Universidade Federal do Paraná

Especialização em Inteligência Artificial Aplicada

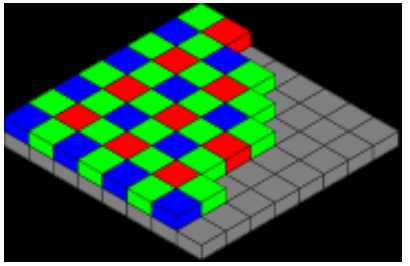
Mobile Robotics

Visão computacional

Prof. Eduardo Todt

2019

Sensores para visão



Sensores

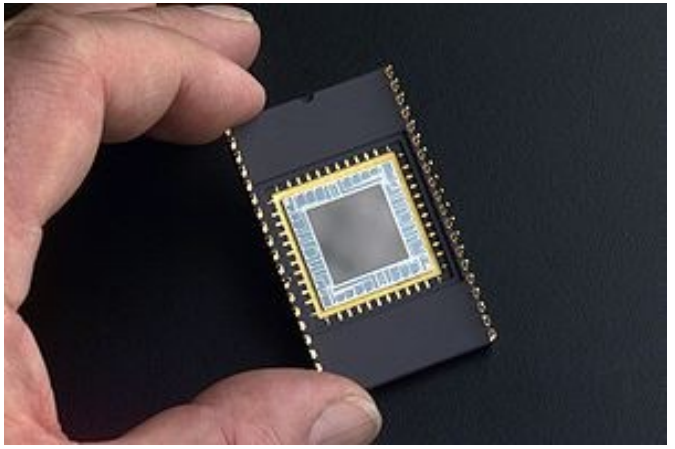
- CCD
  - Array de sensores, carga em capacitores transferida entre pixels, leitura realizada em um extremo
  - Maior sensibilidade próximo a IR
  - Maior faixa dinâmica (ex: 40000:11 → 3600:1 → 35dB)
- CMOS
  - Transistores junto a cada pixel permitem leitura em paralelo dos pixels
  - Menor resolução e menor sensibilidade devido à presença dos transistores adicionais
  - Processo de fabricação mais simples

Imagem em cores

- Um CCD com filtros RGB em pixels vixzinhos
- Três CCDs, cada um com filtro global RGB

Problemas

- Sensibilidades distintas para RGB, azul menor
- White-balance
- Color constancy



http://pt.wikipedia.org/wiki/Dispositivo_de_carga_acoplada

Visão computacional

Inteligência artificial

Robótica

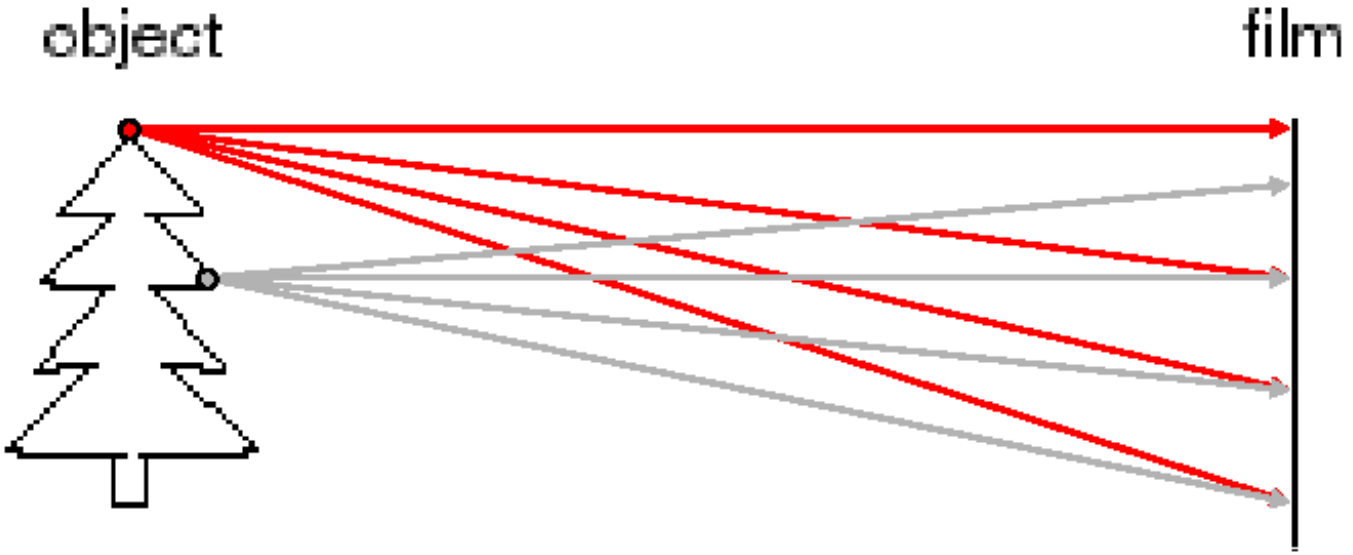
Computação gráfica

Processamento de imagens

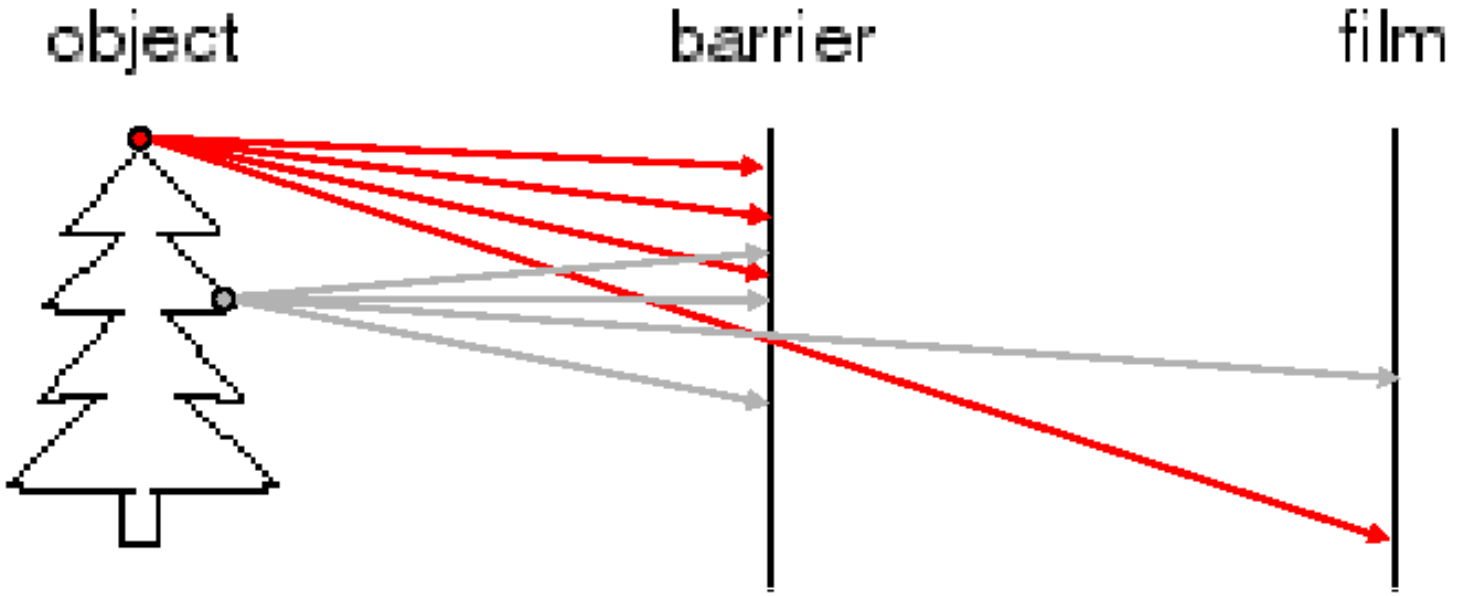
Psicologia

Aprendizagem computacional

Construindo uma câmera



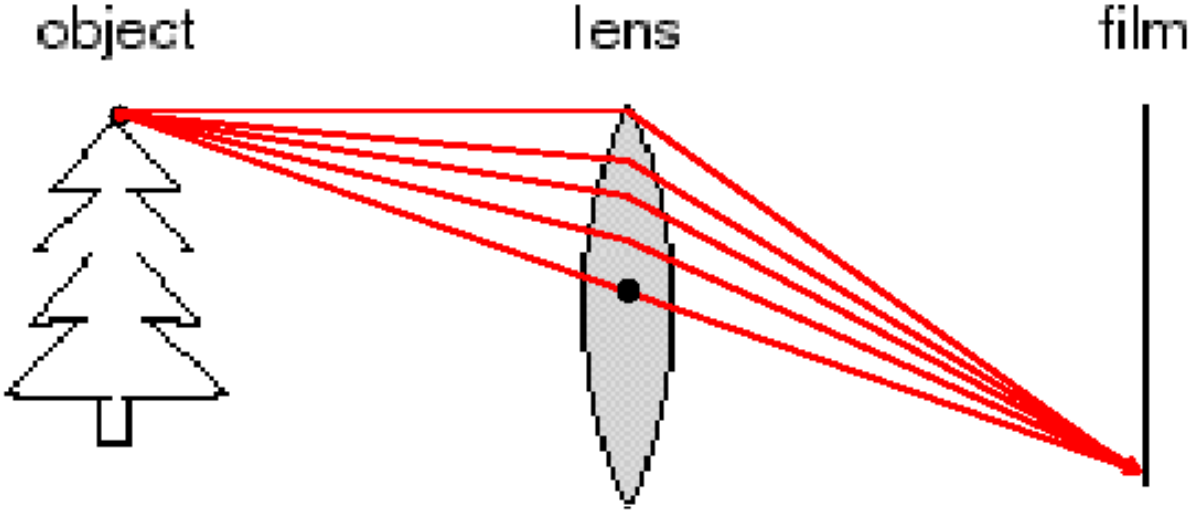
Construindo uma câmera



Adicionando lente

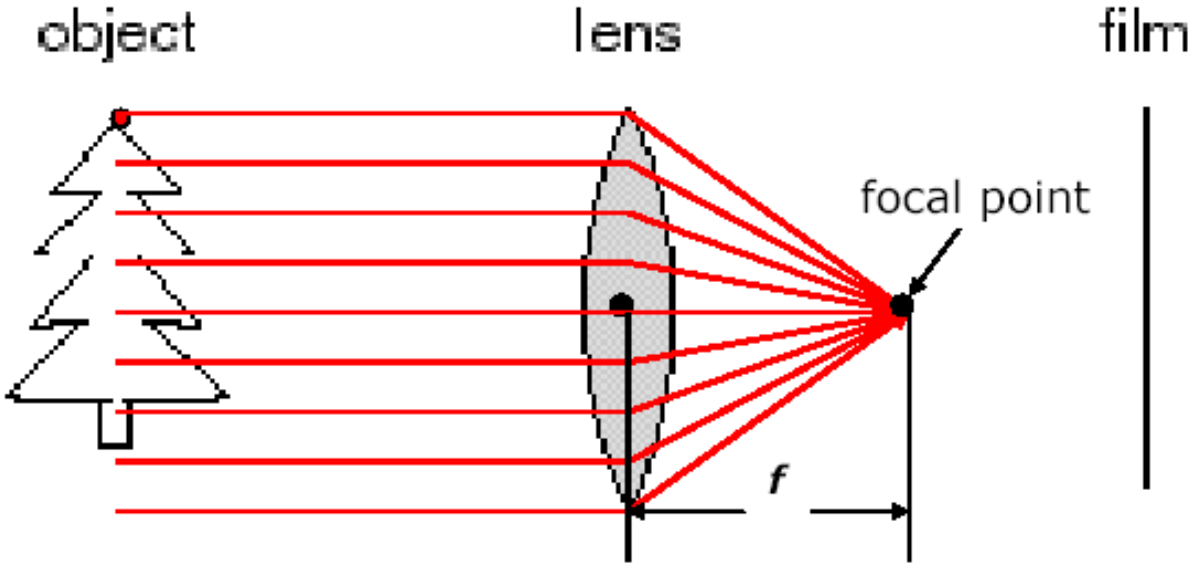
Lente foca os raios no plano de projeção

