ foto\ em\ pixels} \frac{mm}{px} \quad (3)$$

Além do *Cx* e *Cy*, pode-se seguir com a formulação genérica, que considera a existência simultânea dos seis parâmetros (**Figura 14**) (ANDRADE, 1998) (transformação afim, Eqs. 4 e 5):

$$x = Cx.cos\alpha.coluna + Cy.sen\alpha.linha + X_0 \quad (4)$$

$$y = -Cx.sen(\alpha + \varepsilon).coluna + Cy.cos(\alpha + \varepsilon).linha + Y_0 \quad (5)$$

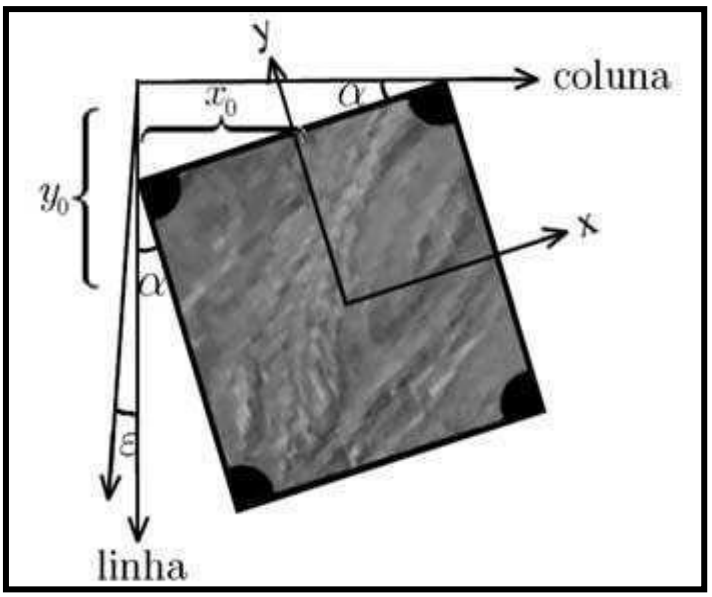


Figura 14. Transformação afim. Fonte: Coelho e Brito (2016).

As expressões acima podem ser descritas de forma linear (COELHO e BRITO, 2016) (Eqs. 6 e 7):

$$x = a_0 + a_1.coluna + a_2.linha \quad (6)$$

$$y = b_0 + b_1.coluna + b_2.linha \quad (7)$$

Desta forma, o modelo matemático mais utilizado é a transformação afim (ANDRADE, 1998) (Eq. 8), na forma matricial, que modela seis incógnitas (parametros a_0, b_0, a_1, b_1, a_2 e b_2), considerando que o sistema de imagem digital apresenta as seguintes características: não-ortogonalidade dos eixos, rotação da

imagem, translação em *x* e *y* e escalas diferentes em *x* e *y*. Esta formulação serve para a resolução de um sistema de equações lineares (COELHO e BRITO, 2016):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & coluna & linha & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & coluna & linha \end{bmatrix} . \begin{bmatrix} a_0 \\ b_0 \\ a_1 \\ b_1 \\ a_2 \\ b_2 \end{bmatrix} \quad (8)$$

Onde: *x* e *y* representam as marcas fiduciais; coluna e linha da imagem; e a_0, b_0, a_1, b_1, a_2 e b_2 representam as seis incógnitas de transformação entre os dois sistemas.

Ainda se faz necessária a eliminação das distorções radial simétrica (*k*) e descentrada (*p*). Para a distorção radial simétrica, as equações são do tipo polinomial (CAMPOS et al., 2015) (Eqs. 9 a 12):

$$\delta x = x''.(k_0 + k_1.r^2 + k_2.r^4 + k_3.r^6) \quad (9)$$

$$\delta y = y''.(k_0 + k_1.r^2 + k_2.r^4 + k_3.r^6) \quad (10)$$

$$x' = x'' - \delta x \quad (11)$$

$$y' = y'' - \delta y \quad (12)$$

Onde: δx e δy são as componentes da distorção radial simétrica; *r* é o raio a partir do ponto principal de simetria, corresponde ao dobro da distância focal; k_0, k_1, k_2 e k_3 são os coeficientes que constam do certificado de calibração de câmara; *x"* e *y"* são as coordenadas do ponto sem correção, referidas ao ponto principal de simetria; e *x'* e *y'* são as coordenadas corrigidas da distorção radial simétrica.

Para a distorção descentrada, as equações são do tipo polinomial (CAMPOS et al., 2015) (Eqs. 13 a 16):

$$\delta x' = p_1.(r^2 + 2x''^2) + 2p_2x''y'' \quad (13)$$

$$\delta y' = p_2.(r^2 + 2y''^2) + 2p_1x''y'' \quad (14)$$

$$x = x' - \delta x' \quad (15)$$

$$y = y' - \delta y' \quad (16)$$

Onde: δx' e δy' são as componentes da distorção descentrada; *r* é o raio a partir do ponto principal de simetria, corresponde ao dobro da distância focal; p_1 e p_2 são os coeficientes que constam do certificado de calibração de câmara; *x"* e *y"* são as coordenadas do ponto sem correção, referidas ao ponto principal de simetria; *x'* e *y'* são as coordenadas corrigidas da distorção radial simétrica; e *x* e *y* são as coordenadas corrigidas das duas distorções.

O aplicativo E-Foto permite a execução da orientação interior com algoritmos que contêm as Eqs. 8 a 16. A **Figura 15** apresenta a tela do E-Foto que possibilita a inclusão de dados para a orientação interior. Nessa fase do E-Foto, é possível obter o vetor dos parâmetros ajustados e sua respectiva matriz variância-covariância. No final do processo, tem-se um relatório dos resíduos da transformação afim (VERMEER e AYEHU, 2017).

2.2.2.2 Orientação exterior

A orientação exterior (**Figura 16)** é a obtenção da posição e da atitude do sensor ao coletar cada imagem em relação ao referencial do espaço-objeto. Uma imagem está orientada exteriormente se são conhecidos os seus seis parâmetros de orientação exterior: as coordenadas no espaço-objeto (*E, N* e *U*) para o centro de perspectiva e os ângulos de rotação ou de atitude do sensor (*φ, ω e κ*) (ANDRADE, 1998).

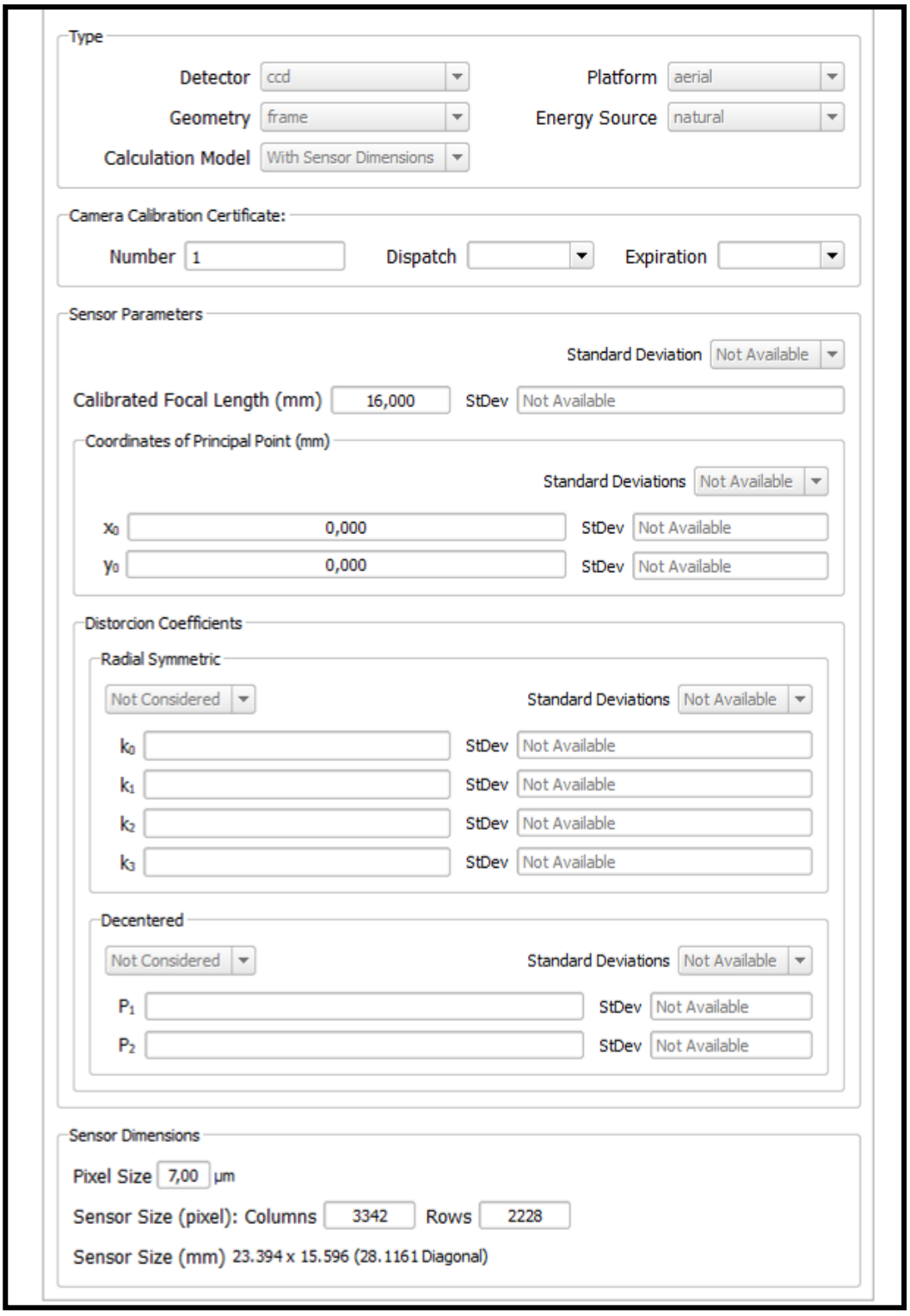


Figura 15. Dados do sensor para orientação interior no E-Foto.

Fonte: http://www.efoto.eng.uerj.br.

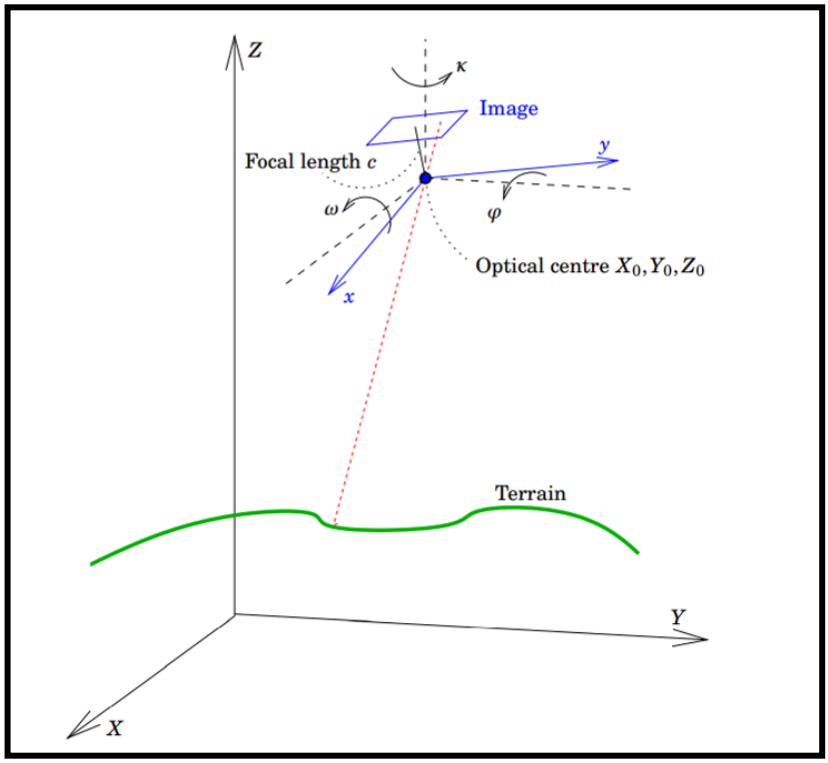


Figura 16. Elementos de orientação exterior. Fonte: Vermeer e Ayehu (2017).

Os ângulos de Euler (*φ*, *ω* e *κ*) significam rotações sofridas pelo sistema local de coordenadas X_m, Y_m e Z_m (de cada câmera) em relação ao referencial do terreno (*E*, *N* e *U*), sistema fotogramétrico. Rotacionando-se X_m, Y_m e Z_m de -*φ*,- *ω* e –*κ*, é possível torná-los paralelos a *E*, *N* e *U*. Já *ω* representa a rotação do eixo X_m, em relação a *E* (WARSI et al., 2013), enquanto *φ* representa a rotação do eixo Y_m em relação a *N*. Esses ângulos devem ser pequenos, não devendo ultrapassar 5º em valor absoluto, no caso de imagens perfeitamente verticais. Por fim, *κ* representa a rotação do eixo Z_m em relação a *U* (ROBERTO et al., 2017) (**Figura 17**).

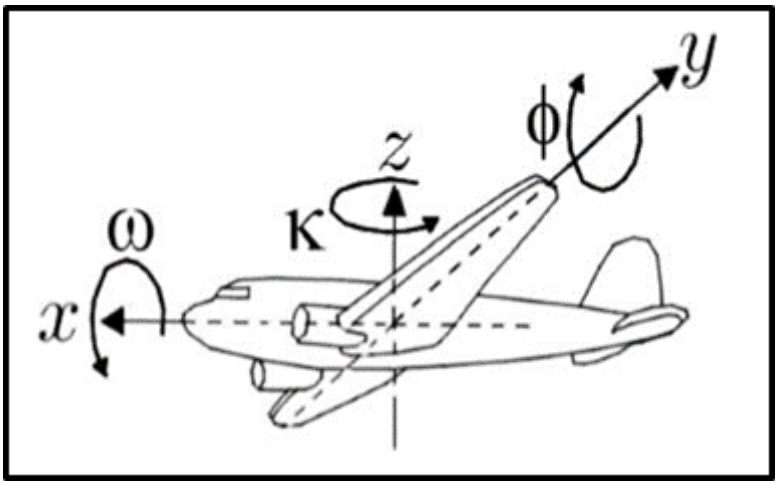


Figura 17. Parâmetros de atitude de um sensor colocado em plataforma aérea.
Fonte: Coelho e Brito (2016).

A matriz de rotação equivalente a cada ângulo expressa a transformação necessária para rotacionar um sistema em relação a outro de tal ângulo. Multiplicando-se todas, pode-se obter a matriz de rotação *R* que equivale aos três movimentos simultaneamente (ANDRADE, 1998; RUZGIENE, 2014) (Eqs. 17 a 19):

$$R_\varphi = \begin{bmatrix} cos\varphi & 0 & -sen\varphi \\ 0 & 1 & 0 \\ sen\varphi & 0 & cos\varphi \end{bmatrix} \quad (17)$$

$$R_\omega = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\omega & sen\omega \\ 0 & -sen\omega & cos\omega \end{bmatrix} \quad (18)$$

$$R_\kappa = \begin{bmatrix} cos\kappa & sen\kappa & 0 \\ -sen\kappa & cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (19)$$

Desta forma, tem-se a Eq. 20 (ANDRADE, 1998; RUZGIENE, 2014):

$$R = R_\kappa . R_\varphi . R_\omega \quad (20)$$

As Eqs. 21 e 22 representam as formas completa e simplificada de *R*, respectivamente (ANDRADE, 1998; RUZGIENE, 2014):

$$R=\begin{bmatrix} cos\varphi cos\kappa & cos\omega sen\kappa + sen\omega sen\varphi cos\kappa & sen\omega sen\kappa - cos\omega sen\varphi cos\kappa \\ -cos\varphi sen\kappa & cos\omega cos\kappa - sen\omega sen\varphi sen\kappa & sen\omega cos\kappa + cos\omega sen\varphi sen\kappa \\ sen\varphi & -sen\omega cos\varphi & cos\omega cos\varphi \end{bmatrix} \quad (21)$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (22)$$

R rotaciona um terno de coordenadas do espaço-imagem para o espaço-objeto. Já *M*, que é igual a R^{-1} ou R^{T}, rotaciona um terno de coordenadas do espaço-objeto para o espaço-imagem. Nota-se que a igualdade $R^{-1} = R^{T}$ é válida se a matriz *R* for ortogonal, que é o caso. *R* é o produto de três matrizes ortogonais (R_{φ}, R_{ω} e R_{κ}), logo, *R* é ortogonal (COELHO e BRITO, 2016).

O desejável na tomada de uma imagem aérea é que a visada seja a mais próxima do nadir. Entretanto, os RPAs, por serem leves, são bastante susceptíveis a turbulências e nem sempre possuem plataformas giro-estabilizadas para manter visada a nadir, principalmente os de asa fixa, de modo que apresentam valores off-nadir consideráveis. A **Figura 18** apresenta os efeitos das atitudes *roll, pitch* e *yaw* de uma plataforma aérea RPA na distorção da imagem da superfície terrestre (ROBERTO et al., 2017; ONUORA et al., 2018).

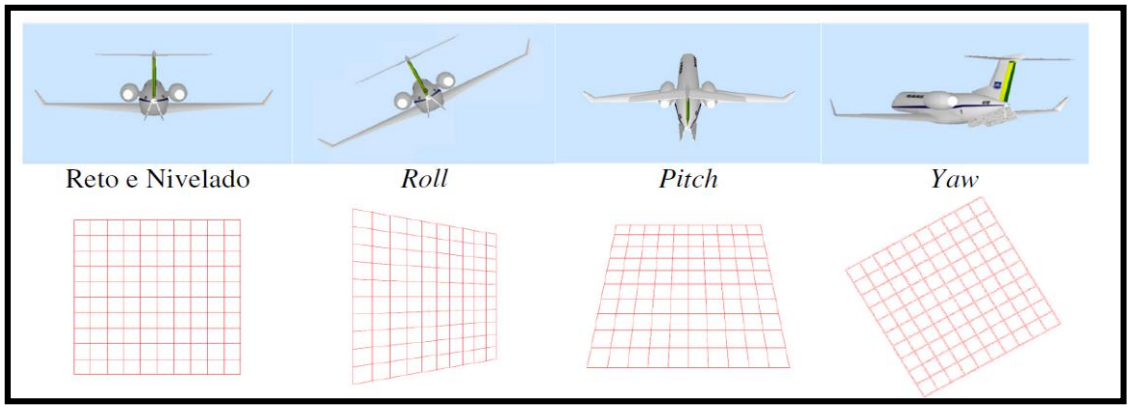


Figura 18. Efeitos das atitudes *roll* (rolamento), *pitch* (arfagem) *e yaw* (guinada) da aeronave na geometria da imagem. Fonte: Roberto et al. (2017).

Concernente aos RPAs, os ângulos de atitude *roll* (rolamento, ϕ) *pitch* (arfagem, θ) e *yaw* (guinada, deriva, azimute, ψ) seguem a definição e são diferentes dos utilizados nos modelos fotogramétricos φ, ω e κ, sendo necessária a conversão do sistema fotogramétrico para o aeronáutico. Os sistemas de navegação de RPAs utilizam, como referência, um sistema de eixos (X_a, Y_a e Z_a) fixos no corpo com origem no centro de gravidade. Já o sistema fotogramétrico baseia-se no sistema de referência da câmera de eixos (X_m, Y_m e Z_m) fixos no corpo, com origem no centro de perspectiva (**Figura 19**). Quando a câmera é alinhada na aeronave, X_m corresponde a Y_a, Y_m a X_a e Z_m a $-Z_a$ (ROBERTO et al., 2017; ONUORA et al., 2018).

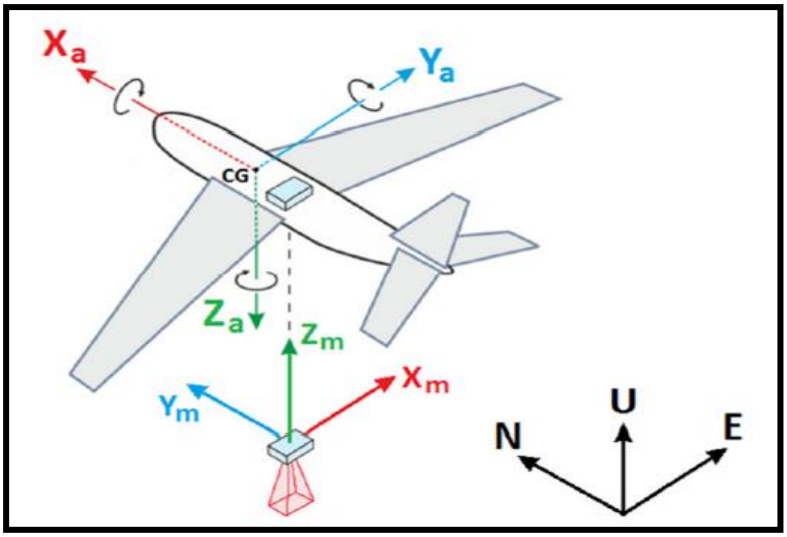


Figura 19. Sistema aeronáutico e fotogramétrico em relação ao ENU.
Fonte: Roberto et al. (2017).

Para o ângulo ψ, o leme é usado como entrada de controle, enquanto o elevador é usado para controlar o ângulo θ e o ângulo ϕ pode ser controlado por meio da deflexão do aileron) (WARSI et al., 2014). Para a conversão do φ, ω e κ (sistema fotogramétrico), para o sistema ϕ, θ e ψ (sistema aeronáutico), deve-se fazer a correspondência entre os sistemas, conforme as Eqs. 23 a 25 (ROBERTO et al., 2017):

$$\theta = atan2(r_{23}\ , \sqrt{r^2{}_{13} + r^2{}_{33}}) \tag{23}$$

$$\phi = atan2\left(\frac{-r_{13}}{cos\theta}, \frac{r_{33}}{cos\theta}\right) \quad (24)$$

$$\psi = atan2\left(\frac{r_{21}}{cos\theta}, \frac{r_{22}}{cos\theta}\right) \quad (25)$$

Todos os pontos que compõem o objeto podem ser representados no espaço-imagem bidimensional pelo princípio da colinearidade, propriedade que indica que, num conjunto de três ou mais pontos, eles estão posicionados de tal forma que se pode traçar uma reta que contenha todos eles (WOLF et al., 2014). Para se chegar à equação de colinearidade, faz-se necessário definir os parâmetros geométricos da câmera e as matrizes de rotação no sistema aeronáutico (ROBERTO et al., 2017). A **Figura 20** mostra os parâmetros geométricos da câmera. Tomando como referência esta figura, é possível extrair as relações matemáticas para a obtenção dos parâmetros geométricos (Eqs. 26 a 29) (ROBERTO et al., 2017):

$$M = \frac{q_l}{L} = \frac{q_c}{C} \quad (26)$$

$$tan\left(\frac{FOV}{2}\right) = \frac{q}{2f} = \frac{Q}{2H} \quad (27)$$

$$GSB = \frac{\mu H}{f} \quad (28)$$

$$D = \frac{Hd}{f} \quad (29)$$

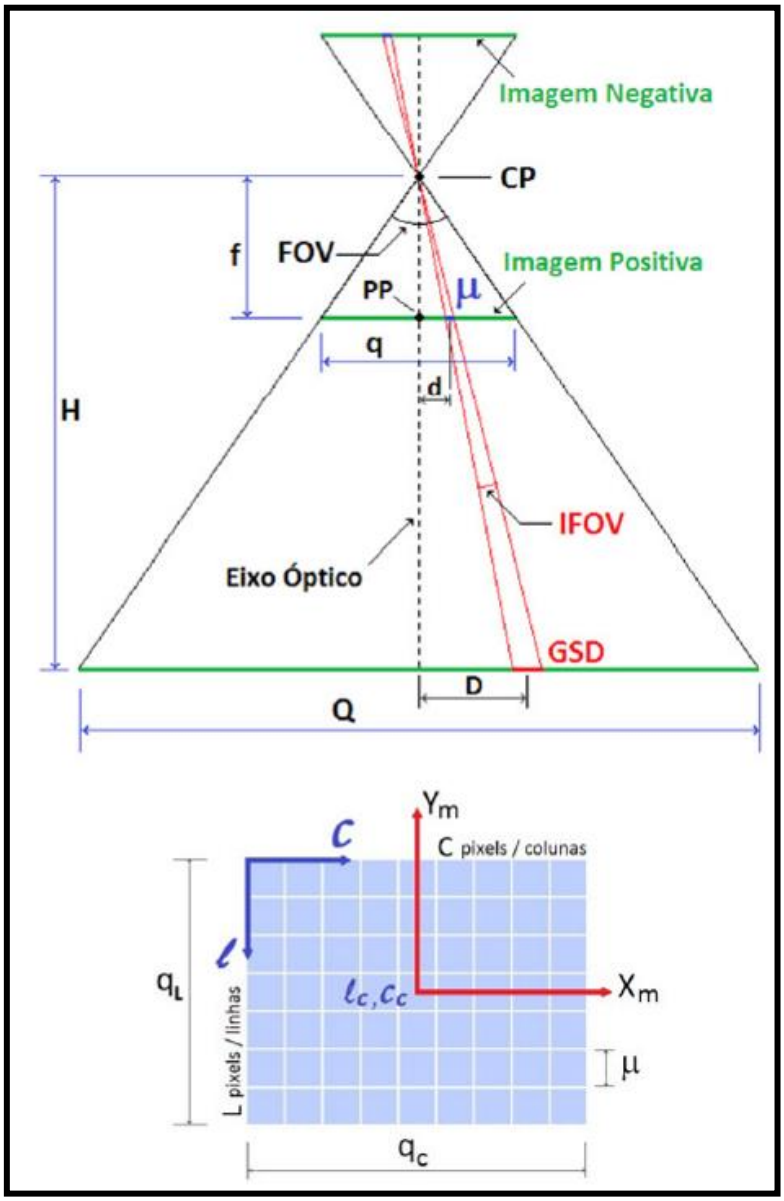


Figura 20. Parâmetros geométricos da câmera. f = distância focal; PP = ponto principal de colimação, interceptação do eixo óptico com o plano da foto; CP = centro de perspectiva, ponto onde se concentra os feixes de luz que saem do terreno e atingem a foto; μ = tamanho do *pixel* da câmera; q = tamanho do quadro da câmera (q_L e q_C); Q = tamanho do quadro da foto projetado no solo (Q_L e Q_C); GSD = *gound sample distance* (tamanho do *pixel* projetado no solo); D = dimensão do objeto no terreno; d = dimensão na foto do objeto com dimensão real D; L e C = quantidade de linhas e colunas da foto, respectivamente; (l, c) = coordenada (linha, coluna) do *pixel*; (l_c, c_c) = coordenada central da câmera, origem do sistema da foto; H = distância projetada do CP no solo, correspondente à altura de voo; FOV (*field of view*) = ângulo de abertura total que representa o campo de visão abrangido pela câmera; e IFOV (*instanteneous field of view*) = ângulo de abertura que representa o campo de visão de um *pixel*. Fonte: Roberto et al. (2017).

As matrizes de rotação do sistema aeronáutico são similares às matrizes do sistema fotogramétrico (Eq. 30 a 32) (ROBERTO et al., 2017):

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sen\phi \\ 0 & -sen\phi & cos\phi \end{bmatrix} \tag{30}$$

$$R_\theta = \begin{bmatrix} -cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & -cos\theta \end{bmatrix} \tag{31}$$

$$R_\psi = \begin{bmatrix} -sen\psi & -cos\psi & 0 \\ cos\psi & -sen\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{32}$$

Dessa forma, tem-se a Eq. 33 (ROBERTO et al., 2017):

$$A = R_\phi \,.\, R_\theta \,.\, R_\psi \tag{33}$$

As Eqs. 34 e 35 representam as formas completa e simplificada de A, respectivamente (ROBERTO et al., 2017):

$$A = \begin{bmatrix} sen\psi cos\theta & cos\psi cos\theta & sen\theta \\ sen\psi sen\theta sen\phi + cos\psi cos\phi & cos\psi sen\theta sen\phi - sen\psi cos\phi & -cos\theta sen\phi \\ sen\psi sen\theta cos\phi - cos\psi sen\phi & cos\psi sen\theta cos\phi + sen\psi sen\phi & -cos\theta cos\phi \end{bmatrix} \tag{34}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{35}$$

Para o sistema aeronáutico, a sequência das rotações nos eixos para transformação do E, N e U para o X_a, Y_a e Z_a é primeiro ψ, depois θ e por último ϕ. A

condição de colinearidade permite a construção da equação de colinearidade. No momento da tomada da imagem, o ponto-objeto P, o centro de projeção CP e o ponto-imagem p formam uma linha reta (**Figura 21**) (COELHO e BRITO, 2016).

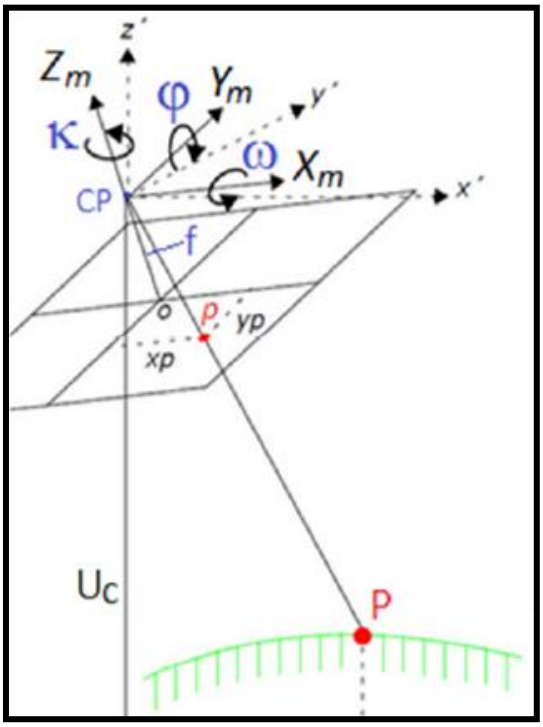


Figura 21. Condição de colinearidade. Fonte: Roberto et al. (2017).).

Por meio da equação de colinearidade, é possível realizar a correção da distorção projetiva, que consiste em reamostrar a imagem manipulando os *pixels* para a situação de visada a nadir. O princípio básico é a relação da Eq. 36. Dividindo-se a primeira e segunda linha pela terceira, resultam as Eqs. 37 e 38 para o sistema aeronáutico (FORLANI et al., 2015; ROBERTO et al., 2017):

$$\lambda \cdot \begin{bmatrix} x_a \\ y_a \\ f \end{bmatrix} = A \cdot \begin{bmatrix} E - E_m \\ N - N_m \\ U - U_m \end{bmatrix} \tag{36}$$

$$x_a = f \cdot \frac{a_{11}(E-E_m)+a_{12}(N-N_m)+a_{13}(U-U_m)}{a_{31}(E-E_m)+a_{32}(N-N_m)+a_{33}(U-U_m)} \tag{37}$$

$$y_a = f.\frac{a_{21}(E-E_m)+a_{22}(N-N_m)+a_{23}(U-U_m)}{a_{31}(E-E_m)+a_{32}(N-N_m)+a_{33}(U-U_m)} \quad (38)$$

Onde: λ = fator de proporcionalidade; x_a e y_a = coordenadas no espaço-imagem; f = distância focal; A = matriz de rotação no sistema aeronáutico; E, N e U = coordenadas no espaço-objeto; e E_m, N_m e U_m = coordenadas no espaço-objeto (coordenadas dos centros respectivos) da posição da câmera.

Por meio dos parâmetros de orientação exterior, é possível estabelecer as coordenadas no terreno. O processo de interseção espacial, que utiliza as equações de colinearidade, para um par de imagens, com os parâmetros da orientação exterior conhecidos, obtém as coordenadas tridimensionais no sistema de espaço-objeto, para qualquer ponto que esteja na área de superposição (COELHO e BRITO, 2016).

O aplicativo E-Foto permite a execução da orientação exterior, com ênfase na ressecção espacial. Nesse método, as coordenadas do centro óptico da câmara e seus ângulos de atitude em relação ao referencial do espaço-objeto são obtidos com, no mínimo, quatro pontos de controle. A ressecção espacial tem, como base, as equações de colinearidade, com ajustamento por mínimos quadrados do modelo paramétrico não linear (VERMEER e AYEHU, 2017). Entretanto, o E-Foto permite a orientação exterior sem a aplicação da ressecção espacial. Nesse caso, o piloto automático do RPA, com *Global Navigation Satellite System* (GNSS) e *Inertial Measurement Unit* (IMU), deve fornecer as coordenadas do centro óptico da câmara e seus ângulos de atitude (KOMAZAKI et al., 2017). A **Figura 22** apresenta a tela do E-Foto que possibilita a inclusão de dados para a orientação exterior. Nessa figura, observa-se que os ângulos de atitude da câmera são do sistema fotogramétrico. É possível ainda obter a matriz de parâmetros da orientação exterior e sua respectiva matriz variância-covariância. No final do processo, tem-se um relatório dos resíduos (VERMEER e AYEHU, 2017).

No aplicativo Agisoft Photoscan, realiza-se o processo de alinhamento onde se orientam as imagens em um espaço arbitrário, relacionando a semelhança entre os *pixels* sem a necessidade de pontos de apoio (AGISOFT, 2019).

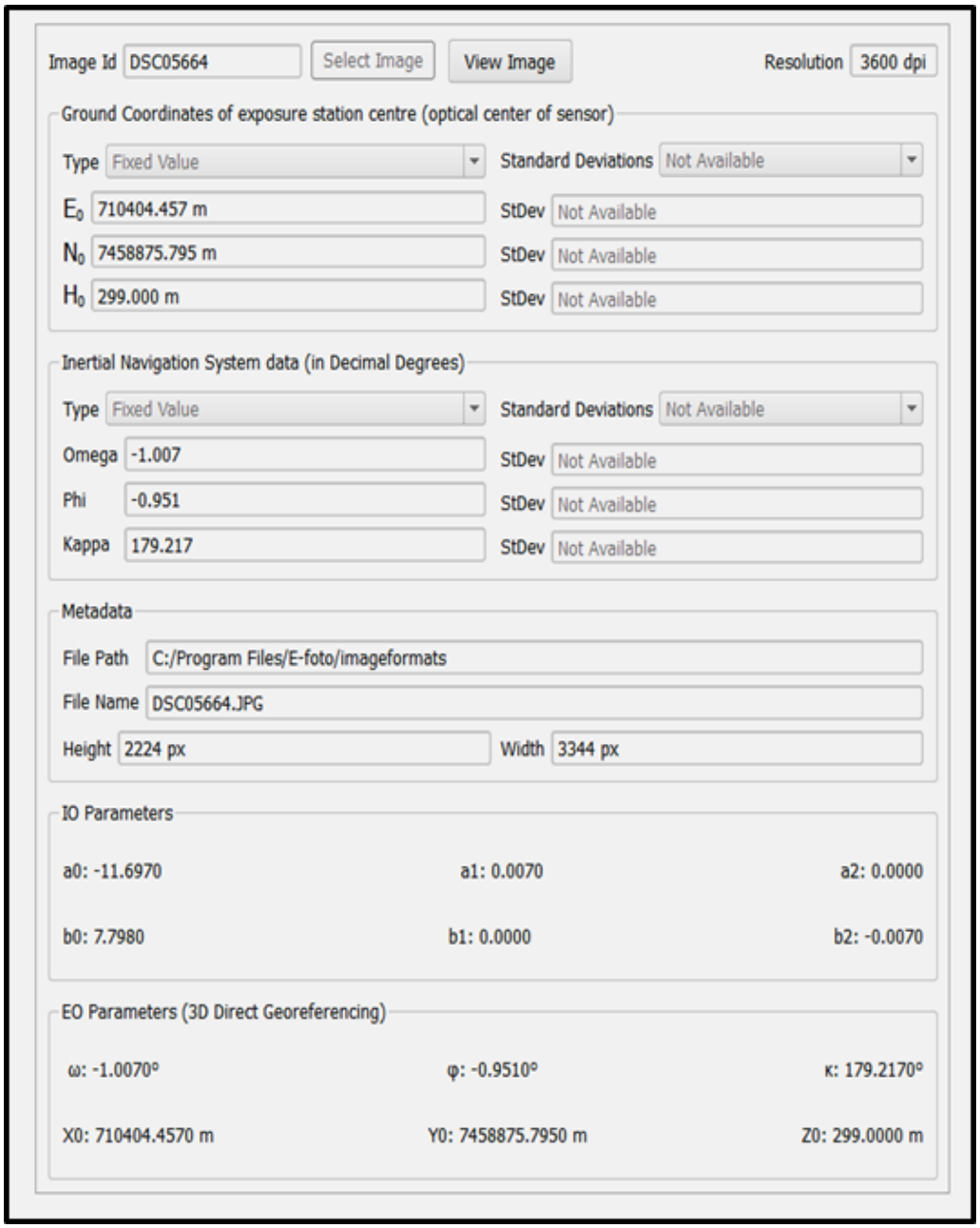


Figura 22. Dados da imagem para orientação exterior no programa E-Foto. Fonte: http://www.efoto.eng.uerj.br.

2.2.2.3 Modelo digital de elevação (MDE)

Um modelo digital é a reconstrução computacional da superfície terrestre por um processo de modelagem matemática realizado por meio da interpolação de um conjunto de pontos amostrados no terreno (MIRANDA et al., 2018). O Modelo Digital de Terreno (MDT) ou em inglês *Digital Terrain Model* (DTM) pode ser definido como a representação estatística da superfície contínua do terreno por um número de pontos selecionados com coordenadas X, Y, Z conhecidas (FORLANI et al., 2015). Quando a informação sobre o terreno for a elevação, tem-se o MDE ou o Modelo Numérico de Elevação (MNE) ou em inglês *Digital Elevation Model* (DEM). Desta forma, o MDE é um subconjunto do MDT (MIRANDA et al., 2018). O DEM foi criado para substituir o DTM. A perfeição do processo de interpolação em um MDE depende da distribuição, densidade e exatidão dos pontos de referência (ANDRADE, 1998).

