

# Princípios de Visão Computacional

## Projeto 1

Gabriel Martins de Miranda

130111350

Universidade de Brasília

Email:gabrielmirandat@hotmail.com

**Abstract**—Infelizmente, câmeras vem com distorção. Porém, existem constantes e com calibração e algum remapeamento podemos consertar isto. Através de estudos em Visão Computacional, hoje é possível determinar a relação entre as unidades naturais da câmera (pixels) e as unidades reais do mundo (por exemplo, milímetros). O presente artigo discute a calibração de câmera e alguns de seus usos, como em Homografia, que relaciona duas imagens na mesma superfície planar e como na estimativa das dimensões de um objeto em coordenadas do mundo (3d) através de sua representação em imagem (2d).

### I. INTRODUÇÃO

#### 1) Calibrando a camera

OpenCV nos oferece funções prontas para lidar com calibração da câmera através de um tabuleiro de xadrez, que está disponível nos próprios diretórios do software. Além do tabuleiro, também é possível usar padrões simétricos ou assimétricos de círculos. O objetivo principal da calibração é gerar os *parametros intrinsecos da camera*, ou a chamada *CameraMatrix*, os *parametros extrinsecos da camera*, que se referem à rotação ou translação da câmera ou do objeto e os chamados *coeficientes de distorção*, que relaciona fatores radiais e tangenciais. Quanto mais próximo de zero estiver o coeficiente de distorção, melhor. Temos que a *cameraMatrix* é única para determinada câmera e só precisa ser encontrada uma vez. Embora os *coeficientes de distorção* sejam os mesmos independente da resolução da câmera utilizada, eles devem ser escalados junto com a resolução atual da resolução calibrada (*podendo serem assim considerados também como parametros intrinsecos*), enquanto que os *parametros extrinsecos* mudam para cada imagem. Para o presente projeto, foi usada apenas uma câmera para obtenção das imagens, ou seja, não tínhamos uma visão estéreo. Devido a este fato, não era possível estimar a profundidade do ponto (*foram usadas profundidades conhecidas*). Os passos para se calibrar uma câmera são simples, podendo ser feito de várias maneras diferentes. O importante é variar bem a posição do tabuleiro para que todos os graus de liberdade da câmera sejam considerados, caso contrário não serão obtidos bons parâmetros. É possível, na mesma aplicação, calibrar a própria webcam ou uma outra câmera qualquer. No caso de outra câmera

qualquer, usamos uma lista de imagens como entrada. Para selecionar uma destas alternativas na aplicação, podemos apenas alterar o arquivo *xml* do diretório, chamado *default.xml*, usando como argumento ou o *ID do dispositivo*, no caso 0, que abre a webcam em tempo real, *< Input > "0" < /Input >*, ou um outro arquivo *xml*, chamado *imagens.xml*, que contém o *path* relativo para uma lista de imagens de entrada, *< Input > "imagens.xml" < /Input >*. Além destes parâmetros, é possível definir as dimensões do tabuleiro sendo utilizado, tais como o número de imagens usadas para a calibração, inicialmente 25 (*deve ser maior ou igual a 6 para considerar todos os graus de liberdade*) e o número de frames a serem pulados no caso de calibração em tempo real. Após a calibração, é gerado o *xml* *out\_camera\_data.xml*, que contém todas as informações da calibração, como a *CameraMatrix*, os *Distortion\_Coefficients* e os *Extrinsic\_Parameters*, sendo estes últimos uma matriz para cada imagem utilizada.

#### 2) Homografia

Quando queremos representar um objeto real do mundo (3D), no plano da imagem (2D), é evidente que nesta passagem temos perda de uma dimensão. Esta perda de dimensão se reflete na projeção *perspectiva*, onde linhas paralelas no objeto parecem se encontrar na imagem e círculos parecem não serem tão redondos assim.

Homografia planar é definida como um mapeamento projetivo de um plano (plano do objeto) para outro (plano da imagem). Duas imagens são relacionadas por uma homografia se e apenas se:

- a) Ambas as imagens estiverem “vendo” o mesmo plano de um ângulo diferente. (*nossa tarefa*).
- b) Ambas as imagens forem tiradas da mesma câmera, mas de um ângulo diferente. (*camera rotacionada de seu centro de projecao sem qualquer translacao*).