Raios que passam pelo centro não são desviados



Adicionando lente

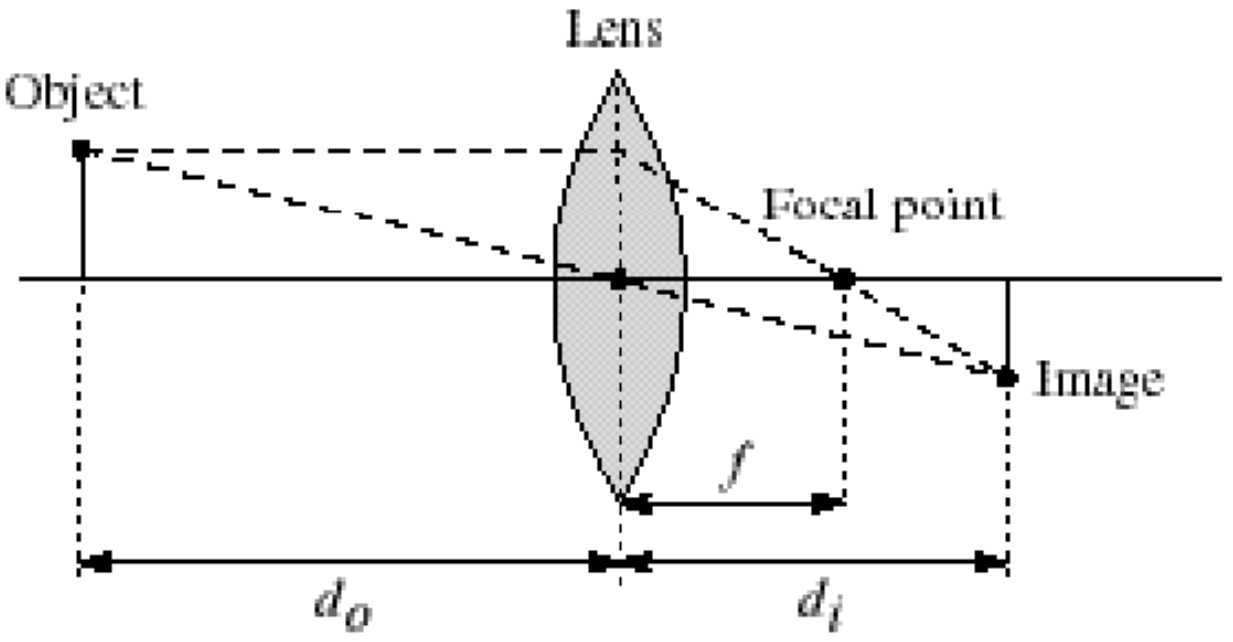
Todos raios paralelos convergem para um ponto, localizado à distância focal *f*



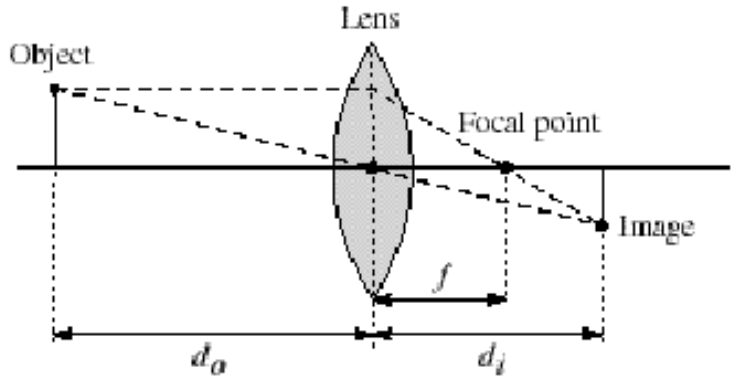
Equação de lente convergente

Pontos satisfazendo equação estão em foco

$$\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f}$$



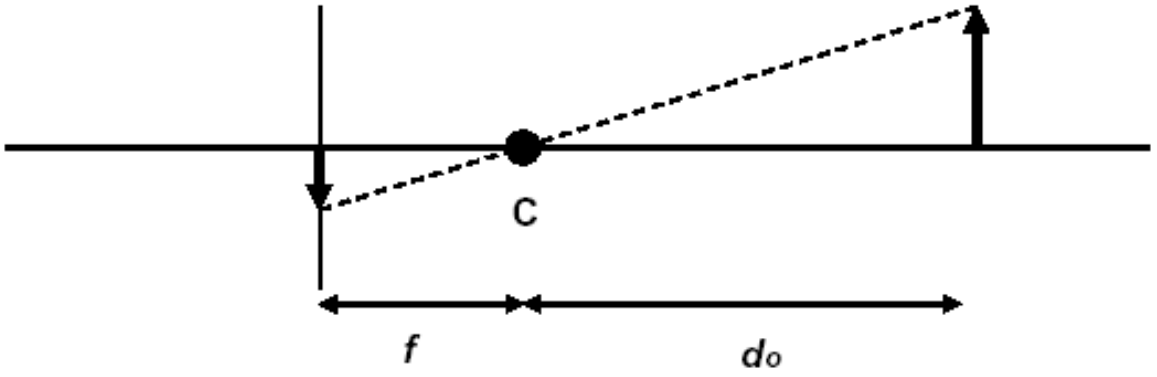
Aproximação pin-hole



$$\frac{1}{f} = \frac{1}{d_i} + \frac{1}{d_o}$$

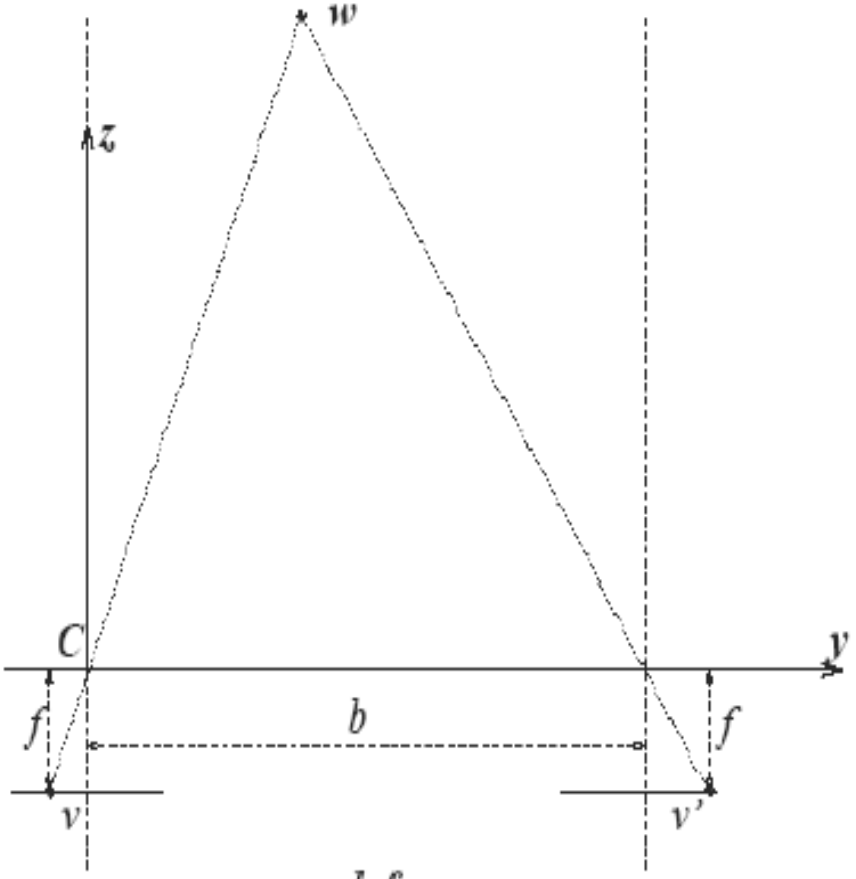
$$d_o >> d_i$$

$$\frac{1}{f} \approx \frac{1}{d_i} \Rightarrow d_i \approx f$$

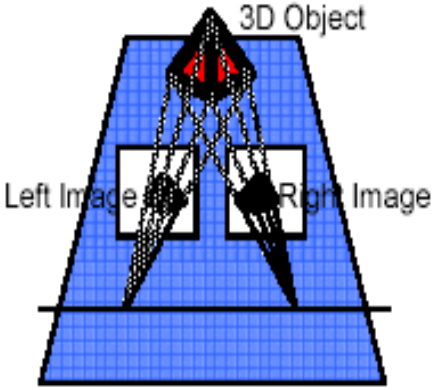


Visão estéreo

$$\begin{cases} \dfrac{-f}{z} = \dfrac{-v}{y} \\ \dfrac{-f}{z} = \dfrac{v'}{b-y} \end{cases}$$