Concernente aos métodos de interpolação para um MDE, podem-se classificá-los em seis grupos, segundo o tipo de representação de superfície que eles produzem. O primeiro grupo representa métodos onde a determinação da altitude de um ponto é dada pela minimização da soma dos quadrados das distâncias para os pontos de referência. No segundo grupo, estão alocados os métodos que definem, em cada ponto de referência, uma superfície com eixo de simetria coincidente com a vertical deste ponto. O terceiro grupo reúne os métodos que determinam uma rede regular de pontos independentes dos pontos de referência. No quarto grupo, semelhante ao terceiro, os pontos de referência estão localizados em uma rede regular. O quinto grupo comporta os métodos que utilizam triângulos cujos vértices são os pontos de referências, com uma rede de pontos regular ou irregular. Por fim, no sexto grupo, estão todos os métodos que exigem pontos de referência situados em pontos característicos do terreno (SCHUT, 1976).

A obtenção do MDE tem como objetivo a reconstrução automática do espaço-objeto a partir de um par estereoscópico de imagens digitais. Este par caracteriza-se pela área de superposição entre duas imagens (SILVEIRA, 2012). Um MDS pode expressar vários tipos de atributos, incluindo, no terreno, prédios, árvores e veículos,

dentre outros alvos. Quando um MDS exprime altitudes, chama-se MDE (**Figura 23**) (COELHO e BRITO, 2016).

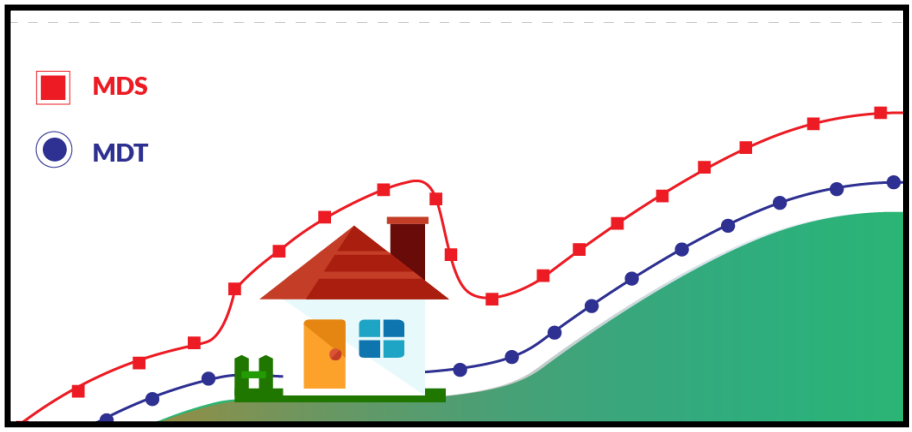


Figura 23. Diferenças entre MDS e MDT. Fonte: https://horusaeronaves.com/

Os métodos para automatizar a identificação de imagens por comparação são o coeficiente de correção de Pearson (ρ), a correspondência por mínimos quadrados, a comparação por feições e a comparação relacional. O coeficiente de correção de Pearson e a correspondência por mínimos quadrados são métodos relevantes utilizados para a extração do MDE (ANDRADE, 1998). O coeficiente de correlação de Pearson é uma medida do grau de relacionamento linear entre duas variáveis aleatórias. Dessa maneira, o coeficiente de correlação tem ênfase na predição do grau de dependência entre essas duas variáveis. O cálculo da correlação é realizado a partir da Eq. 39 (ANDRADE, 1998):

$$\rho = \frac{\frac{\sum_{i=1}^{n}(A_i - \mu_a)(B_i - \mu_b)}{n-1}}{\sqrt{\frac{\sum_{i=1}^{n}(A_i - \mu_a)^2}{n-1}}\sqrt{\frac{\sum_{i=1}^{n}(B_i - \mu_b)^2}{n-1}}} \quad (39)$$

Onde: ρ = coeficiente de correção de *Pearson*; A = recorte de uma área de referência; B = recorte de uma área da foto de busca; μ_a = média aritmética da variável A; μ_e = média aritmética da variável B; e n = número de elementos.

O coeficiente de correlação ρ pode assumir valores entre -1 e 1. Uma correlação positiva indica uma relação diretamente proporcional, ou seja, se uma

variável aumenta, a outra também aumenta. Uma correlação negativa indica uma relação inversamente proporcional, ou seja, se uma variável aumenta, a outra diminui. O valor 0 significa independência linear entre as variáveis analisadas, enquanto que 1 é o maior grau de dependência linear (ANDRADE, 1998).

A partir de um ponto em uma imagem de referência, cria-se uma janela em torno desse ponto (**Figura 24**). Essa janela é chamada de *template*. Como a localização do ponto (*pixel*) homólogo é desconhecida, cria-se uma janela de pesquisa, com dimensões consideravelmente maiores do que as do *template*, em torno da provável localização do ponto em questão na outra imagem (SILVEIRA, 2012).

Por fim, será considerado como candidato a *pixel* homólogo na imagem de pesquisa aquele que tiver o maior grau de correspondência em relação ao *pixel* de referência. O valor da correspondência deverá estar acima de um determinado limiar (SILVEIRA, 2012).

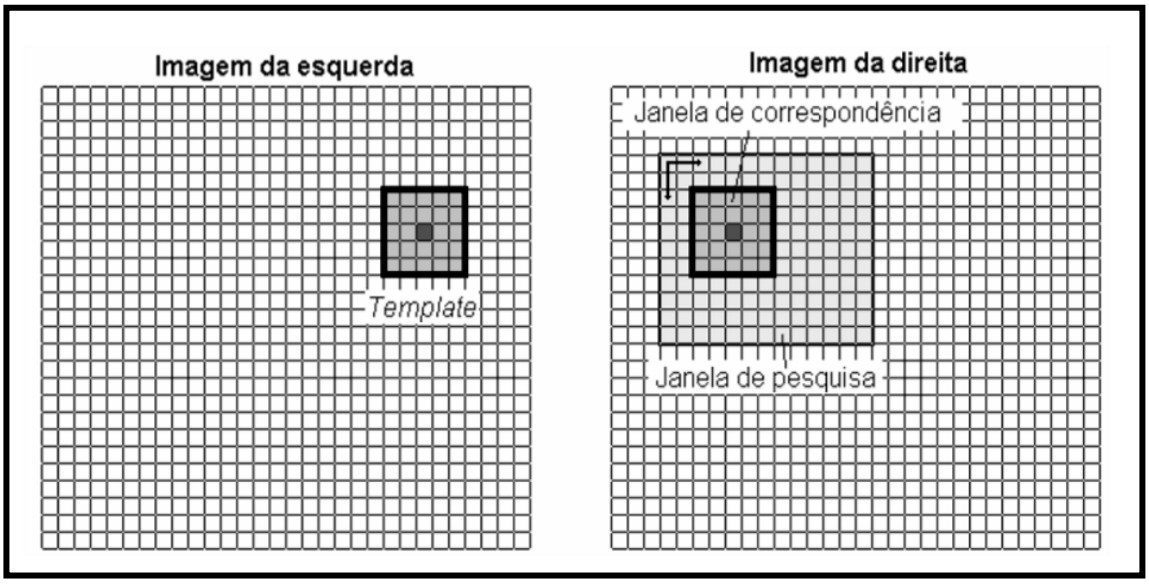


Figura 24. Mecanismo de busca por pontos homólogos. Fonte: Silveira (2012).

A correlação de Pearson não leva em conta os efeitos de rotação e escala entre as fotos. A correspondência por mínimos quadrados considera esses efeitos e utiliza somente uma janela de correspondência, de forma a fazer com que o centro da janela venha a convergir em direção ao ponto homólogo (**Figura 25**). Esta

convergência somente irá ocorrer caso o ponto inicial esteja bem próximo ao ponto homólogo (SILVEIRA, 2012).

O ajustamento da janela de correspondência é realizado a partir de uma transformação geométrica, de forma que os níveis de cinza resultantes fiquem mais semelhantes com os níveis de cinza do *template* a cada iteração. Os parâmetros de transformação são mostrados nas Eq. 40 a 42 (SILVEIRA, 2012):

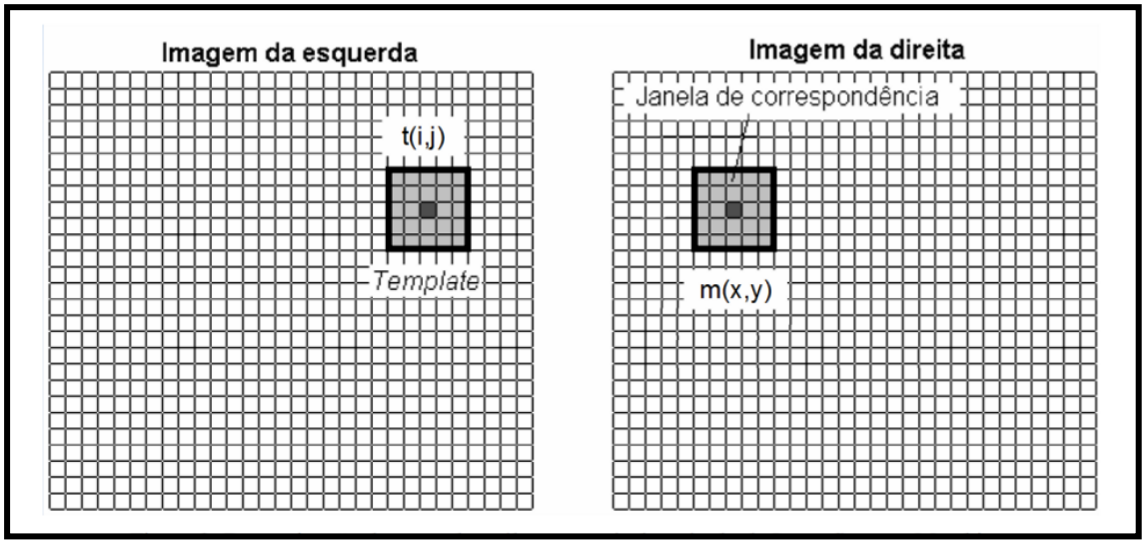


Figura 25. Determinação do ponto homólogo por meio do método de correspondência por mínimos quadrados. Fonte: Silveira (2012).

$$m\,(x\,,\,y) = T_G\,[m(i\,,\,j)] \tag{40}$$

$$x = T_x\,(i,\,j) \tag{41}$$

$$y = T_y\,(i,\,j) \tag{42}$$

Onde: $m(i,\,j)$ = janela de correspondência inicial; T_G = transformação geométrica; T_x = transformação geométrica em x; T_y = transformação geométrica em y; e $m(x,\,y)$ = resultado da transformação geométrica.

A janela de correspondência atual é igual à soma da janela de correspondência da iteração anterior mais a derivada do modelo de transformação geométrica

utilizado. O erro resultante é calculado comparando-se as janelas *m(x, y)* e *t(i, j)* (Eqs. 43 e 44) (SILVEIRA, 2012):

$$m_i(x,y) = m_{i+1}(x,y) + \frac{\partial m_i(x,y)}{\partial T_x}\left[\frac{\partial T_x \Delta a_0}{\partial a_0} + \frac{\partial T_x \Delta a_1}{\partial a_1} + \frac{\partial T_x \Delta a_2}{\partial a_2}\right] + \frac{\partial m_i(x,y)}{\partial T_y}\left[\frac{\partial T_y \Delta a_0}{\partial a_0} + \frac{\partial T_y \Delta a_1}{\partial a_1} + \frac{\partial T_y \Delta a_2}{\partial a_2}\right] \quad (43)$$

$$r(i,j) = t(i,j) - m(x,y) \quad (44)$$

No contexto do projeto E-Foto, utiliza-se o MDE em que o processo fotogramétrico de extração envolve um conjunto de imagens, formando um bloco de imagens. O conhecimento dos parâmetros da orientação interior e exterior de cada uma dessas imagens, a descoberta automática de pares de pontos homólogos ou conjugados em suas áreas de sobreposição longitudinal e o cálculo do algoritmo de intersecção espacial para cada par de pontos conjugados, levam à nuvem resultante de pontos, cujas coordenadas no referencial do terreno são calculadas (COELHO e BRITO, 2016). No módulo de extração do MDE do aplicativo E-Foto, há a possibilidade de selecionar um dos dois tipos de método de correspondência: correlação cruzada ou dos mínimos quadrados (**Figura 26**). Após a execução da extração do MDE, observa-se o processamento do MDE a partir do painel de histograma de precisão do processamento. Esta é a precisão da extração automática, com o intuito de verificar a qualidade das correlações. Quanto mais próximo do valor 1, melhor o desempenho do processo. O arquivo final pode ser salvo no formato ASCII (VERMEER e AYEHU, 2017).

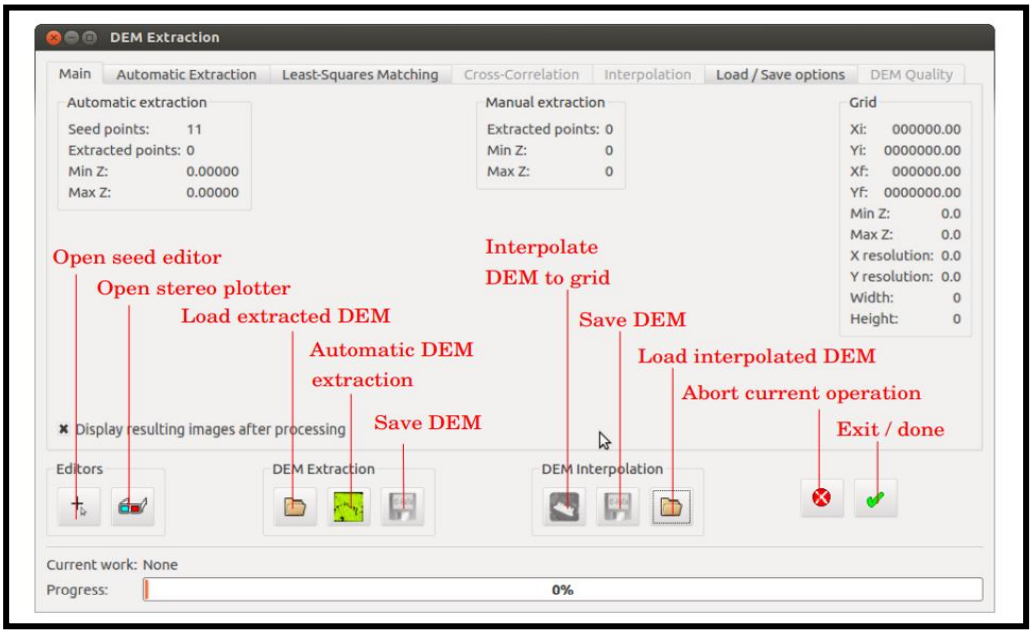


Figura 26. Dados para a extração do MDS no E-Foto.
Fonte: http://www.efoto.eng.uerj.br.

O processo de aerotriangulação gera pares estéreos que podem ser usados para extrair MDE (JENSEN, 2009). No E-Foto, a aerotriangulação é realizada pelo método dos feixes perspectivos (COELHO e BRITO, 2016). O objetivo fim é a geração do ortomosaico, seguindo a orientação interior, a orientação exterior e a extração do MDE.

O aplicativo E-foto usa o método de retificação diferencial para gerar uma ortoimagem. Nesse método, primeiramente se requer que o usuário defina uma ortomatriz vazia sobre o terreno. Essa ortomatriz é associada a uma imagem digital “em branco”, com *pixels* cujas dimensões são da ordem do elemento de resolução do terreno da imagem original em perspectiva cônica ou central. São determinadas as coordenadas tridimensionais do centro de cada *pixel* da ortomatriz vazia no referencial terrestre. A partir dessas coordenadas, por meio das equações de colinearidade, com parâmetros da orientação exterior conhecidos, são definidas as coordenadas no espaço-imagem para aquele ponto. Por meio dos parâmetros da orientação interior, chega-se ao *pixel*. A tonalidade é reamostrada na imagem original e o respectivo valor do nível de cinza é atribuído à ortoimagem vazia. O processo se repete até que toda a ortoimagem vazia tenha sido preenchida com os

respectivos níveis de cinza interpolados da imagem original (**Figura 27**) (COELHO e BRITO, 2016).

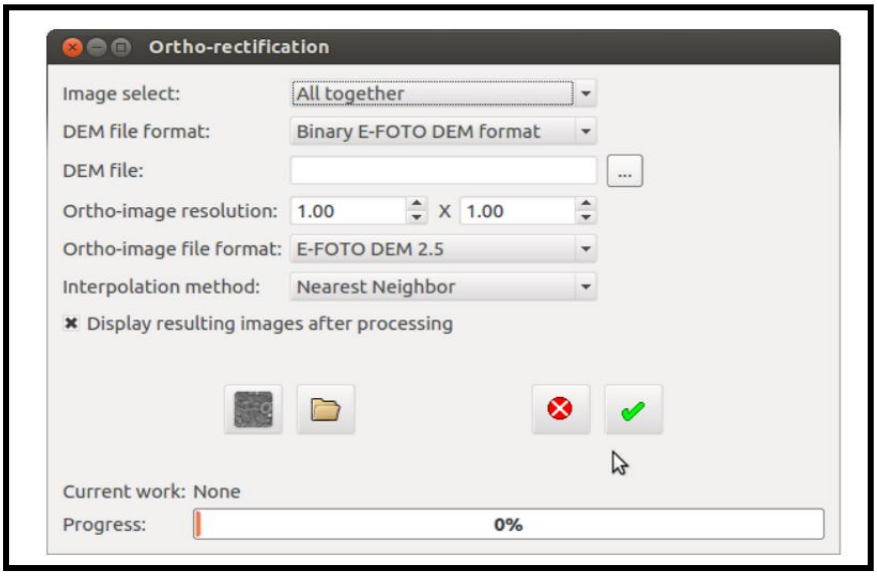


Figura 27. Tela de ortorretificação no ambiente E-Foto. Fonte: http://www.efoto.eng.uerj.br.

No Agisoft Photoscan, tem-se o SfM que é uma técnica que permite a extração de informações tridimensionais a partir de imagens estáticas em 2D e consiste na tomada de diversas imagens de uma cena a partir de pontos de vista diferentes. Nesta técnica, a reconstrução 3D pode ser dividida em: determinação das feições características em cada imagem, determinação de uma estimativa inicial da estrutura da cena e a movimentação relativa da câmera, otimização das estimativas, calibração da câmera, adensamento da nuvem de pontos e geração do modelo tridimensional por meio de rede triangular (SOUZA, 2018).

As imagens obtidas por RPA podem ser processadas utilizando os métodos SfM ou fotogramétrico tradicional. No caso do SfM, existem atualmente aplicativos como Photomodeler, Pix4D, MicMac, VisualSFM, Menci e Agisoft Photoscan, entre outros (SOUZA, 2018). Nesta tese, foi utilizado o E-Foto, fotogramétrico tradicional, e o Agisoft Photoscan, com SfM. Cabe destacar que MDE e ortomosaico são produzidos com controle menor em etapas de processamento como georreferenciamento e formação de blocos no método SfM. Entretanto, o SfM do

Agisoft Photoscan tem apresentado melhores resultados quando comparados com outros aplicativos que utilizam o mesmo método (GINI et al., 2013).

No contexto do aerolevantamento óptico com RPA, os ângulos de atitude (parâmetros externos) geram a distorção projetiva, quando a imagem é obtida com visada *off-nadir* ou a nadir (ROBERTO et al., 2017). Conforme informações do IMU do RPA, os ângulos de guinada (ψ) são os maiores, quando comparadas aos ângulos de rolagem e de arfagem (PEGORARO, 2013; RUZGIENE, 2014; JAIMES, 2016; ROBERTO et al., 2017; CARDOSO, 2018). Os ângulos de rolagem e de arfagem tendem a ser menores que 5°, atendendo os padrões cartográficos (COELHO e BRITO, 2016; HLOTOV et al., 2019). O problema maior consiste em reduzir ou eliminar o ângulo de guinada. Com o aumento do grau de distorção, o tamanho dos *pixels* na imagem é também alterado não uniformemente (GALBRAITH et al., 2005). Com a redução do ângulo de guinada, o ortomosaico criado tenderá a apresentar uma acurácia significativa (HLOTOV et al., 2019). Desta forma, para melhorar a qualidade das informações, é necessária a correção geométrica nas imagens a partir de modelos matemáticos (MAROTTA et al., 2011).

2.3 CORREÇÃO GEOMÉTRICA

Normalmente é a partir de um modelo simples que se faz o aprimoramento na modelagem de fenômenos físicos (ANDRADE, 1998). Modelos matemáticos têm sido gerados, por exemplo, a partir de informações de sensor e modificando equações de colinearidade (MAROTTA et al., 2011).

Em uma correção geométrica, parâmetros correlacionados não podem ser separados num processo de ajustamento (ANDRADE, 1998). Modelos matemáticos com menor número de parâmetros apresentam-se com menor precisão (MAROTTA et al., 2011). Há dois tipos básicos de correções geométricas: a paramétrica e a não-paramétrica. A correção geométrica paramétrica baseia-se no conhecimento dos parâmetros intrínsecos da câmera (parâmetros internos) e as informações da sua posição e orientação (parâmetros externos). Essa correção é independente da imagem e não precisa de nenhum processo de identificação de pontos de apoio. A correção geométrica não-paramétrica utiliza pontos de apoio sobre a imagem. Nesse

método, toma-se como base um mapa georreferenciado e comparam-se as diferenças entre as posições de diferentes pontos escolhidos em duas imagens obtidas por RPA (JAIMES, 2016).

2.3.1 O ângulo *yaw* (de guinada, de deriva, azimute, ψ)

O ângulo de deriva é ocasionado devido ao vento no sentido transversal ou lateral à aeronave, fazendo com que a aeronave descreva uma trajetória diferente daquela planejada no plano de voo (**Figura 28**). O ângulo de guinada é a consequência do movimento de rotação horizontal da aeronave em torno do seu eixo vertical e resulta na redução da cobertura estéreo entre as fotos (LIMA, 2016; ONUORA et al., 2018).

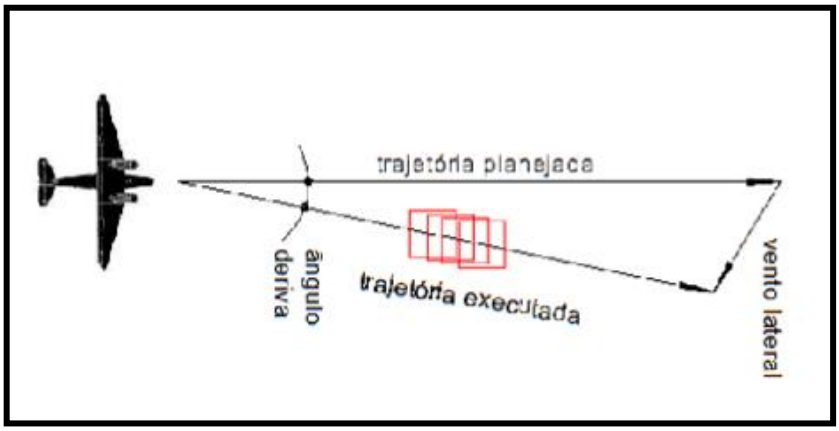


Figura 28. Ângulo de deriva. Fonte: Lima (2016).

Considerando a orientação exterior e que nem todas as plataformas RPA de asa fixa dispõem de plataformas giroestabilizadoras (ROBERTO et al., 2017), como o gimbal, comum em plataformas RPA multirotores, possuindo apenas os sistemas GNSS e IMU, que obtém as informações de atitude da aeronave (DRONENG, 2019), o ângulo de guinada é a principal correção geométrica a ser feita em aerolevantamentos com imagens ópticas em RPA de asa fixa de classe 3. As plataformas giroestabilizadoras compensam os ângulos de guinada até certo ponto (PEPE et al., 2018). O ângulo *yaw* provoca falta de alinhamento das varreduras ou

superposições ou lacunas, também conhecido como efeito “leque” (**Figura 29**) (D’ALGE, 2007).

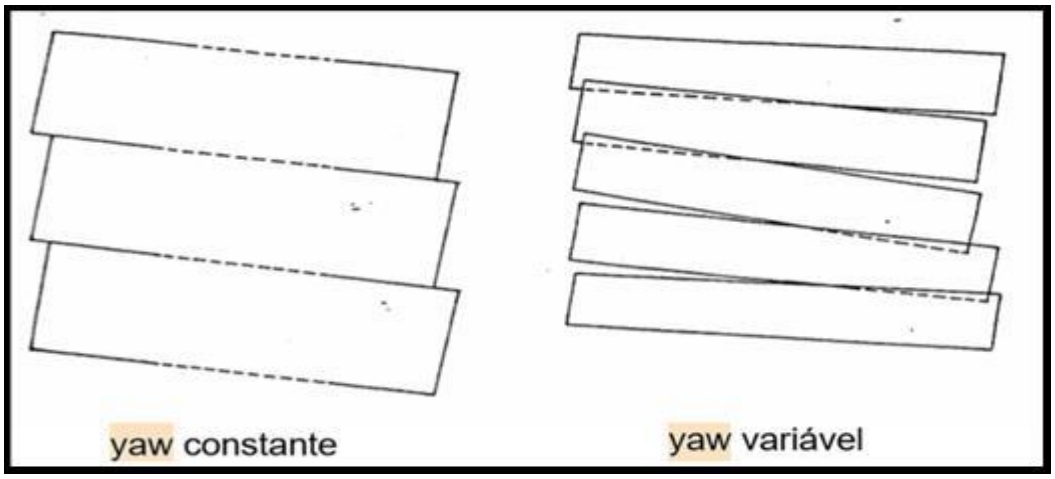


Figura 29. Distorção devido ao *yaw*. Fonte: Machado e Silva (1989).

Os ângulos *yaw* normalmente apresentam valores muito altos de desvio, quando comparados aos ângulos *roll* e *pitch* (Tabelas 2 a 6) (PEGORARO, 2013; RUZGIENE, 2014; JAIMES, 2016; ROBERTO et al., 2017; CARDOSO, 2018).

Tabela 2. Ângulos *roll, pitch* e *yaw* no RPA Echar 20B.

Imagens	*roll*	*pitch*	*yaw*
1	-11,98	13,59	49,23
2	-18,01	10,02	212,71

Fonte: Roberto et al. (2017).

Tabela 3. Ângulos *roll, pitch* e *yaw* no RPA 1.

Imagens	*roll*	*pitch*	*yaw*
1	7,70	4,90	112,60
2	11,40	16,70	224,30
3	-7,60	8,50	250,90
4	12,60	6,90	113,00
5	18,20	11,50	191,10

6	11,40	17,90	234,40

Fonte: Jaimes (2016).

Tabela 4. Ângulos *roll, pitch* e *yaw* no RPA MD4-1000.

Imagens	*roll*	*pitch*	*yaw*
1	5,60	4,20	151,00
2	4,20	5,40	211,71
3	3,50	4,20	154,30
4	5,60	1,60	152,60
5	3,60	4,10	150,80

Fonte: Pegoraro (2013).

Tabela 5. Ângulos *roll, pitch* e *yaw* no RPA MLB BAT3.

Imagens	*roll*	*pitch*	*yaw*
1	-5,15	-10,80	-122,00
2	2,05	0,08	-124,14
3	-0,07	-2,98	-142,63

Fonte: Cardoso (2018).

Tabela 6. Ângulos *roll, pitch* e *yaw* no RPA 2.

Imagens	*roll*	*pitch*	*yaw*
1	0,19	1,22	198,98
2	0,15	0,99	199,48
3	-0,00	1,11	200,45
4	-0,30	0,97	200,33
5	-0,16	1,81	200,11

Fonte: Ruzgiene (2014).

2.3.2 Transformações geométricas

Uma transformação geométrica implica na modificação da disposição dos *pixels* em relação à imagem original (transformação espacial) e definição dos novos valores de intensidade em função da nova disposição (interpolação dos *pixels*) (COELHO e BRITO, 2016). Há dois tipos de transformações espaciais: lineares (Eqs. 45 e 46) e não-lineares (Eqs. 47 e 48) (COELHO e BRITO, 2016). Na resolução de equações lineares, utiliza-se o método matricial. Quando o número de equações for maior do que o número de incógnitas, utiliza-se o MMQ, que estima um valor para as incógnitas de modo a minimizar a soma dos quadrados dos desvios ou resíduos em relação à média (LIMA e BRITO, 2006).

$$x' = a_0 + x \tag{45}$$

$$y' = ky \tag{46}$$

$$x' = a_0 + a_1 y + a_2 x \tag{47}$$

$$y' = b_0 + b_1 y + b_2 x \tag{48}$$

As transformações espaciais lineares representam um modelo linear. Como exemplos, têm-se a translação e a mudança de escala (Eqs. 45 e 46). As transformações espaciais não-lineares podem ser representadas em formato linearizado, por intermédio de um polinômio de 1º grau, ou não, caso em que são utilizados polinômios de grau igual ou maior que 2. Este tipo de transformação é a mais utilizada em fotogrametria digital (Eqs. 47 e 48) (COELHO e BRITO, 2016).

As transformações matemáticas consideradas biunívocas permitem que um sistema referencial bidimensional se transforme em um sistema referencial tridimensional e vice-versa. Há casos que não há transformação inversa, apenas unívoca, como a função de mapeamento polinomial (SANTOS, 2013).

As transformações geométricas 2D entre dois sistemas de coordenadas (E e E') são classificadas em: translação, transformação ortogonal, transformação

isogonal, transformação afim, transformação projetiva e transformação polinomial (**Figura 30**) (SANTOS, 2013).

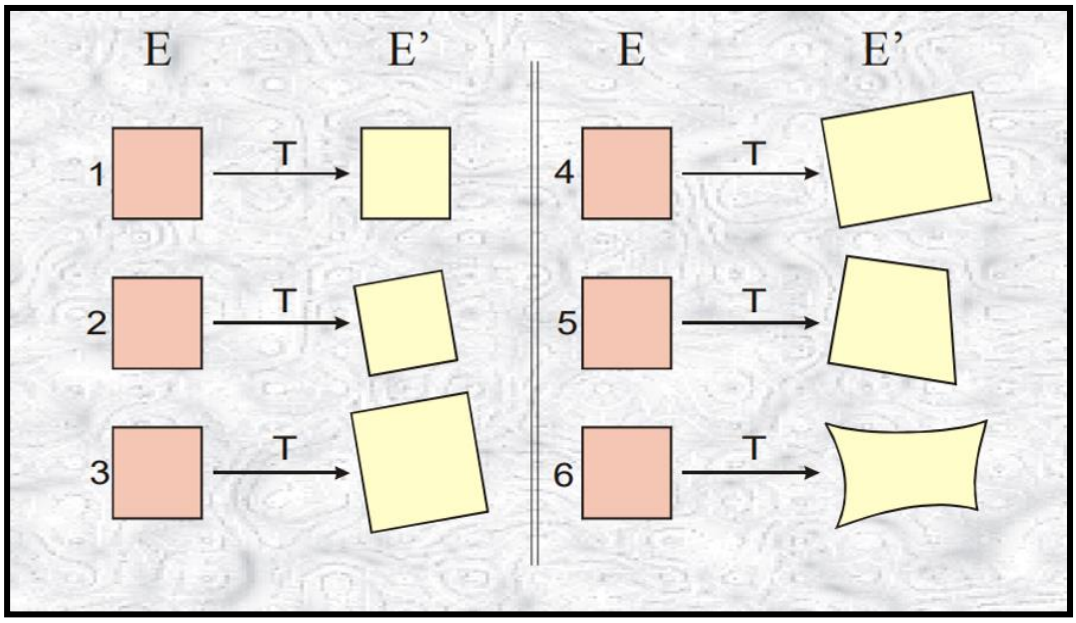


Figura 30. Transformações de um objeto no espaço E para o espaço E': 1- translação, 2- transformação ortogonal, 3- transformação isogonal, 4- transformação afim, 5- transformação projetiva e 6- transformação polinomial. Fonte: Santos (2013).

Uma translação pode não ser considerada uma transformação geométrica entre sistemas. A translação representa um movimento das coordenadas de uma feição em um mesmo sistema de referência. Esta operação tem a propriedade de manter a escala e a orientação da feição transladada (Eq. 49) (ESPINOSA et al., 2013; SANTOS, 2013). Como não permite a rotação entre E e E', não é adequada para realizar a correção do ângulo de guinada em RPA de asa fixa classe 3.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{49}$$

Onde: *x'* e *y'* são as coordenadas do ponto transladado; *x* e *y* são as coordenadas de origem do ponto e *Δx* e *Δy* são as translações.

A transformação ortogonal no plano é linear, também chamada de transformação afim ortogonal (ANDRADE, 1998), e corresponde a uma variação da

transformação afim geral, onde ambos os sistemas são ortogonais (COELHO e BRITO, 2016). Nesta transformação, os pontos são transladados; há uma rotação, com um ângulo relativo *β*, mantendo a mesma escala, em relação à origem O, totalizando três parâmetros (**Figura 31**) (SANTOS, 2013).

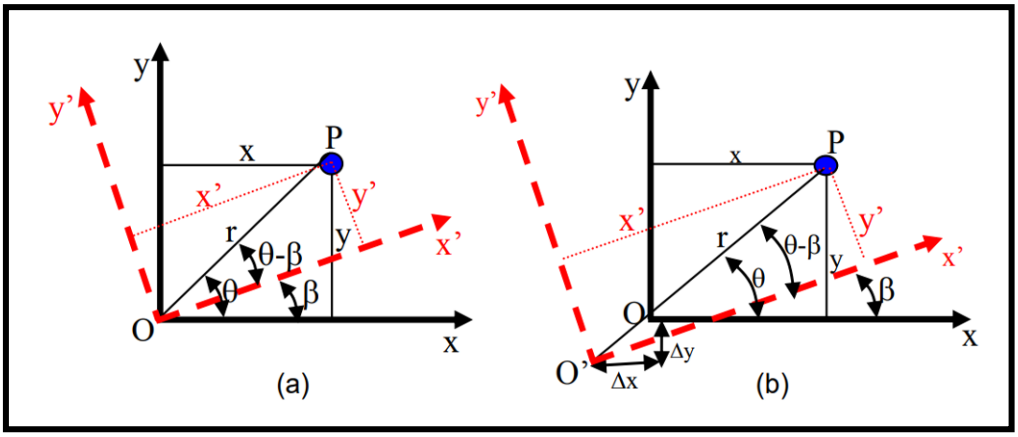


Figura 31. Transformação ortogonal. (a) rotação entre os sistemas bidimensionais; (b) rotação e translação entre os sistemas bidimensionais. Fonte: Santos (2013).

Desta forma, temos a Eq. 50 (ANDRADE, 1998; ESPINOSA et al., 2013; SANTOS, 2013):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\beta & sen\beta \\ -sen\beta & cos\beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \quad (50)$$

Onde: *x'* e *y'* são as coordenadas do ponto no sistema de destino; *β* é o ângulo relativo de rotação entre os sistemas; *x* e *y* são as coordenadas do ponto no sistema de origem e *Δx* e *Δy* são as translações.

Como a matriz de rotação é ortogonal, é possível obter a inversa da Eq. 50 (SANTOS, 2013). Considerando que esta tese visa corrigir o acentuado ângulo de guinada e que a correção deve ser feita em conjunto com os ângulos de rolamento e de arfagem, envolvendo três dimensões, este modelo não seria adequado na correção geométrica de imagens do RPA de asa fixa classe 3, pois envolve duas dimensões (ANDRADE, 1998). A transformação ortogonal é adequadamente aplicada na orientação interior de imagens ópticas (COELHO e BRITO, 2016). As

propriedades invariantes são o comprimento, o ângulo e a área, apresentando três graus de liberdade ou parâmetros (HARTLEY e ZISSERMAN, 2000).

A transformação isogonal no plano é linear, também chamada de transformação afim isogonal de Helmert, de similaridade, conforme, de corpo rígido ou euclidiana. Trata-se de uma variação da transformação afim ortogonal, onde o fator de escala é constante (COELHO e BRITO, 2016). Esta transformação permite a translação dos pontos, uma rotação e um fator de escala uniforme, totalizando quatro parâmetros (**Figura 32**) (SANTOS, 2013).

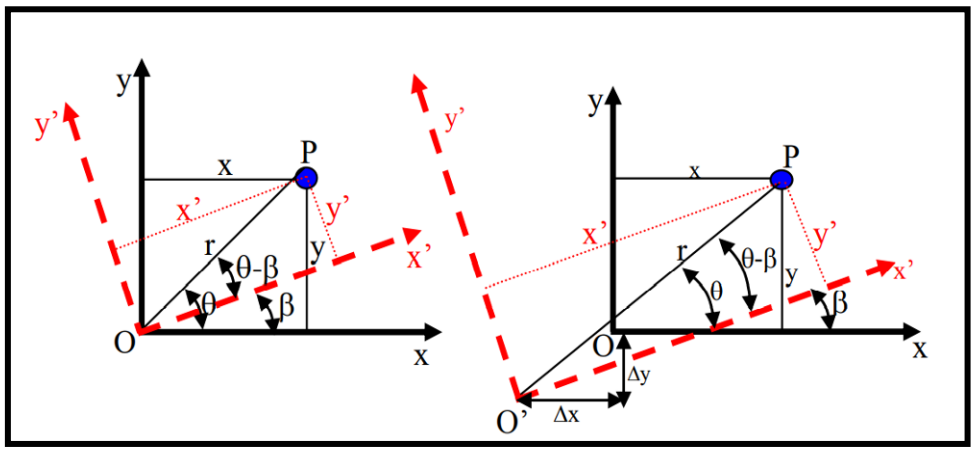


Figura 32. Transformação isogonal. (a) rotação entre os sistemas bidimensionais; (b) rotação, translação e fator de escala entre os sistemas bidimensionais. Fonte: Santos (2013).

Ainda tem-se a Eq. 51 (ANDRADE, 1998; SANTOS, 2013):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \lambda . \begin{bmatrix} cos\beta & sen\beta \\ -sen\beta & cos\beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{51}$$

Onde: *x'* e *y'* são as coordenadas do ponto no sistema conhecido; λ é o fator de escala; β é o ângulo relativo de rotação entre os sistemas; *x* e *y* são as coordenadas do ponto no sistema arbitrário e Δx e Δy são as translações.

Como a matriz de rotação é ortogonal, é possível obter a inversa da Eq. 51 (SANTOS, 2013). Com o mesmo argumento exposto para a transformação ortogonal, este modelo não seria adequado na correção geométrica de imagens do

RPA de asa fixa classe 3, pois envolve duas dimensões (ANDRADE, 1998). A transformação ortogonal é aplicada na orientação interior de imagens ópticas (COELHO e BRITO, 2016). As propriedades invariantes são a razão de comprimento, o ângulo e a razão de áreas e linhas paralelas. Apresenta quatro graus de liberdade (HARTLEY e ZISSERMAN, 2000).