É importante notar que a relação de homografia é independente da estrutura da cena, ou seja, não depende do que a câmera está olhando e este relacionamento

mantém-se independentemente do que é visto nas imagens.

Usando coordenadas homogêneas a relação entre os dois planos pode ser expresso por:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = C \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

- $\tilde{u}/\tilde{w}$  e  $\tilde{v}/\tilde{w}$  representa o ponto em pixels na imagem.
- $C$  inclui duas partes: a transformação física que localiza o plano do objeto e a projeção que tem os parâmetros intrínsecos da câmera.

### 3) Estimando altura e largura de um objeto dada uma profundidade conhecida

Diante de tudo o que foi explicitado, podemos obter a seguinte relação:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Ou simplesmente:

$$s \cdot m' = A \cdot [R|t] \cdot M'$$

- $s$  é o fator de escala.
- $m'$  são as coordenadas do ponto projetado em pixels.
- $A$  é a *CameraMatriz*, ou a matriz de parâmetros intrínsecos.
- $[R|t]$  é a *PoseMatrix*, ou matriz de parâmetros extrínsecos. Ela é a "concatenação" da *matriz de rotacao* ( $R$ ) com a *matriz de translacao*  $t$ .
- $M'$  são as coordenadas 3D do ponto expressas em coordenadas do mundo.

Multiplicando as matrizes  $A$  com  $[R|t]$  obtemos um nova matriz 3x4, que carrega informação tanto dos parâmetros intrínsecos quanto dos extrínsecos, e chamaremos ela de  $C$ .

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Supondo que o plano no mundo se encontra onde  $Z = 0$  em nosso sistema de coordenadas, podemos remover toda a coluna composta dos valores ( $c_{13}, c_{23}, c_{33}$ ), ficando com a seguinte composição de matrizes:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{14} \\ c_{21} & c_{22} & c_{24} \\ c_{31} & c_{32} & c_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix}$$

Rearranjando a matriz  $C$ , obtemos uma nova matriz  $H$ , o qual chamaremos de *matriz de homografia*, uma matriz 3x3.

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & c_{13} \\ h_{21} & h_{22} & c_{23} \\ h_{31} & h_{32} & c_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

A *matriz de homografia*, assim como a matriz  $C$ , carrega informações tanto dos parâmetros intrínsecos como de parâmetros extrínsecos, porém de uma forma mais compacta. Ela será base para a realização do projeto.

## II. METODOLOGIA

Para compilar o programa, foi utilizado o *cmake*, que pode ser baixado com *sudo apt -get install cmake*. Para compilar e executar, basta seguir os seguintes passos no terminal:

- *cmake .* (certifique-se de usar a tecla de espaço antes do ponto).
- *make*
- *./pvc\_130111350\_1*

A *main* aceita dois argumentos de entrada, que devem ser o *path* para duas imagens. Se estiver sendo usado o fluxo da webcam, então estes dois parâmetros são opcionais e se utilizados, a etapa de captura das imagens para a homografia na *Parte 2* a seguir não será usada.

### 1) Parte 1

- Foi definido um contêiner para lidar com as variáveis que serão salvas no arquivo de saída *xml* e também para lidar com o tipo de abordagem que será usada para a calibragem.
- Se o fluxo em tempo real for usado, será pedido ao usuário que posicione o tabuleiro de xadrez em frente à câmera e inicie o processo de captura de snapshots. Para isto é necessário que se pressione *g*.
- Ao final do processo de captura, se tudo tiver ocorrido bem, será possível observar a imagem que a própria câmera vê sem a distorção que o próprio computador faz. Pressionando *u* é possível ver a imagem corrigida utilizando os parâmetros encontrados.
- Caso seja especificada uma lista de imagens como entrada, especificando no próprio *default.xml* o arquivo *imagens.xml*, a câmera será calibrada sem o fluxo em tempo real. Neste caso, como a webcam não abrirá para realizar o *Parte 2*, deve ser mandado para a *main* o *path* de duas imagens para serem usadas no processo da homografia. Pressionando *u* pode-se ver as imagens da lista de imagens sem distorção. Caso seja usada a lista de imagens e não forem especificadas as imagens para se usar na homografia, a *Parte 2* será pulada.