$$z = \frac{bf}{v - v'}$$ Distance

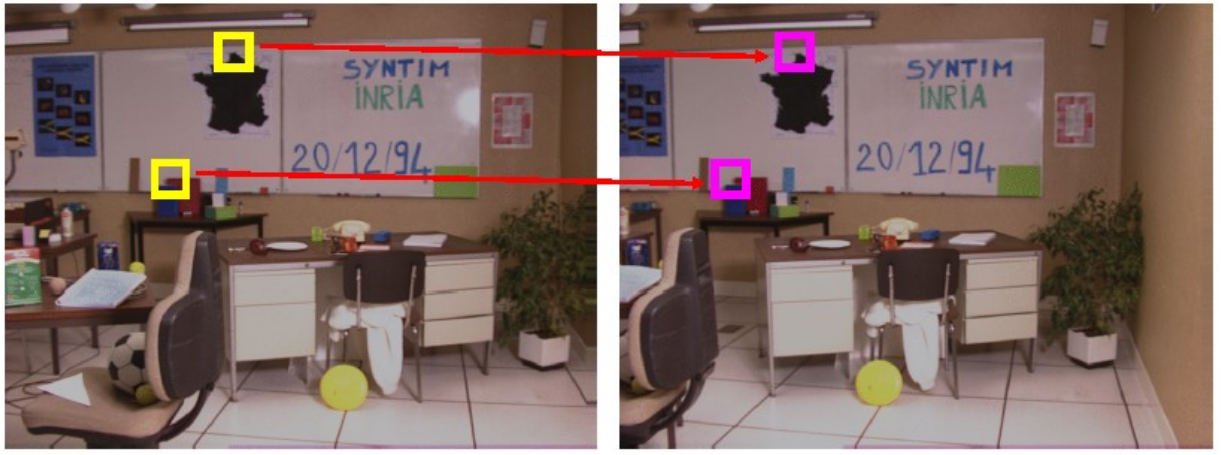


O problema da correspondência

Matching de pontos nas duas imagens que correspondem ao mesmo ponto no mundo 3D

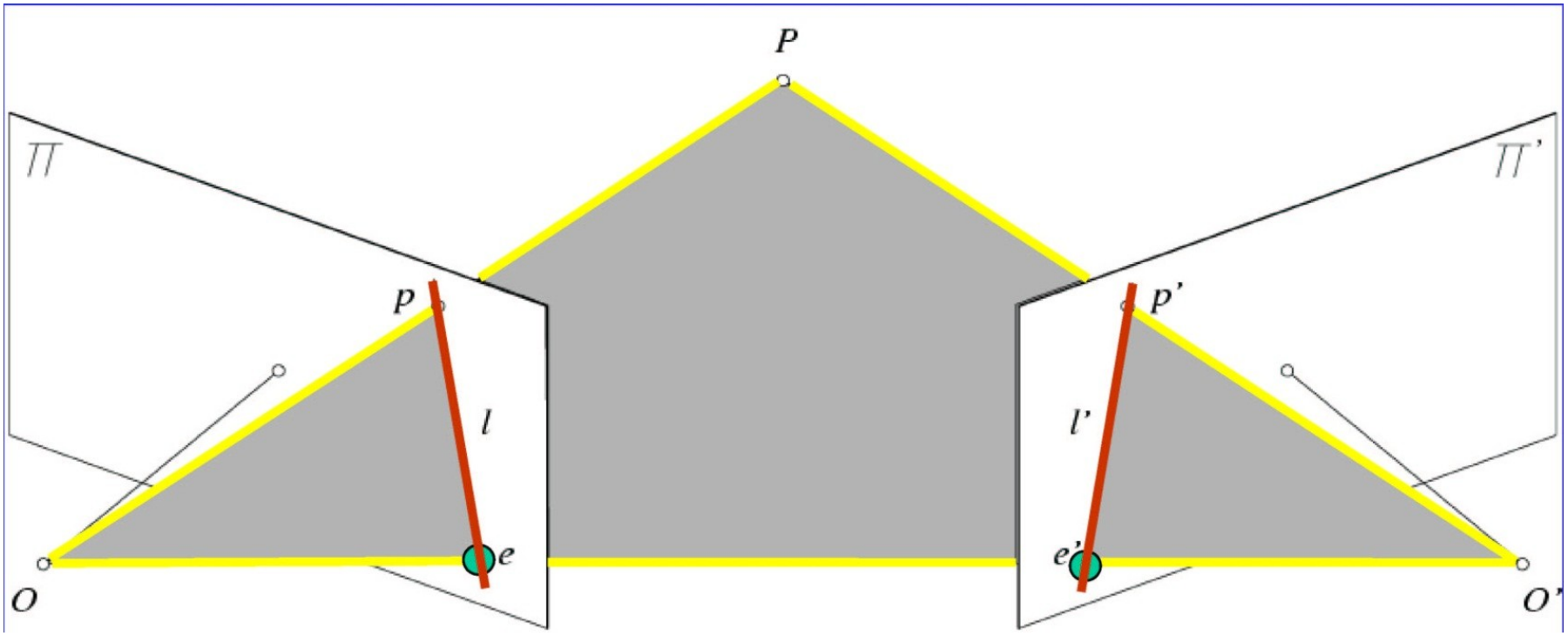
Critérios: correlação e diferença

Custo muito alto para explorar toda imagem



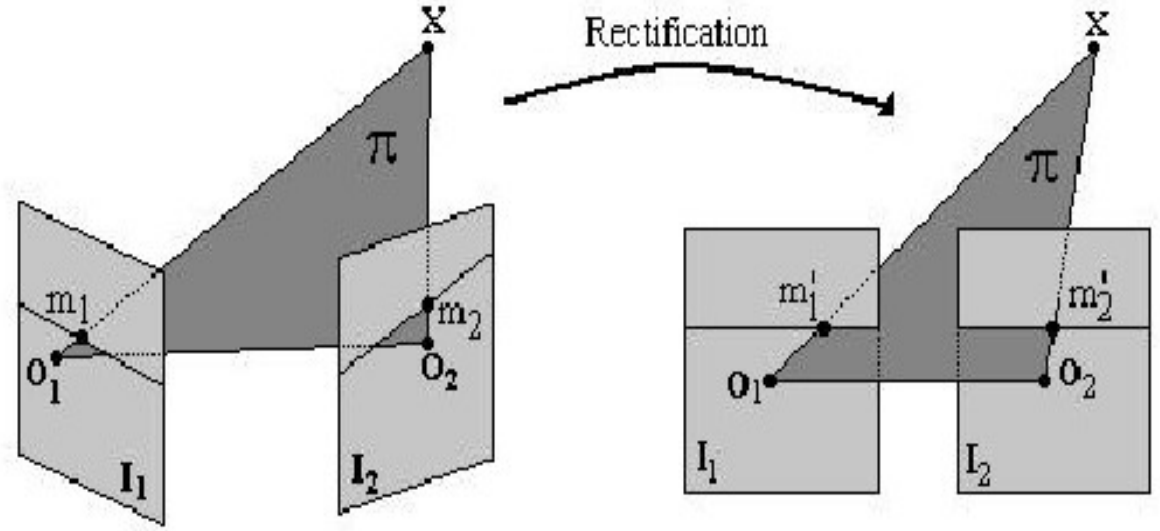
Geometria epipolar

Restringe a correspondência a uma linha



Retificação de imagem

Transformação de cada plano de imagem para que pares conjugados de linhas epipolares fiquem colineares e paralelas a um eixo da imagem



F Moreno 2001

O problema da correspondência

Exemplo de correspondência com geometria epipolar

Mapa de disparidade

1. Achar os pontos conjugados
2. Calcular a disparidade

d=v-v’