A transformação afim no plano é linear (ANDRADE, 1998; COELHO e BRITO, 2016). Esta transformação já foi explicada de forma detalhada no item 2.2.2.1. Cabe frisar que, nesta transformação, ocorre a translação dos pontos, uma rotação, dois fatores de escala, um para *x* e outro para *y*, e um fator de não ortogonalidade entre os eixos, totalizando seis parâmetros (**Figura 33**) (SANTOS, 2013).

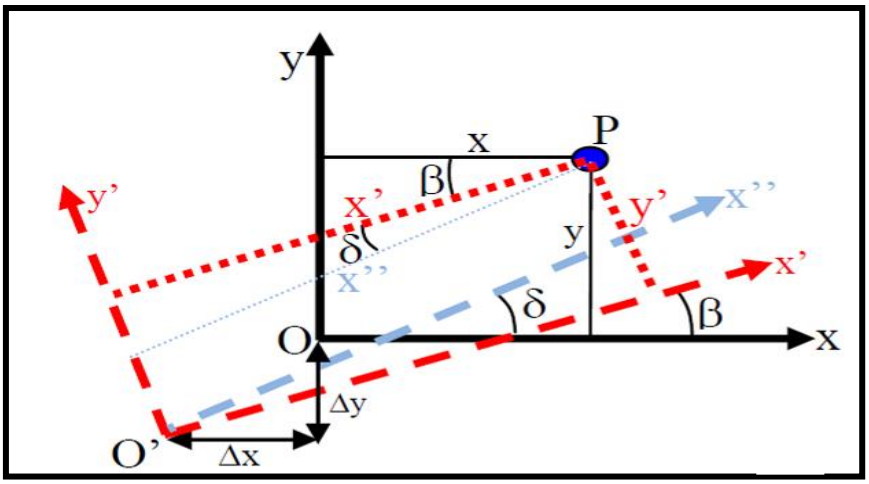


Figura 33. Transformação afim. Fonte: Santos (2013).

Além da Eq. 8 (item 2.2.2.1 orientação interior), tem-se outra forma de representar a transformação afim a partir da Eq. 52 (ANDRADE, 1998; SANTOS, 2013):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\beta & -sen\beta \\ sen\beta & cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \delta & 1 \end{bmatrix} \begin{bmatrix} x\lambda_x \\ y\lambda_y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \quad (52)$$

Onde: *x'* e *y'* são as coordenadas do ponto no sistema conhecido; β é o ângulo relativo de rotação entre os sistemas; δ é o fator de não ortogonalidade entre os

eixos; λ_x e λ_y são os fatores de escala; *x* e *y* são as coordenadas do ponto no sistema arbitrário e *Δx* e *Δy* são as translações.

Como a matriz de rotação é ortogonal, também é possível obter a inversa da Eq. 52 (SANTOS, 2013). Com o mesmo argumento exposto para as transformações ortogonal e isogonal, este modelo não seria adequado para correção geométrica de imagens do RPA de asa fixa classe 3, pois envolve duas dimensões (ANDRADE, 1998). A transformação afim é a mais utilizada na orientação interior de imagens ópticas, sendo a mais completa mencionada até o momento (COELHO e BRITO, 2016). As propriedades invariantes são as linhas paralelas, a relação de comprimentos de segmentos de linhas paralelas e a razão de áreas, apresentando seis graus de liberdade (HARTLEY e ZISSERMAN, 2000).

Ao não considerar a altitude no cálculo dos parâmetros, esta transformação não é indicada para terrenos acidentados (MAROTTA et al., 2011; SARAIVA et al., 2011), sendo utilizado na correção geométrica de imagens orbitais de baixa e média resolução. Comparando o *root mean square* (RMS) ou o erro quadrático médio (EQM) da transformação afim 2D com afim 3D, projetiva 2D e 3D e projetiva 3D modificada, a afim 2D tem apresentado o maior RMS, sendo o pior modelo a ser adotado na transformação geométrica de imagens (MAROTTA et al., 2011; SARAIVA et al., 2011).

A transformação projetiva é não-linear (LIMA e BRITO, 2006). Um modelo matemático linearizado para esta transformação pode ser obtido a partir do princípio de colinearidade (ANDRADE, 1998). Para a aplicação desta transformação, é necessária a coleta de no mínimo cinco pontos (SANTOS, 2013), apresentando oito graus de liberdade (**Figura 34**) (HARTLEY e ZISSERMAN, 2000).

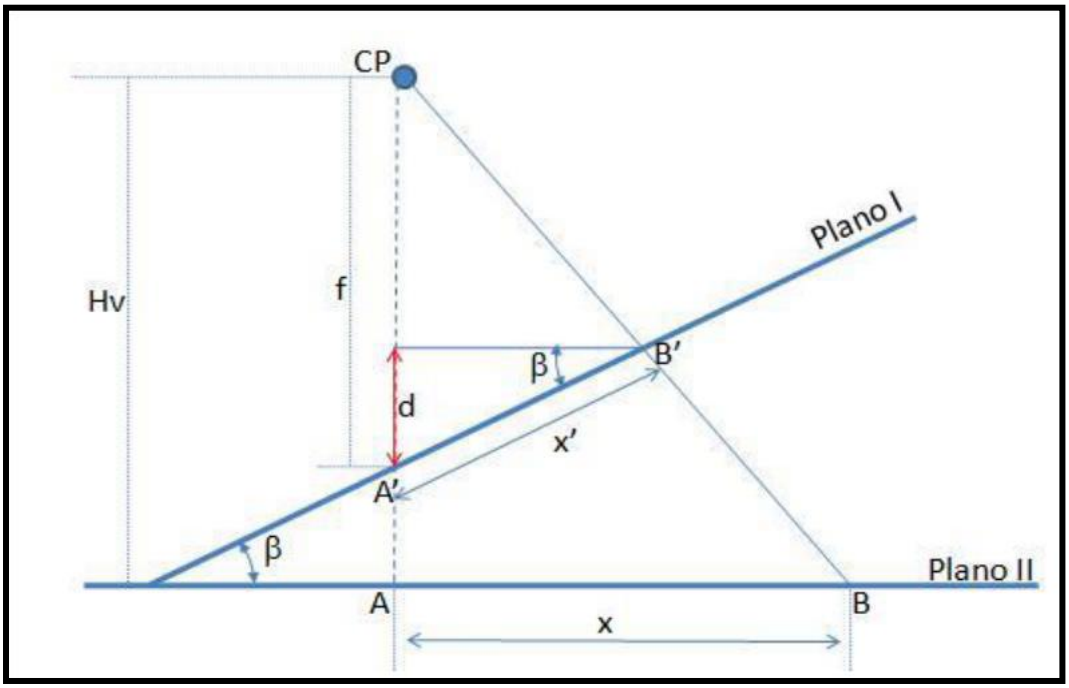


Figura 34. Geometria da transformação projetiva no plano. Fonte: Santos (2013).

A transformação entre os planos é realizada por meio de uma transformação de similaridade, pois existe uma rotação, um fator de escala e duas translações. Desta forma, as primeiras relações obtidas da **Figura 34** seriam as Eqs. 53 e 54, plano I, e Eqs. 55 e 56, plano II, (SANTOS, 2013):

$$x' = xcos\beta + ysen\beta + \Delta x \tag{53}$$

$$y' = -xsen\beta + ycos\beta + \Delta y \tag{54}$$

$$x'' = x'cos\beta + y'sen\beta + \Delta x' \tag{55}$$

$$y'' = -x'sen\beta + y'cos\beta + \Delta y' \tag{56}$$

Fazendo as devidas manipulações matemáticas, chega-se a dois tipos de transformações projetivas: a direta (Eqs. 57 e 58) e a indireta (Eq. 59) (SANTOS, 2013; COELHO e BRITO, 2016).

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1} \tag{57}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1} \tag{58}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_7 x' - a_1 & a_8 x' - a_2 \\ a_7 y' - a_4 & a_8 y' - a_5 \end{bmatrix}^{-1} \begin{bmatrix} a_3 - x' \\ a_6 - y' \end{bmatrix} \tag{59}$$

Onde: *x'* e *y'* são as coordenadas de um ponto no sistema das observações; *β* é o ângulo de rotação entre os planos I e II; *x* e *y* são as coordenadas de um ponto no sistema de medida fixa; a_i são os parâmetros da transformação e *Δx* e *Δy* são as translações.

Esta transformação não é recomendada para a correção geométrica de imagens do RPA de asa fixa classe 3, pois ela mapeia planos em planos, sendo desaconselhável para a ortoretificação de superfícies tridimensionais (COELHO e BRITO, 2016). Comparando RMS da transformação projetiva 2D com afim 2D e 3D, projetiva 3D e projetiva 3D modificada, a projetiva 2D tem apresentado o segundo maior RMS (MAROTTA et al., 2011).

Finalizando as transformações geométricas 2D entre dois sistemas de coordenadas (E e E'), têm-se as funções polinomiais utilizadas em problemas de interpolação; ajustamento fotogramétrico por faixa e em modelação de distorções. Suas equações são bastante arbitrárias, podendo variar o número de parâmetros e grau da função (Eqs. 60 e 61) (SANTOS, 2013).

$$x' = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n \tag{60}$$

$$y' = b_0 + b_1 y + b_2 y^2 + b_3 y^3 + \cdots + b_n y^n \tag{61}$$

Onde: *x'* e *y'* são as coordenadas do ponto transportado; *x* e *y* são as coordenadas do ponto a ser transportado; a_i ... a_n e b_i ... b_n são os parâmetros da transformação.

As principais aplicações envolvendo transformações 2D na fotogrametria são:

orientação interior de imagens; geração de mosaicos semi-controlados; retificação de imagens; determinação de erros sistemáticos em *scanners*; cálculo de deformação de imagens digitalizadas e de faixas fotogramétricas (SANTOS, 2013).

As transformações já citadas são de implementação relativamente simples. Elas não modelam do modo mais eficaz o problema da correção geométrica, pois não têm, como variáveis, os valores dos ângulos de rotação, os quais a câmara é submetida de forma tridimensional (COELHO e BRITO, 2016). Os modelos de transformação geométrica tridimensional são os mais utilizados na fotogrametria. Estas transformações envolvem 3 rotações, 3 translações e um fator de escala uniforme (SANTOS, 2013). Para as transformações geométricas no espaço tridimensional, adota-se o sistema referencial tridimensional em três eixos ordenados (**Figura 35**).

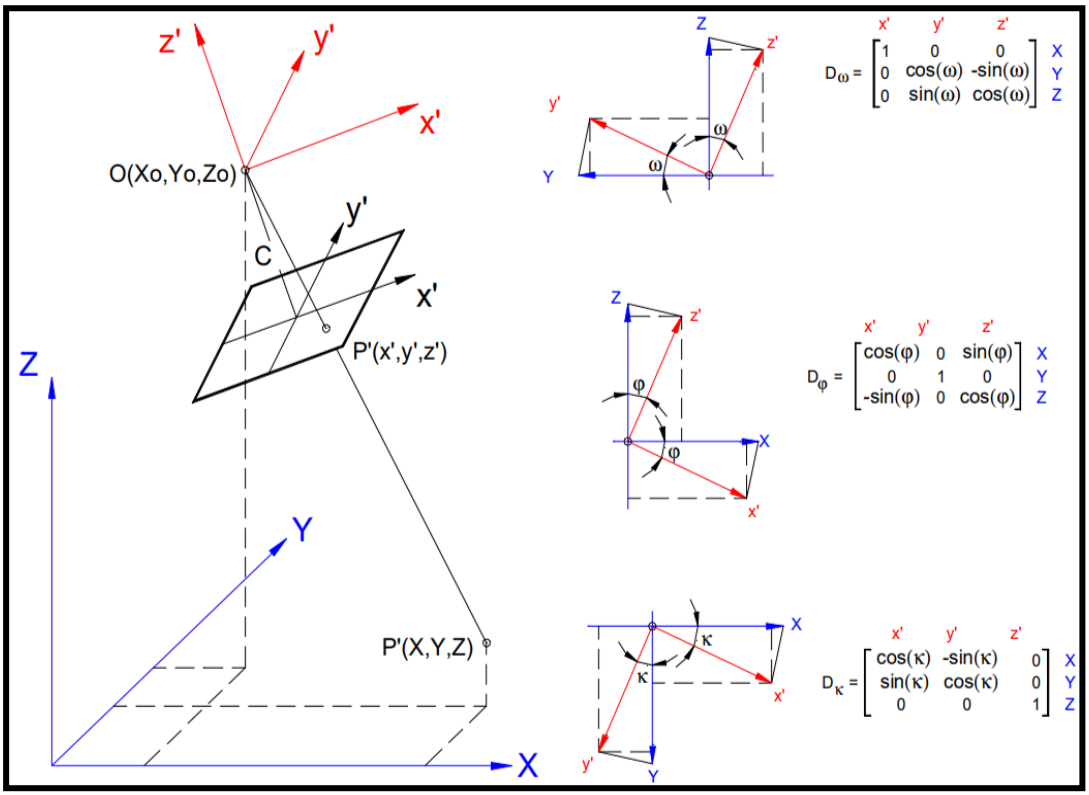


Figura 35. Sistema referencial tridimensional em três eixos em três níveis e as matrizes de rotação. Fonte: Lima et al. (2017).

Os eixos ortogonais *X, Y* e *Z* representam o espaço-objeto e os eixos ortogonais *x', y'* e *z'* representam o espaço-imagem (**Figura 35**). Os ângulos *φ*, *ω* e

κ, sistema fotogramétrico, bem como os ângulos ϕ, θ e ψ, sistema aeronáutico, com suas respectivas matrizes de rotação, foram explicados no item 2.2.2.2 orientação exterior deste trabalho (**Figura 35**).

O efeito da rotação em ω, em torno do eixo x, provoca o não alinhamento de varreduras consecutivas (**Figura 36)** (D'ALGE, 2007). Considerando a matriz de D_ω (**Figura 35**), uma rotação em ω do eixo x no sistema referencial adotado possui efeito unitário para a coordenada *x'* e nenhum efeito para as coordenadas *y', z', Y* e *Z*. O efeito da rotação em ω para o eixo y no sistema referencial 3D não possui nenhum efeito para as coordenadas *x'* e *X*, porém, influencia as coordenadas *y'* e *Y* com cosω e as coordenadas *z'* e *Z* com - senω. Para o eixo z no sistema adotado, o efeito da rotação em ω não possui nenhum efeito para as coordenadas *x'* e *X*, mas provoca efeitos nas coordenadas *y'* e *Y* com senω e nas coordenadas *z'* e *Z* com cosω (SANTOS, 2013; BÖRLIN, 2014).

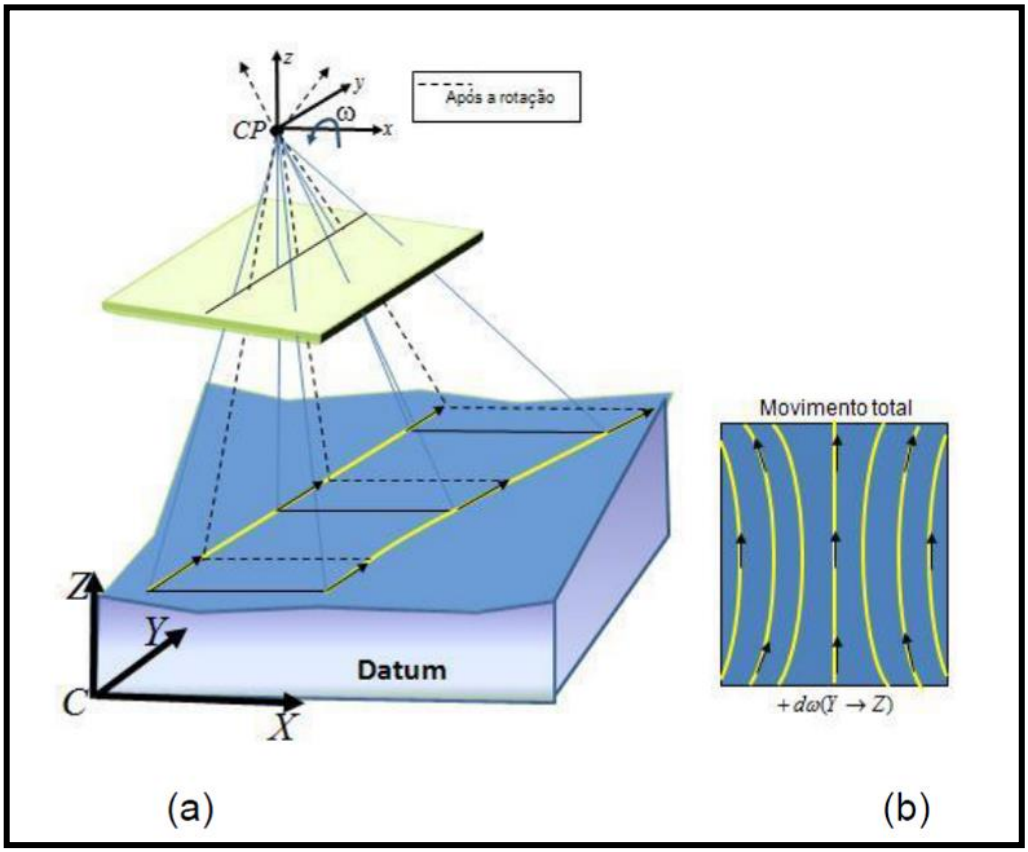


Figura 36. (a) Efeito da rotação em ω sobre a projeção dos pontos. (b) Movimento total das componentes Y e Z no plano de projeção. Fonte: Santos (2013).

O efeito da rotação em φ, em torno do eixo y, provoca superposições ou lacunas entre varreduras consecutivas (**Figura 37**) (D'ALGE, 2007). Considerando a matriz de D_φ (**Figura 35**), uma rotação em φ do eixo y no sistema referencial adotado possui efeito unitário para a coordenada *y'* e nenhum efeito para as coordenadas *x', z', X* e *Z*. O efeito da rotação em φ para o eixo x no sistema referencial 3D não possui nenhum efeito para as coordenadas *y'* e *Y*, porém, influencia as coordenadas *x'* e *X* com cosφ e as coordenadas *z'* e *Z* com senφ. Para o eixo z no sistema adotado, o efeito da rotação em φ não possui nenhum efeito para as coordenadas *y'* e *Y*, mas provoca efeitos nas coordenadas *x'* e *X* com - senφ e nas coordenadas *z'* e *Z* com cosφ (SANTOS, 2013; BÖRLIN, 2014).

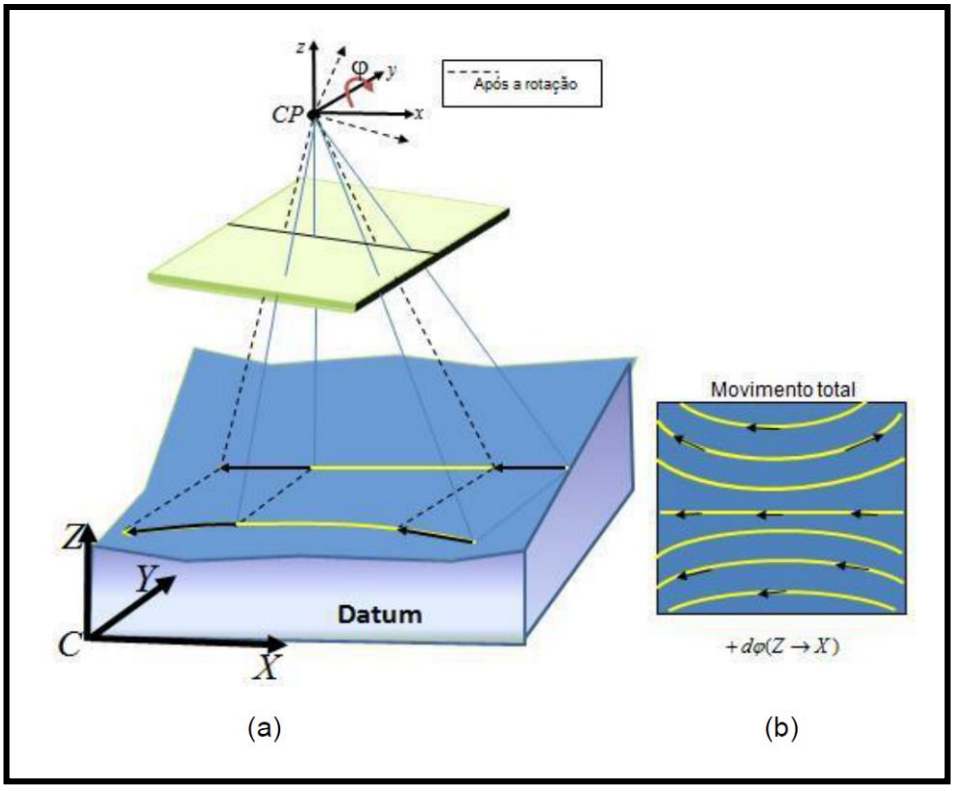


Figura 37. (a) Efeito da rotação em φ sobre a projeção dos pontos. (b) Movimento total das componentes X e Z no plano de projeção. Fonte: Santos (2013).

O efeito da rotação em κ, em torno do eixo z, provoca ainda a falta de alinhamento das varreduras e superposições ou lacunas (**Figura 38**) (D'ALGE, 2007). Considerando a matriz de D_κ (**Figura 35**), a rotação em κ no eixo z do

sistema referencial 3D produz um efeito nas coordenadas *x', y', X* e *Y* e é um somatório dos efeitos produzidos anteriormente (SANTOS, 2013; BÖRLIN, 2014).

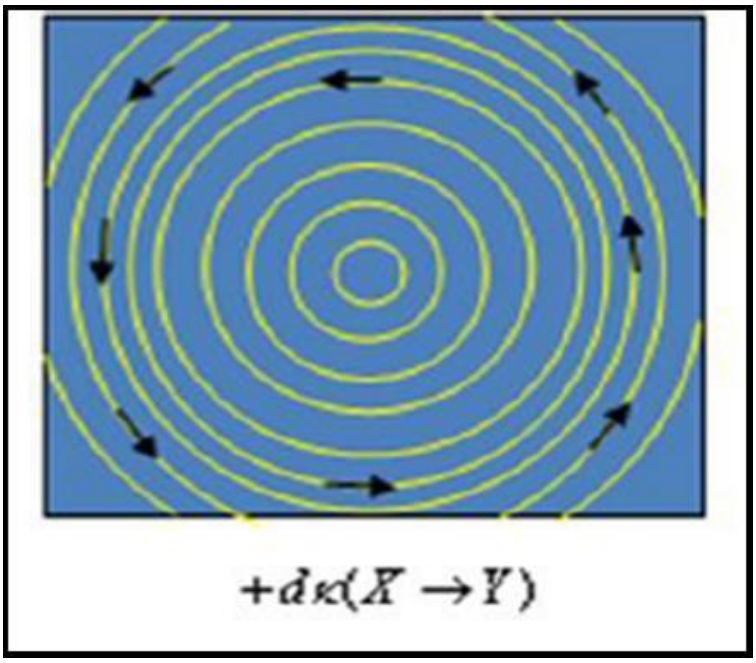


Figura 38. Movimento total das componentes X e Y no plano de projeção.
Fonte: Santos (2013).

As transformações geométricas 3D são empregadas em situações onde é necessária a transformação de pontos contidos em sistemas tridimensionais e as principais utilizadas na fotogrametria são: corpo rígido, afim, projetiva e linear direta (HARTLEY e ZISSERMAN, 2000; LIMA e BRITO, 2006; SANTOS, 2013). A transformação de corpo rígido 3D, também conhecida como transformação de similaridade 3D é representada pela Eq. 62 (ANDRADE, 1998; SANTOS, 2013; COELHO e BRITO, 2016):

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R(\varphi, \omega, \kappa) \begin{bmatrix} X - dX \\ Y - dY \\ Z - dZ \end{bmatrix} \tag{62}$$

Onde: *X', Y'* e *Z'* são as coordenadas 3D finais; *R* é a matriz de rotação dada em função dos ângulos Euler; *X, Y* e *Z* são as coordenadas 3D iniciais; *dX, dY* e *dZ* são translações; e φ, ω e κ são ângulos eulerianos no sistema fotogramétrico.

Consequentemente, obtém-se o vetor *W* (Eq. 63) pelo MMQ e pela linearização de Taylor (GEMAEL, 2004; LIMA e BRITO, 2006; SANTOS, 2013; LIMA et al., 2017):

$$W = [\, dX \; dY \; dZ \;\; \omega \; \varphi \; \kappa \,]^T \tag{63}$$

A aproximação dos valores do vetor *W* é obtida pela Eq. 64 (GEMAEL, 2004; SANTOS, 2013; COELHO e BRITO, 2016; LIMA et al., 2017):

$$W = (A^T P A)^{-1} A^T P L \tag{64}$$

Onde: *W* é o vetor dos valores que, somados ao valor inicial, aproxima-se do valor das incógnitas ou matriz dos parâmetros ajustados ou matriz das incógnitas; *A* é a matriz Jacobiana, matriz das derivadas parciais da função em relação a cada parâmetro que se deseja determinar, ou matriz dos coeficientes das incógnitas; *P* é a matriz peso das observações; e *L* é o vetor ou matriz das observações.

O vetor de resíduos tem, como objetivo, verificar a qualidade dos ajustamentos das observações (Eq. 65) (GEMAEL, 2004; SANTOS, 2013; COELHO e BRITO, 2016; LIMA et al., 2017):

$$V = A.W + L \tag{65}$$

A transformação de similaridade 3D representa o modelo mais simples de transformação 3D, com sete graus de liberdade, quando se acrescenta um fator de escala (HARTLEY e ZISSERMAN, 2000). A transformação afim 3D, também conhecida como equação polinomial do primeiro grau, é representada pela Eq. 66. Este modelo apresenta 12 graus de liberdade, possibilitando maior precisão que a transformação de similaridade (ANDRADE, 1998; MAROTTA et al., 2011; SANTOS, 2013; COELHO e BRITO, 2016):

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} dX \\ dY \\ dZ \end{bmatrix} \tag{66}$$

Onde: *X', Y'* e *Z'* são as coordenadas 3D finais; *a, b, c, d, e, f, g, h, i, dX, dY* e *dZ* são parâmetros da transformação; e *X, Y* e *Z* são as coordenadas 3D iniciais.

Comparando o RMS das transformações afim 3D, projetiva 3D e projetiva 3D modificada, a afim 3D apresentou o maior RMS (MAROTTA et al., 2011).

Modelos matemáticos com maior eficiência na correção geométrica envolvem as equações de colinearidade. As equações de colinearidade, quanto ao aspecto de correção da distorção projetiva, foram explicadas de forma sucinta no item 2.2.2.2 orientação exterior deste trabalho (Eqs. 36 a 38). Neste momento, será explicada a geometria e o desenvolvimento do modelo de colinearidade (ANDRADE, 1998; SANTOS, 2013; COELHO e BRITO, 2016; LIMA et al., 2017). Os elementos ilustrados na **Figura 39** são definidos como:

- *Ox'y'* : sistema fiducial. O eixo x' coincide com a direção da linha de voo;
- *CP*: centro perspectivo da câmara;
- *CP xyz* : sistema fotogramétrico tridimensional de coordenadas. Os eixos x e y são paralelos aos eixos x' e y' e o eixo z coincide com o eixo óptico da câmara;
- *pp* : ponto principal definido pela projeção ortogonal do CP sobre o plano da imagem;
- *f* : distância focal da câmara;
- *C XYZ* : sistema de coordenadas tridimensionais no espaço-objeto;
- n_j : vetor posição imagem;
- N_j : vetor posição no espaço-objeto;
- φ, ω e κ : parâmetros de rotação do CP;
- $X_0Y_0Z_0$: parâmetros de translação do CP;
- *p'* : ponto na imagem; e
- *P* : ponto no espaço-objeto.

A fim de que o vetor n_j seja paralelo ao vetor N_j , é necessário a aplicação de um fator de escala λ^{-1} e uma rotação $M^T_{\varphi,\omega,\kappa}$ (Eq. 67) (SANTOS, 2013):

$$N_j = \lambda^{-1}. M^T_{\varphi,\omega,\kappa} . n_j \qquad (67)$$

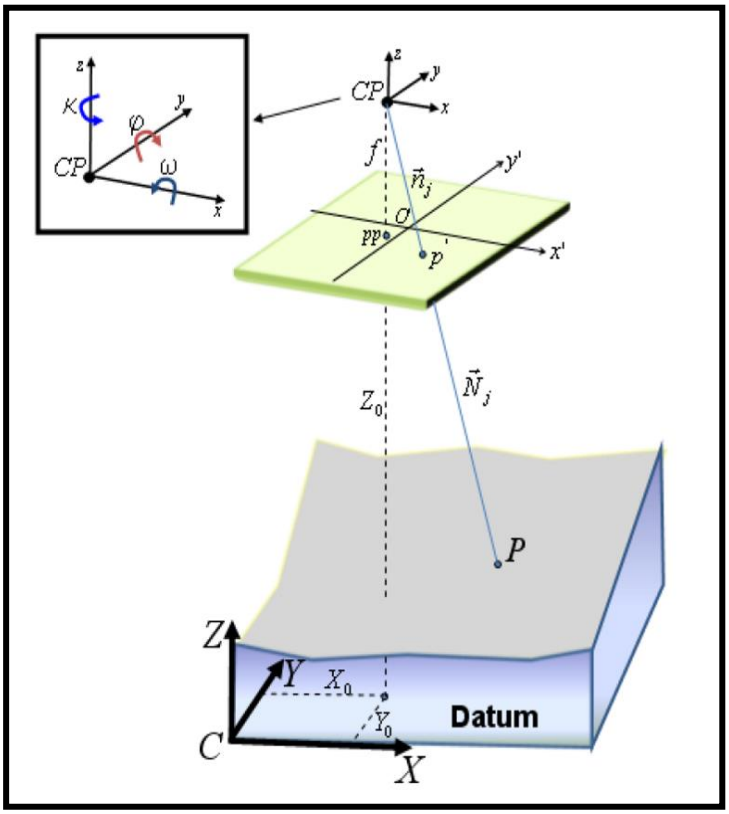


Figura 39. Geometria do modelo de colinearidade. Fonte: Santos (2013).

A forma matricial da Eq. 67, após a transformação inversa, é definida pela Eq. 68 (ANDRADE, 1998; SANTOS, 2013; COELHO e BRITO, 2016):

$$\begin{bmatrix} x \\ y \\ -f \end{bmatrix} = \lambda \cdot \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix} \tag{68}$$

Desenvolvendo a Eq. 68, com a eliminação de λ, chega-se a duas equações fundamentais ou equações de colinearidade direta (Eqs. 69 e 70), semelhantes às Eqs. 37 e 38, diferindo apenas dos sistemas de referências adotados, como por exemplo, o fotogramétrico ou o aeronáutico (ANDRADE, 1998; SANTOS, 2013; COELHO e BRITO, 2016; LIMA et al., 2017; ROBERTO et al., 2017):

$$x = -f \cdot \frac{m_{11}(X-X_0)+m_{12}(Y-Y_0)+m_{13}(Z-Z_0)}{m_{31}(X-X_0)+m_{32}(Y-Y_0)+m_{33}(Z-Z_0)} \tag{69}$$

$$y = -f.\frac{m_{21}(X-X_0)+m_{22}(Y-Y_0)+m_{23}(Z-Z_0)}{m_{31}(X-X_0)+m_{32}(Y-Y_0)+m_{33}(Z-Z_0)} \quad (70)$$

A forma matricial da Eq. 67, sem a transformação inversa, é definida pela Eq. 71 (ANDRADE, 1998; SANTOS, 2013; COELHO e BRITO, 2016):

$$\begin{bmatrix} X-X_0 \\ Y-Y_0 \\ Z-Z_0 \end{bmatrix} = \lambda^{-1}. \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ -f \end{bmatrix} \quad (71)$$

Desenvolvendo a Eq. 71, com a eliminação de λ^{-1}, chegam-se a duas equações de colinearidade inversa (Eqs. 72 e 73) (ANDRADE, 1998; SANTOS, 2013; COELHO e BRITO, 2016). Estas equações permitem a interseção espacial para um par de imagens, definição já explicada no item 2.2.2.2 orientação exterior (COELHO e BRITO, 2016).

$$X = (Z-Z_0)\frac{m_{11}(x)+m_{21}(y)+m_{31}(-f)}{m_{13}(x)+m_{23}(y)+m_{33}(-f)} + X_0 \quad (72)$$

$$Y = (Z-Z_0)\frac{m_{12}(x)+m_{22}(y)+m_{32}(-f)}{m_{13}(x)+m_{23}(y)+m_{33}(-f)} + Y_0 \quad (73)$$

A transformação projetiva 3D utiliza os princípios das equações de colinearidade (LIMA et al., 2017; ROBERTO et al., 2017), que representa um modelo matemático linearizado (ANDRADE, 1998; LIMA e BRITO, 2006). Este modelo apresenta 15 graus de liberdade (três de rotação, três de translação, um para escala isotrópica, cinco para escalas afins e três para a parte projetiva da transformação), possibilitando maior precisão que as transformações de similaridade e afim (HARTLEY e ZISSERMAN, 2000).

Comparando RMS das transformações afim 3D, projetiva 3D e projetiva 3D modificada, a projetiva 3D tem apresentado o segundo menor RMS (MAROTTA et al., 2011).

A transformação linear direta 3D é derivada do princípio da colinearidade. Este modelo leva em consideração parâmetros de escala diferenciados para os eixos x e y e coordenadas ajustadas do centro perspectivo (LIMA e BRITO, 2006). A transformação linear direta 3D possibilita a calibração de câmeras não métricas, dando ênfase aos parâmetros de orientação interior e de orientação exterior, principalmente na correção geométrica de imagens orbitais. A equação linear direta 3D é obtida pela combinação de uma transformação afim geral no plano com a equação de colinearidade (Eqs. 74 e 75) (SANTOS, 2013):

$$x = \frac{L_1X + L_2Y + L_3Z + L_4}{L_9X + L_{10}Y + L_{11}Z + 1} \tag{74}$$

$$y = \frac{L_5X + L_6Y + L_7Z + L_8}{L_9X + L_{10}Y + L_{11}Z + 1} \tag{75}$$

Onde: x e y são as coordenadas planas no referencial da imagem; X, Y e Z são as coordenadas tridimensionais no referencial geodésico local; e L_1 .... L_{11} são os parâmetros de transformação.

Os parâmetros de transformação são determinados pelo processo de ajustamento por MMQ (GEMAEL, 2004). Esta transformação não se aplica diretamente na correção geométrica de imagens do RPA de asa fixa classe 3, pois é adequada à calibração de câmeras não métricas e georreferenciamento de imagens de satélites (SANTOS, 2013).

Considerando as transformações de corpo rígido, afim, projetiva e linear direta, a projetiva possibilita modificações em sua fórmula e apresenta 15 graus de liberdade para uma melhor precisão cartográfica, possibilitando menores RMS (HARTLEY e ZISSERMAN, 2000; MAROTTA et al., 2011; COELHO e BRITO, 2016). Transformações projetivas modificadas como o modelo projetivo modificado, onde há uma correção adicional para coordenadas da imagem para o ajuste de erros

sistemáticos, apresentam menores RMS em relação à transformação projetiva não modificada (MAROTTA et al., 2011).

Esta tese se propõe a modificar o parâmetro matriz de rotação R_ψ da transformação projetiva básica, considerando ser esta transformação a mais eficaz na correção geométrica em imagens de RPA de asa fixa.

2.3.3 Correção geométrica no ângulo de guinada

As imagens com visada lateral possuem um nível considerável de distorção geométrica. À medida que aumenta o ângulo de distorção, o tamanho dos pixels na imagem é também alterado de forma não uniforme. O tipo de terreno, inclinado e elevado, e a curvatura da terra são fatores secundários que podem induzir uma leve distorção na imagem (**Figura 40**) (JAIMES, 2016).

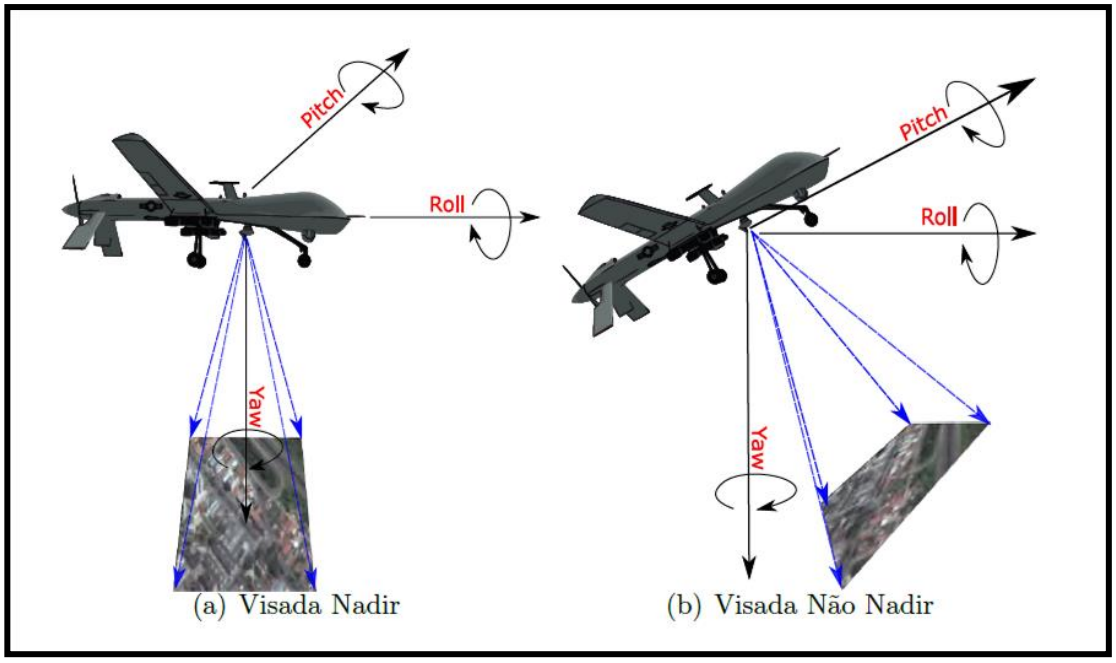


Figura 40. Tipos de visada com o RPA. Fonte: Jaimes (2016).