### 2) Parte 2

- Ainda no caso de ter sido usado o fluxo da webcam, após a *Parte 1*, o usuário precisa pressionar *i* duas vezes para obter as imagens que serão usadas no cálculo da homografia e projeção de uma na outra e, logo em seguida, apertar *esc* e fechar a janela da parte 1. Irá aparecer o primeiro snapshot tirado. Assim, escolhemos quaisquer 4 pontos nesta imagem onde queremos projetar a segunda imagem. A escolha dos pontos é feita clicando com o mouse. Após obtidos 4 pontos, que serão mostradas as coordenadas no terminal, deve-se clicar em um ponto arbitrário dentro da janela para que seja gerada uma terceira imagem, já com os planos projetados.
- O mesmo vale caso sejam utilizadas as imagens passadas como parâmetro na *main*.

### 3) Parte 3

- Clicando para fechar as 3 janelas da segunda parte, tem início um novo fluxo de vídeo da webcam. Lá tentará se localizar um objeto para medir suas dimensões reais em milímetros. É importante esclarecer que para se obter uma boa localização o objeto deve ser escuro e se possível sem muito reflexo.
- Sempre que um objeto for localizado, um retângulo o encobrirá e serão mostradas tanto o seu tamanho em pixels quanto em milímetros em relação às coordenadas do mundo.
- Também é mostrado qual o objeto está sendo avaliado e qual a distância considerada. Para mudar estas configurações o usuário deve pressionar *i*. Pressionando *i* até avaliar o objeto 2 na distância 550, (*a ultima*), o programa é encerrado.
- Para encontrar o ponto em sua coordenadas no mundo real foram realizados os seguintes cálculos:
  - As coordenadas em 2d foram homogeneizadas.
  - foi usada a função *getOptimalNewCameraMatrix*, onde são mandados como parâmetros a matriz de parâmetros intrínsecos e os coeficientes de distorção, não será considerado fatores extrínsecos como rotação e translação.
  - Após obtida a nova *CameraMatrix*, esta com os coeficientes de distorção já embutidos, realizamos a seguinte equação:

*world\_coord=new\_camera\_matrix<sup>-1</sup>\*pixel\_coord*  
onde

- *world\_coord* são as coordenadas do ponto no mundo real (medida em mm).
- *new\_camera\_matrix<sup>-1</sup>* é a inversa da nova matriz de parâmetros intrínsecos que já contém os coeficientes de distorção embutidos.
- *pixel\_coord* são as coordenadas do ponto em

pixels (*no plano da imagem*)

- d) Escolhemos, então, 3 dos quatro pontos do *retangulo* que encobre o objeto em questão e encontramos suas representações em coordenadas do mundo.
- e) Obtida a *world\_coord*, multiplicamo-a por uma das distâncias pré-determinadas, no caso foram consideradas as distâncias 350, 450 e 550 milímetros, estas podendo ser mudadas no vetor *dist* que se encontra no início da *main*. As distâncias servem como um fator de escala para nossa equação.
- f) O último passo então resume-se a encontrar a distância entre estes 3 pontos, 2 a 2, que representam a *altura* e a *largura* do nosso objeto.

## III. RESULTADOS

Algumas saídas gráficas das etapas do programa:

Calibração

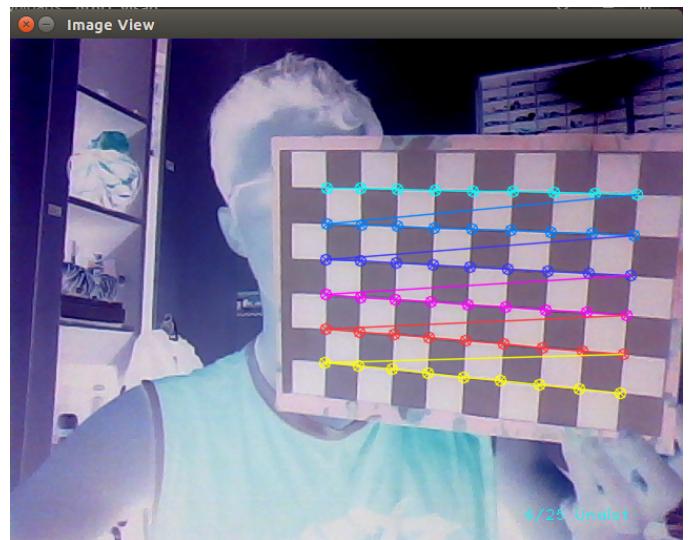


Fig. 1: Capturando frame para a calibração.

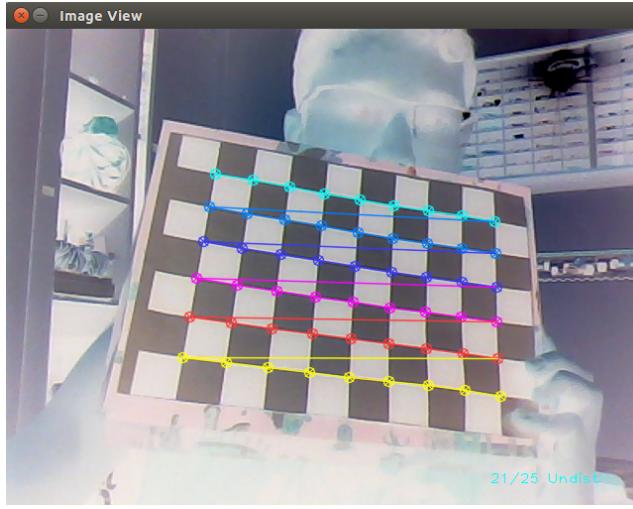


Fig. 2: Capturando em angulações diferentes.



Fig. 5: Depois da calibração, visão da câmera. Parâmetros ruins.

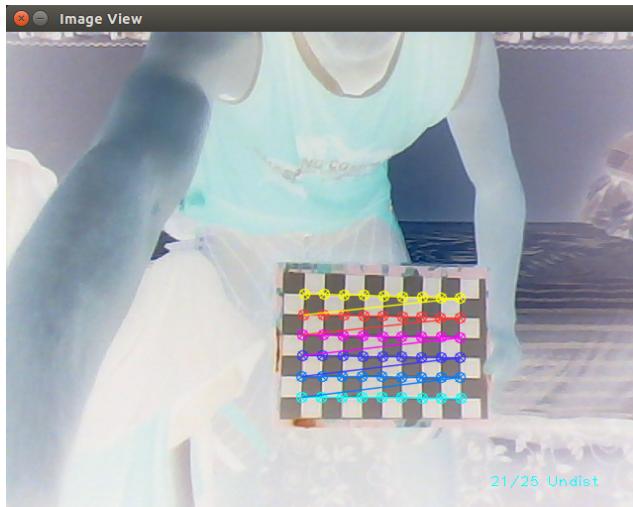


Fig. 3: Considerando snapshots longínquos.



Fig. 6: Depois da calibração, visão da câmera. Pouca variação dos ângulos e maioria com o tabuleiro longe da webcam.



Fig. 4: Depois da calibração, visão da câmera. Bons parâmetros



Fig. 7: Imagem corrigida com os parâmetros da calibração.



Fig. 8: Imagem capturada para a homografia. Imagem fonte.



Fig. 9: Resultado da projeção perspectiva entre os planos.