3. Objetos mais próximos são representados mais claros

Left image

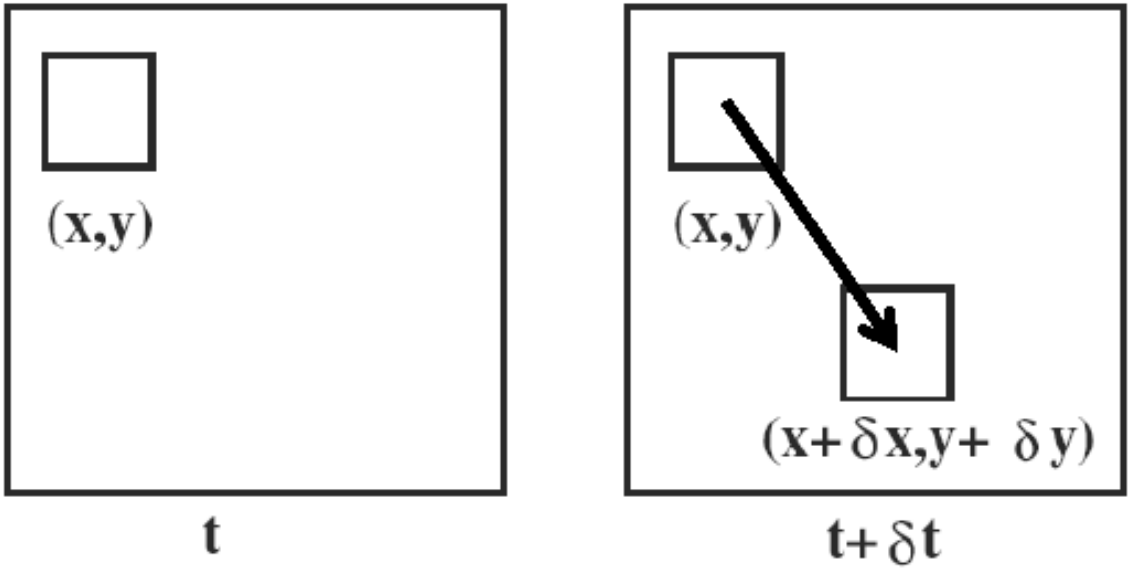
Right image

Disparity map

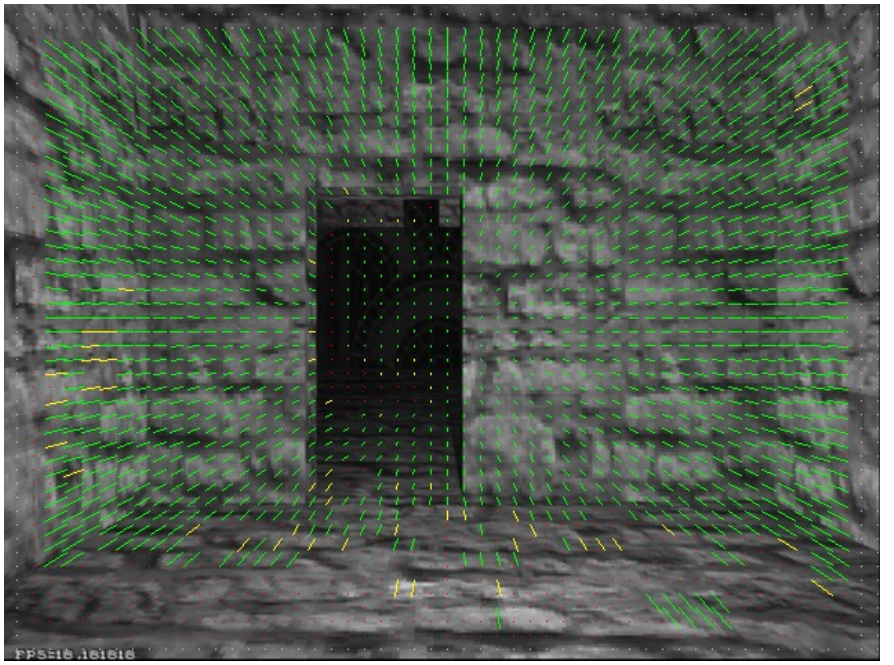
Fluxo óptico

Aproximação do movimento aparente dos objetos em uma imagem

Correlação entre frames sucessivos



Fluxo óptico



http://people.csail.mit.edu/lpk/mars/temizer_2001/Optical_Flow/

Calibração de Câmeras

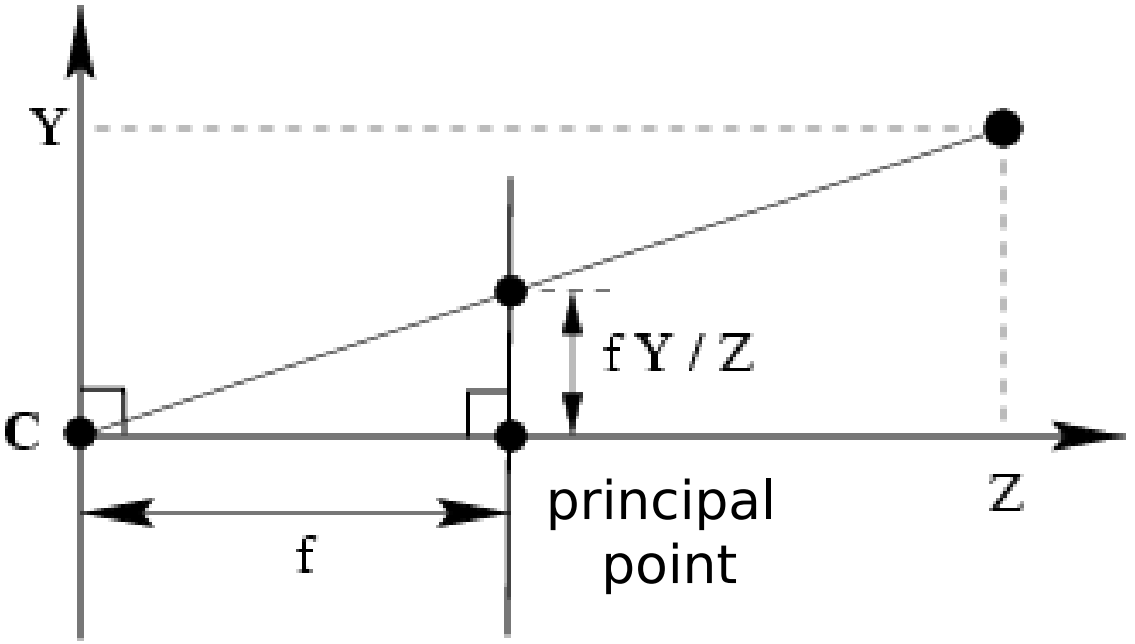
parâmetros intrínsecos

- geometria da câmera
- características óticas

parâmetros extrínsecos

- posição e orientação do frame da câmera relativo ao mundo

Pinhole camera model

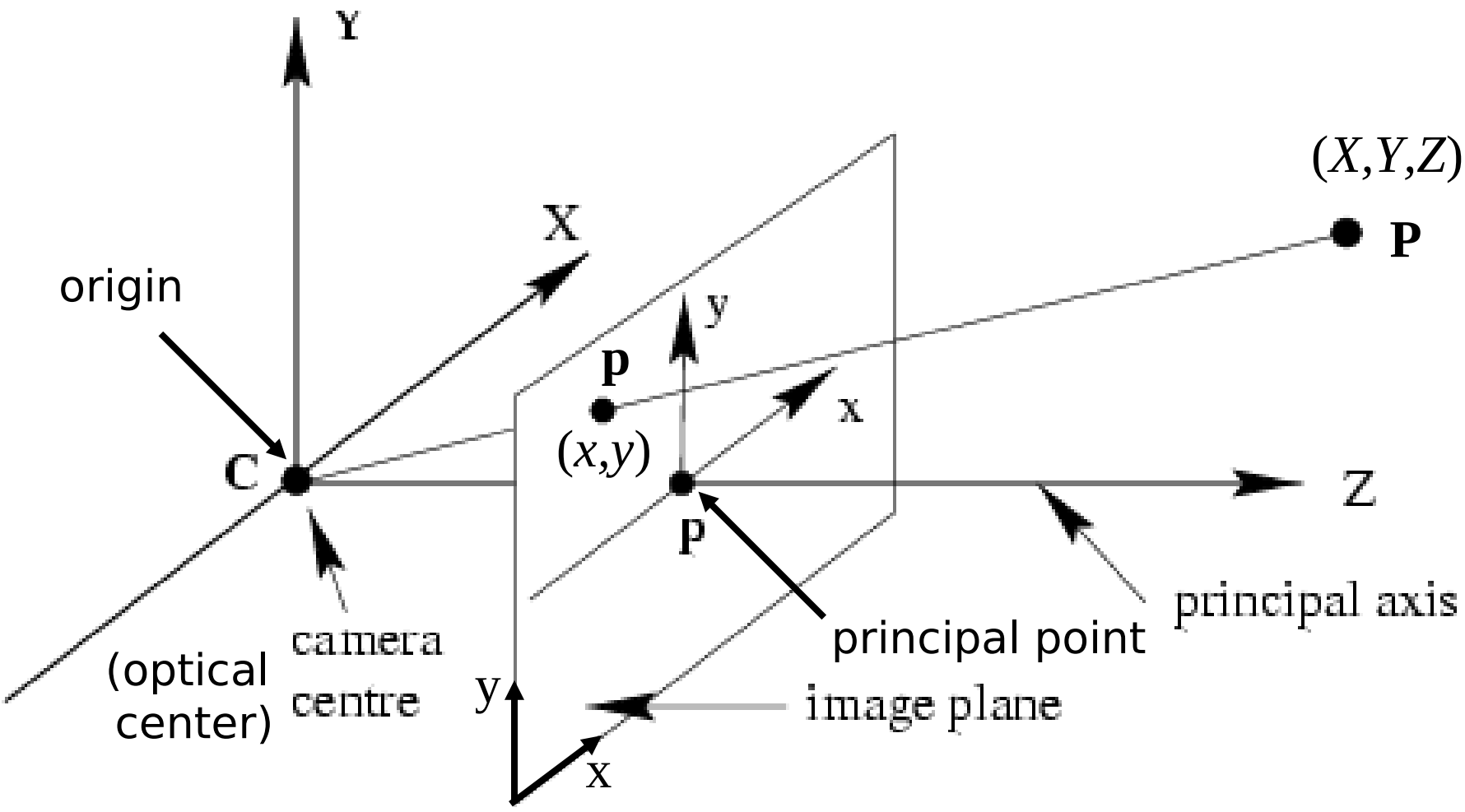


$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Pinhole camera model



$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Intrinsic matrix

Is this form of K good enough?

$$K = \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

- non-square pixels (digital video)

$$K = \begin{bmatrix} f_x & 0 & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

Intrinsic matrix

Is this form of K good enough?

$$K = \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

- non-square pixels (digital video)
- skew

$$K = \begin{bmatrix} f_x & s & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

Intrinsic matrix

Is this form of K good enough?

$$K = \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

- non-square pixels (digital video)
- skew
- radial distortion

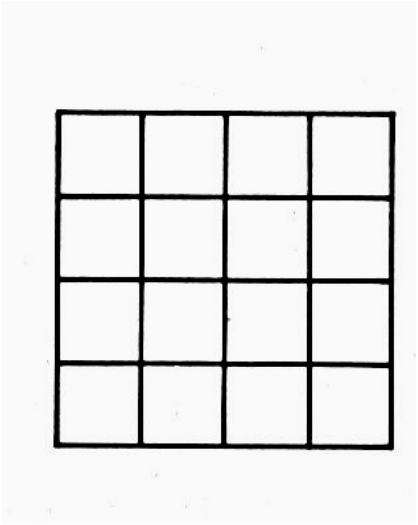
$$K = \begin{bmatrix} f_x & s & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

Distortion

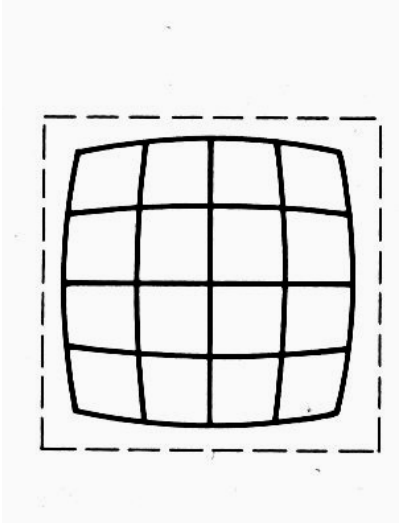
Radial distortion of the image

- Caused by imperfect lenses
- Deviations are most noticeable for rays that pass through the edge of the lens

Barrel Distortion



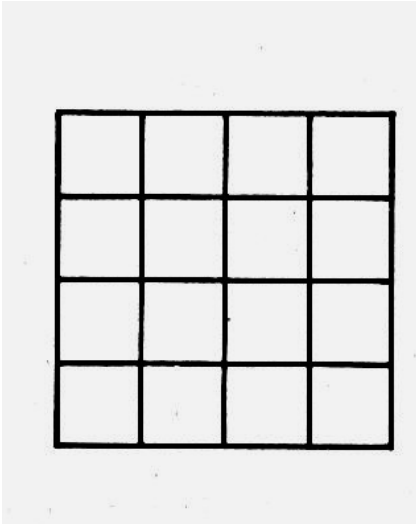
No distortion



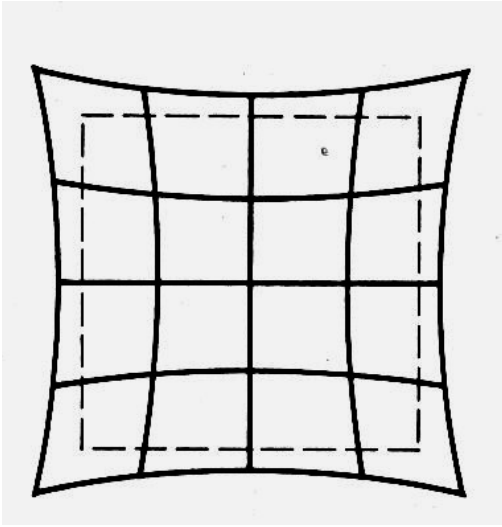
Barrel

Wide Angle Lens

Pin Cushion Distortion



No distortion



Pin cushion

Telephoto lens

Modeling distortion

Distortion-Free:

$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

With Distortion:

1. Project (X, Y, Z) to "normalized" image coordinates

$$x_n = \frac{X}{Z}$$

$$y_n = \frac{Y}{Z}$$

$$r^2 = x_n^2 + y_n^2$$

2. Apply radial distortion

$$x_d = x_n\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)$$

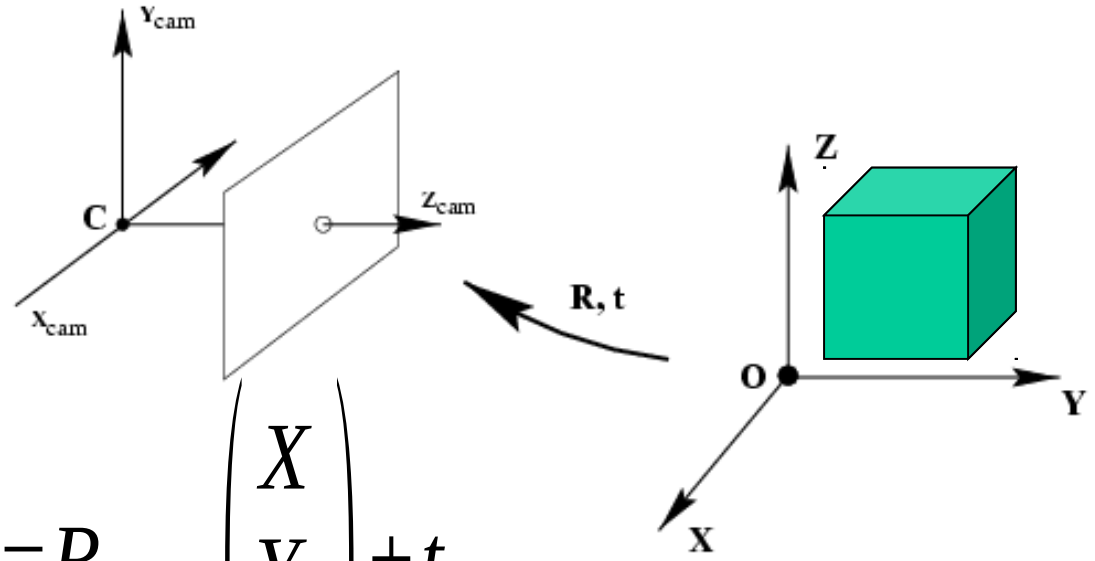
$$y_d = y_n\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)$$

3. Apply focal length translate image center

$$x = f x_d + x_c$$

$$y = f y_d + y_c$$

Camera rotation and translation



$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = R_{3\times 3} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + t$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R|t \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$x \sim K \underbrace{\begin{bmatrix} R|t \end{bmatrix}}_{\text{extrinsic matrix}} X$$

Two kinds of parameters

internal or *intrinsic* parameters: focal length, optical center, skew

external or *extrinsic* (pose): rotation and translation:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R|t \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Camera calibration

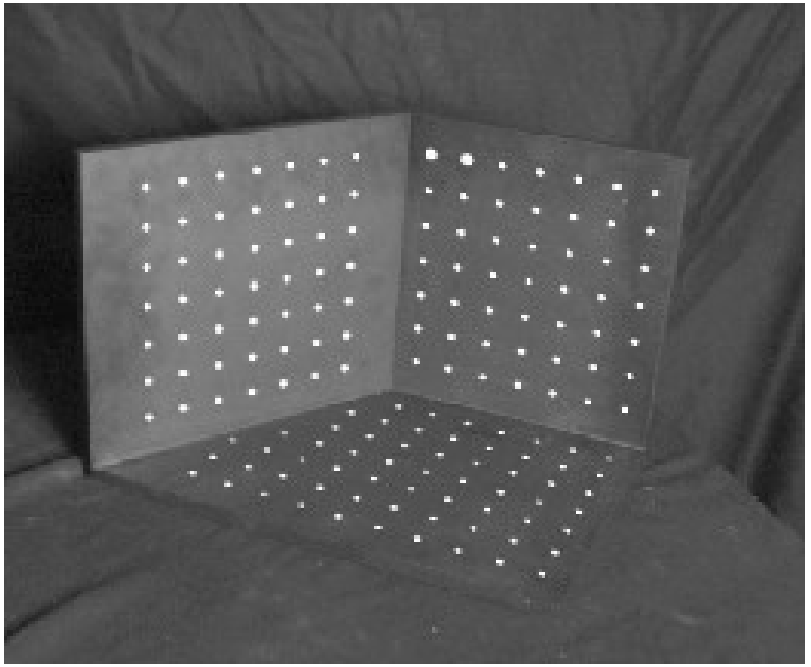
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} [R|t] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Estimate both intrinsic and extrinsic parameters