Considerando as atitudes *roll, pitch* e *yaw* da aeronave, o ângulo *yaw* é afetado de maneira acentuada devido ao vento no sentido transversal ou lateral da aeronave (LIMA, 2016; ONUORA et al., 2018). No aerolevantamento da primeira

área deste trabalho, observa-se a ação do vento nas três faixas, provocando o efeito leque (**Figura 41**).

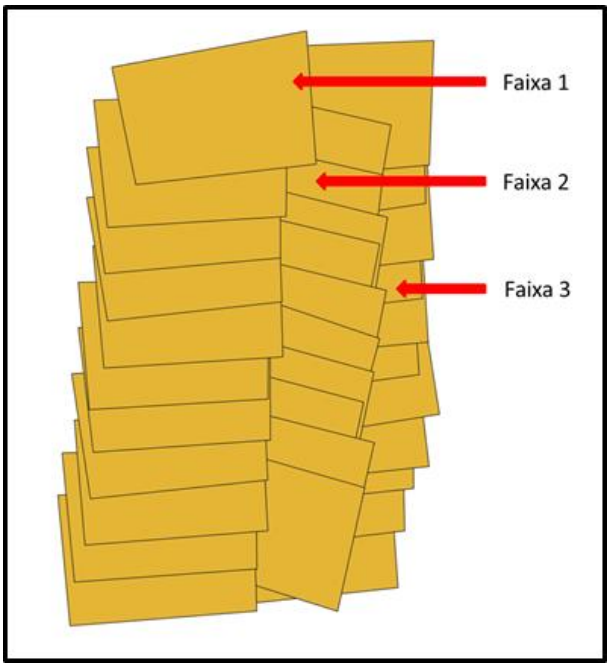


Figura 41. Mosaico da primeira área com três faixas de voo do RPA. Fonte: Dado vetorial criado no aplicativo de sistema de informações geográficas QGIS 3.4.0.

Tomando como referência as Eqs. 36 a 38, onde é adotado o sistema aeronáutico, pois os sistemas de navegação de RPAs utilizam como referência um sistema de eixos (X_a, $\underline{Y_a}$ e Z_a) fixos no corpo com origem no centro de gravidade, conforme já explicado no item 2.2.2.2 orientação exterior (ROBERTO et al., 2017), a matriz de rotação no sistema aeronáutica A é composta de três matrizes de rotação (Eqs. 30 a 35).

A proposta deste trabalho é a alteração da matriz R_ψ (Eq 32) que é uma das componentes de A (matriz original), onde se tem a correção do ângulo ψ. A alteração será no sentido de trocar primeiramente todos os sinais dos termos da primeira linha. Em seguida, trocar as posições da linha 1 com a linha 2. Finalmente, trocar sinais do seno da diagonal secundária e derivar o termo a_{21} de R_ψ (Eq. 76) ou trocar os sinais do cosseno da diagonal principal e derivar o termo a_{22} de R_ψ (Eq. 77) (HALLERT, 1960):

$$Rd1_{\psi} = \begin{bmatrix} cos\psi & -(-sen\psi) & 0 \\ -(sen\psi)' & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (76)$$

$$Rd2_{\psi} = \begin{bmatrix} -(cos\psi) & -sen\psi & 0 \\ sen\psi & -(cos\psi)' & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (77)$$

Conforme mostrado nas Tabelas de 2 a 6 e o sentido anti-horário ou positivo, os ângulos de guinada elevados estão na sua maioria nos quadrantes 1°, 2° e 3° do ciclo trigonométrico (**Figura 42**) (IEZZI, 1977).

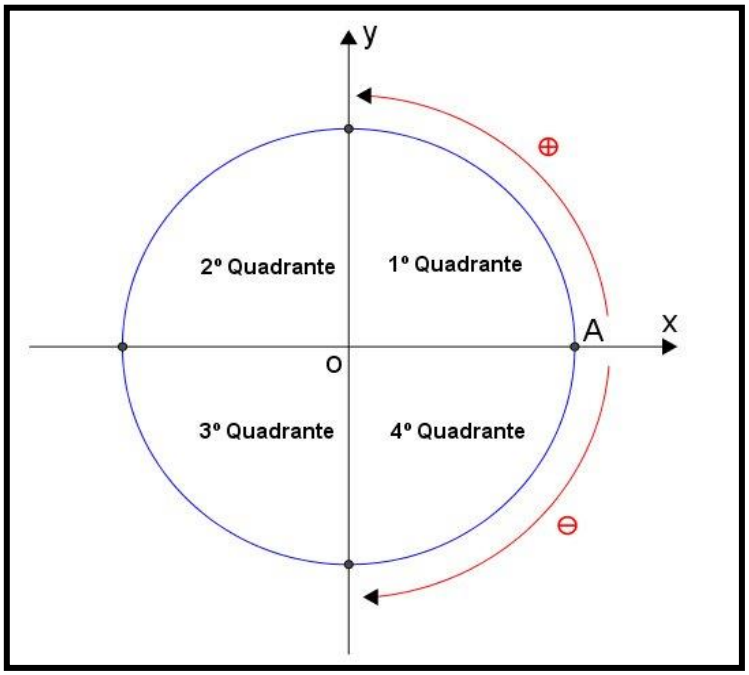


Figura 42. Ciclo trigonométrico com os quatro quadrantes.
Fonte: https://www.infoescola.com/matematica/circulo-trigonometrico/.

Considerando o ângulo central *α* do ciclo trigonométrico como sendo o ângulo de guinada, o eixo X representa os cossenos e o eixo Y representa os senos, com base em um ângulo agudo de um triangulo retângulo inscrito neste ciclo trigonométrico. É importante relacionar medidas de arcos trigonométricos com extremidades simétricas do ponto A (A' e A'') em relação a um dos eixos de

coordenadas, possibilitando o cálculo do seno ou cosseno do ângulo de guinada. Desta forma, com o valor π radianos igual a 180°, tem-se que a redução de um ângulo de guinada do 2° para o 1° quadrante será sen (π - ψ) = senψ e cos (π - ψ) = –cosψ e a redução de um ângulo de guinada do 3° para o 1° quadrante será sen (π + ψ) = –senψ e cos (π + ψ) = –cosψ (**Figura 43**) (IEZZI, 1977).

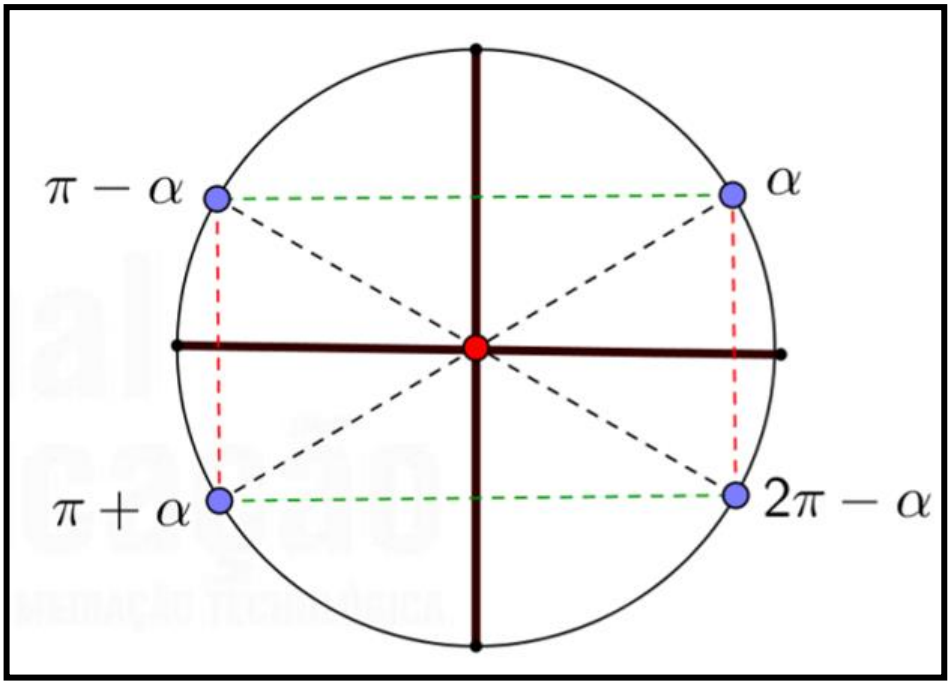


Figura 43. Ciclo trigonométrico com a redução de quadrantes.
Fonte: https://guiadoestudante.abril.com.br.

As derivadas trigonométricas são: (senψ)' = cosψ, (–senψ)' = - cosψ, (cosψ)' = - senψ e (–cosψ)' = senψ (LEITHOLD, 1994). As Eqs. 76 e 77 permitem que se faça a maior rotação de correção possível, dependendo do ângulo de guinada. As Eqs. 78 e 79 se aplicam a ângulos do 1° quadrante (0° ≤ α ≤ 90°)(ANTON e RORRES, 2012):

$$Rd11_{\psi} = \begin{bmatrix} cos\psi & sen\psi & 0 \\ -cos\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (78)$$

$$Rd21\psi = \begin{bmatrix} -cos\psi & -sen\psi & 0 \\ sen\psi & sen\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (79)$$

Para o 2° quadrante (90° < α ≤ 180°), realizando a redução nas Eqs. 80 e 81, aplicam-se as Eqs. 81 e 82:

$$R'd1_{\psi} = \begin{bmatrix} cos(\pi - \psi) & -(-sen(\pi - \psi)) & 0 \\ -(sen(\pi - \psi))' & cos(\pi - \psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (80)$$

$$R'd2_{\psi} = \begin{bmatrix} -(cos(\pi - \psi)) & -sen(\pi - \psi) & 0 \\ sen(\pi - \psi) & -(cos(\pi - \psi))' & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (81)$$

$$Rd12_{\psi} = \begin{bmatrix} -cos\psi & sen\psi & 0 \\ -cos\psi & -cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (82)$$

$$Rd22_{\psi} = \begin{bmatrix} cos\psi & -sen\psi & 0 \\ sen\psi & -sen\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (83)$$

Para o 3° quadrante (180° ≤ α ≤ 270°), realizando a redução nas Eqs. 84 e 85, aplicam-se as Eqs. 86 e 87:

$$R''d1_{\psi} = \begin{bmatrix} cos(\pi + \psi) & -(-sen(\pi + \psi)) & 0 \\ -(sen(\pi + \psi))' & cos(\pi + \psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (84)$$

$$R''d2_{\psi} = \begin{bmatrix} -(cos(\pi + \psi)) & -sen(\pi + \psi) & 0 \\ sen(\pi + \psi) & -(cos(\pi + \psi))' & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (85)$$

$$Rd13_{\psi} = \begin{bmatrix} -cos\psi & -sen\psi & 0 \\ cos\psi & -cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (86)$$

$$Rd23_{\psi} = \begin{bmatrix} cos\psi & sen\psi & 0 \\ -sen\psi & -sen\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (87)$$

Desenvolvendo $A = R_{\phi} \cdot R_{\theta} \cdot R_{\psi}$, tem-se que $A' = R_{\phi} \cdot R_{\theta} \cdot Rd11_{\psi}$ ou $A'' = R_{\phi} \cdot R_{\theta} \cdot Rd21_{\psi}$ e as Eqs. 34 e 35 são remodeladas para as Eqs. 88 a 91:

$$A' = \begin{bmatrix} -cos\theta cos\psi & -cos\theta sen\psi & sen\theta \\ -sen\phi sen\theta cos\psi - cos\phi cos\psi & -sen\phi sen\theta sen\psi + cos\phi cos\psi & -sen\phi cos\theta \\ -sen\theta cos\phi cos\psi + sen\phi cos\psi & -sen\theta cos\phi cos\psi - sen\phi cos\psi & -cos\phi cos\theta \end{bmatrix} \quad (88)$$

$$A' = \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} \\ a'_{21} & a'_{22} & a'_{23} \\ a'_{31} & a'_{32} & a'_{33} \end{bmatrix} \quad (89)$$

$$A'' = \begin{bmatrix} cos\theta cos\psi & cos\theta sen\psi & sen\theta \\ sen\phi sen\theta cos\psi + cos\phi sen\psi & sen\phi sen\theta sen\psi + cos\phi sen\psi & -sen\phi cos\theta \\ sen\theta cos\phi cos\psi - sen\phi sen\psi & sen\theta cos\phi sen\psi - sen\phi cos\psi & -cos\phi cos\theta \end{bmatrix} \quad (90)$$

$$A'' = \begin{bmatrix} a''_{11} & a''_{12} & a''_{13} \\ a''_{21} & a''_{22} & a''_{23} \\ a''_{31} & a''_{32} & a''_{33} \end{bmatrix} \quad (91)$$

Por fim, as Eqs. 36 a 38 são refeitas e são geradas as Eqs. 92 a 97, devido aos dois modos de operação obtidos pelas as duas matrizes derivadas $Rd11_{\psi}$ e $Rd21_{\psi}$.:

$$\lambda \cdot \begin{bmatrix} x_a \\ y_a \\ f \end{bmatrix} = A' \cdot \begin{bmatrix} E - E_m \\ N - N_m \\ U - U_m \end{bmatrix} \quad 36) \ (92)$$

$$x_a = f. \frac{a'_{11}(E-E_m)+a'_{12}(N-N_m)+a'_{13}(U-U_m)}{a'_{31}(E-E_m)+a'_{32}(N-N_m)+a'_{33}(U-U_m)} \quad (93)$$

$$y_a = f. \frac{a'_{21}(E-E_m)+a'_{22}(N-N_m)+a'_{23}(U-U_m)}{a'_{31}(E-E_m)+a'_{32}(N-N_m)+a'_{33}(U-U_m)} \quad 38) \ (94)$$

$$\lambda \cdot \begin{bmatrix} x_a \\ y_a \\ f \end{bmatrix} = A'' \cdot \begin{bmatrix} E - E_m \\ N - N_m \\ U - U_m \end{bmatrix} \quad 36) \ (95)$$

$$x_a = f. \frac{a''_{11}(E-E_m)+a''_{12}(N-N_m)+a''_{13}(U-U_m)}{a''_{31}(E-E_m)+a''_{32}(N-N_m)+a''_{33}(U-U_m)} \quad 37) \ (96)$$

$$y_a = f. \frac{a''_{21}(E-E_m)+a''_{22}(N-N_m)+a''_{23}(U-U_m)}{a''_{31}(E-E_m)+a''_{32}(N-N_m)+a''_{33}(U-U_m)} \quad 38) \ (97)$$

2.4 MÉTODO PARA AVALIAÇÃO DOS ORTOMOSAICOS

O método aplicado na avaliação dos ortomosaicos foi o proposto no aplicativo GeoPEC (SANTOS et al., 2016). Nesse contexto, o importante em qualquer aerolevantamento espacial não é eliminar, mas gerenciar a incerteza inerente ao processo de geração do ortomosaico. Para avaliar a acurácia posicional no aplicativo, é necessária a entrada das coordenadas e/ou distâncias de referência e das coordenadas e/ou distâncias do dado em que se quer avaliar (SANTOS, 2018).

A acurácia posicional refere-se a quão próxima a posição de um dado espacial está em relação à sua realidade no terreno (MONICO et al., 2009; ZHOU e REICHLE, 2010). Uma amostra de checagem com o padrão de distribuição espacial do tipo agrupado pode comprometer a avaliação da acurácia posicional desse dado espacial. Dessa forma, técnicas de estatística espacial, como métodos do vizinho mais próximo e a função *K* de Ripley, podem ser bastante úteis para a análise das amostras quanto à distribuição espacial das mesmas, onde $R < 1$ é agrupado, $R = 1$ é aleatório e $R > 1$ é disperso (SANTOS et al., 2016).

Na análise da distribuição espacial, o método do vizinho mais próximo de alta ordem utilizado no GeoPEC compara a distância média entre os vizinhos mais próximos com um conjunto de pontos que têm um padrão definido teoricamente. Normalmente, o método do vizinho mais próximo é utilizado apenas na primeira ordem ($k = 1$). O cálculo se inicia com a determinação do índice *R*. As Eqs. 98 a 100 representam o índice *R* (SANTOS et al., 2016):

$$R(K) = \frac{R_{obs}(K)}{R_{esp}(K)} \tag{98}$$

$$R_{obs}(K) = \frac{\sum dv_i(K)}{n}, i \text{ a } n \text{ no somatório,} \tag{99}$$

$$R_{esp}(K) = \gamma_{1(k)}\sqrt{(A/n)} \tag{100}$$

Onde $R_{obs}(k)$ é a média observada das distâncias de cada ponto ao seu *k* vizinho mais próximo; $R_{esp}(k)$ é a média esperada das distâncias entre os *k* vizinhos mais próximos para a distribuição aleatória; $dv_i(k)$ é a distância de um ponto *i* ao seu *k* vizinho mais próximo; *n* é o número de pontos; e *A* é a área da região em estudo.

Para inferir se o índice *R* é estatisticamente igual ao valor da distribuição aleatória, aplica-se o teste *Z*, onde, na hipótese nula, admite-se que o padrão da distribuição espacial dos dados siga o padrão aleatório. Se a estatística *Z* calculada para a ordem *k* ($Z_R(k)$), por meio das Eqs. 101 e 102, for maior que o valor tabelado

para Z ($Z_{crítico}$), rejeita-se a hipótese nula. Para K = 1, $\gamma 1(K)$ = 0,5 e $\gamma 2(K)$ = 0,2613 (SANTOS et al., 2016).

$$Z_R(K) = \frac{R_{obs}(K) - R_{esp}(K)}{SE_r(K)} \quad (101)$$

$$SE_r(K) = \gamma_{2(K)} \sqrt{(A/n^2)} \quad (102)$$

Onde a variável $SE_r(K)$ representa o EP da diferença entre as distâncias médias observadas e as esperadas entre os vizinhos mais próximos para a ordem k. A primeira ordem realiza uma análise puramente local.

Não foi utilizada a função K de *Ripley* neste trabalho, pois a sua aplicabilidade é limitada, além de não apresentar uniformidade em todas as direções (anisotrópico).

A acurácia expressa o grau de proximidade de uma estimativa com o parâmetro para qual ela foi estimada. Assim, a acurácia incorpora efeitos sistemáticos e aleatórios (MONICO et al., 2009). Portanto, entende-se que a acurácia envolve tanto a tendência (efeitos sistemáticos) como a precisão (efeitos aleatórios) (IBGE, 2019). Primeiramente, ao realizar o teste de tendência baseado no teste t de *Student* e o teste de precisão (qui-quadrado - χ^2), tem-se, como requisito básico, que a amostra siga uma distribuição normal ou gaussiana. O teste de normalidade *Bera-Jarques* ou *Bowman-Shelton* baseia-se na diferença entre os coeficientes de assimetria e curtose da amostra quando comparados aos valores de assimetria e curtose de uma distribuição normal. As hipóteses desse teste são:

H_0: a amostra segue distribuição normal (assimetria (*Ass*) = 0 e curtose (*Cur*) = 3); e

H_1: a amostra não segue distribuição normal.

As Eqs. 103 a 105 representam o teste de normalidade (YULIANTO, 2012).

$$B_{calc} = n\left[\frac{(Ass^2/6 + (Cur-3)^2)}{24}\right] \quad (103)$$

$$Ass = \frac{\sum(x_i - x')^3}{nS^3} \quad (104)$$

$$Cur = \frac{\sum(x_i - x')^4}{nS^4} \quad (105)$$

No controle de qualidade de dados espaciais, a análise de tendências é de fundamental importância. A maioria das avaliações de acurácia posicional que analisam tendências utiliza o teste de hipóteses *t* de *Student*. A análise de tendência é realizada com base nas discrepâncias entre as coordenadas observadas e as coordenadas de referência (Eq. 106), de onde se obtêm as estatísticas como média (Eq. 107) e desvio-padrão (Eqs. 108 e 109), sendo *n* o número de elementos da amostra (SILVA et al., 2015).

$$\Delta X = (X_o - X_r) \quad e \quad \Delta Y = (Y_o - Y_r) \quad (106)$$

$$\Delta X' = \left(\frac{1}{n}\right)\sum \Delta X \quad e \quad \Delta Y' = \left(\frac{1}{n}\right)\sum \Delta Y,\ i = 1 \text{ a } n \text{ no somatório.} \quad (107)$$

$$S_{\Delta X1} = \frac{1}{(n-1)}\sum(\Delta X - \Delta X')^2 \quad e \quad S_{\Delta Y1} = \frac{1}{(n-1)}\sum(\Delta Y - \Delta Y')^2 \quad (108)$$

$$S_{\Delta X} = \sqrt{S_{\Delta X1}} \quad e \quad S_{\Delta Y} = \sqrt{S_{\Delta Y1}} \quad (109)$$

Para o teste de tendência, são avaliadas as seguintes hipóteses:

H_0: se $\Delta X' = 0$, então *X* não é tendencioso; H_1: se $\Delta X' \neq 0$, então *X* é tendencioso; e H_0: se $\Delta Y' = 0$, então *Y* não é tendencioso; H_1: se $\Delta Y' \neq 0$, então *Y* é tendencioso.

A partir do número de pontos de referência utilizados na análise, obtém-se o valor limite tabelado $t_{n-1,\alpha/2}$. Se o valor do teste *t* de *Student* calculado (Eqs. 110 e 111), for inferior ao $t_{crítico}$ (tabelado), pode-se afirmar que o ortomosaico está livre de erros sistemáticos (ALVES JÚNIOR et al., 2015; SILVA et al., 2015).

$$t_x = \frac{\Delta X'}{S_{\Delta X}}\sqrt{n} \quad e \quad t_y = \frac{\Delta Y'}{S_{\Delta Y}}\sqrt{n} \tag{110}$$

$$|t_{calc}| < t_{n-1,\alpha/2} \tag{111}$$

Há técnicas de estatística espacial que analisam a presença de tendência nos dados, independentemente da distribuição estatística dos dados: a média direcional e a variância circular. O objetivo da média direcional é obter uma medida de tendência central da direção de um conjunto de vetores, cujas componentes são as discrepâncias nas ordenadas e nas abscissas. A média direcional apenas descreve uma tendência da direção, mas não consegue distinguir a variabilidade desta direção. Essa variabilidade é dada pela variância circular que é calculada a partir do comprimento do vetor resultante (SANTOS et al., 2016).

A análise de tendência na avaliação da acurácia posicional é de grande importância, já que esses efeitos sistemáticos podem ser modelados (MONICO et al., 2009). Não se descarta um dado espacial porque ele é tendencioso, já que este pode servir para determinação de áreas, distâncias e ângulos entre feições, se a escala estiver consistente (SANTOS et al., 2016).

Para a análise da precisão, foi utilizado o teste de χ^2, obedecendo aos valores do EP determinados no Decreto-Lei Nº 87.817/84 e ET-CQDG, que estabelece três e quatro classes, respectivamente. Para realizar o teste χ^2, inicialmente calcula-se o EP esperado (σ) de cada componente (Eq. 112) (SILVA et al., 2015; OLIVEIRA e BRITO, 2019).

$$\sigma_x = \sigma_y = \frac{EP}{\sqrt{2}} \tag{112}$$

Posteriormente, aplica-se um teste de hipótese, comparando-se o desvio-padrão das discrepâncias com o EP esperado para as classes do PEC e do PEC-PCD:

H_0: $S_{\Delta x}^2 = \sigma_x^2$; H_1: $S_{\Delta x}^2 > \sigma_x^2$; e H_0: $S_{\Delta y}^2 = \sigma_y^2$; H_1: $S_{\Delta y}^2 > \sigma_y^2$.

Para atender a precisão de uma determinada classe, o valor do teste χ^2 calculado (χ''^2_x e Y''^2_y) (Eqs. 113 e 114), deve ser inferior ao teste χ^2 *crítico* ($\chi''^2_{n-1,\alpha}$) (Eqs. 115 e 116), onde *n* é o tamanho da amostra (ALVES JÚNIOR et al., 2015; SILVA et al., 2015).

$$\chi''^2_x = (n-1)\left(\frac{S_{\Delta x^2}}{\sigma_{x^2}}\right) \quad (113)$$

$$\chi''^2_y = (n-1)\left(\frac{S_{\Delta y^2}}{\sigma_{y^2}}\right) \quad (114)$$

$$\chi''^2_x \leq \chi''^2_{n-1,\alpha} \quad (115)$$

$$\chi''^2_y \leq \chi''^2_{n-1,\alpha} \quad (116)$$

2.5 APLICATIVO PYTHON E BIBLIOTECA DIGITAL OPENCV

Python é uma linguagem de programação de alto nível, lançada por Guido van Rossum em 1991. Possui um modelo de desenvolvimento comunitário e gerenciado pela organização sem fins lucrativos *Python Software Foundation*. A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Combina uma sintaxe concisa e clara com os recursos

poderosos de sua biblioteca padrão e por módulos e *frameworks* desenvolvidos por terceiros (SEVERANCE, 2009; MATTHES, 2017; PYTHON, 2019).

OpenCV foi desenvolvida pela Intel em 2000. É uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de visão computacional. Esta biblioteca foi desenvolvida nas linguagens de programação C/C++, Java, Python e Visual Basic. Neste trabalho, o OpenCV foi utilizado como biblioteca básica para o Python (OPENCV, 2019).

No capítulo seguinte, são realizadas as fases primeira a quarta: planejamento e execução dos voos, determinação dos pontos de apoio e de verificação, geração de ortomosaicos iniciais e avaliação dos ortomosaicos iniciais, sem a correção do ângulo de guinada.

3 EXPERIMENTO E RESULTADOS PRELIMINARES

3.1 PLANEJAMENTO E EXECUÇÃO DOS VOOS

Na primeira fase desta tese, o imageamento da primeira área foi feito em três faixas de voo, com 300 metros de altura em média, totalizando 17 imagens, sendo 5 imagens na faixa 1 (marcadores amarelos), 6 imagens na faixa 2 (marcadores verdes) e 6 imagens na faixa 3 (marcadores azuis), com 22 pontos de campo (marcadores vermelhos) (**Figura 44**).

Figura 44. Imageamento da primeira área, região de Itaipuaçu, Maricá, RJ, envolvendo três faixas de voo (amarelo, verde e azul) e pontos de campo (vermelho). Fonte: Elaborada pelo autor.

O imageamento da segunda área foi feito, com 1000 metros de altura em média, totalizando 17 imagens em uma única faixa de voo, totalizando 28 imagens (marcadores amarelos), com 22 pontos de campo (marcadores vermelhos) (**Figura 45**).

Figura 45. Imageamento da segunda área, região de garimpo em Capoeirana, Nova Era, MG, envolvendo apenas uma faixa de voo (amarelo) e pontos de campo (vermelho). Fonte: Elaborada pelo autor.

Na Tabela 7 são apresentados as faixas, pontos, coordenadas dos centros perspectivos, alturas de voo e ângulos de orientação *pitch*, *roll* e *yaw*, totalizando 17 pontos analisados da primeira área (Maricá, RJ). Os valores máximos para *pitch*, *roll* e *yaw* foram, respectivamente, de 1,623º; 2,742º e 179,497º, enquanto os valores mínimos foram, respectivamente, de -3,749º; -1,119º e -179,441º. As amplitudes (diferença entre os valores máximo e mínimo) para o *pitch*, *row* e *yaw* foram, respectivamente, de 5,372º; 3,861º; e 358,938º, com os correspondentes valores de desvio-padrão de 1,284º; 1,059º e 147,715º. As médias foram, respectivamente, de -0,306º, 0,592º e -9,449º.

Tabela 7. Dados relativos a centros perspectivos, altura, *pitch*, *roll* e *yaw* do aerolevantamento fotogramétrico efetuado na primeira área (Maricá, RJ).

Faixa	Ponto	Longitude (graus)	Latitude (graus)	Altura (m)	*Pitch*	*Roll*	*Yaw*
1	P4	-42,947738	-22,965074	299	-0,951	-1,007	179,217

1	P5	-42,947784	-22,966016	298	-0,168	0,336	177,539
1	P6	-42,947826	-22,967004	297	1,231	1,063	179,497
1	P7	-42,947910	-22,967864	298	0,504	-1,063	-178,042
1	P8	-42,948001	-22,968801	299	0,839	0,783	-179,161
2	P11	-42,946044	-22,968894	301	0,056	0,224	5,707
2	P12	-42,945957	-22,968107	301	-1,567	-0,112	5,539
2	P13	-42,945884	-22,967353	299	0,839	-1,119	3,301
2	P14	-42,945827	-22,966522	299	-0,504	1,287	2,910
2	P15	-42,945774	-22,965740	300	-0,895	0,504	3,245
2	P16	-42,945709	-22,964771	300	-3,749	1,343	2,518
3	P4	-42,944522	-22,965061	297	1,623	1,567	-179,441
3	P5	-42,944610	-22,965845	299	-0,560	0,168	-178,322
3	P6	-42,944690	-22,966693	299	0,280	1,902	-179,385
3	P7	-42,944763	-22,967540	300	0,056	0,392	177,595
3	P8	-42,944740	-22,968338	299	-0,504	2,742	176,028
3	P9	-42,944763	-22,969133	300	-1,735	1,063	-179,385

Fonte: Elaborada pelo autor.

Na Tabela 8 são apresentados os pontos, as coordenadas dos centros perspectivos, as alturas de voo e os ângulos de orientação *pitch*, *roll* e *yaw* da segunda área (Nova Era, MG), totalizando 28 pontos analisados. Os valores máximos para *pitch*, *roll* e *yaw* foram, respectivamente, de 10,260º; 7,060º e 95,320º e os valores mínimos foram, respectivamente, de -8,080º, -5,820º e 72,120º. As amplitudes para o *pitch*, *row* e *yaw* foram, respectivamente, de 18,340º; 12,880º; e 23,200º, com os correspondentes valores de desvio-padrão de 4,047º; 3,229º e 6,013º. As médias foram, respectivamente, de 0,891º; 2,762º e 82,559º.

Tabela 8. Dados relativos a centros perspectivos, altura, *pitch*, *roll* e *yaw* do aerolevantamento fotogramétrico efetuado na segunda área (Nova Era, MG).

Ponto	Longitude (graus)	Latitude (graus)	Altura (m)	*Pitch*	*Roll*	*Yaw*
P1	-43,053943	-19,703799	1084,732	1,426	-2,817	80,176
P2	-43,054430	-19,703749	1082,129	1,864	0,567	76,441
P3	-43,055099	-19,703722	1081,179	1,269	-0,462	80,360
P4	-43,055654	-19,703704	1080,198	2,512	3,771	78,383
P5	-43,055977	-19,703691	1080,985	1,711	5,204	80,619
P6	-43,056831	-19,703730	1082,380	-2,211	3,697	90,001
P7	-43,057107	-19,703730	1082,637	-5,747	5,010	88,156
P8	-43,057741	-19,703683	1084,319	0,466	2,564	77,031
P9	-43,057970	-19,703672	1082,913	4,539	3,333	77,502
P10	-43,058291	-19,703616	1083,780	6,843	3,307	82,062
P11	-43,058924	-19,703637	1082,210	1,619	4,755	93,524
P12	-43,059207	-19,703637	1082,430	-6,625	7,014	88,948
P13	-43,059810	-19,703626	1083,386	-3,369	4,159	84,732
P14	-43,060695	-19,703580	1084,272	-0,730	2,444	84,910
P15	-43,061572	-19,703547	1087,923	-0,248	-5,820	79,595
P16	-43,062120	-19,703537	1081,923	-0,740	-4,448	74,863
P17	-43,062461	-19,703514	1078,978	-0,179	-1,054	75,715
P18	-43,062742	-19,703464	1078,559	2,405	0,986	72,115
P19	-43,063039	-19,703431	1078,161	6,317	3,246	77,766
P20	-43,063311	-19,703411	1079,180	10,258	3,243	85,763

P21	-43,063698	-19,703417	1079,109	5,343	2,112	92,971
P22	-43,063979	-19,703439	1079,469	-1,324	3,274	95,316
P23	-43,064284	-19,703474	1080,611	-8,084	4,764	86,569
P24	-43,064828	-19,703445	1080,429	2,069	4,890	78,211
P25	-43,065457	-19,703416	1081,363	2,820	4,135	85,747
P26	-43,065717	-19,703456	1081,629	-2,897	6,725	83,342
P27	-43,066281	-19,703442	1081,716	2,327	5,678	78,226
P28	-43,066617	-19,703439	1082,191	3,313	7,058	82,631

Fonte: Elaborada pelo autor.

A maior amplitude ocorreu no ângulo *yaw*, enquanto a maioria dos resultados se concentraram próximos aos valores máximos e mínimos. Com relação à curtose, os dados de *pitch* e *roll* apresentaram distribuições leptocúrticas (pico mais alto que a distribuição normal), enquanto os dados de *yaw* apresentaram uma distribuição platicúrtica (distribuição mais achatada do que a distribuição normal). Esses dados estatísticos foram obtidos por meio do pacote estatístico SPSS Statistics, versão 22.0 (BISQUERRA et al., 2004).

3.2 DETERMINAÇÃO DOS PONTOS DE APOIO E DE VERIFICAÇÃO

Para a segunda fase desta tese, os resultados da determinação dos pontos de apoio (HV) e de verificação (C), em coordenadas UTM e com as elevações em metros, são mostrados nas Tabelas 9 e 10. Na primeira área, foram obtidos 22 pontos de campo, sendo 7 pontos de apoio e 15 pontos de verificação (optou-se por 15 pontos para melhor avaliar a distorção provocada pelo ângulo *yaw* nas faixas de voo). Os valores máximo e mínimo para as elevações foram, respectivamente, 12,959 m e 1,567 m (amplitude de 11,392 m) (Tabela 9). Os valores de média e desvio-padrão foram, respectivamente, de 6,938 m e 3,483 m. A curtose dos dados foi representada por uma distribuição platicúrtica.

Tabela 9. Coordenadas dos pontos de apoio e de verificação no campo e elevações da primeira área.

Pontos	UTM Leste (m)	UTM Norte (m)	Elevação (m)
C-1	710.303,001	7.458.540,884	6,58.6
C-2	710.395,067	7.458.58.8,307	8,878
C-3	710.309,610	7.458.726,593	11,966
C-4	710.395,024	7.458.682,402	12,959
C-5	710.556,127	7.458.662,809	12,577
C-6	710.579,864	7.458.510,927	5,448
C-7	710.672,303	7.458.674,758	11,747
C-8	710.669,969	7.458.577,359	9,741
C-9	710.862,759	7.458.640,788	12,212
C-10	710.770,173	7.458.537,567	8,069
C-12	710.301,555	7.458.478,201	3,878
C-13	710.395,148	7.458.530,693	6,973
C-14	710.493,933	7.458.476,826	3,565
C-17	710.648,682	7.458.432,527	2,117
C-18	710.837,990	7.458.383,259	2,137
HV-3	710.58.5,134	7.458.476,260	3,493
HV-4	710.488,402	7.458.563,665	8,267
HV-5	710.488,897	7.458.857,402	8,703
HV-6	710.680,642	7.458.815,403	8,207
HV-9	710.304,241	7.458.863,154	8,848

HV-11	710.860,090	7.458.462,742	2,856
HV-12	710.864,972	7.458.809,663	8,280

Fonte: Elaborada pelo autor.

Na segunda área, foram obtidos 22 pontos de campo, com realização de aerotriangulação, sendo 14 pontos de apoio (optou-se por 14 pontos por ser uma área acidentada) e 8 pontos de verificação. Os valores máximo e mínimo para as elevações foram, respectivamente, de 684,400 m e 603,430 m (amplitude de 80,970 m) (Tabela 10). Os valores de média e desvio-padrão foram, respectivamente, de 634,708 m e 24,592 m. A curtose dos dados foi representada por uma distribuição platicúrtica.

Tabela 10. Coordenadas dos pontos de apoio e de verificação no campo e elevações na segunda área.

Pontos	UTM Leste (m)	UTM Norte (m)	Elevação (m)
C53	702743,150	7820185,950	626,990
C54	702572,940	7820202,810	614,800
C55	703642,330	7820151,660	619,800
C56	703463,390	7820163,470	616,850
C-57	702672,640	7820181,500	645,780
C-58	702957,570	7820178,900	678,420
C-59	703537,790	7820156,730	603,430
C-60	703818,100	7820146,620	624,400
HV1151	702588,050	7820275,060	604,660
HV1152	702755,060	7820267,030	640,950
HV1153	702727,400	7820105,360	645,780

HV1154	702593,670	7820132,710	619,370
HV1155	703043,040	7820270,140	631,120
HV1156	703032,010	7820099,730	684,400
HV1157	702941,090	7820139,500	678,420
HV1158	702879,060	7820257,890	658,540
HV1159	703465,800	7820252,970	603,430
HV1160	703645,130	7820237,140	617,390
HV1161	703658,910	7820066,470	652,870
HV1162	703466,610	7820088,820	637,070
HV1164	703897,150	7820223,050	624,400
HV1165	703732,080	7820149,970	647,580

Fonte: Elaborada pelo autor.

Os pontos HV e C das áreas 1 e 2 estão representados, respectivamente, nas Figuras 1 e 2 (marcadores vermelhos), considerando que no trabalho de campo na área 2, os pontos foram sinalizados.

3.3. GERAÇÃO DE ORTOMOSAICOS INICIAIS

Na terceira fase desta tese, na criação dos ortomosaicos no E-Foto, primeiramente foi definido o projeto, configurando-se todas as informações necessárias ao mesmo (PIRAS et al., 2017). Em seguida, foi necessário realizar a orientação interior que permite a reconstrução do feixe perspectivo que geraram as imagens. Na sequência, realizou-se a orientação exterior pelo método da ressecção espacial que orienta, em posição e atitude, cada imagem em relação ao referencial de coordenadas tridimensionais do terreno. O penúltimo passo foi a extração dos MDE e, finalmente, a criação dos ortomosaicos (RIBEIRO et al., 2014).

Foram levadas em consideração todas as informações sobre os voos e o trabalho de campo, isto é, valores de altitude máxima (12,959 m e 684,400 m), mínima (1,567 m e 603,430 m) e média (6,938 m e 634,700 m), sistema de projeção cartográfica (UTM), sistema de referência geodésica (SIRGAS 2000), distâncias focais (16 mm e 35 mm), tamanho do pixel (7 µm), tamanho do sensor (23,394 mm x 15,596 mm), altura de voo (300 m e 1080 m), escalas nominais (1:18.750 e 1:67.500), recobrimento longitudinal (70%) e recobrimento transversal (50%).