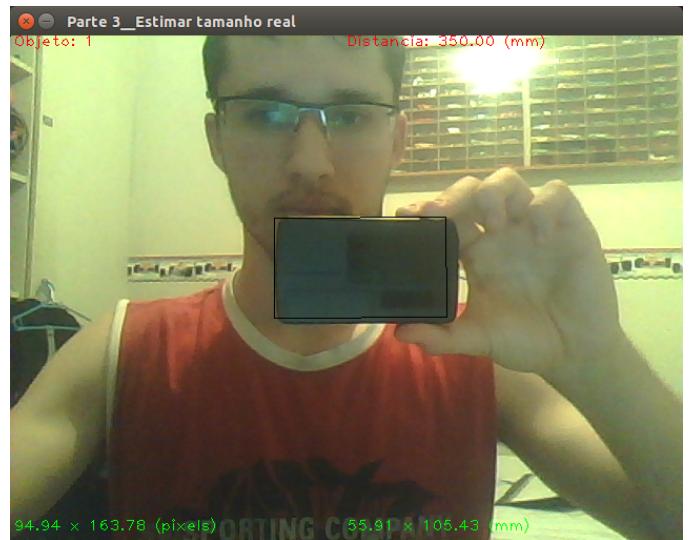


Fig. 10: Estimadas as medidas do objeto 1 a 350mm. Vale ressaltar que o objeto possui 57x102 mm.



Fig. 11: Estimadas as medidas do objeto 2 a 350mm. Vale ressaltar que o objeto possui 63x154 mm.



Fig. 12: Estimadas as medidas do objeto 1 a 450mm.

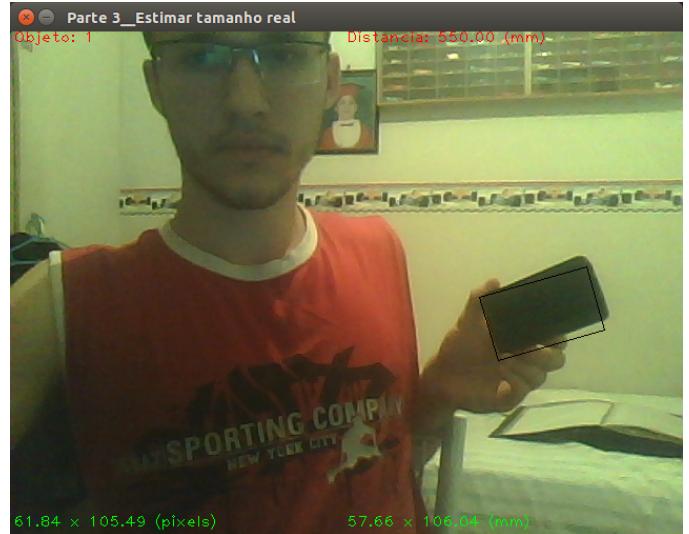


Fig. 14: Estimadas as medidas do objeto 1 a 550mm.

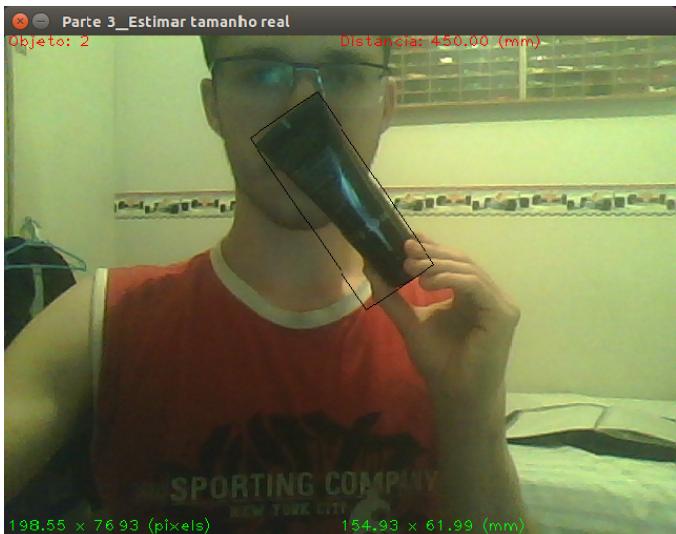


Fig. 13: Estimadas as medidas do objeto 2 a 450mm.