Mainly, two categories:

- Using objects with known geometry as reference
- Self calibration (structure from motion)

Camera calibration approaches

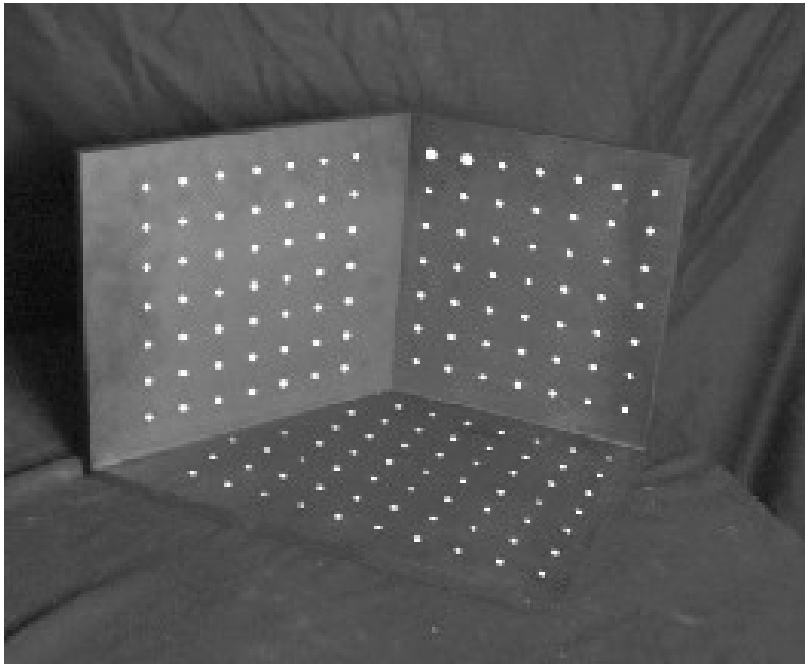


Directly estimate 11 unknowns in the **M** matrix using known 3D points (*Xi*,*Yi*,*Zi*) and measured feature positions (*ui*,*vi*)

$$x \sim K\left[R \middle| t\right] X = \mathbf{M}\mathbf{X}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Calibration



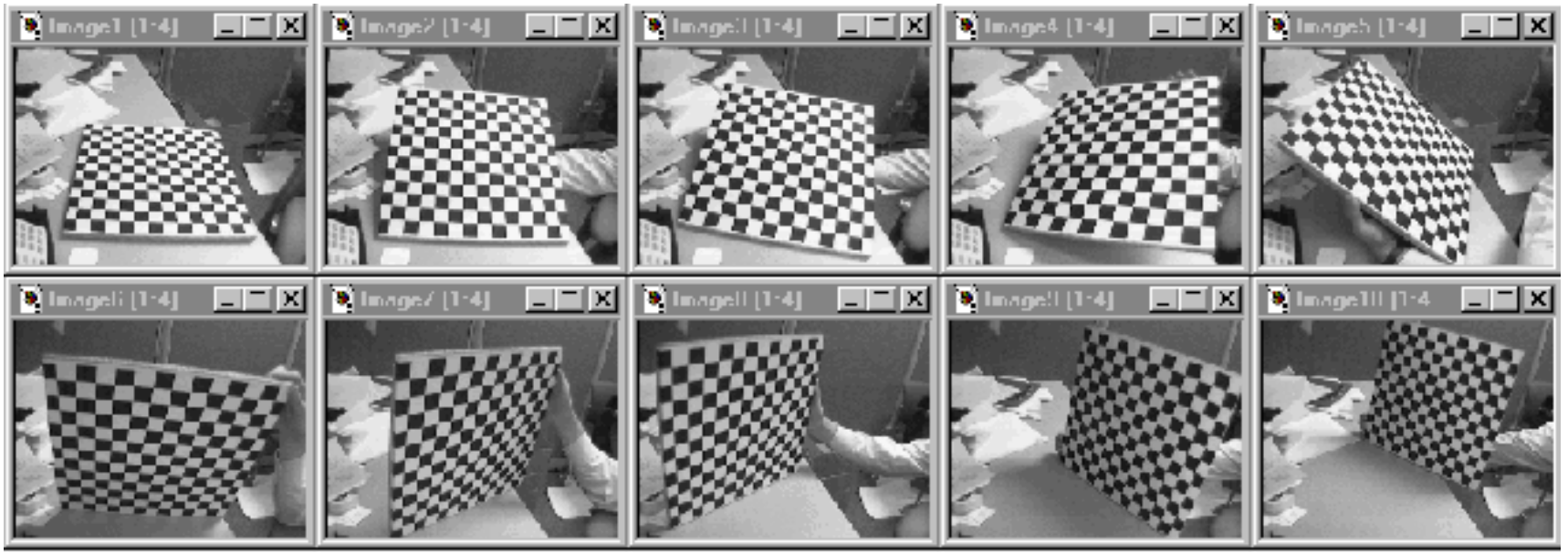
Directly estimate 11 unknowns in the **M** matrix using known 3D points (*Xi*,*Yi*,*Zi*) and measured feature positions (*ui*,*vi*)

$$x \sim K \left[R \middle| t \right] X = \mathbf{M}\mathbf{X}$$

NonLinear Approach:

$$\sum_{i=1}^{N} \left(u_i - \frac{m_{00} X_i + m_{01} Y_i + m_{02} Z_i + m_{03}}{m_{20} X_i + m_{21} Y_i + m_{22} Z_i + 1} \right)^2 + \left(v_i - \frac{m_{10} X_i + m_{11} Y_i + m_{12} Z_i + m_{13}}{m_{20} X_i + m_{21} Y_i + m_{22} Z_i + 1} \right)^2$$

Multi-plane calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

- Only requires a plane
- Don’t have to know positions/orientations
- Good code available online!
  - Intel’s OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Zhengyou Zhang’s web site: http://research.microsoft.com/~zhang/Calib/

Linear regression

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

Linear regression

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

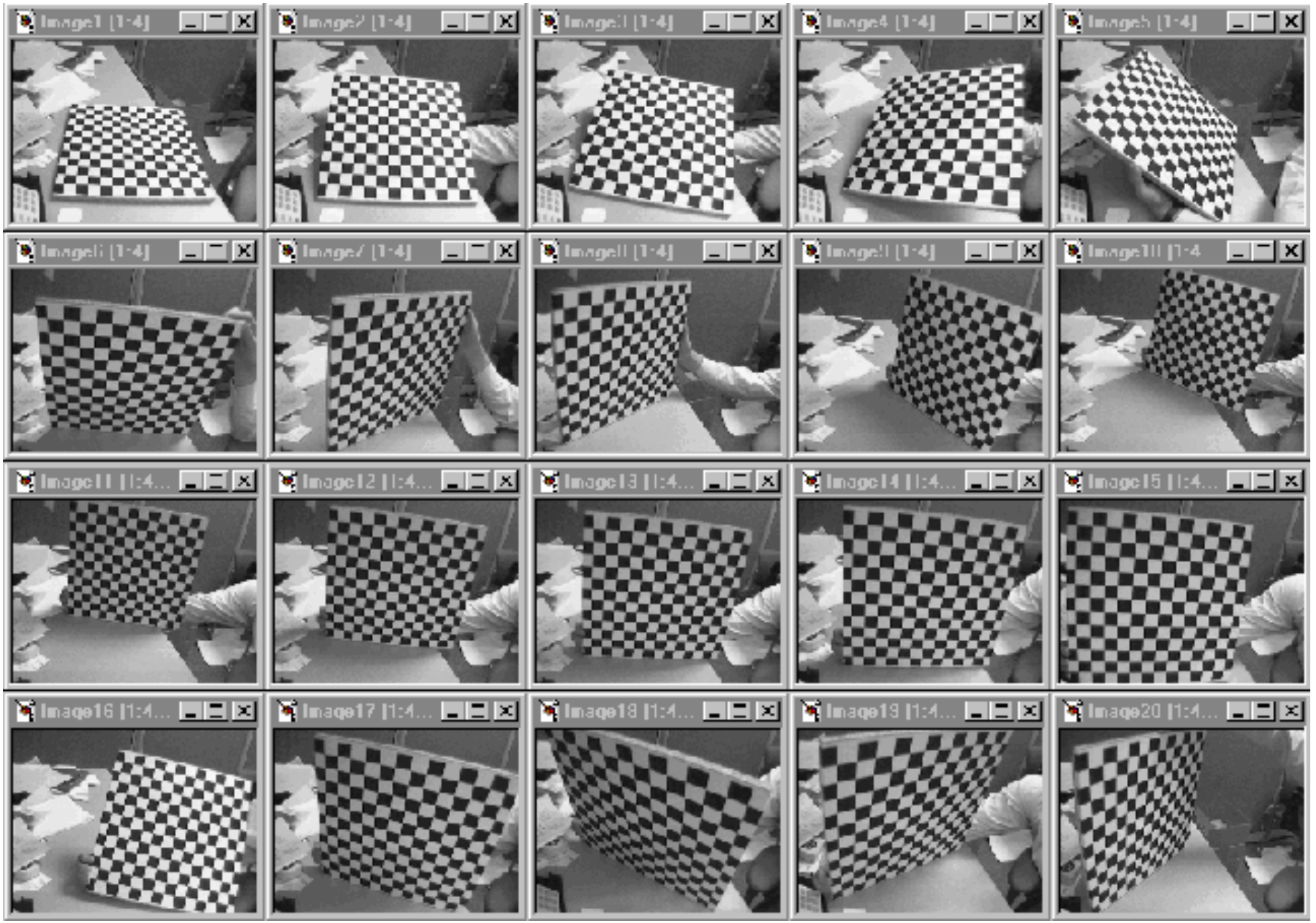
$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iY_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

Linear regression

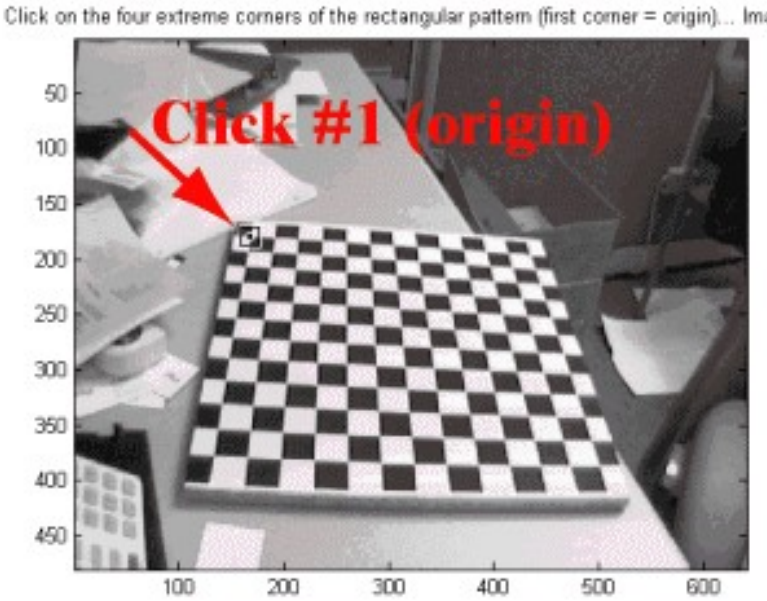
$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Y_1 \\ & & & & & & \vdots & & & & \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -u_NX_N & -u_NY_N & -u_NZ_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -v_NX_N & -v_NY_N & -v_NY_N \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