O modelo matemático para a orientação interior utilizado no E-Foto foi a transformação afim geral (TAG). A TAG modela seis parâmetros que consideram que o sistema inicial e o sistema de imagem digital podem apresentar características de não ortogonalidade dos eixos, rotação da imagem, translação em *x* e *y* e escalas diferentes em *x* e *y*. A forma linear da TAG é representada pelas Eqs. 6 e 7 (COELHO e BRITO, 2016).

As incógnitas são os valores de a_0, a_1, a_2, b_0, b_1 e b_2, que correspondem a parâmetros de transformação entre os dois sistemas. Os valores de *x* e *y* das marcas fiduciais advêm do certificado de calibração. Entretanto, nas imagens provenientes de câmeras não métricas, normalmente não são encontradas as marcas fiduciais. Nesses casos, basta que se trabalhe com as dimensões do sensor e o respectivo tamanho do seu pixel (VERMEER e AYEHU, 2017). Rearranjando as Eq. 6 e 7 em forma matricial e isolando o vetor que contém as incógnitas, chega-se à Eq. 8. Dessa forma, o modelo matemático utilizado no E-Foto é definido conforme as Eqs. 98 a 100:

$$X_a = (A^T \,.\, P \,.\, A)^{-1} \,.\, (A^T \,.\, P \,.\, L_B) \tag{98}$$

$$V = A \,.\, X_a - L_b \tag{99}$$

$$La = L_b + V \tag{100}$$

Onde X_a é o vetor dos parâmetros; A é a matriz dos coeficientes dos parâmetros; L_b é o vetor das observações; P é a matriz-peso das observações; L_a é

o vetor das observações ajustadas; e *V* é o vetor dos resíduos (VERMEER e AYEHU, 2017).

Por meio da ressecção espacial no aplicativo E-Foto e por intermédio das equações de colinearidade e ajustamento por mínimos quadrados do modelo paramétrico não-linear, podem-se determinar os seis elementos de orientação exterior de uma imagem, a partir de, no mínimo, três pontos de controle não-colineares (KIM et al., 2017). Para este trabalho, os seis parâmetros já foram fornecidos, o que facilitou a orientação exterior (Tabelas 7 e 8).

No contexto do E-Foto, o MDE consiste de um conjunto de pontos presentes em um modelo estereoscópico em que suas respectivas coordenadas 3D no referencial do espaço objeto (*X, Y* e *Z*) são calculadas automaticamente a partir de suas respectivas projeções no espaço imagem (LIZARAZO et al., 2017). Neste trabalho, foi utilizada a correlação de *Pearson* como método de correspondência (JINGXIONG e NA, 2008). O método dos mínimos quadrados apresentou acurácia menor que a correlação de *Pearson*, não sendo utilizado neste trabalho. Desta forma, os histogramas da acurácia no processamento dos MDEs apresentaram desempenhos para a área 1 de 0,9 (49%), 0,8 (26%), 0,7 (14%) e 0,6 (9%) (**Figura 46**) e, para a área 2, de 0,9 (42%), 0,8 (32%), 0,7 (13%) e 0,6 (11%) (**Figura 47**). Foi criada uma grade regular de uma nuvem de pontos, de forma automática, por meio do método de interpolação por média móvel, com 30 cm de resolução para cada MDE (SILVEIRA, 2012).

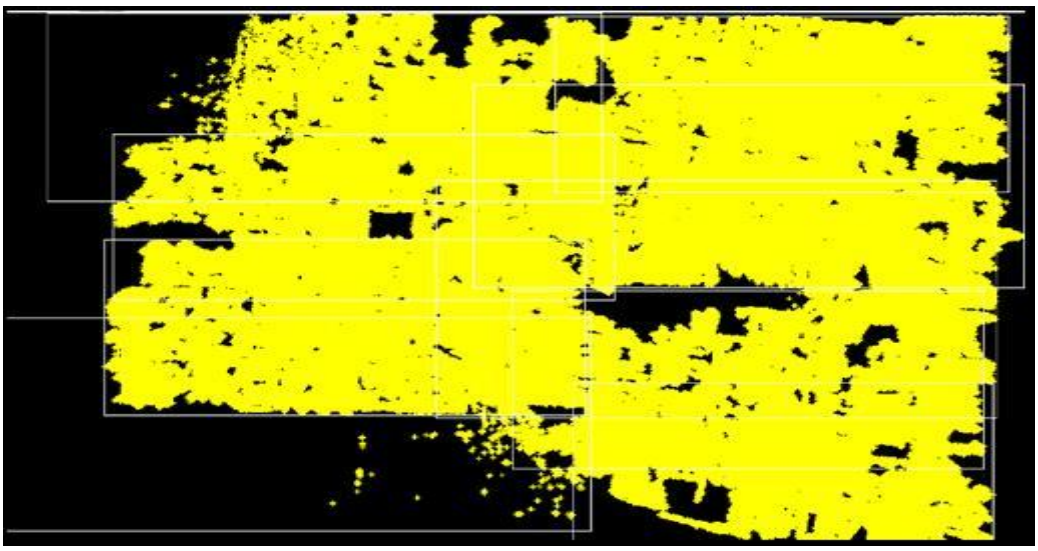


Figura 46. MDE da área 1 com 30 cm de resolução. Fonte: Elaborada pelo autor.

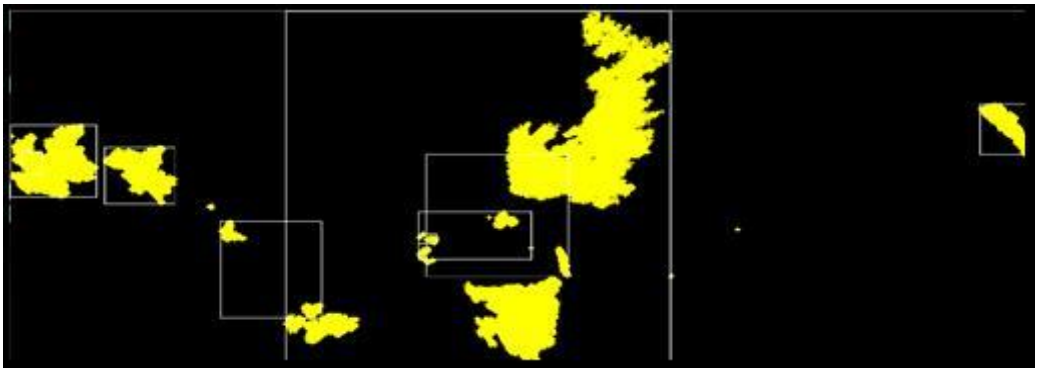


Figura 47. MDE da área 2 com 30 cm de resolução. Fonte: Elaborada pelo autor.

O aplicativo E-Foto usa o método de retificação diferencial para gerar um ortomosaico (COELHO e BRITO, 2015). Na execução da ortorretificação, foi utilizada a grade regular de uma nuvem de pontos. Foram gerados ortomosaicos com 50 cm de resolução, definindo-se o geotiff como o formato dos ortomosaicos e o Lagrange como o método de interpolação para as áreas 1 (**Figura 48**) e 2 (**Figura 49**). O E-Foto não conseguiu de maneira eficiente corrigir a distorção provocada pelo ângulo *yaw*.

Figura 48. Ortomosaico da subárea noroeste da área 1 com 50 cm de resolução. Fonte: Elaborada pelo autor.

Figura 49. Ortomosaico da área 2 com 50 cm de resolução. Fonte: Elaborada pelo autor.

A criação dos ortomosaicos no Agisoft Photoscan seguiu quatro fases. A primeira fase é o alinhamento da câmara. Procura-se por pontos em comum em imagens e combina-os, formando uma nuvem de pontos. A segunda fase é a construção da nuvem de pontos densa. Com base nas posições da câmera estimadas e das próprias imagens, uma nuvem de pontos densa é definida. A terceira fase é a construção de uma malha poligonal 3D que representa a superfície de objeto com base na nuvem de pontos densa. Na quarta fase, a malha pode ser texturizada (AGISOFT, 2019). Desta forma, na quarta fase temos a geração dos MDE, áreas 1 (**Figura 50**) e 2 (**Figura 51**), com 12 cm e 7 cm de resolução, respectivamente.

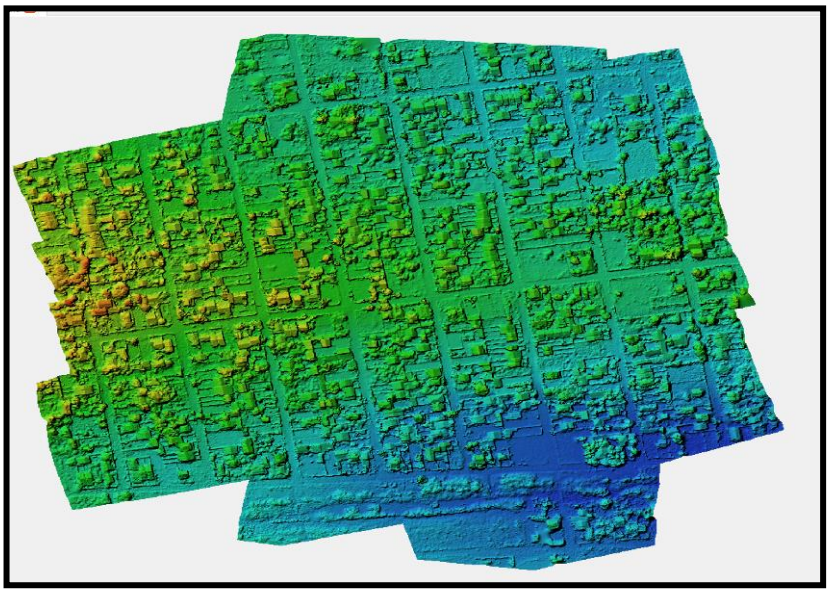


Figura 50. MDE da área 1 com 12 cm de resolução. Fonte: Elaborada pelo autor.

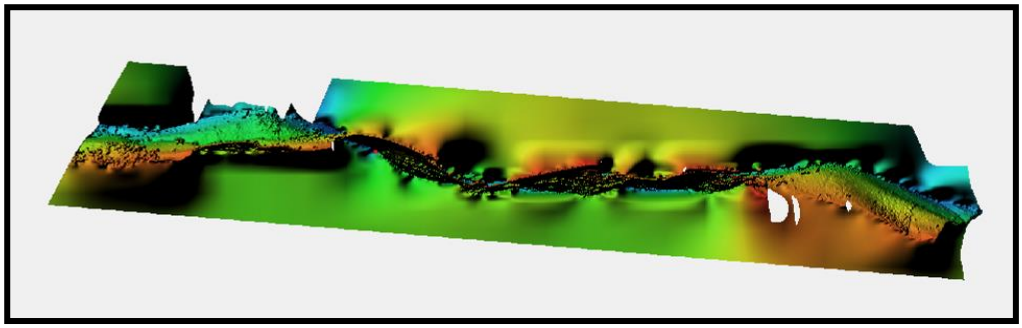


Figura 51. MDE da área 2 com 7 cm de resolução. Fonte: Elaborada pelo autor.

Por fim, os MDE são utilizados para a geração de ortomosaicos da área 1 (**Figura 52**) e 2 (**Figura 53**), com 12 cm e 7 cm de resolução, respectivamente. O Agisoft Photoscan apresentou menores distorções provocadas pelo ângulo *yaw*.

Figura 52. Ortomosaico da área 1 com 12 cm de resolução. Fonte: Elaborada pelo autor.

Figura 53. Ortomosaico da área 2 com 7 cm de resolução. Fonte: Elaborada pelo autor.

3.4. AVALIAÇÃO DOS ORTOMOSAICOS INICIAIS

Na quarta fase desta tese, para a planimetria da Área 1 (processamento no Agisoft PhotoScan), com 13 pontos de verificação (o aplicativo GeoPEC rejeitou 2 pontos) e considerando a junção do teste de tendência com o teste de precisão, as amostras apresentaram padrão disperso, distribuição normal, com resultado tendencioso. A precisão resultou em classe A (PEC) e classe B (PEC-PCD), escala 1:5.000 (Tabela 11).

Tabela 11. Dados estatísticos planimétricos da área 1 (processamento no Agisoft PhotoScan) do GeoPEC.

Estatística	Leste	Norte	Posicional
Média	-0,4720	0,8362	1,1246
Desvio-padrão	0,7408	0,8240	0,9255
Erro médio quadrático	0,8540	1,1515	1,4337
Máximo	0,5170	2,7410	3,3466
Mínimo	-1,9200	-0,1870	0,2731
Assimetria	-0,7130	1,0410	1,0880

Fonte: Fonte: Elaborada pelo autor.

Para a altimetria da Área 1 (processamento no Agisoft PhotoScan), com 15 pontos de verificação e considerando-se a junção do teste de tendência com o teste de precisão, as amostras apresentaram distribuição normal, com resultado tendencioso. A precisão resultou em classe C (PEC) e classe D (PEC-PCD), na equidistância de 70 metros de curva de nível (Tabela 12).

Tabela 12. Dados estatísticos altimétricos da área 1 (processamento no Agisoft PhotoScan) do GeoPEC.

Estatística	Altitude (m)
Média	31,1305
Desvio-padrão	3,5951
Erro médio quadrático	31,3236
Máximo	38,4520
Mínimo	26,5080
Assimetria	0,3846

Fonte: Fonte: Elaborada pelo autor.

Considerando a planimetria da Área 2 (processamento no Agisoft PhotoScan), com 8 pontos de verificação e considerando a junção do teste de tendência com o teste de precisão, as amostras apresentaram padrão disperso, distribuição normal, com resultado tendencioso. A precisão resultou em classe C (PEC) e classe D (PEC-PCD), escala 1:2.000 (Tabela 13).

Tabela 13. Dados estatísticos planimétricos da área 2 (processamento no Agisoft PhotoScan) do GeoPEC.

Estatística	Leste	Norte	Posicional
Média	-0,1338	0,8375	1,0016

Desvio-padrão	0,3915	0,6527	0,5049
Erro médio quadrático	0,3899	1,0364	1,1073
Máximo	0,4000	1,9600	1,9637
Mínimo	-0,7600	-0,1900	0,1910
Assimetria	-0,1700	0,0720	0,3190

Fonte: Elaborada pelo autor.

Para a altimetria da Área 2 (processamento no Agisoft PhotoScan), com 8 pontos de verificação e considerando-se a junção do teste de tendência com o teste de precisão, as amostras apresentaram distribuição normal, com resultado tendencioso. A precisão resultou em classe C (PEC) e classe D (PEC-PCD), na equidistância de 50 metros de curva de nível (Tabela 14).

Tabela 14. Dados estatísticos altimétricos da área 2 (processamento no Agisoft PhotoScan) do GeoPEC.

Estatística	Altitude (m)
Média	-14,9462
Desvio-padrão	15,8102
Erro médio quadrático	21,0263
Máximo	7,8700
Mínimo	-30,7300
Assimetria	0,4481

Fonte: Elaborada pelo autor.

Desta forma, para a planimetria da Área 1 (processamento no E-Foto), com 10 pontos de verificação (o aplicativo GeoPEC rejeitou 5 pontos) e considerando a junção do teste de tendência com o teste de precisão, as amostras apresentaram

padrão disperso, distribuição normal, com resultado tendencioso. A precisão resultou em classe C (PEC) e classe D (PEC-PCD), escala 1:50.000 (Tabela 15).

Tabela 15. Dados estatísticos planimétricos da área 1 (processamento no E-Foto) do GeoPEC.

Estatística	Leste	Norte	Posicional
Média	-0,9863	24,2913	26,1313
Desvio-padrão	7,3664	11,0173	8,7913
Erro médio quadrático	7,1847	26,5209	27,4769
Máximo	14,3880	37,3230	37,9489
Mínimo	-16,9300	-5,1840	5,8678
Assimetria	-0,2580	-1,147	-0,6380

Fonte: Elaborada pelo autor.

Para a altimetria da Área 1 (processamento no E-Foto), com 10 pontos de verificação (o aplicativo GeoPEC rejeitou 5 pontos) e considerando-se a junção do teste de tendência com o teste de precisão, as amostras apresentaram distribuição normal, com resultado tendencioso. A precisão resultou em classe C (PEC) e classe D (PEC-PCD), na equidistância de 25 metros de curva de nível (Tabela 16).

Tabela 16. Dados estatísticos altimétricos da área 1 (processamento no E-Foto) do GeoPEC.

Estatística	Altitude (m)
Média	-3,5261
Desvio-padrão	9,3424
Erro médio quadrático	9,5387

Máximo	7,9640
Mínimo	-17,3180
Assimetria	-0,2060

Fonte: Elaborada pelo autor.

Considerando a planimetria da Área 2 (processamento no E-Foto), com 8 pontos de verificação e considerando a junção do teste de tendência com o teste de precisão, as amostras apresentaram padrão disperso, distribuição normal, com resultado tendencioso. A precisão resultou em classe B (PEC) e classe C (PEC-PCD), escala 1:25.000 (Tabela 17).

Tabela 17. Dados estatísticos planimétricos da área 2 (processamento no E-Foto) do GeoPEC.

Estatística	Leste	Norte	Posicional
Média	-3,4888	-0,9550	5,8218
Desvio-padrão	6,3638	3,3032	5,2560
Erro médio quadrático	6,8998	3,2340	7,6201
Máximo	2,4300	3,3800	18,4129
Mínimo	-18,1000	-5,9500	2,6674
Assimetria	-1,3210	0,0630	1,6400

Fonte: Elaborada pelo autor.

Para a altimetria da Área 2 (processamento no E-Foto), com 8 pontos de verificação e considerando-se a junção do teste de tendência com o teste de precisão, as amostras apresentaram distribuição normal, com resultado tendencioso. A precisão resultou em classe C (PEC) e classe D (PEC-PCD), na equidistância de 60 metros de curva de nível (Tabela 18).

Tabela 18. Dados estatísticos altimétricos da área 2 (processamento no E-Foto) do GeoPEC.

Estatística	Altitude (m)
Média	5,8992
Desvio-padrão	23,4110
Erro médio quadrático	22,6797
Máximo	31,2780
Mínimo	-43,7120
Assimetria	-1,0265

Fonte: Elaborada pelo autor.

Em todas as análises feitas nas duas áreas, foram padronizados o nível de confiança de 90% para o teste de normalidade, nível de confiança de 90% para o teste *t* de *Student* (tendência) e nível de confiança de 90% para χ^2 (precisão)**.** A Tabela 19 representa um resumo dos erros médios quadráticos e classes PEC e PEC-PCD das Áreas 1 e 2 obtidas pelos aplicativos Agisoft Photoscan e E-Foto.

Tabela 19. Dados estatísticos (erro médio quadrático) das Áreas 1 e 2. PEC = Padrão de Exatidão Cartográfica; PCD = Produtos Cartográficos Digitais; Ep = Equidistância; e Es = Escala.

Tipos	Agisoft Photoscan	E-Foto	Areas	Classes PEC	Classes PEC-PCD	Es e Ep
Planimetria	1,4337 m	27,4769 m	1	A/C	B/D	1:5000/1:50000
Planimetria	1,1073 m	7,6201 m	2	C/B	D/C	1:2000/1:25000
Altimetria	31,3236 m	9,5387 m	1	C/C	D/D	70/25 m
Altimetria	21,0263 m	22,6797 m	2	C/C	D/D	50/60 m

Fonte: Elaborada pelo autor.

Conforme as Tabelas 11 a 19 apresentadas, verificaram-se melhores resultados nas planimetrias das áreas 1 e 2 no aplicativo Agisoft Photoscan. Entretanto, na altimetria da área 1, o aplicativo E-Foto apresentou melhor resultado. Na altimetria da área 2, os aplicativos E-Foto e Agisoft Photoscan apresentaram resultados próximos.

De forma resumida, na avaliação do PEC em ortomosaicos obtidos por RPA por meio dos aplicativos Agisoft Photoscan, E-Foto e GeoPEC (quarta fase desta tese), as amostras apresentaram padrões dispersos, distribuições normais, com resultados tendenciosos. Os melhores resultados nas planimetrias das áreas foram no aplicativo Agisoft Photoscan. Na altimetria da área 1, o aplicativo E-Foto apresentou melhor resultado. As precisões nas planimetrias e nas altimetrias podem ser aperfeiçoadas, obtendo-se mosaicos de escalas maiores com classe A por meio da aplicação da correção geométrica no ângulo *yaw*, presentes nas imagens distorcidas, a fim de reduzir a guinada e facilitar a ortorretificação. O modelo de transformação projetiva modificada proposto pode ser aplicado com algoritmos na linguagem de programação Python.

A primeira parte da quinta fase deste trabalho foi apresentado na fundamentação teórica: criação matemática do método paramétrico de transformação projetiva com derivação dos termos da matriz de rotação R_{ψ}. O capítulo seguinte representa a segunda parte da quinta fase desta tese: criação de algoritmos em linguagem de programação *Python*, capazes de minimizar erros geométricos.

4 ALGORITMO DE CORREÇÃO GEOMÉTRICA NO ÂNGULO DE GUINADA

Este capítulo representa a segunda parte da quinta fase desta tese em que foi aplicado o modelo matemático de transformação projetiva modificada para o ângulo de guinada. Este trabalho propõe, no final do Capítulo 3, um modelo matemático de transformação projetiva modificada completo. Entretanto, a fim de simplificar o processo, este capítulo propõe a aplicação da modificação de sua matriz de rotação R_{ψ}, por meio de um algoritmo específico na linguagem de programação Python.

A proposta é a aplicação deste algoritmo na fase de pré-processamento, reduzindo ou eliminando os desvios maiores que 5°, permitindo a redução do efeito leque nas faixas de imageamento no aerolevantamento óptico com RPAs. Após o pré-processamento, as imagens devem ser processadas em aplicativos específicos para geração de ortomosaicos, como o Agisoft Photoscan.

Concernente a demanda computacional, para a aplicação da correção nas imagens com o Python no pré-processamento, o E-Foto ou Agisoft Photoscan no processamento e o GeoPEC na avaliação da acurácia posicional, foi utilizado um computador Dell, processador Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz, memória 16 GB, sistema operacional de 64 bits, Placa de Vídeo integrada NVIDIA GeForce GTX 1060 6 GB, SSD 256 GB, HDD de 2 TB, dois monitores Led Samsung e AOC de 21,5” e sistema operacional Windowns 10 Home Single Language, com pacote Microsoft Office Professional Plus 2019, todos originais. Neste contexto, o tempo médio de processamento para geração dos ortomosáicos foi de uma hora diária.

Para trabalhos futuros, pode-se criar um algoritmo contendo todo o modelo matemático de transformação projetiva modificada proposto, com as três matrizes de rotação.

4.1 EXPERIMENTO INICIAL COM ALGORITMO NO EXCEL

Inicialmente, antes da criação do algoritmo específico na linguagem de programação Python, foram realizados testes em imagens formadas por coordenadas cartesianas no plano cartesiano com um algoritmo criado no aplicativo

Excel, contendo as matrizes de rotação modificadas $Rd11_{\psi}$ e $Rd21_{\psi}$, primeiro quadrante do plano cartesiano (Eqs. 117 e 118), respectivamente.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} cos\psi & sen\psi & 0 \\ -cos\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (117)$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} -cos\psi & -sen\psi & 0 \\ sen\psi & sen\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (118)$$

Onde: *x', y'* e *z'* são as coordenadas do ponto corrigido; *x, y* e *z* são as coordenadas do ponto não corrigido e as matrizes 3x3 são as matrizes de rotação modificadas $Rd11_{\psi}$ e $Rd21_{\psi}$, respectivamente, no primeiro quadrante do plano cartesiano.

A imagem RPA teste com ângulo de guinada 0° no aplicativo Excel é formada por 18 coordenadas cartesianas representadas no plano cartesiano no primeiro quadrante (**Figura 54**), gerada por uma matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 0° (Tabela 20).

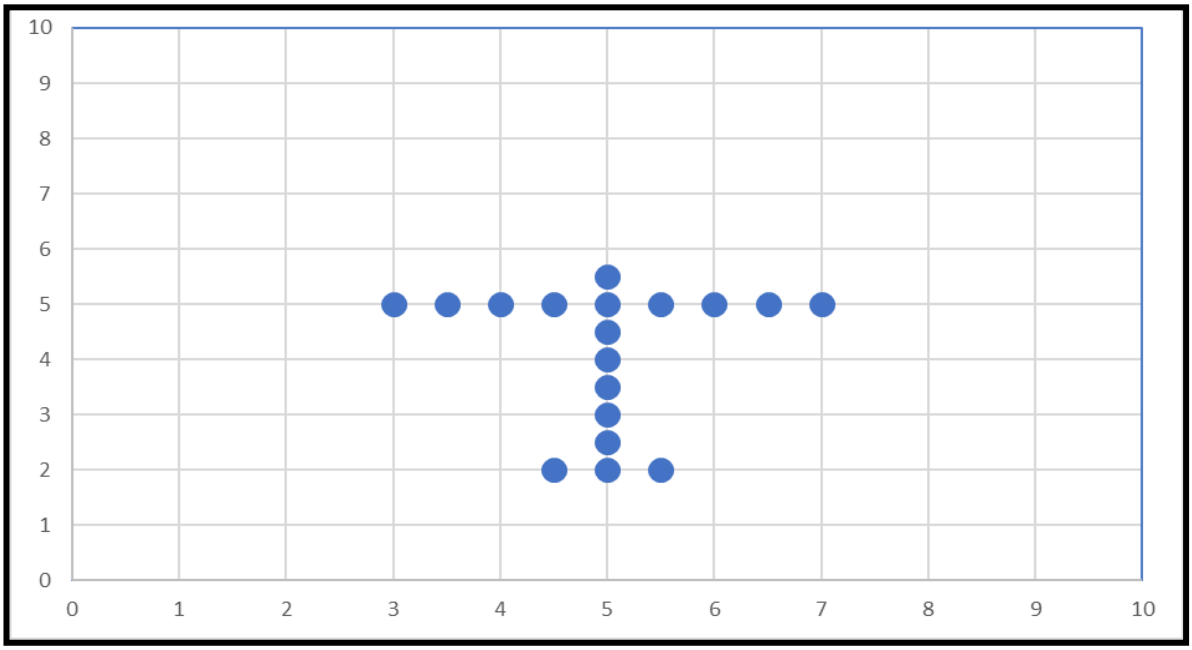


Figura 54. Imagem RPA teste com ângulo de guinada 0° no aplicativo Excel.

Fonte: Elaborada pelo autor.

Tabela 20. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 0°.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	5	2	0
B	5	3	0
C	5	4	0
D	5	5	0
E	4,5	2	0
F	5	2,5	0
G	5	3,5	0
H	5	4,5	0
I	5	5,5	0
J	4	5	0
K	6	5	0
L	4,5	5	0
M	5,5	5	0
N	5,5	2	0
O	3	5	0

P	7	5	0
Q	3,5	5	0
R	6,5	5	0

Fonte: Elaborada pelo autor.

Considerando os valores do ângulo de guinada das Tabelas 2 a 8, os testes foram realizados com ângulos que estão no primeiro, no segundo e no terceiro quadrantes do plano cartesiano. Desta forma, o primeiro teste foi com ângulo de distorção de 13,19° no primeiro quadrante. A **Figura 55** representa a imagem RPA teste com ângulo de guinada 13,19° no aplicativo Excel e a Tabela 21 apresenta sua matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 13,19°.

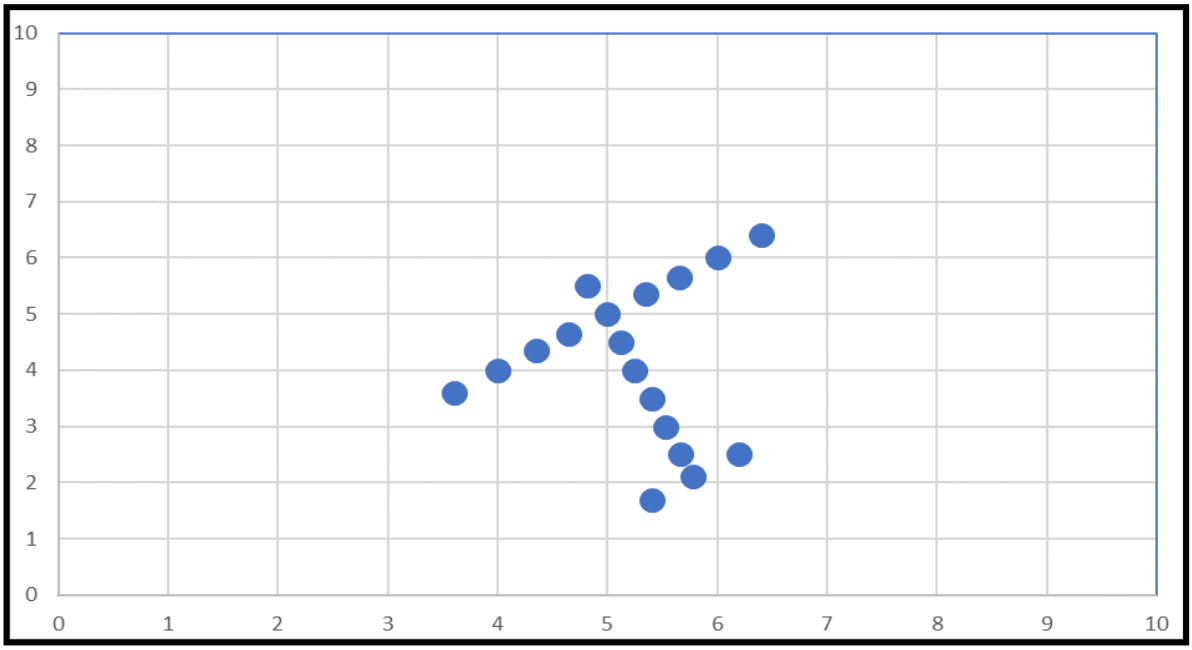


Figura 55. Imagem RPA teste com ângulo de guinada 13,19° no aplicativo Excel.

Fonte: Elaborada pelo autor.

Tabela 21. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 13,19°.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	5	5	0
B	3,6	3,6	0
C	6,4	6,4	0

D	4,35	4,35	0
E	5,35	5,35	0
F	4	4	0
G	6	6	0
H	4,65	4,65	0
I	5,65	5,65	0
J	5,78	2,1	0
K	5,67	2,5	0
L	5,53	3	0
M	5,4	3,5	0
N	5,25	4	0
O	5,12	4,5	0
P	4,82	5,5	0
Q	6,2	2,5	0
R	5,4	1,7	0

Fonte: Elaborada pelo autor.

Após a multiplicação da matriz *Rd11*$_{\psi}$ com ângulo de guinada de 13,19° (Tabela 22), com a matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 13,19° (Tabela 21), é gerada a imagem RPA teste *Rd11*$_{\psi}$ (**Figura 56**) e a matriz de coordenadas (x, y, z) 3x18 *Rd11*$_{\psi}$ (Tabela 23), corrigidas conforme Eq. 117. O resultado foi correção satisfatória para o primeiro quadrante.

Tabela 22. Matriz *Rd11*$_{\psi}$ com ângulo de guinada de 13,19°.

Linha/Coluna	1	2	3
1	=COS(RADIANOS(13,19))	=SEN(RADIANOS(13,19))	0
2	=-COS(RADIANOS(13,19))	=COS(RADIANOS(13,19))	0
3	0	0	1

Fonte: Elaborada pelo autor.

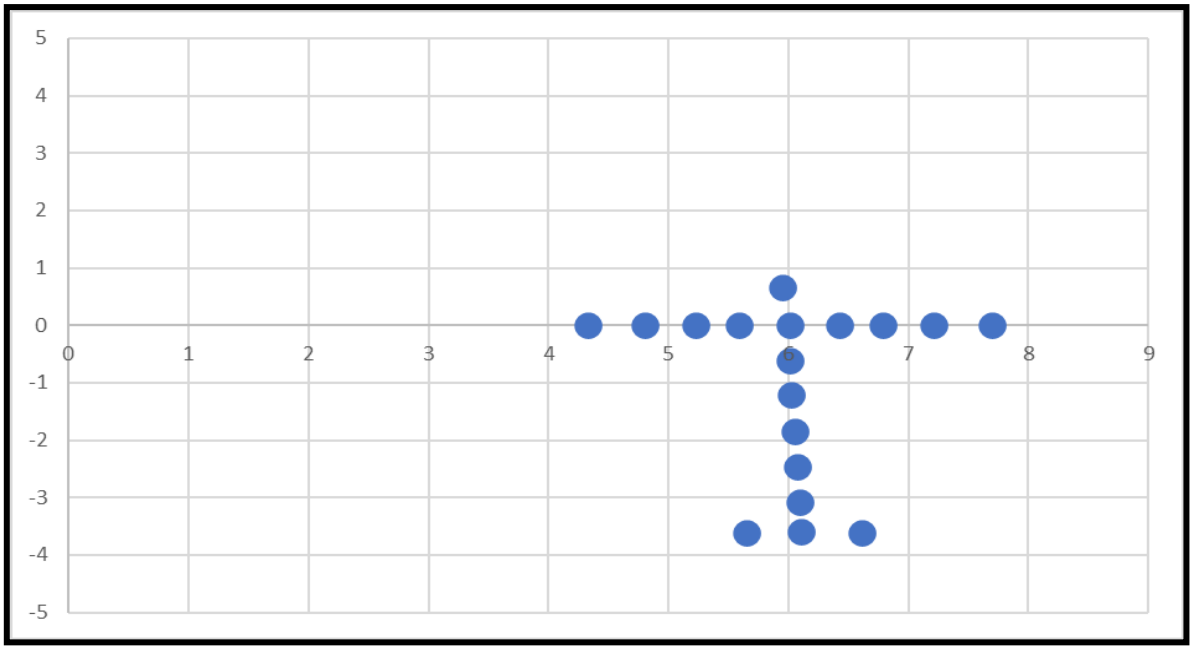


Figura 56. Imagem RPA teste *Rd11*$_{\psi}$ com ângulo de guinada 13,19° corrigida no aplicativo Excel. Fonte: Elaborada pelo autor.

Tabela 23. Matriz de coordenadas (x, y, z) 3x18 *Rd11*$_{\psi}$ para o ângulo de guinada 13,19° corrigida.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	6,008998	0	0
B	4,326479	0	0
C	7,691518	0	0
D	5,227829	0	0
E	6,429628	0	0
F	4,807199	0	0
G	7,210798	0	0
H	5,588369	0	0
I	6,790168	0	0
J	6,106696	-3,58292	0
K	6,090871	-3,08637	0
L	6,068654	-2,46326	0
M	6,056175	-1,84988	0
N	6,024222	-1,21702	0

O	6,011742	-0,60364	0
P	5,947838	0,662061	0
Q	6,606889	-3,60239	0
R	5,645449	-3,60239	0

Fonte: Elaborada pelo autor.

Após a multiplicação da matriz *Rd21*$_{\psi}$ com ângulo de guinada de 13,19° (Tabela 24), com a matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 13,19° (Tabela 21), é gerada a imagem RPA teste *Rd21*$_{\psi}$ (**Figura 57**) e a matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ (Tabela 25), corrigidas conforme Eq 118. O resultado se aproximou da correção satisfatória para o primeiro quadrante.

Tabela 24. Matriz *Rd21*$_{\psi}$ com ângulo de guinada de 13,19°.

Linha/Coluna	1	2	3
1	=-COS(RADIANOS(13,19))	=-SEN(RADIANOS(13,19))	0
2	=SEN(RADIANOS(13,19))	=SEN(RADIANOS(13,19))	0
3	0	0	1

Fonte: Elaborada pelo autor.

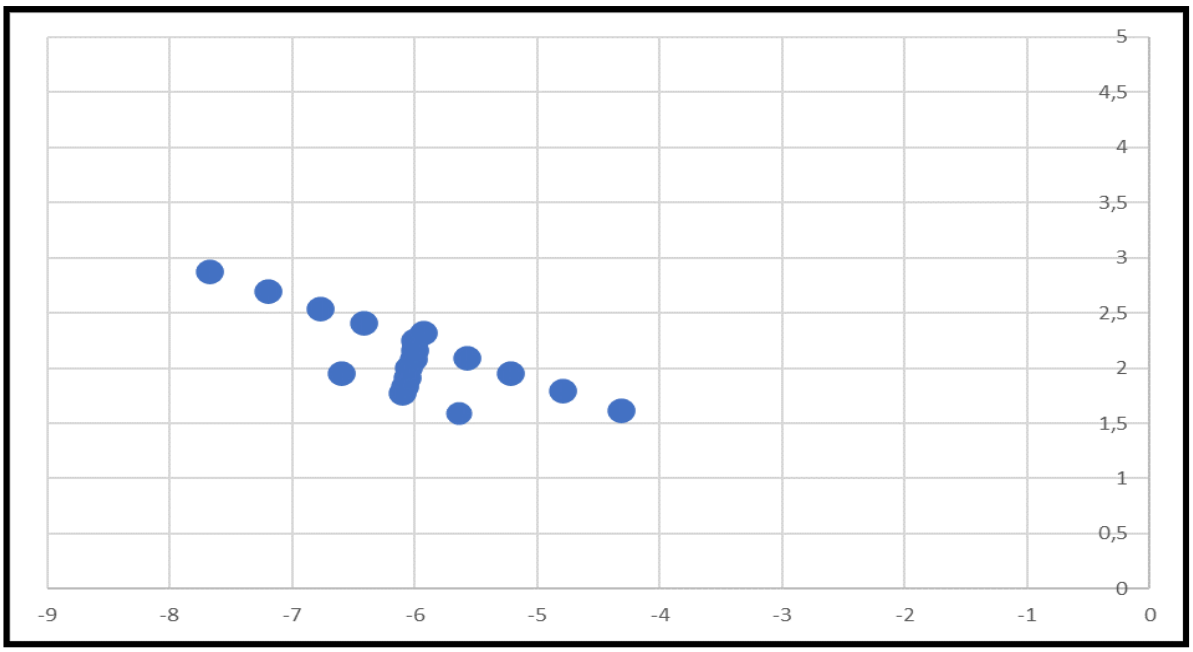


Figura 57. Imagem RPA teste *Rd21*$_{\psi}$ com ângulo de guinada 13,19° corrigida no aplicativo Excel. Fonte: Elaborada pelo autor.