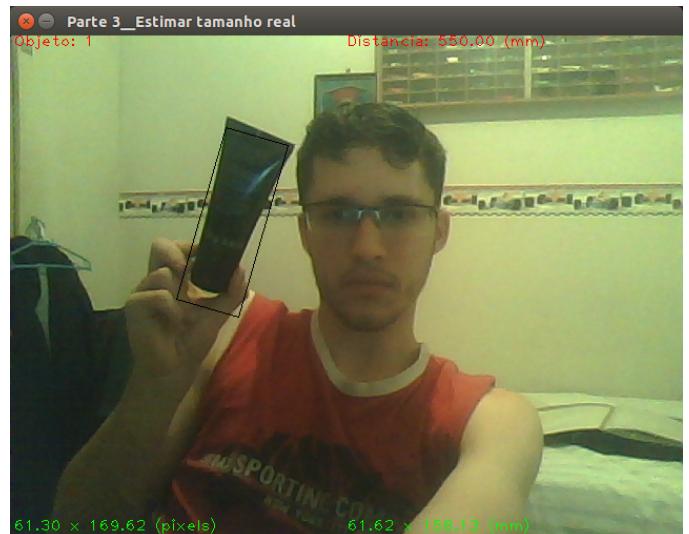


Fig. 15: Estimadas as medidas do objeto 2 a 550mm. Apesar de no índice estar escrito que se trata do objeto 1, isto não faz diferença, o que importa é a distância.

#### IV. CONCLUSÃO

Foi possível observar a importância da calibração de uma câmera, que dada uma imagem, pode nos dar um visão ampla com projeções e movimentação de uma "câmera virtual" na cena. Foi visto também que é possível medir a localização real dos pontos de objetos dada apenas uma imagem e os parâmetros da câmera que a tirou, sendo possível até determinar a distância que se encontrava da câmera (*no caso de se possuir uma visao estereo*). Pudemos concluir que duas imagens podem estar relacionadas por uma homografia, e com isto é possível projetar uma no plano da outra.

## V. REFERÊNCIAS

- [1][http://docs.opencv.org/doc/tutorials/calib3d/camera\\_calibration/camera\\_calibration.html#cameracalibrationonopencv](http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html#cameracalibrationonopencv)
- [2][http://docs.opencv.org/modules/calib3d/doc/camera\\_calibration\\_and\\_3d\\_reconstruction.html#void%20Rodrigues%28InputArray%20src,%20OutputArray%20dst,%20OutputArray%20jacobian%29](http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#void%20Rodrigues%28InputArray%20src,%20OutputArray%20dst,%20OutputArray%20jacobian%29)
- [3][http://docs.opencv.org/doc/tutorials/calib3d/camera\\_calibration\\_square\\_chess/camera\\_calibration\\_square\\_chess.html#cameracalibrationsquarechessboardtutorial](http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration_square_chess/camera_calibration_square_chess.html#cameracalibrationsquarechessboardtutorial)
- [4][http://en.wikipedia.org/wiki/Homography\\_%28computer\\_vision%29](http://en.wikipedia.org/wiki/Homography_%28computer_vision%29)
- [5]<http://stackoverflow.com/questions/7836134/get-3d-coord-from-2d-image-pixel-if-we-know-extrinsic-and-intrinsic-parameters?rq=1>
- [6]<https://www.youtube.com/watch?v=bcszwZLwhTUI>
- [7]<https://www.youtube.com/watch?v=R9mvvylyUY0>
- [8][http://docs.opencv.org/doc/tutorials/imgproc/shapedescriptors/bounding\\_rotated\\_ellipses/bounding\\_rotated\\_ellipses.html#bounding-rotated-ellipses](http://docs.opencv.org/doc/tutorials/imgproc/shapedescriptors/bounding_rotated_ellipses/bounding_rotated_ellipses.html#bounding-rotated-ellipses)
- [9][http://docs.opencv.org/doc/tutorials/core/file\\_input\\_output\\_with\\_xml\\_yml/file\\_input\\_output\\_with\\_xml\\_yml.html#fileinputoutputxmlyaml](http://docs.opencv.org/doc/tutorials/core/file_input_output_with_xml_yml/file_input_output_with_xml_yml.html#fileinputoutputxmlyaml)
- [10][http://docs.opencv.org/doc/tutorials/features2d/feature\\_homography/feature\\_homography.html#feature-homography](http://docs.opencv.org/doc/tutorials/features2d/feature_homography/feature_homography.html#feature-homography)
- [11]<http://stackoverflow.com/questions/13419605/how-to-map-x-y-pixel-to-world-coordinates>