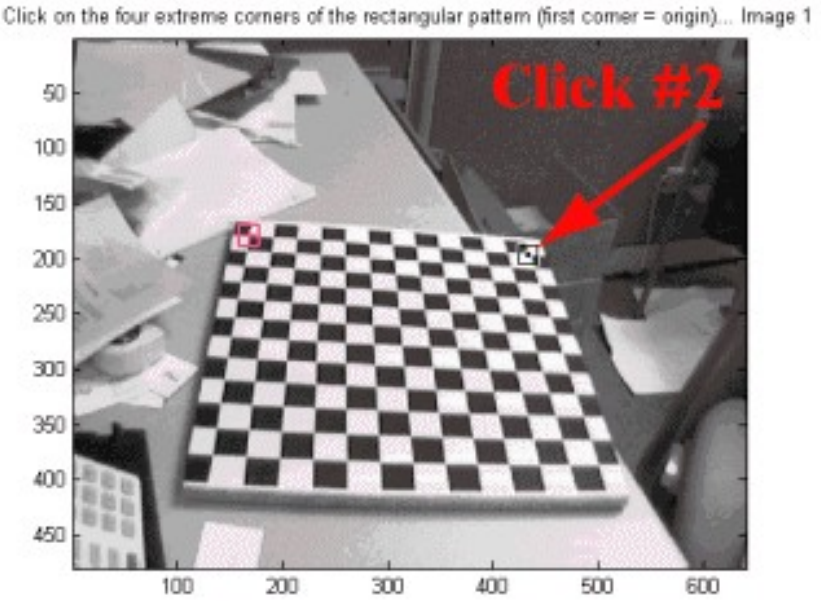
Solve for Projection Matrix M using least-square techniques