Tabela 25. Matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ para o ângulo de guinada 13,19° corrigida.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	-6,009	2,281809	0
B	-4,32648	1,642903	0
C	-7,69152	2,920716	0
D	-5,22783	1,985174	0
E	-6,42963	2,441536	0
F	-4,8072	1,825448	0
G	-7,2108	2,738171	0
H	-5,58837	2,122083	0
I	-6,79017	2,578445	0
J	-6,1067	1,798066	0
K	-6,09087	1,864238	0
L	-6,06865	1,946383	0
M	-6,05617	2,03081	0
N	-6,02422	2,110674	0
O	-6,01174	2,195101	0
P	-5,94784	2,354827	0
Q	-6,60689	1,985174	0
R	-5,64545	1,620085	0

Fonte: Elaborada pelo autor.

O segundo teste foi com ângulo de distorção de 56,75° no primeiro quadrante. A **Figura 58** representa a imagem RPA teste com ângulo de guinada 56,75° no aplicativo Excel e a Tabela 26 apresenta sua matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 56,75°.

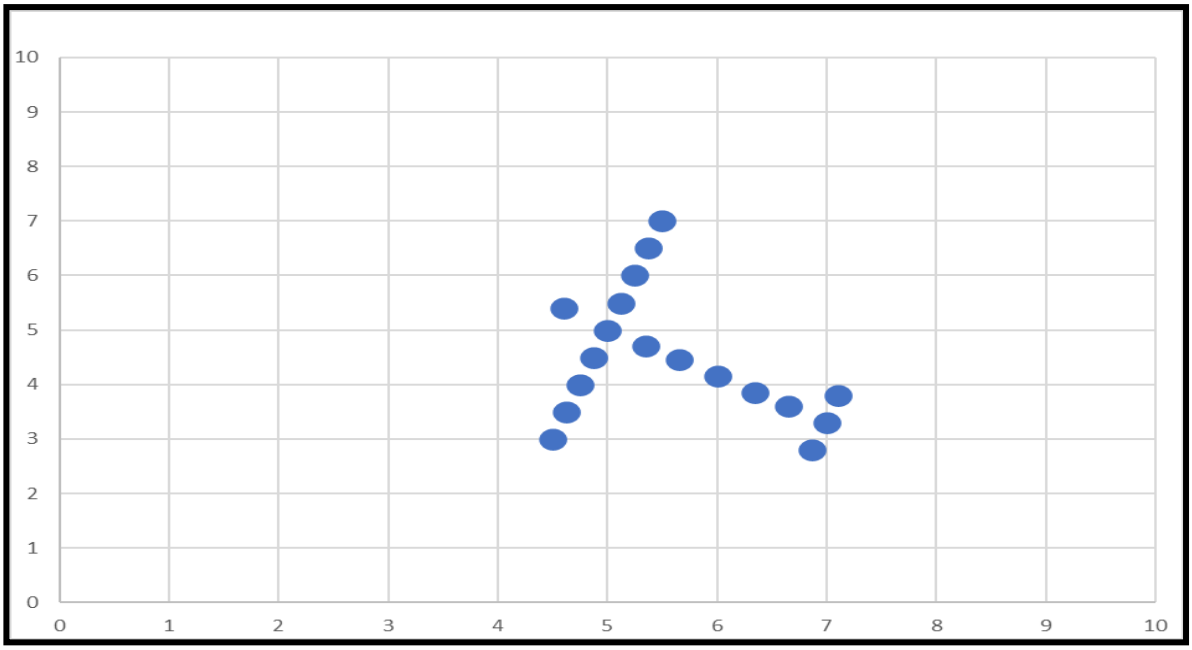


Figura 58. Imagem RPA teste com ângulo de guinada 56,75° no aplicativo Excel.

Fonte: Elaborada pelo autor.

Tabela 26. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 56,75°.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	5	5	0
B	4,5	3	0
C	6	4,15	0
D	7	3,3	0
E	5,35	4,7	0
F	5,65	4,45	0
G	6,35	3,85	0
H	6,65	3,6	0
I	4,75	4	0
J	4,6	5,4	0
K	4,62	3,5	0
L	4,87	4,5	0
M	5,25	6	0
N	5,12	5,5	0
O	5,37	6,5	0

P	5,5	7	0
Q	6,87	2,8	0
R	7,1	3,8	0

Fonte: Elaborada pelo autor.

Após a multiplicação da matriz $Rd11_{\psi}$ com ângulo de guinada de 56,75° (Tabela 27), com a matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 56,75° (Tabela 26), é gerada a imagem RPA teste $Rd11_{\psi}$ (**Figura 59**) e a matriz de coordenadas (x, y, z) 3x18 $Rd11_{\psi}$ (Tabela 28), corrigidas conforme Eq 117. O resultado se aproximou da correção satisfatória para o primeiro quadrante.

Tabela 27. Matriz $Rd11_{\psi}$ com ângulo de guinada de 56,75°.

Linha/Coluna	1	2	3
1	=COS(RADIANOS(56,75))	=SEN(RADIANOS(56,75))	0
2	=-COS(RADIANOS(56,75))	=COS(RADIANOS(56,75))	0
3	0	0	1

Fonte: Elaborada pelos autores.

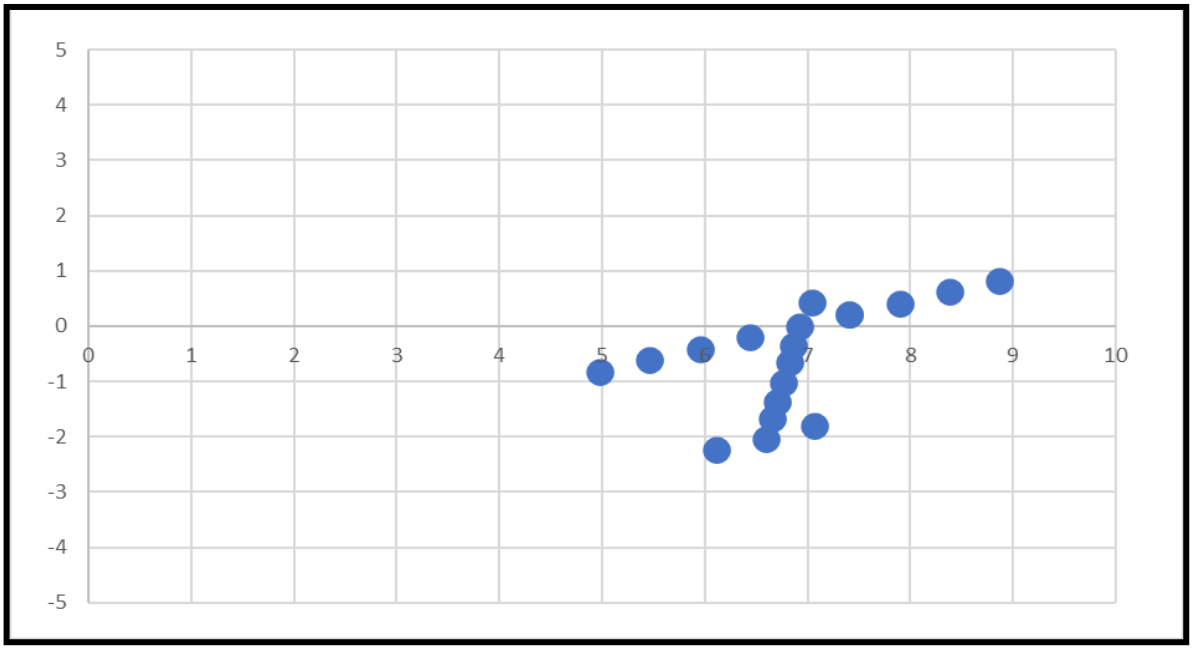


Figura 59. Imagem RPA teste $Rd11_{\psi}$ com ângulo de guinada 56,75° corrigida no aplicativo Excel. Fonte: Elaborada pelo autor.

Tabela 28. Matriz de coordenadas (x, y, z) 3x18 *Rd11*$_{\psi}$ para o ângulo de guinada 56,75° corrigida.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	6,922897	0	0
B	4,976178	-0,82244	0
C	6,760347	-1,01434	0
D	6,597797	-2,02868	0
E	6,863914	-0,35639	0
F	6,81933	-0,65795	0
G	6,701364	-1,37073	0
H	6,65678	-1,67229	0
I	5,949537	-0,41122	0
J	7,038094	0,438635	0
K	5,460116	-0,61409	0
L	6,433476	-0,20287	0
M	7,896256	0,41122	0
N	7,406835	0,208351	0
O	8,380195	0,619571	0
P	8,869616	0,82244	0
Q	6,108376	6,108376	0
R	7,070769	-1,80937	0

Fonte: Elaborada pelo autor.

Após a multiplicação da matriz *Rd21*$_{\psi}$ com ângulo de guinada de 56,75° (Tabela 29), com a matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 56,75° (Tabela 26), é gerada a imagem RPA teste *Rd21*$_{\psi}$ (**Figura 60**) e a matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ (Tabela 30), corrigidas conforme Eq 118. O resultado da correção não foi satisfatório para o primeiro quadrante.

Tabela 29. Matriz *Rd21*$_{\psi}$ com ângulo de guinada de 56,75°.

Linha/Coluna	1	2	3
1	=-COS(RADIANOS(56,75))	=-SEN(RADIANOS(56,75))	0

2	=SEN(RADIANOS(56,75))	=SEN(RADIANOS(56,75))	0
3	0	0	1

Fonte: Elaborada pelos autores.

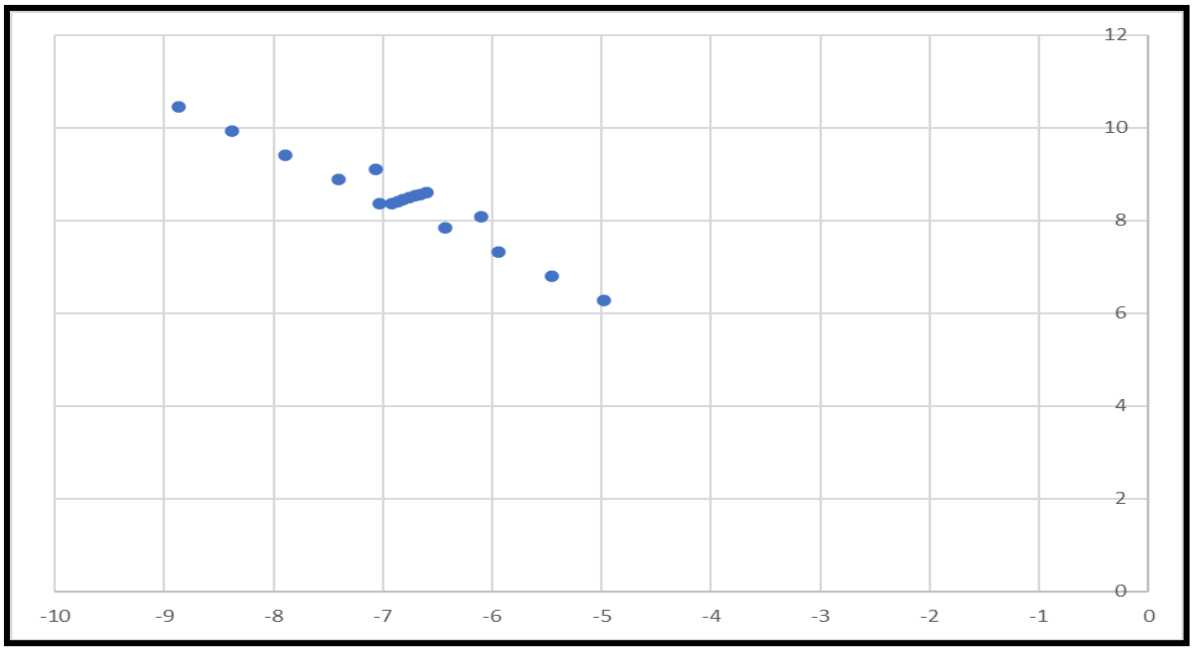


Figura 60. Imagem RPA teste *Rd21*$_{\psi}$ com ângulo de guinada 56,75° corrigida no aplicativo Excel. Fonte: Elaborada pelo autor.

Tabela 30. Matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ para o ângulo de guinada 56,75° corrigida.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	-6,9229	8,362862	0
B	-4,97618	6,272146	0
C	-6,76035	8,488304	0
D	-6,5978	8,613747	0
E	-6,86391	8,404676	0
F	-6,81933	8,44649	0
G	-6,70136	8,530119	0
H	-6,65678	8,571933	0
I	-5,94954	7,317504	0
J	-7,03809	8,362862	0

K	-5,46012	6,790644	0
L	-6,43348	7,836001	0
M	-7,89626	9,408219	0
N	-7,40684	8,881359	0
O	-8,38019	9,926717	0
P	-8,86962	10,45358	0
Q	-6,10838	-6,10838	0
R	-7,07077	9,115519	0

Fonte: Elaborada pelo autor.

O terceiro teste foi com ângulo de distorção de 90° no primeiro quadrante. A **Figura 61** representa a imagem RPA teste com ângulo de guinada 90° no aplicativo Excel e a Tabela 31 apresenta sua matriz coordenadas (x, y, z) 3x18 para o ângulo de guinada 90°.

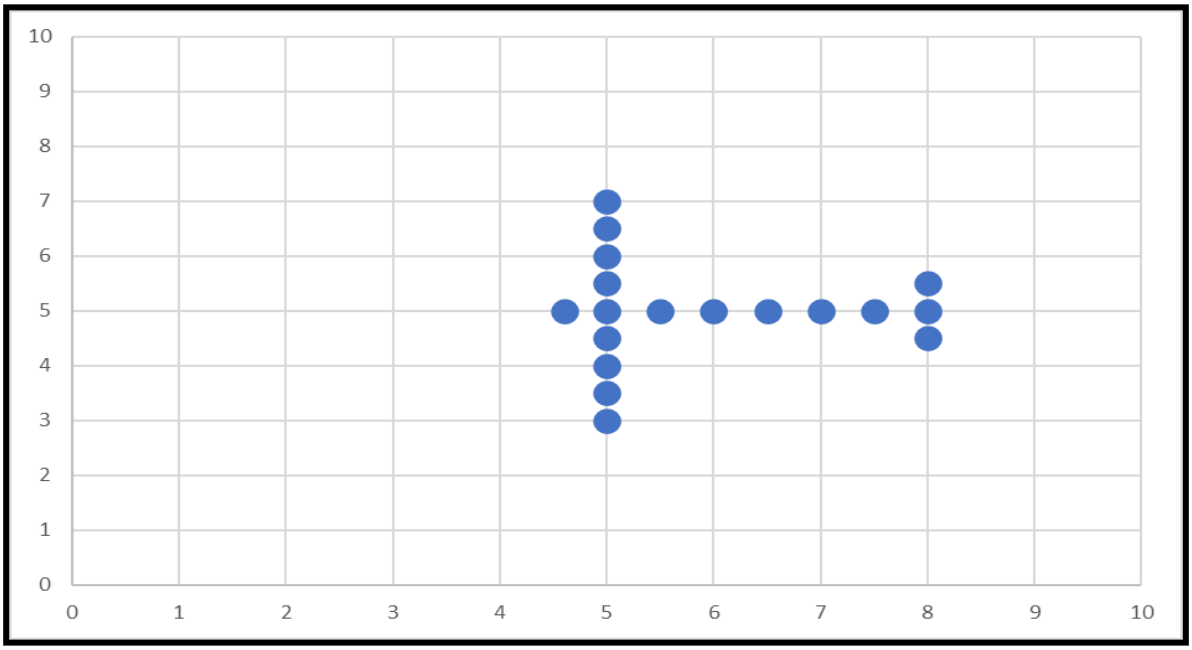


Figura 61. Imagem RPA teste com ângulo de guinada 90° no aplicativo Excel.

Fonte: Elaborada pelo autor.

Tabela 31. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 90°.

Ponto	Coordenada x	Coordenada y	Coordenada z

A	5	5	0
B	5	3	0
C	6	5	0
D	8	5	0
E	5,5	5	0
F	6,5	5	0
G	7	5	0
H	7,5	5	0
I	5	4	0
J	4,6	5	0
K	5	3,5	0
L	5	4,5	0
M	5	6	0
N	5	5,5	0
O	5	6,5	0
P	5	7	0
Q	8	4,5	0
R	8	5,5	0

Fonte: Elaborada pelo autor.

Após a multiplicação da matriz $Rd11_{\psi}$ com ângulo de guinada de 90° (Tabela 32), com a matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 90° (Tabela 31), é gerada a imagem RPA teste $Rd11_{\psi}$ (**Figura 62**) e a matriz de coordenadas (x, y, z) 3x18 $Rd11_{\psi}$ (Tabela 33), corrigidas conforme Eq 117. O resultado se aproximou da correção satisfatória para o primeiro quadrante.

Tabela 32. Matriz $Rd11_{\psi}$ com ângulo de guinada de 90°.

Linha/Coluna	1	2	3
1	=COS(RADIANOS(90))	=SEN(RADIANOS(90))	0
2	=-COS(RADIANOS(90))	=COS(RADIANOS(90))	0
3	0	0	1

Fonte: Elaborada pelo autor.

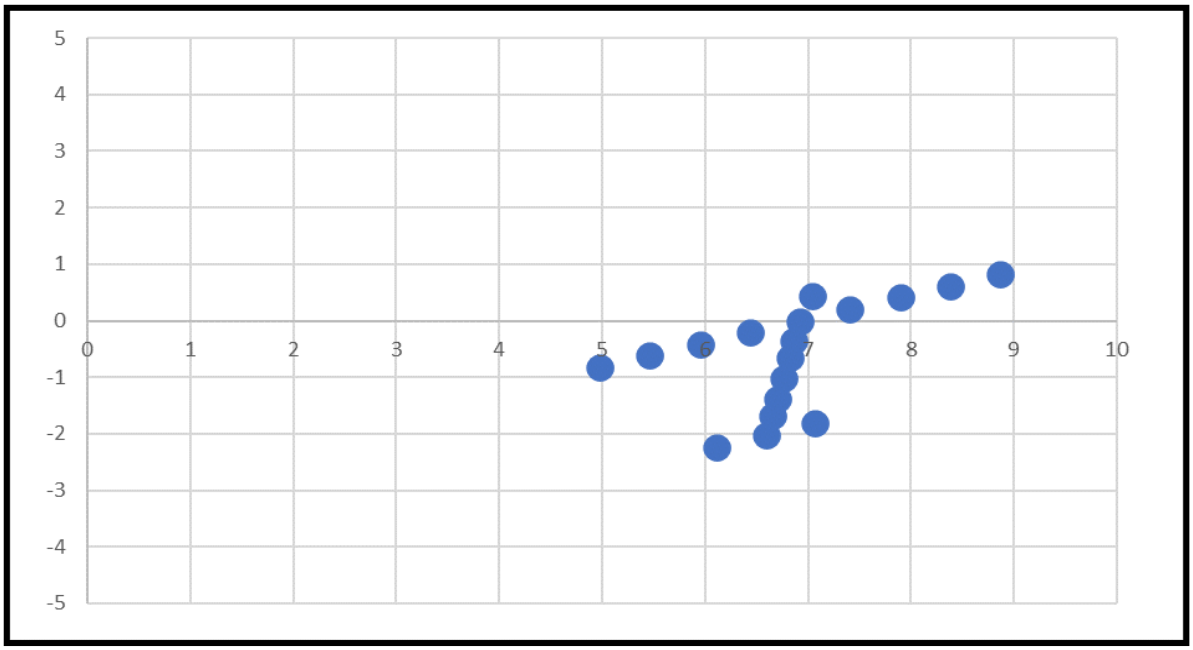


Figura 62. Imagem RPA teste *Rd11℣* com ângulo de guinada 90° corrigida no aplicativo Excel. Fonte: Elaborada pelo autor.

Tabela 33. Matriz de coordenadas (x, y, z) 3x18 *Rd11℣* para o ângulo de guinada 90° corrigida.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	5	0	0
B	3	-1,2E-16	0
C	5	-6,1E-17	0
D	5	-1,8E-16	0
E	5	-3,1E-17	0
F	5	-9,2E-17	0
G	5	-1,2E-16	0
H	5	-1,5E-16	0
I	4	-6,1E-17	0
J	5	2,45E-17	0
K	3,5	-9,2E-17	0
L	4,5	-3,1E-17	0
M	6	6,13E-17	0

N	5,5	3,06E-17	0
O	6,5	9,19E-17	0
P	7	1,23E-16	0
Q	4,5	-2,1E-16	0
R	5,5	-1,5E-16	0

Fonte: Elaborada pelo autor.

Após a multiplicação da matriz *Rd21*$_{\psi}$ com ângulo de guinada de 90° (Tabela 34), com a matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 90° (Tabela 31), é gerada a imagem RPA teste *Rd21*$_{\psi}$ (**Figura 63**) e a matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ (Tabela 35), corrigidas conforme Eq 118. O resultado da correção não foi satisfatório para o primeiro quadrante.

Tabela 34. Matriz *Rd21*$_{\psi}$ com ângulo de guinada de 90°.

Linha/Coluna	1	2	3
1	=-COS(RADIANOS(90))	=-SEN(RADIANOS(90))	0
2	=SEN(RADIANOS(90))	=SEN(RADIANOS(90))	0
3	0	0	1

Fonte: Elaborada pelo autor.

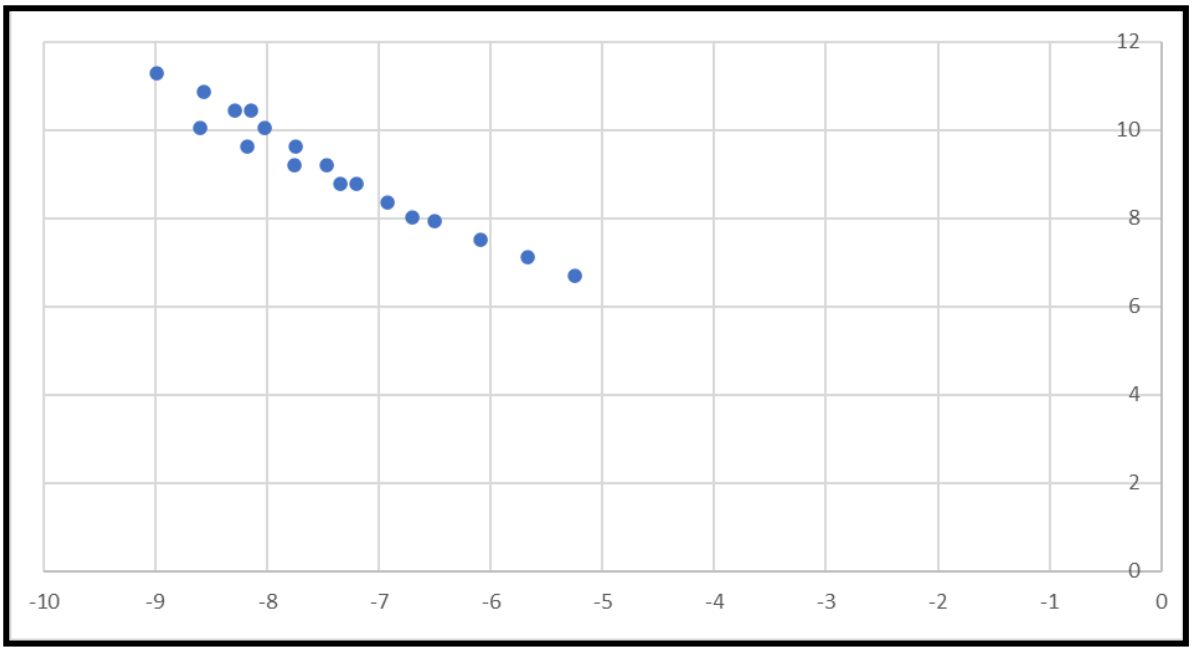


Figura 63. Imagem RPA teste *Rd21*$_{\psi}$ com ângulo de guinada 90° corrigida no aplicativo Excel. Fonte: Elaborada pelo autor.

Tabela 35. Matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ para o ângulo de guinada 90° corrigida.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	-6,9229	8,362862	0
B	-5,25032	6,690289	0
C	-7,47119	9,199148	0
D	-8,56778	10,87172	0
E	-7,19704	8,781005	0
F	-7,74534	9,617291	0
G	-8,01948	10,03543	0
H	-8,29363	10,45358	0
I	-6,08661	7,526575	0
J	-6,70358	8,028347	0
K	-5,66847	7,108432	0
L	-6,50475	7,944718	0
M	-7,75918	9,199148	0
N	-7,34104	8,781005	0

O	-8,17733	9,617291	0
P	-8,59547	10,03543	0
Q	-8,14963	10,45358	0
R	-8,98592	11,28986	0

Fonte: Elaborada pelo autor.

O quarto teste foi com ângulo de distorção de 168,70° no segundo quadrante. A **Figura 64** representa a imagem RPA teste com ângulo de guinada 168,70° no aplicativo Excel e a Tabela 36 apresenta sua matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 168,70°.

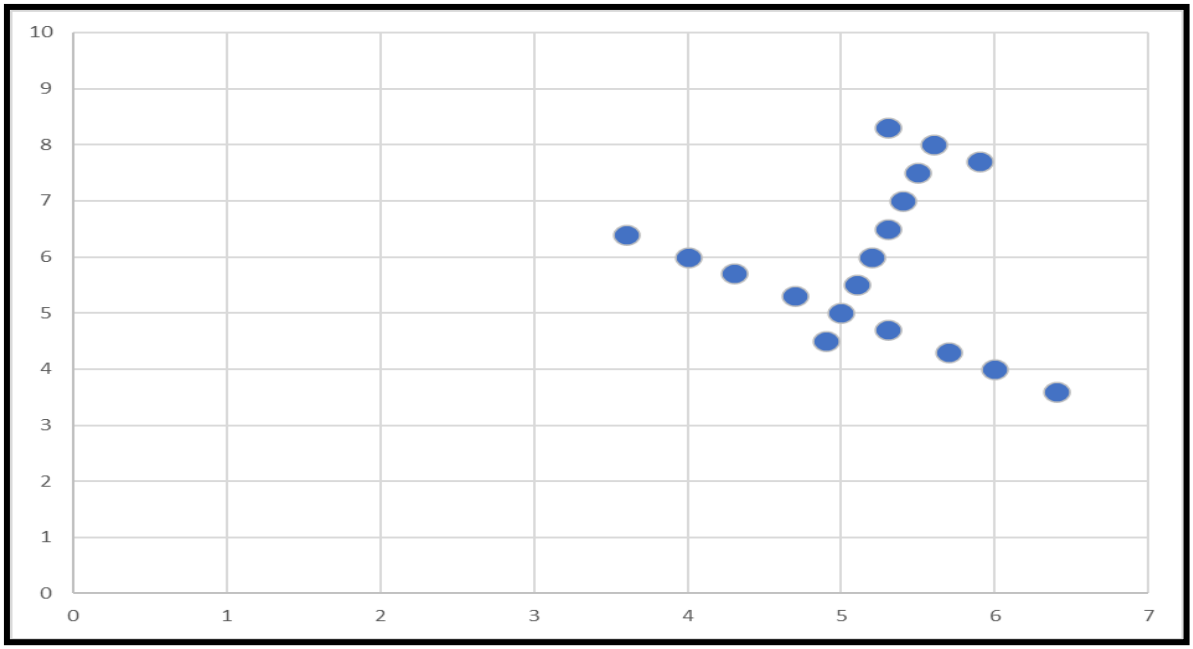


Figura 64. Imagem RPA teste com ângulo de guinada 168,70° no aplicativo Excel.

Fonte: Elaborada pelo autor.

Tabela 36. Matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 168,70°.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	5	5	0
B	5,3	8,3	0
C	5,5	7,5	0
D	3,6	6,4	0

E	5,9	7,7	0
F	5,3	4,7	0
G	6,4	3,6	0
H	4	6	0
I	5,6	8	0
J	4,3	5,7	0
K	4,7	5,3	0
L	4,9	4,5	0
M	5,2	6	0
N	5,1	5,5	0
O	5,3	6,5	0
P	5,4	7	0
Q	5,7	4,3	0
R	6	4	0

Fonte: Elaborada pelo autor.

Após a multiplicação da matriz *Rd11*$_{\psi}$ com ângulo de guinada de 168,70° (Tabela 37), com a matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 168,70° (Tabela 36), é gerada a imagem RPA teste *Rd11*$_{\psi}$ (**Figura 65**) e a matriz de coordenadas (x, y, z) 3x18 *Rd11*$_{\psi}$ (Tabela 38), corrigidas conforme Eq 117. Neste teste não foi realizada mudança do segundo para o primeiro quadrante, pois os resultados da correção geométrica no segundo ou no primeiro quadrante são idênticos. O resultado foi correção satisfatória para o segundo quadrante.

Tabela 37. Matriz *Rd11*$_{\psi}$ com ângulo de guinada de 168,70°.

Linha/Coluna	1	2	3
1	=COS(RADIANOS(168,70))	=SEN(RADIANOS(168,70))	0
2	=-COS(RADIANOS(168,70))	=COS(RADIANOS(168,70))	0
3	0	0	1

Fonte: Elaborada pelos autores

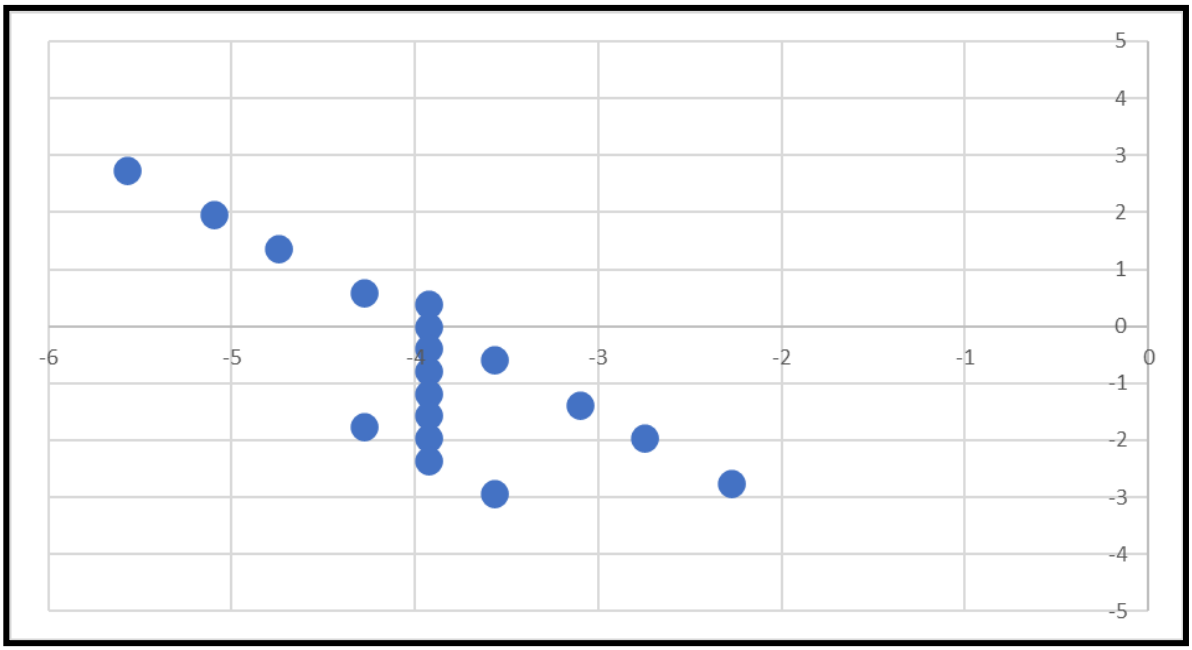


Figura 65. Imagem RPA teste *Rd11*$_{\psi}$ com ângulo de guinada 168,70° corrigida no aplicativo Excel. Fonte: Elaborada pelo autor.

Tabela 38. Matriz de coordenadas (x, y, z) 3x18 *Rd11*$_{\psi}$ para o ângulo de guinada 168,70° corrigida.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	-3,92334	0	0
B	-3,5709	-2,94184	0
C	-3,92378	-1,96123	0
D	-2,27616	-2,74572	0
E	-4,27684	-1,76511	0
F	-4,27631	0,588369	0
G	-5,57053	2,745721	0
H	-2,74678	-1,96123	0
I	-3,92387	-2,35348	0
J	-3,09975	-1,37286	0
K	-3,57037	-0,58837	0
L	-3,92325	0,392246	0
M	-3,92352	-0,78449	0
N	-3,92343	-0,39225	0
O	-3,92361	-1,17674	0

P	-3,9237	-1,56898	0
Q	-4,74694	1,372861	0
R	-5,0999	1,961229	0

Fonte: Elaborada pelo autor.

Após a multiplicação da matriz *Rd21*$_{\psi}$ com ângulo de guinada de 168,70° (Tabela 39), com a matriz de coordenadas (x, y, z) 3x18 para o ângulo de guinada 168,70° (Tabela 36), é gerada a imagem RPA teste *Rd21*$_{\psi}$ (**Figura 66**) e a matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ (Tabela 40), corrigidas conforme Eq. 118. Neste teste não foi realizada mudança do segundo para o primeiro quadrante, pois os resultados da correção geométrica no segundo ou no primeiro quadrante são idênticos. O resultado se aproximou da correção satisfatória para o segundo quadrante.

Tabela 39. Matriz *Rd21*$_{\psi}$ com ângulo de guinada de 168,70°.

Linha/Coluna	1	2	3
1	=-COS(RADIANOS(168,70))	=-SEN(RADIANOS(168,70))	0
2	=SEN(RADIANOS(168,70))	=SEN(RADIANOS(168,70))	0
3	0	0	1

Fonte: Elaborada pelo autor.

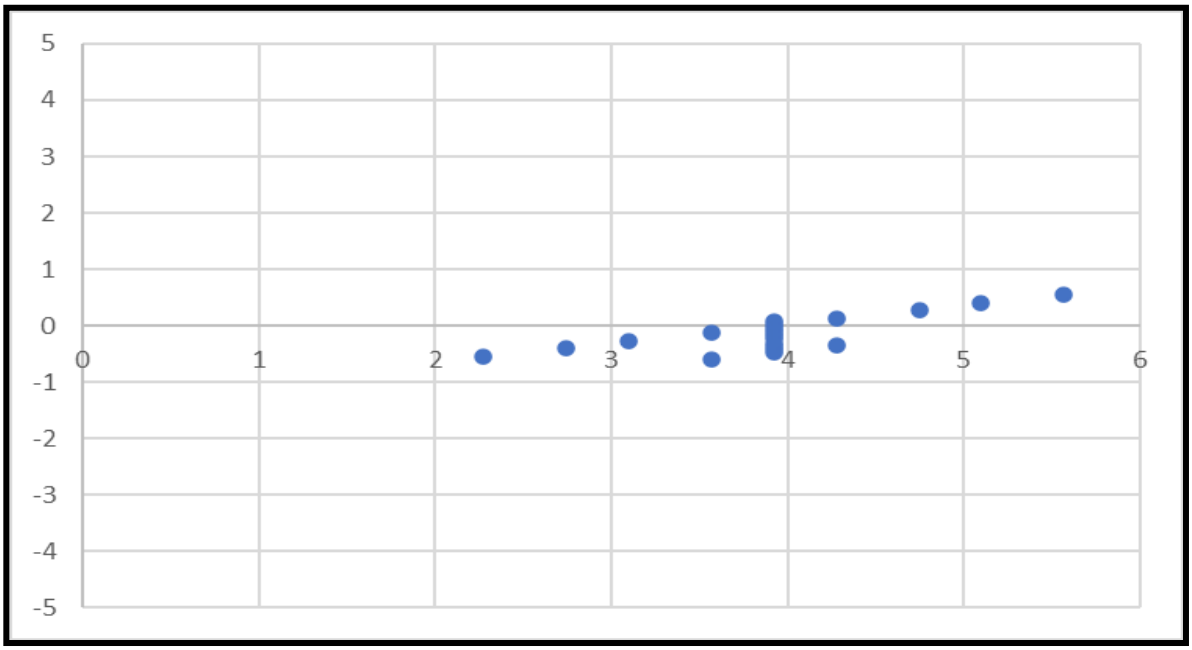


Figura 66. Imagem RPA teste *Rd21*$_{\psi}$ com ângulo de guinada 168,70° corrigida no aplicativo Excel. Fonte: Elaborada pelo autor.

Tabela 40. Matriz de coordenadas (x, y, z) 3x18 *Rd21*$_{\psi}$ para o ângulo de guinada 168,70° corrigida.

Ponto	Coordenada x	Coordenada y	Coordenada z
A	3,923343	0	0
B	3,570905	-0,58784	0
C	3,923785	-0,39189	0
D	2,276157	-0,54865	0
E	4,276841	-0,3527	0
F	4,276311	0,117568	0
G	5,570528	0,548649	0
H	2,746782	-0,39189	0
I	3,923873	-0,47027	0
J	3,09975	-0,27432	0
K	3,570374	-0,11757	0
L	3,923254	0,078378	0
M	3,923519	-0,15676	0
N	3,923431	-0,07838	0
O	3,923608	-0,23514	0

P	3,923696	-0,31351	0
Q	4,746935	0,274325	0
R	5,099903	0,391892	0

Fonte: Elaborada pelo autor.

Os resultados dos testes no aplicativo Excel apresentaram correção satisfatória para o primeiro e o segundo quadrante com o modelo matemático $Rd11_{\psi}$ nos casos com ângulos de guinada de 13,19° e de 168,70°, bem como resultados próximo da correção satisfatória com ângulos de guinada de 56,75° e de 90°. Para o modelo matemático $Rd21_{\psi}$ no primeiro e no segundo quadrante, os resultados foram próximos da correção satisfatória para os ângulos de guinada de 13,19° e de 168,70°. Entretanto, para os ângulos de guinada de 56,75° e de 90°, os resultados da correção não foram satisfatórios.

Neste contexto, o modelo matemático $Rd11_{\psi}$ foi mais eficaz que o modelo matemático $Rd21_{\psi}$. Desta forma, será apresentado no próximo item, o algoritmo de correção do ângulo de guinada com o modelo matemático $Rd11_{\psi}$. Contudo, se há a intensão de testar o modelo matemático $Rd21_{\psi}$, basta alterar a matriz $Rd11_{\psi}$ pela matriz $Rd21_{\psi}$ no algoritmo de correção do ângulo de guinada.

4.2 ALGORITMO DE CORREÇÃO GEOMÉTRICA PARA O ÂNGULO DE GUINADA NA LINGUAGEM DE PROGRAMAÇÃO PYTHON

O algoritmo na linguagem de programação Python, que permite a visualização inicial na tela de trabalho das imagens originais da RPA de forma redimensionada, é apresentado abaixo como algoritmo de visualização. O objetivo deste algoritmo é apresentar as imagens de uma faixa de voo da RPA de forma redimensionada para uma análise inicial.

```
#Algoritmo de visualização.
#Redução do tamanho da imagem na tela de trabalho.

import numpy as np  #importar numpy.
```

```
import cv2              #importar cv2.

imagem1 = cv2.imread('imagem teste.jpg')  #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation =
cv2.INTER_AREA) #interpolação.
cv2.imshow('Imagem_menor', imagem1_redimensionada)   #nova imagem na tela.
```

O algoritmo de correção geométrica proposto nesta tese é composto de duas partes básicas. A primeira parte permite a visualização da imagem original redimensionada na tela de trabalho. A segunda parte permite a correção do ângulo de guinada com a visualização da imagem corrigida redimensionada na tela de trabalho. Na imagem corrigida salva é mantido o tamanho original. É aplicada a transformação M (Eqs. 78 ou 79). Nesta transformação, trabalha-se com o tamanho 2x2, coordenadas x e y, pois a coordenada z não sofre mudança. Após a transformação M, matriz de rotação R_ψ modificada, são aplicadas as Eqs. 117 ou 118 por meio da função *warpAffine*. Também, a fim de manter a imagem planificada, usou-se o cosseno do complemento do suplemento do ângulo a ser corrigido no termo a_{21}.

O algoritmo na linguagem de programação Python, que permite a correção geométrica no ângulo de guinada por meio da matriz de rotação R_ψ das imagens originais da RPA, é apresentado no Apêndice A.

O capítulo seguinte representa a sexta e a sétima fase desta tese: aplicação dos algoritmos de correção nos mosaicos iniciais, geração de ortomosaicos e MDEs no Agisoft Photoscan e avaliação do PEC-PCD no GeoPEC, onde é verificado o comportamento da distribuição espacial, a normalidade e a acurácia posicional das amostras, além das análises de tendências e de precisão.

5 EXPERIMENTO E RESULTADOS FINAIS

Para a sexta fase desta tese, o aplicativo livre E-Foto exige que os ângulos de atitude da câmera sejam do sistema fotogramétrico. Esta condição inicial traz a necessidade de converter os ângulos de atitude do RPA, sistema aeronáutico, para os ângulos de atitude da câmera, sistema fotogramétrico, tornando o processo de correção do ângulo de guinada mais demorado.

Além disso, conforme resultados obtidos na Seção 3 desta tese, na avaliação do PEC, o aplicativo Agisoft Photoscan apresentou menores RMS na planimetria e na altimetria em relação ao E-Foto. Desta forma, nesta seção, após as imagens da primeira e da segunda área serem corrigidas pelo algoritmo de correção do ângulo de guinada, serão gerados MDEs e ortomosaicos corrigidos no Agisoft Photoscan.

A **Figura 67** apresenta as três faixas de voo da primeira área com suas imagens ópticas originais, sem correção, num total de 17 imagens, sendo 5 imagens na faixa 1 (P4 a P8), coluna da esquerda, 6 imagens na faixa 2 (P11 a P16), coluna central, e 6 imagens na faixa 3 (P4 a P9), coluna da direita. Os ângulos *yaw* estão apresentados na Tabela 7.

Faixa 1 - Imagem P4 | Faixa 2 - Imagem P16 | Faixa 3 - Imagem P4



Faixa 1 - Imagem P5 | Faixa 2 - Imagem P15 | Faixa 3 - Imagem P5



Figura 67. Três faixas de voo da primeira área de estudo, com suas imagens ópticas originais, sendo 5 imagens na faixa 1 (P4 a P8), coluna da esquerda, 6 imagens na faixa 2 (P11 a P16), coluna central, e 6 imagens na faixa 3 (P4 a P9), coluna da direita. Fonte: Elaborada pelo autor.

A **Figura 68** apresenta as três faixas de voo da primeira área, com suas imagens ópticas corrigidas nas faixas 1 e 3, e imagens originais na faixa 2, num total de 17 imagens, sendo 5 imagens na faixa 1 (P4 a P8), coluna da esquerda, 6 imagens na faixa 2 (P11 a P16), coluna central, e 6 imagens na faixa 3 (P4 a P9), coluna da direita.

Faixa 1 - Imagem P4

Faixa 2 - Imagem P16

Faixa 3 - Imagem P4

Faixa 1 - Imagem P5

Faixa 2 - Imagem P15

Faixa 3 - Imagem P5

Faixa 1 - Imagem P6

Faixa 2 - Imagem P14

Faixa 3 - Imagem P6

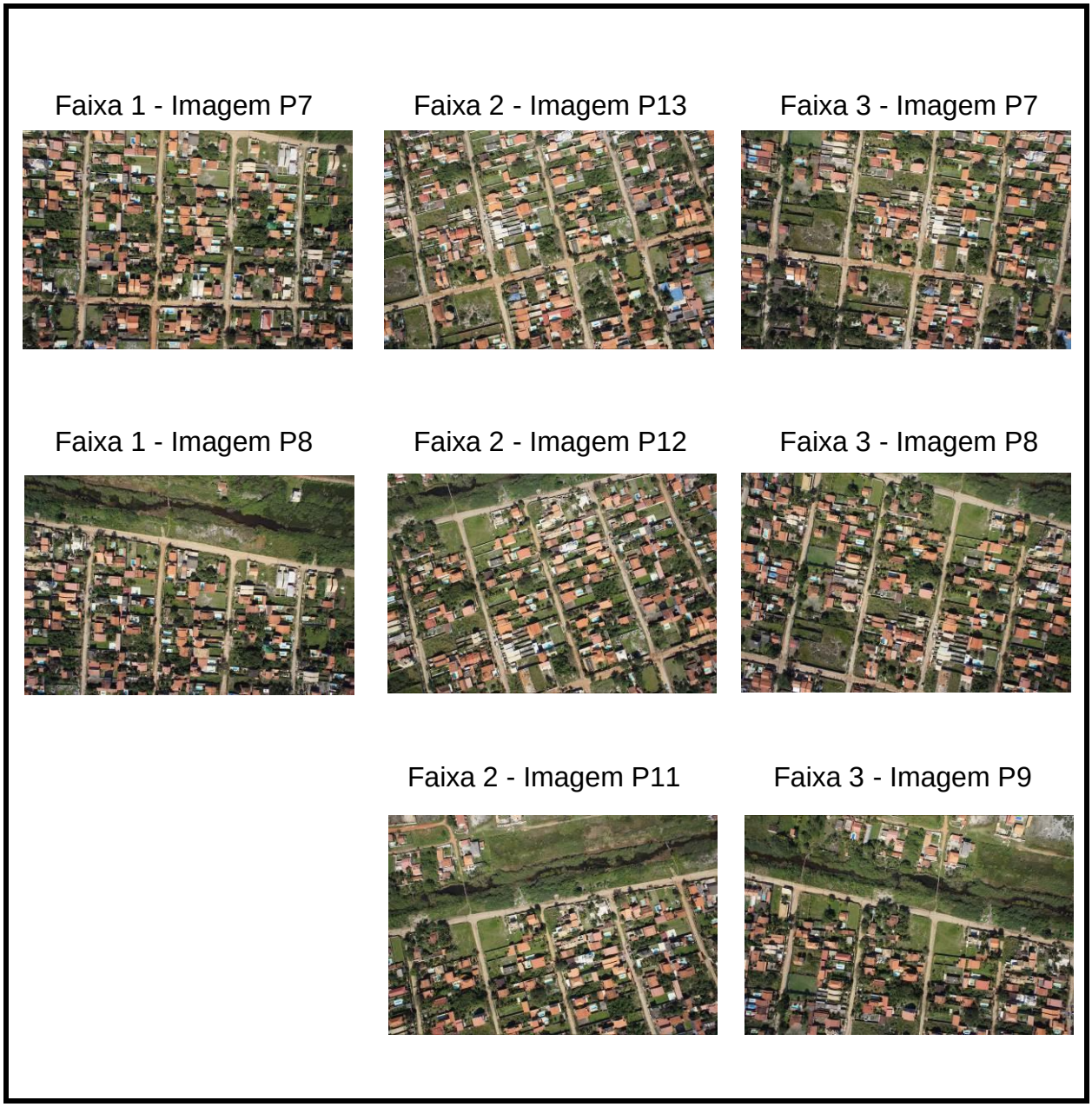


Figura 68. Três faixas de voo da primeira área com suas imagens ópticas corrigidas nas faixas 1 e 3, e imagens originais na faixa 2, sendo 5 imagens na faixa 1 (P4 a P8), coluna da esquerda, 6 imagens na faixa 2 (P11 a P16), coluna central, e 6 imagens na faixa 3 (P4 a P9), coluna da direita. Fonte: Elaborada pelo autor.

A **Figura 69** apresenta a faixa de voo da segunda área com suas imagens ópticas originais, sem correção, num total de 28 imagens (P1 a P28). Os ângulos *yaw* estão apresentados na Tabela 8.

Faixa 1 - Imagem P1

Faixa 1 - Imagem P11

Faixa 1 - Imagem P20

Faixa 1 - Imagem P2

Faixa 1 - Imagem P12

Faixa 1 - Imagem P21

Faixa 1 - Imagem P3

Faixa 1 - Imagem P13

Faixa 1 - Imagem P22

Faixa 1 - Imagem P4

Faixa 1 - Imagem P14

Faixa 1 - Imagem P23

Faixa 1 - Imagem P5

Faixa 1 - Imagem P15

Faixa 1 - Imagem P24

Faixa 1 - Imagem P6

Faixa 1 - Imagem P16

Faixa 1 - Imagem P25

Faixa 1 - Imagem P7

Faixa 1 - Imagem P17

Faixa 1 - Imagem P26

Faixa 1 - Imagem P8

Faixa 1 - Imagem P18

Faixa 1 - Imagem P27

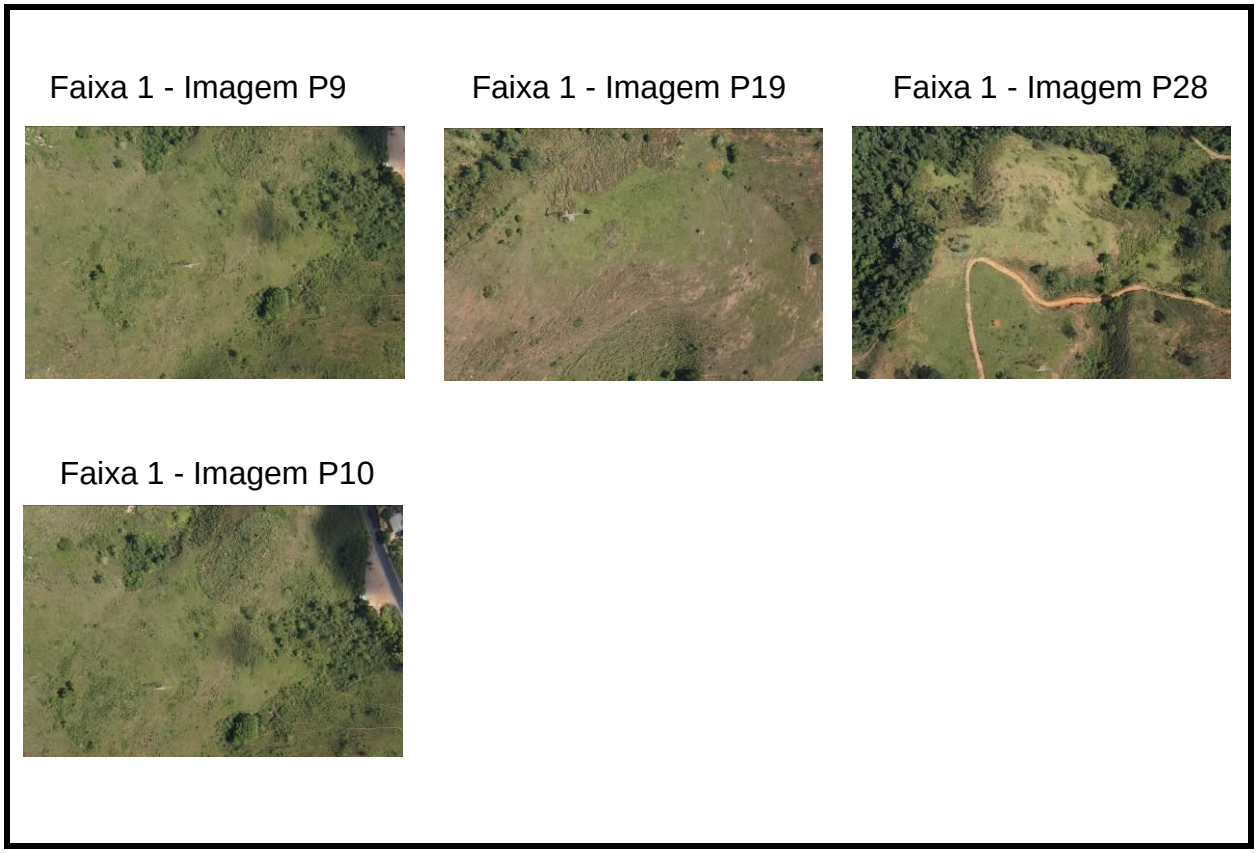
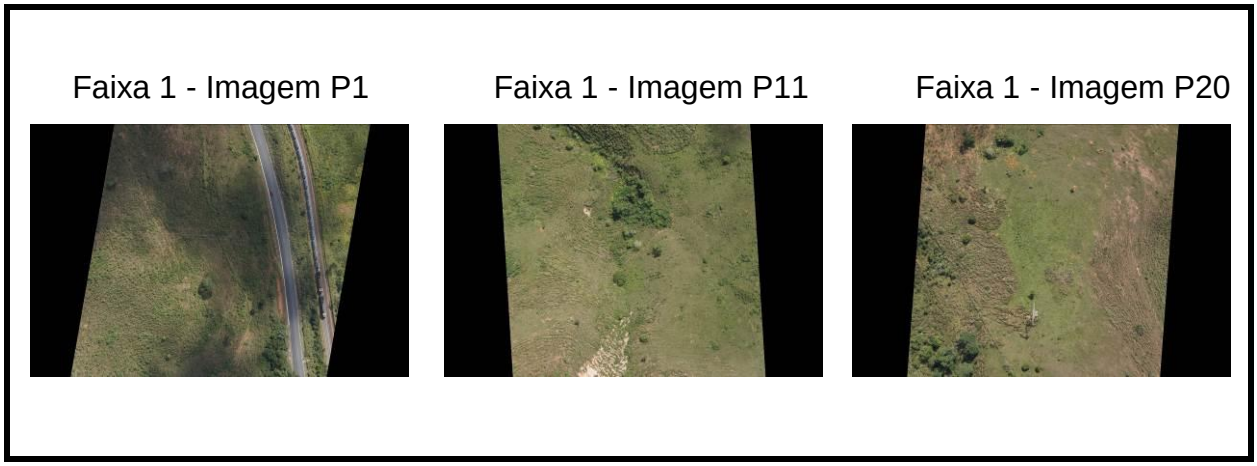


Figura 69. Faixa de voo da segunda área com suas imagens ópticas originais, sem correção, num total de 28 imagens (P1 a P28). Fonte: Elaborada pelo autor.

A **Figura 70** apresenta a faixa de voo da segunda área com suas imagens ópticas corrigidas num total de 28 imagens (P1 a P28).



Faixa 1 - Imagem P2

Faixa 1 - Imagem P12

Faixa 1 - Imagem P21

Faixa 1 - Imagem P3

Faixa 1 - Imagem P13

Faixa 1 - Imagem P22

Faixa 1 - Imagem P4

Faixa 1 - Imagem P14

Faixa 1 - Imagem P23

Faixa 1 - Imagem P5

Faixa 1 - Imagem P15

Faixa 1 - Imagem P24

Faixa 1 - Imagem P6

Faixa 1 - Imagem P16

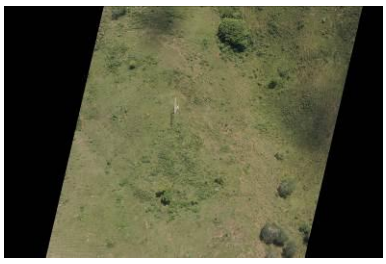
Faixa 1 - Imagem P25

Faixa 1 - Imagem P7

Faixa 1 - Imagem P17

Faixa 1 - Imagem P26

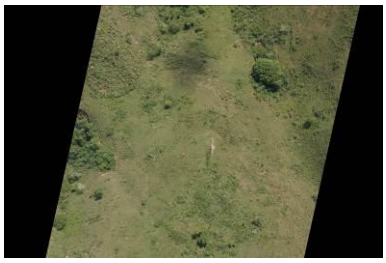
Faixa 1 - Imagem P8



Faixa 1 - Imagem P18

Faixa 1 - Imagem P27

Faixa 1 - Imagem P9



Faixa 1 - Imagem P19

Faixa 1 - Imagem P28

Figura 70. Faixa de voo da segunda área com suas imagens ópticas corrigidas num total de 28 imagens (P1 a P28). Fonte: Elaborada pelo autor.

5.1. GERAÇÃO DE ORTOMOSAICOS CORRIGIDOS

A criação dos ortomosaicos no Agisoft Photoscan seguiu quatro fases, conforme sequência realizada na Seção 3 deste trabalho, no experimento e seus resultados. A primeira fase é o alinhamento da câmara. A segunda fase é a construção da nuvem de pontos densa. A terceira fase é a construção de uma malha poligonal 3D. Na quarta fase, a malha pode ser texturizada (AGISOFT, 2019). Desta forma, na quarta fase tem-se a geração dos MDE das áreas 1 (**Figura 71**) e 2 (**Figura 72**), com 12,6 cm e 6 cm de resolução, respectivamente.

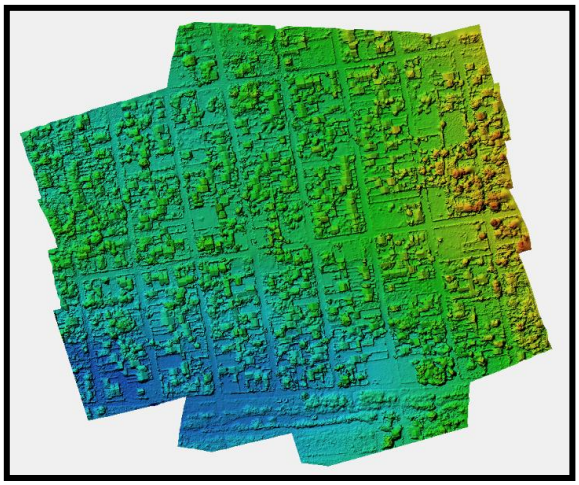


Figura 71. MDE corrigido da área 1, com 12,6 cm de resolução espacial. Fonte: Elaborada pelo autor.

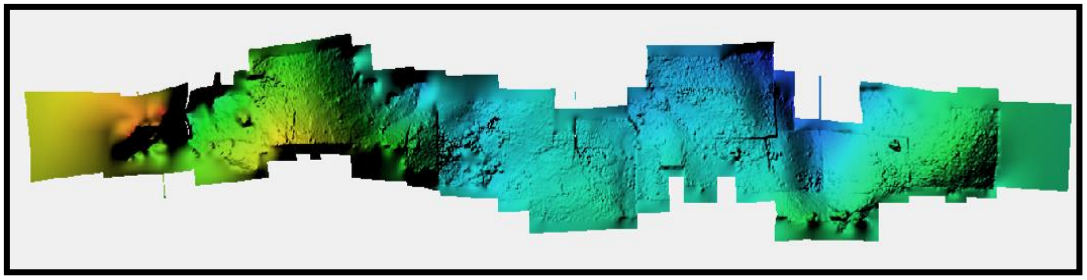


Figura 72. MDE corrigido da área 2, com 6 cm de resolução espacial. Fonte: Elaborada pelo autor.

Por fim, os MDE são utilizados para a geração de ortomosaicos da área 1 (**Figura 73**) e 2 (**Figura 74**), com 12,6 cm e 6 cm de resolução, respectivamente.

Figura 73. Ortomosaico corrigido da área 1, com 12,6 cm de resolução espacial. Fonte: Elaborada pelo autor.

Figura 74. Ortomosaico corrigido da área 2, com 6 cm de resolução espacial. Fonte: Elaborada pelo autor.

5.2. AVALIAÇÃO DOS ORTOMOSAICOS CORRIGIDOS

Na sétima fase desta tese, para a planimetria da Área 1 corrigida, com 13 pontos de verificação (o aplicativo GeoPEC rejeitou 2 pontos), e considerando-se a junção do teste de tendência com o teste de precisão, as amostras apresentaram padrão disperso, distribuição normal, com resultado tendencioso somente na direção norte, com redução do efeito sistemático. A precisão resultou em classe A (PEC) e classe A (PEC-PCD), escala 1:1.000 (Tabela 41).

Tabela 41. Dados estatísticos planimétricos da área 1 corrigida do GeoPEC.

Estatística	Leste	Norte	Posicional
Média	-0,0024	0,0685	0,0812
Desvio-padrão	0,0346	0,0604	0,0531
Erro médio quadrático	0,0336	0,0898	0,0959
Máximo	0,0700	0,1780	0,1802
Mínimo	-0,0640	-0,0290	0,0070
Assimetria	0,0020	0,1360	0,3110

Fonte: Fonte: Elaborada pelo autor.

Para a altimetria da Área 1 corrigida, com 15 pontos de verificação, e considerando-se a junção do teste de tendência com o teste de precisão, as amostras apresentaram distribuição normal, com resultado tendencioso, com redução do efeito sistemático. A precisão resultou em classe A (PEC) e classe A (PEC-PCD), na equidistância de 60 metros de curva de nível (Tabela 42).

Tabela 42. Dados estatísticos altimétricos da área 1 corrigida do GeoPEC.

Estatística	Altitude (m)

Média	8,6808
Desvio-padrão	4,4138
Erro médio quadrático	9,6716
Máximo	17,6390
Mínimo	0,0000
Assimetria	0,0578

Fonte: Fonte: Elaborada pelo autor.

Considerando a planimetria da Área 2 corrigida, com 8 pontos de verificação, e considerando-se a junção do teste de tendência com o teste de precisão, as amostras apresentaram padrão disperso, distribuição normal, com resultado tendencioso. A precisão resultou em classe B (PEC) e classe C (PEC-PCD), escala 1:2.000 (Tabela 43).

Tabela 43. Dados estatísticos planimétricos da área 2 corrigida do GeoPEC.

Estatística	Leste	Norte	Posicional
Média	-0,246	0,8684	0,9265
Desvio-padrão	0,2309	0,3193	0,3243
Erro médio quadrático	0,3273	0,9183	0,9749
Máximo	0,0440	1,4120	1,4120
Mínimo	-0,5790	0,2640	0,2721
Assimetria	-0,0500	-0,2320	-0,6280

Fonte: Elaborada pelo autor.

Para a altimetria da Área 2 corrigida, com 7 pontos de verificação (o aplicativo GeoPEC rejeitou 1 ponto), e considerando-se a junção do teste de tendência com o

teste de precisão, as amostras apresentaram distribuição normal, com resultado não tendencioso. A precisão resultou em classe C (PEC) e classe D (PEC-PCD), na equidistância de 50 metros de curva de nível (Tabela 44).

Tabela 44. Dados estatísticos altimétricos da área 2 corrigida do GeoPEC.

Estatística	Altitude (m)
Média	4,008
Desvio-padrão	17,5750
Erro médio quadrático	16,7577
Máximo	31,8500
Mínimo	-24,7880
Assimetria	-0,0167

Fonte: Elaborada pelos autores.

Em todas as análises feitas nas duas áreas, foram padronizados o nível de confiança de 95% para o teste de normalidade, nível de confiança de 90% para o teste *t* de *Student* (tendência) e nível de confiança de 90% para χ^2 (precisão)**.** A Tabela 45 representa um resumo dos erros médios quadráticos da planimetria e classes PEC e PEC-PCD das Áreas 1 e 2 no aplicativo Agisoft Photoscan, considerando os mosaicos não corrigidos e corrigidos.

Tabela 45. Dados estatísticos (erro médio quadrático) da planimetria das Áreas 1 e 2 no Agisoft Photoscan, com mosaicos não corrigidos e corrigidos. PEC = Padrão de Exatidão Cartográfica; PCD = Produtos Cartográficos Digitais; e Es = Escala.

Tipos	Planimetria RMS	Es	Classes PEC	Classes PEC-PCD

Área 1 não corrigida	1,4337	1:5.000	A	B
Área 1 corrigida	0,0959	1:1.000	A	A
Área 2 não corrigida	1,1073	1:2.000	C	D
Área 2 corrigida	0,9749	1:2.000	B	C

Fonte: Elaborada pelo autor.

A Tabela 46 representa um resumo dos erros médios quadráticos da altimetria e classes PEC e PEC-PCD das Áreas 1 e 2 não corrigidos e corrigidos obtidas pelo aplicativo Agisoft Photoscan.

Tabela 46. Dados estatísticos erro médio quadrático da altimetria das Áreas 1 e 2 não corrigidos e corrigidos no Agisoft Photoscan. PEC = Padrão de Exatidão Cartográfica; PCD = Produtos Cartográficos Digitais; e Eq = Equidistância.

Tipos	Altimetria RMS	Eq (m)	Classes PEC	Classes PEC-PCD
Área 1 não corrigida	31,3236	70	C	D
Área 1 corrigida	9,6716	60	A	A
Área 2 não corrigida	21,0263	50	C	D
Área 2 corrigida	16,7577	50	C	D

Fonte: Elaborada pelo autor.

Conforme a Tabela 45, verificaram-se os melhores resultados nas planimetrias das áreas 1 e 2 no aplicativo Agisoft Photoscan, após a correção geométrica no ângulo de guinada. A área 1 corrigida apresentou menor RMS, com uma escala maior que 1:1.000; também foi possível obter PEC A, analógico e digital. Apesar do relevo ondulado da área 2 corrigida, mantendo-se a mesma escala de 1:2.000, o RMS foi menor, com melhoria no PEC, B para o analógico e C para o digital.

Concernente à Tabela 46, verifica-se também melhores resultados nas altimetrias das áreas 1 e 2 no aplicativo Agisoft Photoscan, após a correção

geométrica no ângulo de guinada. A área 1 corrigida apresentou menor RMS, com uma equidistância menor de 60 metros; também foi possível obter PEC A, analógico e digital. Para a área 2 corrigida, mantendo-se a mesma equidistância de 50 metros e os PECs, o RMS foi menor.

Cabe frisar que, todas as avaliações foram realizadas com padrão disperso e distribuição normal, obtendo-se redução no resultado tendencioso ou resultado não tendencioso.

Concernente a escala das fotos, a relação escala igual a distância focal dividida pela altura se mante constante. As distâncias focais foram 16 mm para a área 1 e 35 mm para a área 2. Conforme tabelas 7 e 8, as alturas estão mantendo a média de 300 metros para a área 1 e 1080 metros para área 2 (ANDRADE, 1998). A resolução espacial nos ortomosaicos sem correção na área 1 foi de 12 cm e na área 2 de 7 cm no Agisoft Photoscan. A resolução espacial nos ortomosaicos com correção na área 1 foi de 12,6 cm e na área 2 de 6 cm no Agisoft Photoscan.

Desta forma, o modelo matemático de transformação projetiva modificada para o ângulo de guinada, através da modificação de sua matriz de rotação R_{ψ}, por meio de um algoritmo específico na linguagem de programação Python, apresentou resultados significativos na redução do RMS e melhores PEC, analógico e digital.

6 CONCLUSÕES

Conforme o primeiro objetivo específico proposto, considerando a necessidade de corrigir erros geométricos causados pelas variações nos ângulos de guinada em ortomosaicos de imagens ópticas, obtidas por RPAs de asa-fixa de classe 3, os quais normalmente não utilizam plataformas *gimbal* giro-estabilizadas, esta tese demonstrou que é possível aplicar o método matemático paramétrico de transformação projetiva, com derivação dos termos da matriz de rotação $R_{\psi,}$ que modifica a fórmula da transformação projetiva clássica, aumentando a rotação da imagem, a fim de aumentar o grau de correção.

De acordo com o segundo objetivo específico desta tese, foi possível criar um algoritmo de rotação de imagem em linguagem de programação Python, com a nova fórmula da transformação projetiva, capaz de minimizar erros geométricos em ortomosaicos de imagens ópticas obtidas por RPAs.

Em conformidade com o terceiro objetivo específico deste trabalho, ao avaliar o PEC analógico e digital (por meio do aplicativo GeoPEC) de ortomosaicos corrigidos pelo algoritmo de rotação de imagem em linguagem de programação *Python*, os resultados foram satisfatórios, obtendo-se menores erros quadráticos médios.

As áreas de estudo foram em Maricá (área 1), no estado do Rio de Janeiro, com relevo plano, e em Nova Era (área 2), no estado de Minas Gerais, com relevo ondulado. O RPA de asa-fixa de classe 3 utilizado neste projeto foi o PT-UAV, pertencente à empresa Esteio Engenharia e Aerolevantamentos S.A. O sensor empregado foi a câmera Alfa Nex 3 Sony. Para o registro dos pontos de apoio e de verificação, foi utilizado o receptor GPS Hiper da Topcon e a antena Hiper GD. A metodologia empregada neste trabalho foi dividida em sete fases.

Após a apresentação da metodologia na introdução, este trabalho partiu para a fundamentação teórica, considerando todo o processo de geração do ortomosaico.

Nas fases primeira à quarta, isto é, planejamento e execução dos voos; determinação dos pontos de apoio e de verificação; geração dos ortomosaicos iniciais por meio dos aplicativos Agisoft Photoscan e E-Foto; e avaliação dos ortomosaicos, por meio do aplicativo Geo PEC, sem a correção do ângulo de guinada, verificou-se que as amostras apresentaram padrões dispersos,

distribuições normais, com resultados tendenciosos. Os melhores resultados nas planimetrias das áreas foram obtidos no aplicativo Agisoft Photoscan. Na altimetria da área 1, o aplicativo E-Foto apresentou melhor resultado. Entretanto, os RMS, as escalas e os PECs não foram satisfatórios, sendo necessária a aplicação de um modelo de correção geométrico.

Na quinta fase, foi proposto um modelo matemático de transformação projetiva modificada completo. Todavia, a fim de simplificar o processo de correção, foi proposta apenas a aplicação da modificação de sua matriz de rotação R_{ψ}, por meio de um algoritmo específico na linguagem de programação Python. A criação de um algoritmo contendo todas matrizes de rotação estenderia o tempo desta pesquisa, limitada em quatro anos. As distorções dos ângulos *roll* e *pitch* em sua grande maioria são menores que 5 graus, o que atende os padrões cartográficos. O ângulo *yaw* é o ângulo principal a ser corrigido devido aos altos valores maiores que 5 graus.

Na sexta fase, foi aplicado o algoritmo de correção nos mosaicos iniciais, seguido pela geração dos ortomosaicos e MDEs corrigidos no Agisoft Photoscan. Por fim, na sétima fase, os ortomosaicos foram reavaliados no GeoPEC.

Conclui-se que, com as reavaliações com padrão disperso e distribuição normal, obteve-se uma redução no resultado tendencioso ou resultado não tendencioso. A escala na planimetria da área 1 foi aumentada. A equidistância da área 1 foi reduzida. Os RMS foram reduzidos em todas as áreas. Os PECs na maioria das áreas foram aumentados de nível. Desta forma, RMS, escala, equidistância e PECs foram todos melhorados em relação aos ortomosaicos não corrigidos.

Para trabalhos futuros, pode-se criar um algoritmo contento todo o modelo matemático de transformação projetiva modificada proposto, com as três matrizes de rotação. Este modelo completo poderá ser utilizado para criar um novo aplicativo livre para geração de ortomosaicos corrigidos de imagens ópticas de RPAs, colaborando com o E-Foto e o Agisoft Photoscan.

7. REFERÊNCIAS

AGISOFT. **Agisoft PhotoScan User Manual: Professional Edition, Version 1.2**. Disponível em: <www.agisoft.com/downloads/usermanuals/>. Acesso: 15 de junho de 2019.

ALIPRANDI, D. C.; CAPOTE, G.; FARIA, J. R. F.; NEVES, E. M.; SÁ, R. **Análise** tipomorfológica da paisagem e do sistema de espaços livres urbanos do município de Maricá (RJ): Escala urbana - bacia. **Paisagem e Ambiente: Ensaios**, n. 33, p. 83-96, 2014.

ALMEIDA, J. A. **Da segurança operacional para implantação de VANT em espaço aéreo não-agregado no Brasil: capacidade de perceber e evitar** (dissertação de Mestrado em Engenharia Aeronáutica e Mecânica), São José dos Campos: ITA, 2012.

ALVES JÚNIOR, L. R.; FERREIRA, M. E.; CÔRTES, J. B. R.; JORGE, L. A. C. High accuracy mapping with cartographic assessment for a fixed-wing remotely piloted aircraft system. **Journal of Applied Remote Sensing,** v. 12/1, 2018.

ALVES JÚNIOR, L. R.; CÔRTES, J. B. R.; SILVA, J. R.; FERREIRA, M. E. Validação de ortomosaicos e modelos digitais de terreno utilizando fotografias obtidas com câmera digital não métrica acoplada a um VANT. **Revista Brasileira de Cartografia,** v. 67/7, p. 1453-1466, 2015.

ALVES JÚNIOR, L. R. **Análise de produtos cartográficos obtidos com câmera digital não métrica acoplada a um veículo aéreo não tripulado em áreas urbanas e rurais no estado de Goiás** (dissertação de Mestrado em Geografia). Goiânia: UFG/IESA, 2015.

ANDRADE, J. B.; **Fotogrametria**. Curitiba: SBEE, 1998.

ANTON, H; RORRES, C. **Álgebra Linear com aplicações. Porto Alegre**: Bookman, 10ª ed. 2012.

ARAÚJO, E. H. G.; KUX, H. J. H.; ALBUQUERQUE, P. C. G. **Ortorretificação de imagens QuickBird standard – levantamento planialtimétrico de pontos de apoio e métodos de correção geométrica**. São José dos Campos: INPE, 2007.

AKAR, O. Mapping land use with using rotation forest algorithm from UAV images. **European Journal of Remote Sensing,** v. 50, p. 269-279, 2017.

BERTEŠKA, T. E.; RUZGIENĖ, B. Photogrammetric mapping based on UAV imager. **Geodesy and Cartography,** v**.** 39, p**.** 158-163, 2013.

BISQUERRA, R.; SARRIERA, J. C.; MARTÍNEZ, F. **Introdução à Estatística. Enfoque Informático com o Pacote Estatístico SPSS**. Porto Alegre. 2004.

BÖRLIN, N. **Fundamentals of Photogrammetry**. Department of Computing Science. Umeã University, Sweden. 2014.

BOERY, M. N. O. Veículos aéreos não tripulados e a segurança de voo: uma análise no contexto internacional. **In:** SEMINÁRIO DE RELAÇÕES INTERNACIONAIS DA ASSOCIAÇÃO BRASILEIRA DE RELAÇÕES INTERNACIONAIS, 3. **Anais...** Florianópolis, SC. 2016.

BRASIL. Ministério da Defesa. **Sistemas de aeronaves remotamente pilotadas e o acesso ao espaço aéreo brasileiro**. Tráfego Aéreo, ICA 100-40**,** de 20 de novembro de 2018. Disponível em: <https:// https://publicacao&id=4944&refresh=74BF0EC9-5C52-4974-8BC43E6188C82079>.

BRASIL. Ministério da Defesa. **Decreto-Lei Nº 1.177 de 21 de junho de 1971.** Dispõe sobre aerolevantamentos no território nacional e dá outras providências. Disponível em: <http://www.planalto.gov.br/ccivil_03/Decreto-Lei/1965-1988/Del1177.htm.>.

BRASIL. **Decreto nº 6.666, de 27 de novembro de 2008.** Institui, no âmbito do Poder Executivo federal, a Infra-Estrutura Nacional de Dados Espaciais - INDE, e dá outras providências. Disponível em: www.planalto.gov.br/ccivil_03/_Ato2007-010/2008/Decreto/D6666.htm.

BRASIL. Ministério da Defesa. **Portaria Nº 101/GM-MD, de 26 de dezembro de 2018**. Dispõe sobre a adoção de procedimentos para a atividade de aerolevantamento no território nacional. Disponível em: < http://www.planalto.gov.br/ccivil_03/Decreto-Lei/1965-1988/Del1177.htm.>.

BRASIL. Ministério da Defesa. **Decreto Nº 89.817 de 20 de junho de 1984**. Estabelece as instruções reguladoras das normas técnicas da cartografia nacional. Disponível em: <http://www.planalto.gov.br/ccivil_03/decreto/1980-1989/D89817.htm>.

BRASIL. Portal Ministério da Defesa. **Geoinformação e aerolevantamento**. Disponível em: <http://www.defesa.gov.br/cartografia-e-aerolevantamento-claten>. Acesso em 28 de abril de 2018.

CAMPOS, M. B.; TOMMASELLI, A. M. G.; MORAES, M. V. A.; MARCATO JÚNIOR, J. Análise comparativa dos resultados obtidos pelos métodos de calibração de campo tridimensional e bidimensional. **Boletim de Ciências Geodésicas**, v. 21, n. 2, p. 308-328, 2015.

CARDOSO, R. J. **Construção de um mosaico utilizando um veículo aéreo não-tripulado através da fundamentação cartográfica** (trabalho de conclusão de curso, Faculdade de Computação da Universidade Federal de Uberlândia), 2018.

COEHO, L.; BRITO, J. N. **Fotogrametria Digital**. Rio de Janeiro: UERJ, 2016. Disponível em: <http://www.efoto.eng.uerj.br/>.

COVENEY, S.; ROBERTS, K. Lightweight UAV digital elevation models and orthoimagery for environmental applications: data accuracy evaluation and potential for river flood risk modelling. **International Journal of Remote Sensing,** v. 38, p. 3159-3180, 2017.