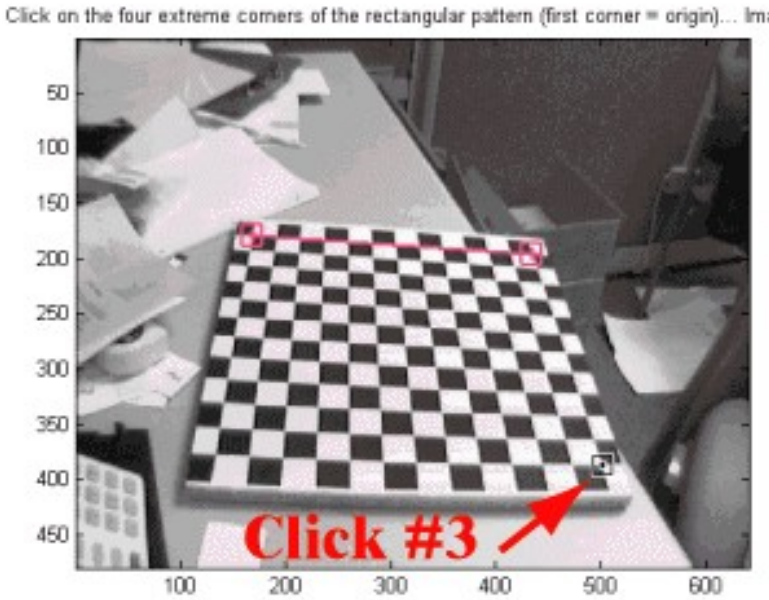
Step 1: data acquisition

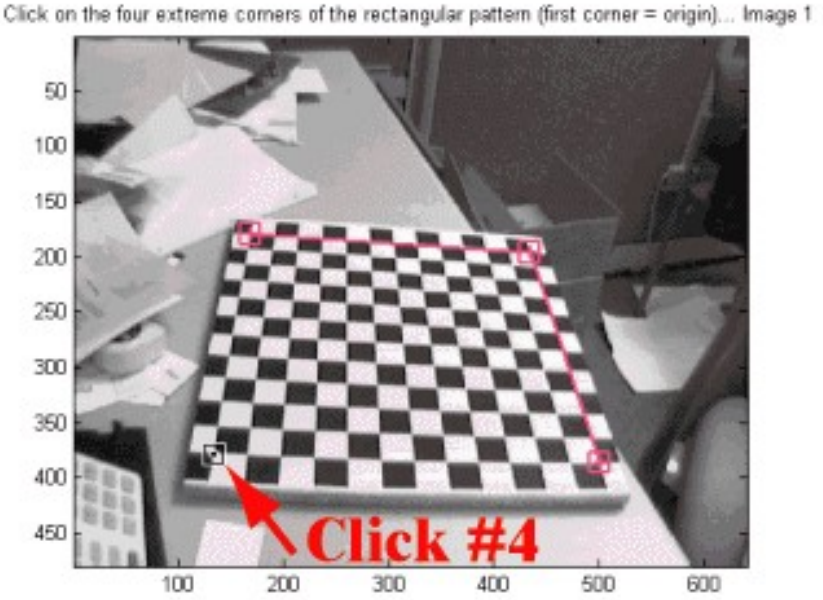


Step 2: specify corner order

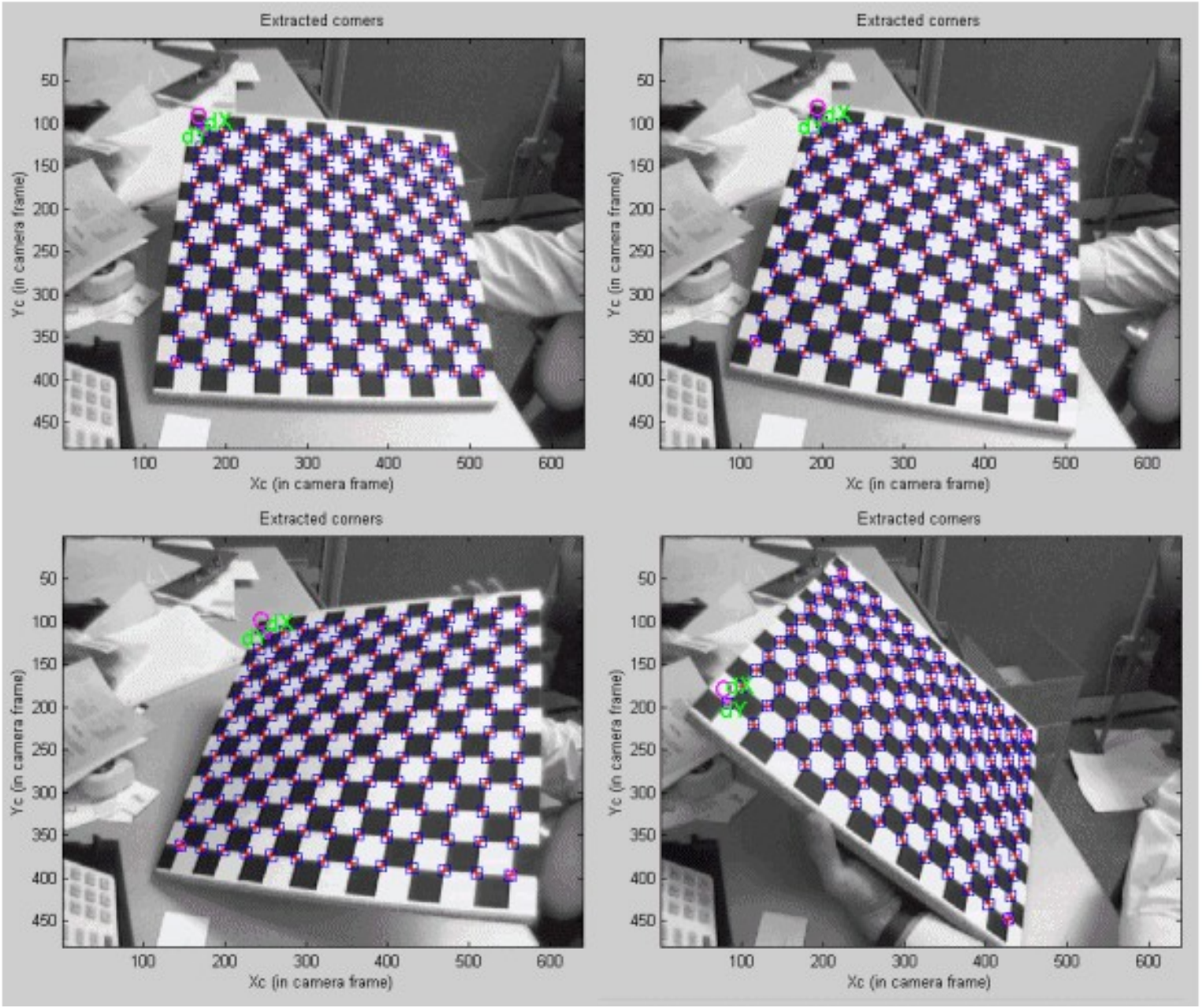




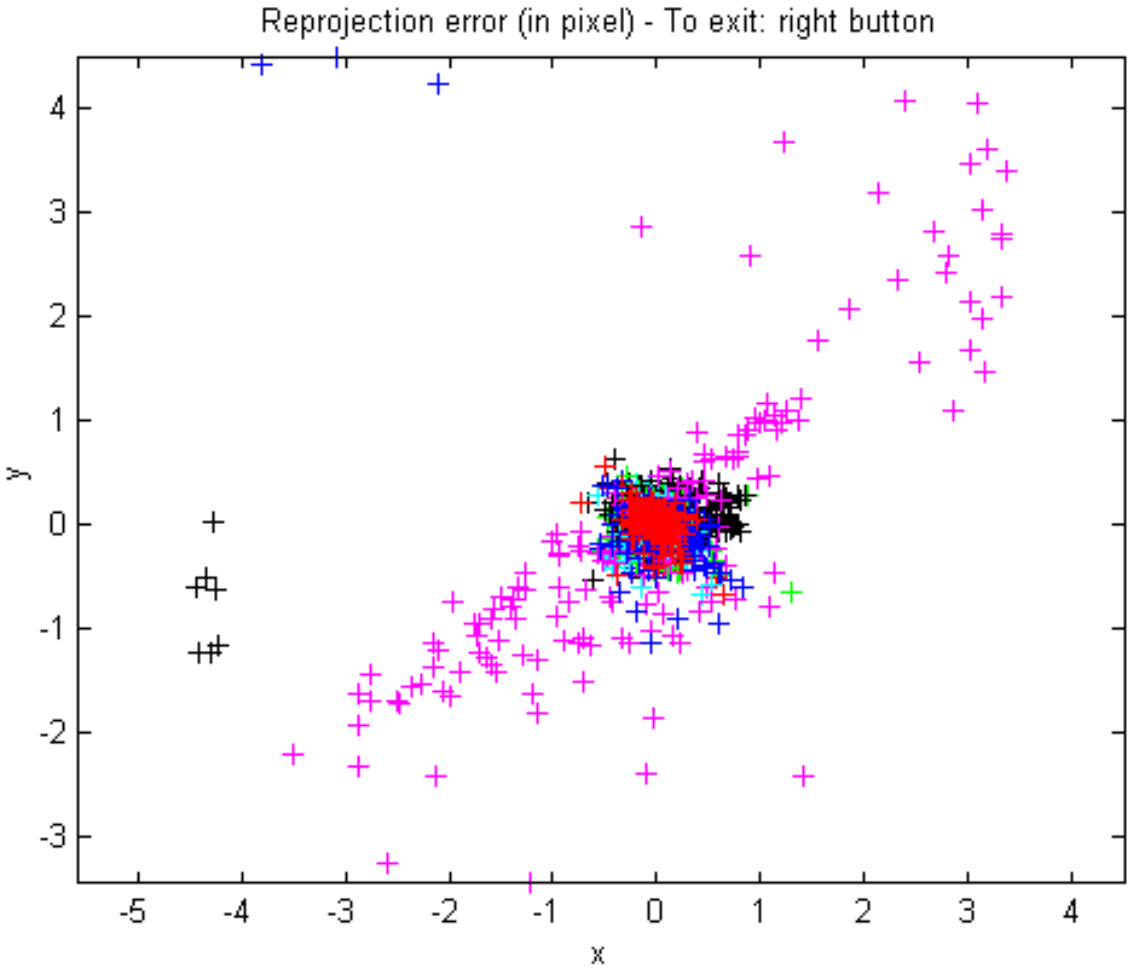




Step 3: corner extraction

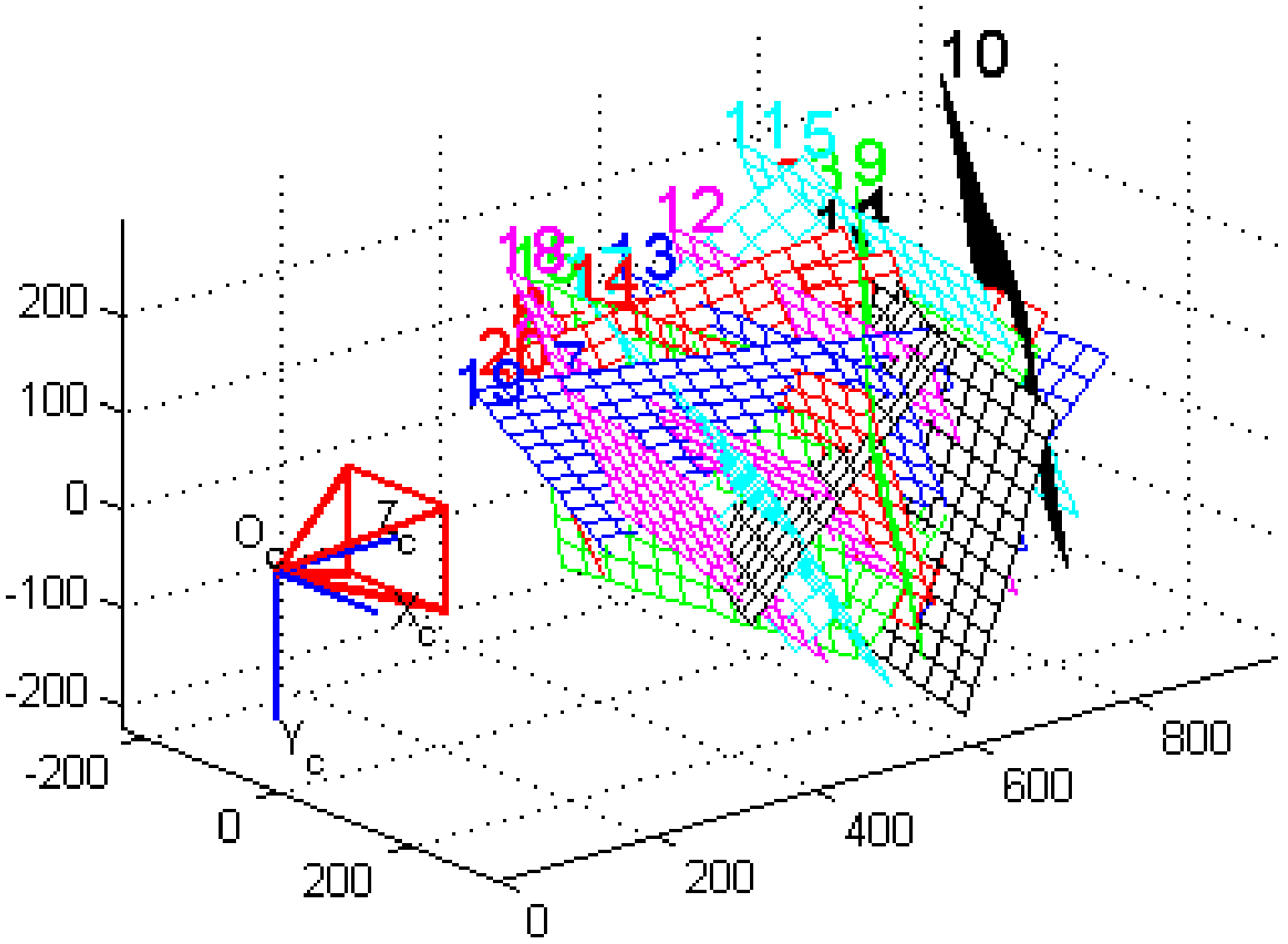


Step 4: minimize projection error

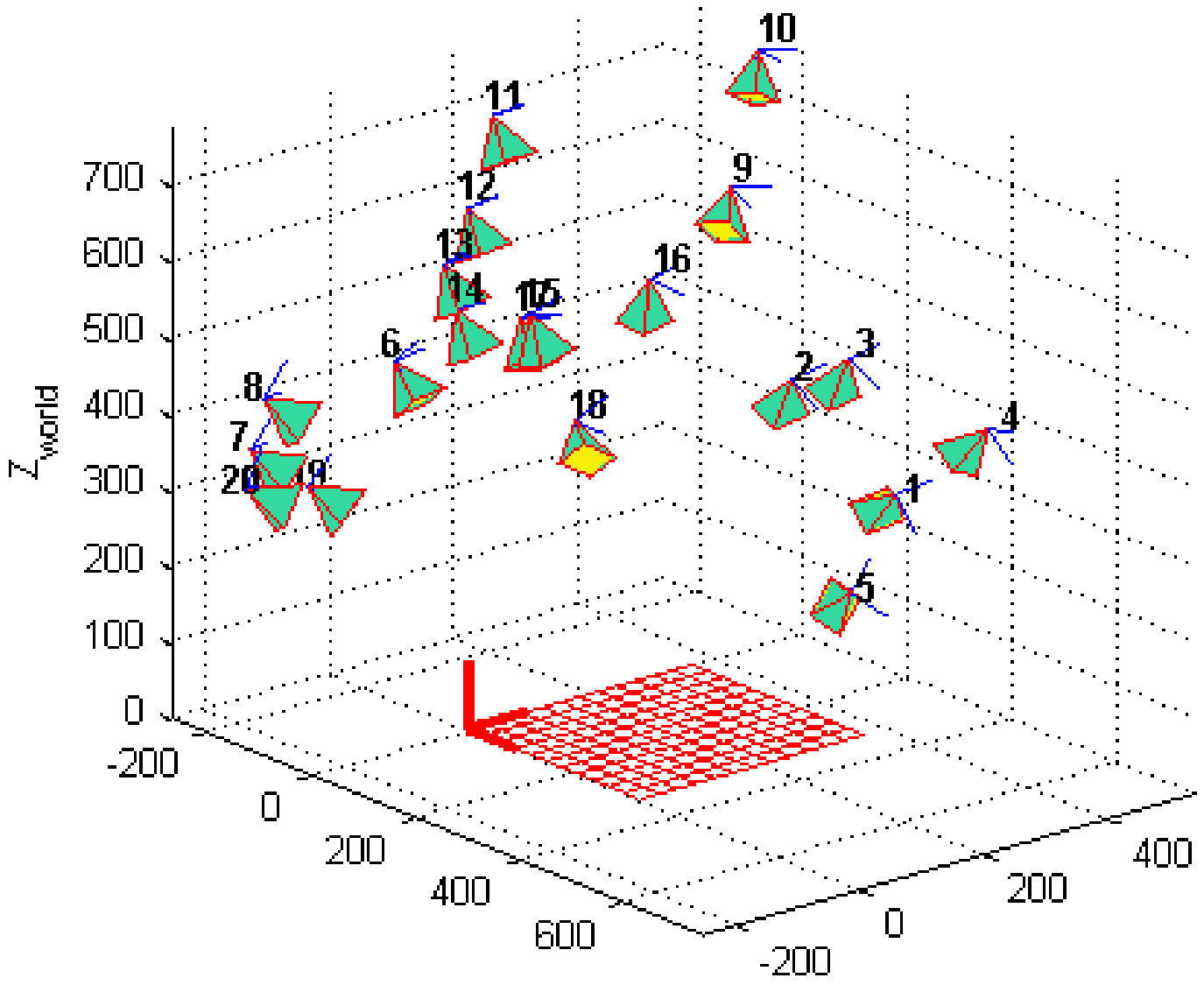



```
Calibration res

Focal Length:          fc = [ 657.46290   657.94673 ] ± [ 0.31819   0.34046 ]
Principal point:       cc = [ 303.13665   242.56935 ] ± [ 0.64682   0.59218 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
Distortion:            kc = [ -0.25403   0.12143   -0.00021   0.00002  0.00000 ]
Pixel error:          err = [ 0.11689   0.11500 ]
```