D'ALGE, J. C. L. **Correção geométrica de imagens de sensoriamento remoto.** Aula da matéria Introdução ao Sensoriamento Remoto do curso de Sensoriamento Remoto – INPE, abril 2007.

DRONENG. **Drones e Engenharia**. Disponível em: <http://blog.droneng.com.br/drones-para-mapeamento-aereo-qual-modelo-comprar/>. Acesso em 03 de julho de 2019.

E-FOTO. **Uma Estação Fotogramétrica Digital Educacional Livre.** Disponível em: <http://www.efoto.eng.uerj.br/>. Acesso em 11 de junho de 2019.

EISENBEIß. H. **UAV Photogrammetry**. Zürich: Institut für Geodäsie und Photogrammetrie, 2009.

ESPINOSA, I. C. O. V.; BISCOLLA, L. M. C. C. O.; FILHO, P. B. **Álgebra Linear para Computação**. LTC. Rio de Janeiro, RJ, 2013.

FITZ, P. R. **Cartografia Básica**. São Paulo: Oficina de Textos, 4ª ed.,2009.

FONSECA NETO, F. D.; GRIPP JÚNIOR, G.; BOTELHO, M. F.; SANTOS, A. P.; NASCIMENTO, L. A.; FONSECA, A. L. B. Avaliação da qualidade posicional de dados espaciais gerados por VANT utilizando feições pontuais e lineares para aplicações cadastrais. **Boletim de Ciências Geodésicas**, v. 23, n. 1, p. 134-149, 2017.

FORLANI, G.; RONCELLA, R.; DIOTRI, F. Production of high-resolution digital terrain models in mountain regions to support risk assessment. **Geomatics, Natural Hazards and Risk**, v**.** 6, p**.** 379-397, 2015.

GALBRAITH, A.; THEILER, J.; THOME, K.; ZIOLKOWSKI, R. Resolution enhancement of multilook imagery for the multispectral thermal imager. **IEEE Transactions on Geoscience and Remote Sensing**, v. 43, p. 1964–1977, 2005.

GEMAEL, C.; MACHADO, A. M. L.; WANDRESEN, R. **Introdução ao ajustamento de observações. Aplicações Geodésicas.** Curitiba: UFPR, 2015.

GINI, R.; PAGLIARI, D.; PASSONI, D.; PINTO, L.; SONA, G.; DOSSO, P. UAV photogrammetry: Block triangulation comparisons. **International Archives of the**

Photogrammetry, Remote Sensing and Spatial Information Sciences, v. XL-1/W2. Rostock, Germany, 2013.

HARTLEY, R. I.; ZISSERMAN, A. **Multiple View Geometry in Computer Visio**n. Cambridge University Press, 2ª ed., 2000.

HALLERT, B. **Photogrammetry. Basic Principles and General Survey.** McGraw-Hill, 1960.

HLOTOV, V.; HUNINA, A.; YURKIV, M.; SIEJKA, Z. Determining of correlation relationship between roll, pitch, and yaw for UAVs. **Reports on Geodesy and Geoinformatics**, v.107, p. 13-18, 2019.

IBGE. Instituto Brasileiro de Geografia e Estatística. **Avaliação da Qualidade de Dados Geoespaciais.** Manuais Técnicos em Geociências. Rio de Janeiro: IBGE, 2017.

IBGE. Instituto Brasileira de Geografia e Estatística. **Noções de Cartografia.** Disponível em: <https://ww2.ibge.gov.br/home/geociencias/cartografia/manual_nocoes/representacao.htm>. Acesso em 14 de junho de 2019.

IBGE. Instituto Brasileiro de Geografia e Estatística. **Manual Técnico de Geomorfologia**. Manuais Técnicos em Geociências. Rio de Janeiro: IBGE, 2009.

IEZZI, G. **Fundamentos da Matemática elementar 3**. Trigonometria. 2ª ed., 1977.

JACOBSEN, K. Geometric potential of IKONOS and QuickBird images. **Photogrammetric Weeks**, n. 3, p. 101-110, 2003.

JAIMES, B. R. A. **Estratégias para aumentar a robustez de estimação de posição geográfica em VANTS através de imagens** (dissertação de Mestrado, Escola de Engenharia da Universidade Federal de Minas Gerais), 2016.

JENSEN, J. R. **Sensoriamento Remoto do Ambiente**. Uma Perspectiva em Recursos Terrestres. Upper Saddle River: Prentice Hall, 2ª ed., 2009.

JINGXIONG, Z.; NA, Y. Indicator and multivariate geostatistics for spatial prediction**. Geo-spatial Information Science**, v.11, p.243-246, 2008.

JORGE, J.; VALLBÉ, M.; SOLER, J. A. Detection of irrigation inhomogeneities in an olive grove using the NDRE vegetation index obtained from UAV images. **European Journal of Remote Sensing,** v. 52, n. 1, p. 169-177, 2019.

KIM, J. I.; KIM, T.; SHIN, T.; KIM, S. Fast and robust geometric correction for mosaicking UAV images with narrow overlaps. **International Journal of Remote Sensing**, v. 38, p. 2557-2576, 2017.

KOEVA, M.; MUNEZA, M.; GEVAERT, C.; GERKE, M.; NEX, F. Using UAVs for map creation and updating. A case study in Rwanda. **Survey Review,** v. 50, p. 312-325, 2016.

KOMAZAKI, J. M.; CAMARGO, P. O.; GALO, M.; AMORIM, A. Avaliação da qualidade geométrica de modelos digitais do terreno obtidos a partir de imagens adquiridas com VANT. In: CONGRESSO BRASILEIRO DE CARTOGRAFIA, 27. **Anais...** Rio de Janeiro: SBC, p. 576-580, 2017.

LEITHOLD, L. **O Cálculo com Geometria Analítica**. V. 1, Ed. Harbra, 3ª ed., 1994.

LEMES, I. R.; AMORIM, A.; LEMES, E. R.; JORGE, L, C.; TOLEDO, L. F.; CHECON, M. M. S. Implementação de um portal *geoweb* para auxiliar na gestão territorial. In: SIMPÓSIO BRASILEIRO DE GEOMÁTICA, 4. **Anais...** Presidente Prudente: UNESP, p. 104-109, 2017.

LIMA, S. A.; BRITO, J. L. N. S. Estratégias para retificação de imagens digitais. In: CONGRESSO BRASILEIRO DE CADASTRO TÉCNICO MULTIFINALITÁRIO. **Anais...** Florianópolis: UFSC, 2006.

LIMA, S. A.; ROBERTO, L.; SHIGUEMORI, E. H.; KUX, H. J. H.; NUNES, L. J.; BRITO, S. Determinação da posição e atitudes de VANT por fotogrametria. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 17., Santos, SP. **Anais...** São José dos Campos: INPE, p. 5392-5399, 2017.

LIMA, A. L. **Avaliação da técnica de georreferenciamento direto em mapeamento aerofotogramétrico** (dissertação de Mestrado). Campinas: UNICAMP, 2016.

LIZARAZO, I.; ÂNGULO, V.; RODRÍGUEZ, J. Automatic mapping of land surface elevation changes from UAV-based imagery. **International Journal of Remote Sensing**, v. 38, p. 2603-2622, 2017.

MACHADO e SILVA, A. J. F. M. **Modelos de Correção Geométrica para imagens HRV-SPOT** (dissertação de Mestrado), São José dos Campos: INPE, 1989.

MAROTTA, G. S.; RODRIGUES, D. D.; VIEIRA, C. A. O.; FRANÇA, G. S. L. A. Análise de diferentes transformações para a correção geométrica de imagens orbitais de altíssima resolução. **Revista Brasileira de Cartografia,** n. 63/5, p. 619-631, 2011.

MATTHES, E. **Curso Intensivo de Python. Uma Introdução Prática e Baseada em Projetos à Programação.** NOVATEC, São Paulo, SP. 2017.

MIRANDA, G. H. B.; MEDEIROS, N. G.; SANTOS, A. P.; SANTOS, G. R. Análise de qualidade de amostragem e interpolação na geração de MDE. **Revista Brasileira de Cartografia,** v. 70, n. 1, p. 226-257, 2018.

MOHLER, J. P. **Small camera vertical aerial photography.** Universidade de Montana. Montana, EUA, 1968.

MONICO, J. F. G.; PÓZ, A. P. D.; GALO, M.; SANTOS, M. C.; OLIVEIRA, L. C. Acurácia e precisão: Revendo os conceitos de forma acurada. **Boletim de Ciências Geodésicas**, v. 15, n. 3, p. 469-483, 2009.

MUNARETTO, L. **Vants e Drones. A Aeronáutica ao Alcance de Todos.** 2ª Ed., São José dos Campos, SP, 2017.

NARDINI, E. Da guerra à paz, uma incursão pelo mundo dos drones. **Revista Eletrônica de Jornalismo Científico,** 2016.

NOVAIS, N. A. **Aeronaves remotamente pilotadas: uma proposta para elaboração de regulação nacional** (dissertação de Mestrado em Engenharia Aeronáutica e Mecânica). São José dos Campos: ITA, 2011.

NUNES, J. L.; BRITO, S. **Fluxograma do processamento fotogramétrico no E-Foto.** Laboratório de Fotogrametria e Sensoriamento Remoto. Rio de Janeiro: UERJ, 2017.

OLIVEIRA, D. V.; BRITO, J. L. S. Avaliação da acurácia posicional de dados gerados por aeronave remotamente pilotada. **Revista Brasileira de Cartografia,** v. 71, n. 4, p. 934-959, 2019.

OLIVEIRA, G. D.; ALMEIDA, M. S.; MEDEIROS, N. G.; SANTOS, A. P.; POZ, W. R. D. Correção geométrica de imagens orbitais a partir das coordenadas de vértices de imóveis certificados pelo INCRA. **Revista Brasileira de Cartografia,** v. 70, n. 1, p. 290-324, 2018.

ONUORA, A. E.; MBAOCHA, C. C.; EZE, P. C.; UCHEGBU, V. C. Unmanned aerial vehicle pitch optimization for fast response of elevator control system. **International Journal of Scientific Engineering and Science,** v. 2, n. 3, p. 16-19, 2018.

OPENCV. **OpencCV Team**. Disponível em: <https://opencv.org/>. Acesso: 01 de fevereiro de 2019.

PEGORARO, A. J. **Estudo do potencial de um veículo aéreo não tripulado/quadrotor, como plataforma na obtenção de dados cadastrais** (tese de doutorado em Engenharia Civil), Florianópolis: UFSC, 2013.

PEPE, M.; FREGONESE, L.; SCAIONI, M. Planning airborne photogrammetry and remote sensing missions with modern platforms and sensors. **European Journal of Remote Sensing**, v. 51, n. 1, p. 412-436, 2018.

POOLE, D. **Álgebra Linear**. Cengage Learning. São Paulo, SP, 2004.

PIRAS, M.; TADDIA, G.; FORNO, M. G.; GATTIGLIO, M.; AICARDI, I.; DABOVE, P.; RUSSO, L. S.; LINGUA, A. Detailed geological mapping in mountain areas using an unmanned aerial vehicle: application to the Rodoretto Valley, NW Italian Alps. **Geomatics, Natural Hazards and Risk**, v. 8, p. 137-149, 2017.

PYTHON. Python Software Foundation. Disponível em: <https://www.python.org/>. Acesso: 01 de fevereiro de 2019.

QAYYUM, A.; MALIK, A. S.; SAAD, N. M.; ABDULLAD, M. F. B.; IQBAL, M.; RASHEED, W.; ABDULLAH, A. R. B. A.; JAAFAR, M. Y. H. Measuring height of high-voltage transmission poles using unmanned aerial vehicle (UAV) imagery. **The Imaging Science Journal**, v. 65, p. 137-150, 2017.

RIBEIRO, J. A.; DESEILLIGNY, M. P.; BRITO, J. L. N. S.; BERNARDO FILHO, O.; MOTA, G. L. A. **E-Foto and MicMac: synergetic benefits of integrating open-source digital photogrammetry software.** 2014.

ROBERTO, L.; LIMA, S. A.; SANT'ANNA, S. J. S.; SHIGUEMORI, E. H. Correção de distorção projetiva em imagens obtidas com câmera a bordo de VANT. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 17., Santos, 28-31 maio 2017. **Anais...** São José dos Campos: INPE, p. 7467-7474, 2017.

RUZGIENE, B. **Analysis of camera orientation variation in airborne photogrammetry: images under tilt (roll-pitch-yaw) angles**. Vilnius Gediminas Technical University, Vilnius, Lithuania, 2014.

SANTOS, A. P. **Software desenvolvidos**. Viçosa: UFV, 2018.

SANTOS, D. R. **Fotogrametria II**. Curitiba: UFPR, 2013.

SANTOS, A. P.; RODRIGUES, D. D.; SANTOS, N. T.; GRIPP JÚNIOR, J. Avaliação da acurácia posicional em dados espaciais utilizando técnicas de estatística espacial: proposta de método e exemplo utilizando a norma brasileira. **Boletim de Ciências Geodésicas**, v. 22, n. 4, p. 630-650, 2016.

SARAIVA, C. C.; MITISHITA, E. A.; CENTENO, J. S. Transformação afim no plano e afim paralela para a correção geométrica de imagens IKONOS no processo de monorrestituição digital para a atualização cartográfica de mapas municipais na escala de 1:25.000. **Revista Brasileira de Cartografia,** n. 63/2, p. 293-304, 2011.

SEVERANCE, C. **Python for Informatics.** Exploring Information. Version 2.7.3, 2009.

SILVA, C. A.; DUARTE, C. R.; SOUTO, M. V. S.; SABADIA, J. A. B. Utilização de VANT para geração de ortomosaicos e aplicação do Padrão de Exatidão

Cartográfica (PEC). In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 17., João Pessoa, 25-29 abril 2015. **Anais...** São José dos Campos: INPE, p. 1137-1144, 2015.

SILVEIRA, M. T. **Manual Técnico do Submódulo de Extração do MDS**. Rio de Janeiro: UERJ, 2012.

SOUZA, G. **Análise da influência das configurações dos pontos de apoio e do voo na acurácia de ortofotomosaicos elaborados a partir de dados de VANT** (dissertação de Mestrado em Sensoriamento Remoto). Porto Alegre: UFRGS, 2018.

SCHUT, G. H. **Review of interpolation methods for digital terrain models.** The Canadian Surveyor. Dec. Canada. 1976.

TOMMASELLI, A. M. G.; HASEGAWA, J. K.; GALO, M.; IMAI, N. N.; RUY, R. S. Sensoriamento remoto aerotransportado: uma abordagem usando câmaras digitais. In: SANTIL, F. L. P.; SILVEIRA, H.; SOUZA, M. L.; SANTOS, F. R. (Orgs.). **Recursos Tecnológicos Aplicados à Cartografia.** Maringá: Sthampa, v. 1, p. 81-116, 2010.

TRAMONTINA, J.; COSTA, C. C.; CORREA, A. N.; PEGORARO, A. J. Análise de usabilidade da plataforma fotogramétrica educacional E-Foto para o ensino da fotogrametria digital. **Revista Brasileira de Cartografia,** n°. 69/6, p. 1029-1040, 2017.

VERMEER, M.; AYEHU, G. T. **Digital Aerial Mapping. A Hands-On Course.** 2017.

WARSI, F. A.; HAZRY, D.; AHMED, S. F.; JOYO, M. K.; TANVEER, M. H. Roll angle stabilization of fixed-wing UAVs in occurrence of noises by using PID with EKF controller. **Jökull Journal**, v. 63, p. 189-200, 2013.

WARSI, F. A.; HAZRY, D.; AHMED, S. F.; JOYO, M. K.; TANVEER, M. H.; KAMARUDIN, H.; ZURADZMAN, M. Yaw, pitch and roll controller design for fixed-wing UAV under uncertainty and perturbed condition. **IEEE 10th International Colloquium on Signal Processing & its Applications,** p. 7-9, 2014.

WOLF, P. R.; DEWITT B. A.; WILKINSON, B. E. **Elements of Photogrammetry with Applications in GIS**. 4ª ed., Mc-Graw Hill, 2014.

YULIANTO, M. A. **The Bowman-Shelton Test for Normality Testing**. 2012. Disponível em: <https://digensia.wordpress.com/2012/05/01/the-bowman-shelton-test-for-normality-testing/>.

ZANETTI, J.; GRIPP JUNIOR, J.; SANTOS, A. P. Influência do número e distribuição de pontos de controle em ortofotos geradas a partir de um levantamento por VANT. **Revista Brasileira de Cartografia,** n. 69/2, p. 263-277, 2017.

ZHOU, G.; REICHLE, S. UAV-based multi-sensor data fusion processing. **International Journal of Image and Data Fusion**, v. 1, p. 283-291, 2010.

APÊNDICE A

ALGORITMO NA LINGUAGEM DE PROGRAMAÇÃO PYTHON QUE PERMITE A CORREÇÃO GEOMÉTRICA NO ÂNGULO DE GUINADA POR MEIO DA MATRIZ DE ROTAÇÃO R_{ψ} DAS IMAGENS ORIGINAIS DA RPA

```
# Algoritmo de correção ângulo de guinada.
# Algoritmo para correção da distorção geométrica em ortomosaicos de imagens ópticas obtidas por RPA e causada por variações no ângulo de guinada por meio do método paramétrico de transformação projetiva com derivação dos termos da matriz de rotação Rψ.

### Correção da imagem teste.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('imagem teste.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)
```

```
#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('Imagem_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original.
img1 = cv2.imread('imagem teste.jpg')

#Definindo a transformação.
M = np.float32([
            [   np.cos(np.pi*(ψ/180)),      np.sin(np.pi*( ψ /180)), X],
            [  -np.cos(np.pi*( ψ /180)),      np.cos(np.pi*( ψ /180)), Y]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('Imagem_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)
```

```
#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('Imagem_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
```

O algoritmo na linguagem de programação Python que permite a visualização inicial na tela de trabalho das imagens originais da faixa 1 da primeira área é definido como algoritmo_visualização_faixa 1_ primeira área:

```
#Redução do tamanho da imagem na tela de trabalho.

import numpy as np   #importar numpy.
import cv2               #importar cv2.

imagem1 = cv2.imread('P4.jpg')   #carregar imagem.
largura = imagem1.shape[1]       #variável largura.
altura = imagem1.shape[0]        #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                 #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP4_menor', imagem1_redimensionada)   #nova imagem na tela.

imagem1 = cv2.imread('P5.jpg')   #carregar imagem.
largura = imagem1.shape[1]       #variável largura.
altura = imagem1.shape[0]        #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                 #nova largura em pixels.
```

```
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP5_menor', imagem1_redimensionada)   #nova imagem na tela.

imagem1 = cv2.imread('P6.jpg')   #carregar imagem.
largura = imagem1.shape[1]       #variável largura.
altura = imagem1.shape[0]        #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600               #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP6_menor', imagem1_redimensionada)   #nova imagem na tela.

imagem1 = cv2.imread('P7.jpg')   #carregar imagem.
largura = imagem1.shape[1]       #variável largura.
altura = imagem1.shape[0]        #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600               #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP7_menor', imagem1_redimensionada)   #nova imagem na tela.

imagem1 = cv2.imread('P8.jpg')   #carregar imagem.
largura = imagem1.shape[1]       #variável largura.
altura = imagem1.shape[0]        #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
```

```
largura_nova = 600                    #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP8_menor', imagem1_redimensionada)   #nova imagem na tela.

cv2.waitKey(0)
```

O algoritmo na linguagem de programação Python que permite a correção geométrica no ângulo de guinada, por meio da matriz de rotação R_{ψ}, das imagens originais da faixa 1 da primeira área, cinco imagens, é definido como algoritmo_correção ângulo de guinada_faixa 1_ primeira área:

```
### Correção da imagem P4.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P4.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
```

```
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP4_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P4.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(179.217/180)),      np.sin(np.pi*(179.217/180)), 3320],
        [  -np.cos(np.pi*(89.217/180)),      np.cos(np.pi*(179.217/180)), 2250]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP4_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
```

```
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP4_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P5.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P5.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)
```

```
#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP5_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original.
img1 = cv2.imread('P5.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(177.539/180)),      np.sin(np.pi*(177.539/180)), 3320],
        [  -np.cos(np.pi*(87.539/180)),      np.cos(np.pi*(177.539/180)), 2250]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP5_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
```

```
cv2.imshow('ImagemP5_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P6.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P6.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP6_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
```

```
#Carregamento da imagem original.
img1 = cv2.imread('P6.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(179.497/180)),      np.sin(np.pi*(179.497/180)), 3320],
        [  -np.cos(np.pi*(89.497/180)),      np.cos(np.pi*(179.497/180)), 2250]])
#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP6_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP6_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
### Correção da imagem P7.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P7.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)
#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP7_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original.
img1 = cv2.imread('P7.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(-178.042/180)),      np.sin(np.pi*(-178.042/180)), 3360],
```

```
[  -np.cos(np.pi*(88.042/180)),      np.cos(np.pi*(-178.042/180)), 2270]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))
#Salvar a imagem corrigida.
cv2.imwrite('ImagemP7_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP7_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P8.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np
```

```
#Carregamento da imagem original.
img1 = cv2.imread('P8.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP8_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P8.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(-179.161/180)),      np.sin(np.pi*(-179.161/180)), 3360],
        [  -np.cos(np.pi*(89.161/180)),      np.cos(np.pi*(-179.161/180)), 2250]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape
```

```
#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP8_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP8_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Considerando que a faixa 2 da primeira área não apresenta distorções elevadas no ângulo de guinada, não será feita correção geométrica nesta faixa. Conforme Tabela 7, o ângulo *yaw* máximo é de 5,707°, o que pode ser facilmente corrigido por aplicativos que geram ortomosaicos.

O algoritmo na linguagem de programação Python que permite a visualização inicial na tela de trabalho das imagens originais da faixa 3 da primeira área é definido como algoritmo_visualização_faixa 3_ primeira área:

```
#Redução do tamanho da imagem na tela de trabalho.
```

```
import numpy as np   #importar numpy.
import cv2               #importar cv2.

imagem1 = cv2.imread('P04.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP04_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P05.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP05_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P06.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
```

```
largura_nova = 600                  #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP06_menor', imagem1_redimensionada)     #nova imagem na tela.
imagem1 = cv2.imread('P07.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                  #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP07_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P08.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                  #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP08_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P09.jpg')   #carregar imagem.
```

```
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP09_menor', imagem1_redimensionada)    #nova imagem na tela.

cv2.waitKey(0)
```

O algoritmo na linguagem de programação Python, que permite a correção geométrica no ângulo de guinada por meio da matriz de rotação R_{ψ} das imagens originais da faixa 3 da primeira área, seis imagens, é definido como algoritmo_correção ângulo de guinada_faixa 3_ primeira área:

```
### Correção da imagem P04.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P04.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
```

```
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP04_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P04.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(-179.441/180)),      np.sin(np.pi*(-179.441/180)), 3350],
        [  -np.cos(np.pi*(89.441/180)),      np.cos(np.pi*(-179.441/180)), 2250]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP04_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
```

```
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP04_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P05.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P05.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
```

```
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP05_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original.
img1 = cv2.imread('P05.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(-178.322/180)),      np.sin(np.pi*(-178.322/180)), 3350],
        [  -np.cos(np.pi*(88.322/180)),      np.cos(np.pi*(-178.322/180)), 2250]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP05_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
```

```
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP05_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P06.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P06.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)
```

```
#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP06_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original.
img1 = cv2.imread('P06.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(-179.385/180)),      np.sin(np.pi*(-179.385/180)), 3350],
        [  -np.cos(np.pi*(89.385/180)),      np.cos(np.pi*(-179.385/180)), 2250]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP06_corrigida.jpg', img1)
#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP06_corrigida', img1_redimensionada)
```

```
#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
### Correção da imagem P07.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P07.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP07_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original.
img1 = cv2.imread('P07.jpg')
```

```
#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(177.595/180)),      np.sin(np.pi*(177.595/180)), 3360],
        [  -np.cos(np.pi*(87.595/180)),      np.cos(np.pi*(177.595/180)), 2320]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP07_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP07_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
### Correção da imagem P08.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P08.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP08_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P08.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(176.028/180)),      np.sin(np.pi*(176.028/180)), 3250],
```

```
[  -np.cos(np.pi*(86.028/180)),      np.cos(np.pi*(176.028/180)), 2350]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape
#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP08_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP08_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P09.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np
```

```
#Carregamento da imagem original.
img1 = cv2.imread('P09.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP09_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P09.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(-179.385/180)),      np.sin(np.pi*(-179.385/180)), 3350],
        [  -np.cos(np.pi*(89.385/180)),      np.cos(np.pi*(-179.385/180)), 2250]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape
```

```
#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP09_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP09_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
```

O algoritmo na linguagem de programação Python, que permite a visualização inicial na tela de trabalho das imagens originais da faixa 1, da segunda área, é definido como algoritmo_visualização_faixa 1_ segunda área:

```
#Redução do tamanho da imagem na tela de trabalho.

import numpy as np   #importar numpy.
import cv2              #importar cv2.
```

```
imagem1 = cv2.imread('P01.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP01_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P02.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP02_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P03.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
```

```
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP03_menor', imagem1_redimensionada)    #nova imagem na tela.

imagem1 = cv2.imread('P04.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP04_menor', imagem1_redimensionada)    #nova imagem na tela.

imagem1 = cv2.imread('P05.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP05_menor', imagem1_redimensionada)    #nova imagem na tela.

imagem1 = cv2.imread('P06.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
```

```
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                 #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP06_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P07.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP07_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P08.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP08_menor', imagem1_redimensionada)     #nova imagem na tela.
```

```
imagem1 = cv2.imread('P09.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP09_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P10.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP10_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P11.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
```

```
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP11_menor', imagem1_redimensionada)    #nova imagem na tela.
imagem1 = cv2.imread('P12.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP12_menor', imagem1_redimensionada)    #nova imagem na tela.

imagem1 = cv2.imread('P13.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP13_menor', imagem1_redimensionada)    #nova imagem na tela.

imagem1 = cv2.imread('P14.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
```

```
largura_nova = 600                  #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP14_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P15.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                  #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP15_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P16.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                  #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP16_menor', imagem1_redimensionada)     #nova imagem na tela.
```

```
imagem1 = cv2.imread('P17.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP17_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P18.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP18_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P19.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
```

```
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP19_menor', imagem1_redimensionada)    #nova imagem na tela.
imagem1 = cv2.imread('P20.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP20_menor', imagem1_redimensionada)    #nova imagem na tela.

imagem1 = cv2.imread('P21.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP21_menor', imagem1_redimensionada)    #nova imagem na tela.

imagem1 = cv2.imread('P22.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
```

```
largura_nova = 600                  #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP22_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P23.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                  #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP23_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P24.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                  #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP24_menor', imagem1_redimensionada)     #nova imagem na tela.
```

```
imagem1 = cv2.imread('P25.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP25_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P26.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP26_menor', imagem1_redimensionada)     #nova imagem na tela.

imagem1 = cv2.imread('P27.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
```

```
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP27_menor', imagem1_redimensionada)    #nova imagem na tela.

imagem1 = cv2.imread('P28.jpg')   #carregar imagem.
largura = imagem1.shape[1]        #variável largura.
altura = imagem1.shape[0]         #variável altura.
proporcao = float(altura/largura)  #fórmula proporção.
largura_nova = 600                #nova largura em pixels.
altura_nova = int(largura_nova*proporcao)      #nova altura em pixels.
tamanho_novo = (largura_nova, altura_nova)  #novo tamanho.
imagem1_redimensionada = cv2.resize(imagem1, tamanho_novo, interpolation = cv2.INTER_AREA) #interpolação.
cv2.imshow('ImagemP28_menor', imagem1_redimensionada)    #nova imagem na tela.

cv2.waitKey(0)
```

O algoritmo na linguagem de programação Python, que permite a correção geométrica no ângulo de guinada por meio da matriz de rotação R_{ψ} das imagens originais da faixa 1, da segunda área, vinte e oito imagens, é definido como algoritmo_correção ângulo de guinada_faixa 1_ segunda área:

```
### Correção da imagem P01.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P01.jpg')
```

```
#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP01_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P01.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(80.176/180)),      np.sin(np.pi*(80.176/180)), 450],
        [  -np.cos(np.pi*(9.824/180)),      np.cos(np.pi*(80.176/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))
```

```
#Salvar a imagem corrigida.
cv2.imwrite('ImagemP01_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP01_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
### Correção da imagem P02.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P02.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
```

```
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP02_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P02.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(76.441/180)),      np.sin(np.pi*(76.441/180)), 450],
        [  -np.cos(np.pi*(13.559/180)),      np.cos(np.pi*(76.441/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP02_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
```

```
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP02_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P03.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P03.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
```

```
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)
#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP03_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P03.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(80.360/180)),      np.sin(np.pi*(80.360/180)), 450],
        [  -np.cos(np.pi*(9.64/180)),      np.cos(np.pi*(80.360/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP03_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
```

```
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP03_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P04.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P04.jpg')
#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
```

```
cv2.imshow('ImagemP04_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P04.jpg')

#Definindo a transformação.
M = np.float32([
            [   np.cos(np.pi*(78.383/180)),      np.sin(np.pi*(78.383/180)), 450],
            [  -np.cos(np.pi*(11.617/180)),      np.cos(np.pi*(78.383/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP04_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP04_corrigida', img1_redimensionada)
```

```
#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P05.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P05.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP05_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
```

```
img1 = cv2.imread('P05.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(80.619/180)),      np.sin(np.pi*(80.619/180)), 450],
        [  -np.cos(np.pi*(9.381/180)),      np.cos(np.pi*(80.619/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP05_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP05_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
### Correção da imagem P06.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P06.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP06_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P06.jpg')

#Definindo a transformação.
M = np.float32([
```

```
            [   np.cos(np.pi*(90.001/180)),      np.sin(np.pi*(90.001/180)), 750],
            [  -np.cos(np.pi*(0.001/180)),      np.cos(np.pi*(90.001/180)), 3300])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP06_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP06_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P07.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
```

```
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P07.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP07_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P07.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(88.156/180)),      np.sin(np.pi*(88.156/180)), 650],
        [  -np.cos(np.pi*(1.844/180)),      np.cos(np.pi*(88.156/180)), 3300]])

#Guardando as dimensões da imagem.
```

```
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))
#Salvar a imagem corrigida.
cv2.imwrite('ImagemP07_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP07_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P08.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P08.jpg')
```

```
#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP08_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P08.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(77.031/180)),      np.sin(np.pi*(77.031/180)), 450],
        [  -np.cos(np.pi*(12.969/180)),      np.cos(np.pi*(77.031/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))
```

```
#Salvar a imagem corrigida.
cv2.imwrite('ImagemP08_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP08_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P09.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P09.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape
```

```
#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP09_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P09.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(77.502/180)),      np.sin(np.pi*(77.502/180)), 450],
        [  -np.cos(np.pi*(12.969/180)),      np.cos(np.pi*(77.502/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP09_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
```

```
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP09_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P10.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P10.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
```

```
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP10_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P10.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(82.062/180)),      np.sin(np.pi*(82.062/180)), 450],
        [  -np.cos(np.pi*(7.938/180)),      np.cos(np.pi*(82.062/180)), 3600]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP10_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
```

```
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP10_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P11.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np
#Carregamento da imagem original.
img1 = cv2.imread('P11.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)
```

```
#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP11_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P11.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(93.524/180)),      np.sin(np.pi*(93.524/180)), 850],
        [  -np.cos(np.pi*(3.524/180)),      np.cos(np.pi*(93.524/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP11_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)
```

```
#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP11_corrigida', img1_redimensionada)
#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P12.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P12.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP12_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
```

```
#Carregamento da imagem original
img1 = cv2.imread('P12.jpg')
#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(88.948/180)),      np.sin(np.pi*(88.948/180)), 850],
        [  -np.cos(np.pi*(1.052/180)),      np.cos(np.pi*(88.948/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP12_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP12_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
### Correção da imagem P13.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P13.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP13_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P13.jpg')

#Definindo a transformação.
M = np.float32([
```

```
            [   np.cos(np.pi*(84.732/180)),      np.sin(np.pi*(84.732/180)), 650],
            [  -np.cos(np.pi*(5.268/180)),      np.cos(np.pi*(84.732/180)), 3300]])
#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP13_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP13_corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P14.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
```

```
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P14.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP14_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P14.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(84.910/180)),      np.sin(np.pi*(84.910/180)), 650],
        [  -np.cos(np.pi*(5.09/180)),      np.cos(np.pi*(84.910/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape
```

```
#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP14_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP14corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P15.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P15.jpg')
```

```
#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP15_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P15.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(79.595/180)),      np.sin(np.pi*(79.595/180)), 550],
        [  -np.cos(np.pi*(10.405/180)),      np.cos(np.pi*(79.595/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))
```

```
#Salvar a imagem corrigida.
cv2.imwrite('ImagemP15_corrigida.jpg', img1)
#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP15corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P16.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P16.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
```

```
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP16_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P16.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(74.863/180)),      np.sin(np.pi*(74.863/180)), 450],
        [  -np.cos(np.pi*(15.137/180)),      np.cos(np.pi*(74.863/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP16_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
```

```
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP16corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P17.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P17.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
```

```
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP17_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P17.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(75.715/180)),      np.sin(np.pi*(75.715/180)), 450],
        [  -np.cos(np.pi*(14.285/180)),      np.cos(np.pi*(75.715/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP17_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
```

```
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP17corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P18.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P18.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)
```

```
#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP18_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P18.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(72.115/180)),      np.sin(np.pi*(72.115/180)), 250],
        [  -np.cos(np.pi*(17.885/180)),      np.cos(np.pi*(72.115/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP18_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)
```

```
#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP18corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P19.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P19.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP19_original', img1_redimensionada)
```

```
## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P19.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(77.766/180)),      np.sin(np.pi*(77.766/180)), 450],
        [  -np.cos(np.pi*(12.234/180)),      np.cos(np.pi*(77.766/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP19_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP19corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P20.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P20.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP20_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P20.jpg')
```

```
#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(85.763/180)),      np.sin(np.pi*(85.763/180)), 650],
        [  -np.cos(np.pi*(4.237/180)),      np.cos(np.pi*(85.763/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP20_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP20corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P21.
```

```
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P21.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP21_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P21.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(92.971/180)),      np.sin(np.pi*(92.971/180)), 850],
        [  -np.cos(np.pi*(2.971/180)),      np.cos(np.pi*(92.971/180)), 3300]])
```

```
#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP21_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP21corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P22.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np
```

```
#Carregamento da imagem original.
img1 = cv2.imread('P22.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP22_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P22.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(95.316/180)),      np.sin(np.pi*(95.316/180)), 950],
        [  -np.cos(np.pi*(5.316/180)),      np.cos(np.pi*(95.316/180)), 3800]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape
```

```
#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP22_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP22corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P23.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P23.jpg')
```

```
#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP23_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P23.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(86.569/180)),      np.sin(np.pi*(86.569/180)), 650],
        [  -np.cos(np.pi*(5.316/180)),      np.cos(np.pi*(86.569/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
```

```
cv2.imwrite('ImagemP23_corrigida.jpg', img1)
#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP23corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P24.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P24.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
```

```
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP24_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P24.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(78.211/180)),      np.sin(np.pi*(78.211/180)), 450],
        [  -np.cos(np.pi*(11.789/180)),      np.cos(np.pi*(78.211/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP24_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
```

```
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP24corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P25.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P25.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
```

```
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP25_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P25.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(85.747/180)),      np.sin(np.pi*(85.747/180)), 650],
        [  -np.cos(np.pi*(4.253/180)),      np.cos(np.pi*(85.747/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP25_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
```

```
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP25corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()

### Correção da imagem P26.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P26.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)
```

```
#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP26_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P26.jpg')

#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(83.342/180)),      np.sin(np.pi*(83.342/180)), 450],
        [  -np.cos(np.pi*(83.342/180)),      np.cos(np.pi*(83.342/180)), 1600]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP26_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
```

```
cv2.imshow('ImagemP26corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()



### Correção da imagem P27.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P27.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP27_original', img1_redimensionada)
```

```
## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P27.jpg')

#Definindo a transformação.
M = np.float32([
            [   np.cos(np.pi*(78.226/180)),      np.sin(np.pi*(78.226/180)), 550],
            [  -np.cos(np.pi*(11.774/180)),      np.cos(np.pi*(78.226/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP27_corrigida.jpg', img1)

#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP27corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()


### Correção da imagem P28.
## Parte 1 - Visualização da imagem original.
#Importação do cv2 e numpy.
import cv2
import numpy as np

#Carregamento da imagem original.
img1 = cv2.imread('P28.jpg')

#Guardando as dimensões da imagem original.
largura, altura, camadas = img1.shape

#Redimensionamento da imagem original.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation = cv2.INTER_AREA)

#Apresenta imagem original redimensionada na tela de trabalho.
cv2.imshow('ImagemP28_original', img1_redimensionada)

## Parte 2 - Visualização e registro da imagem corrigida.
#Carregamento da imagem original
img1 = cv2.imread('P28.jpg')
```

```
#Definindo a transformação.
M = np.float32([
        [   np.cos(np.pi*(82.631/180)),      np.sin(np.pi*(82.631/180)), 550],
        [  -np.cos(np.pi*(7.369/180)),      np.cos(np.pi*(82.631/180)), 3300]])

#Guardando as dimensões da imagem.
largura, altura, camadas = img1.shape

#Aplicando a transformação.
img1 = cv2.warpAffine(img1,M,(altura, largura))

#Salvar a imagem corrigida.
cv2.imwrite('ImagemP28_corrigida.jpg', img1)
#Redimensionamento da imagem corrigida.
largura = img1.shape[1]
altura = img1.shape[0]
proporcao = float(altura/largura)
largura_nova = 600
altura_nova = int(largura_nova*proporcao)
tamanho_novo = (largura_nova, altura_nova)
img1_redimensionada = cv2.resize(img1, tamanho_novo, interpolation =
cv2.INTER_AREA)

#Apresenta imagem corrigida redimensionada na tela de trabalho.
cv2.imshow('ImagemP28corrigida', img1_redimensionada)

#Evitar que a janela da imagem feche rapidamente e limpar a tela.
cv2.waitKey(0)
cv2.destroyAllWindows()
```