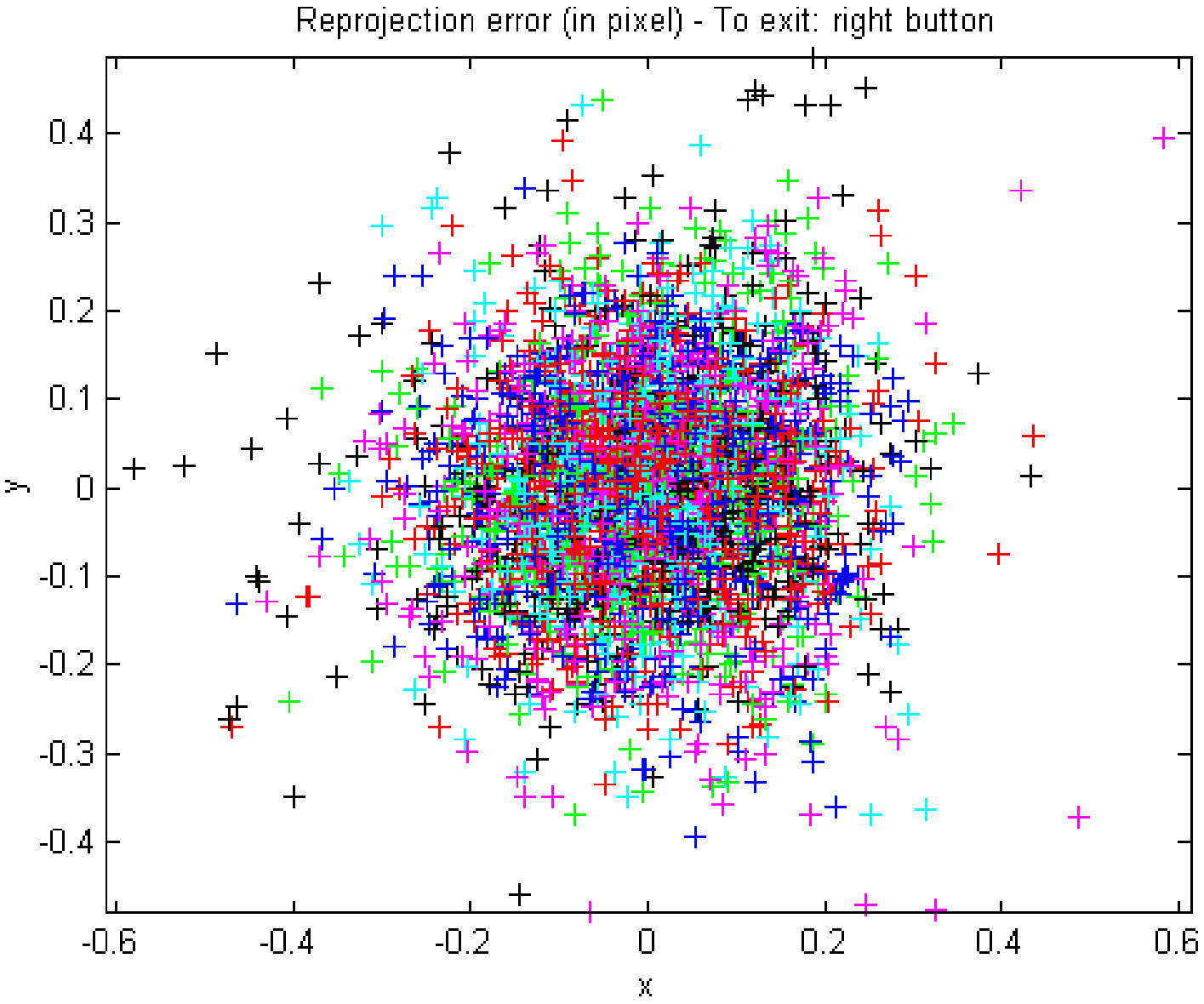
Step 4: camera calibration



Step 4: camera calibration

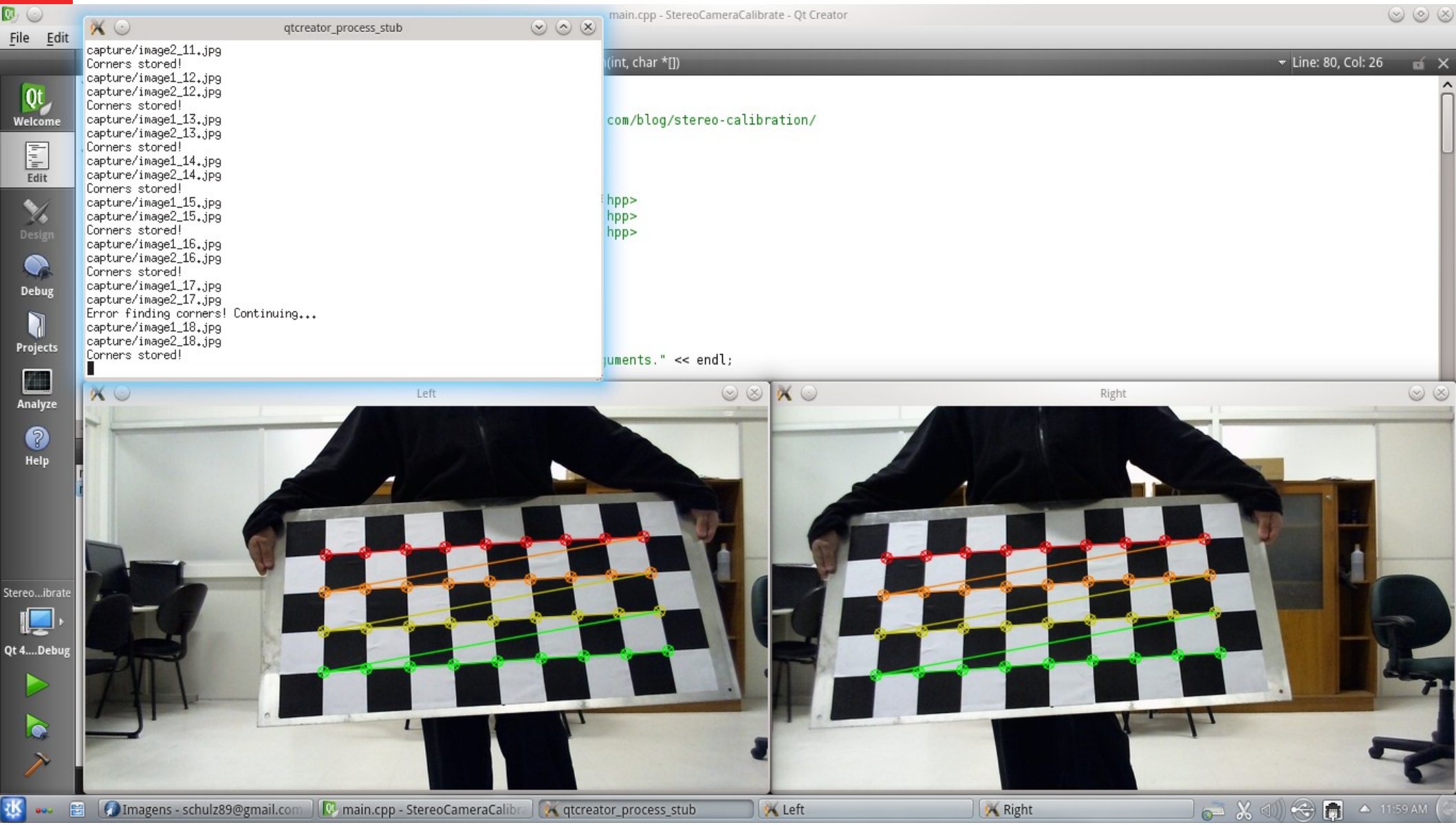


Step 5: refinement

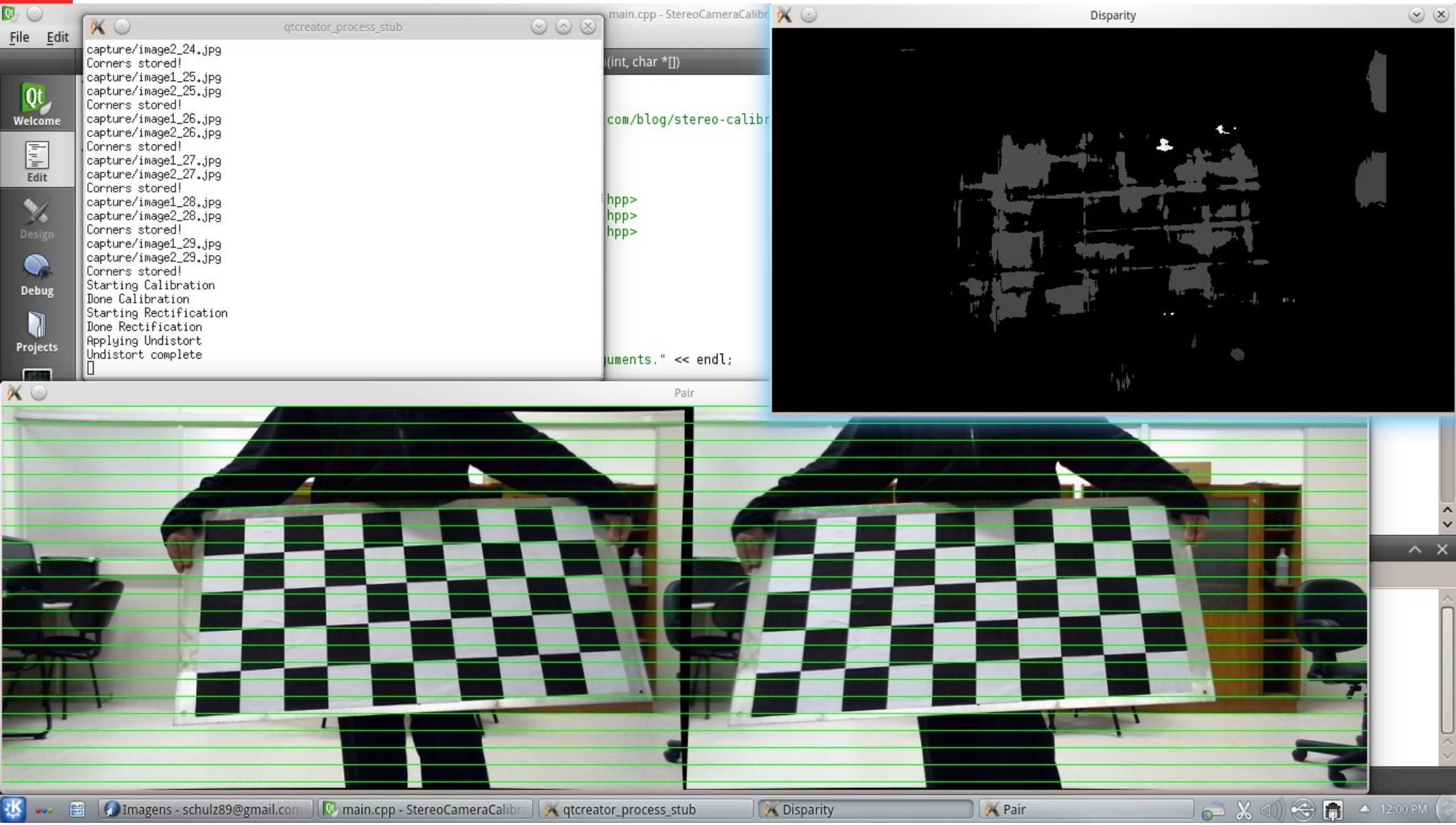


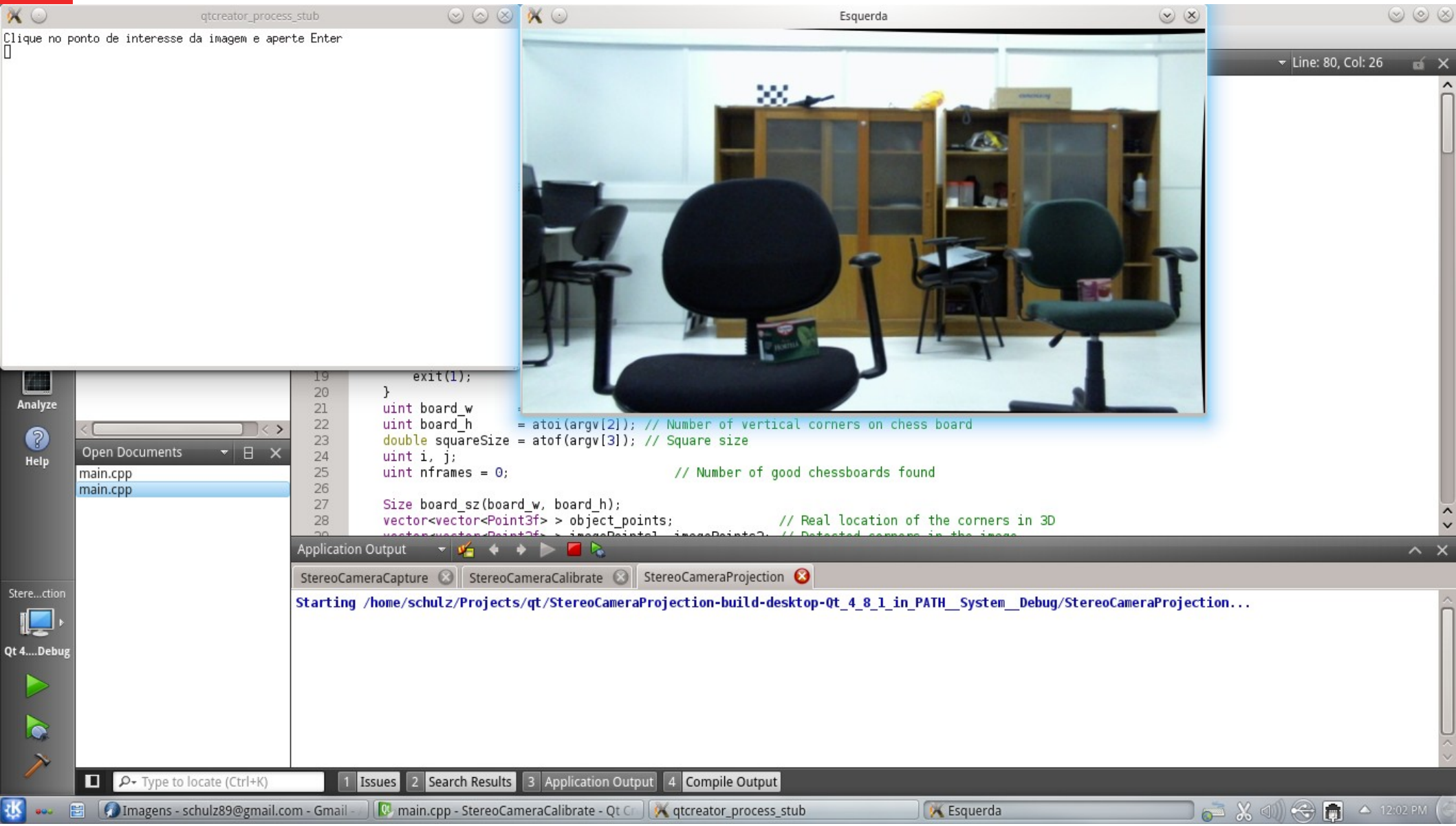
Camera calibration With OpenCV

http://docs.opencv.org/master/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

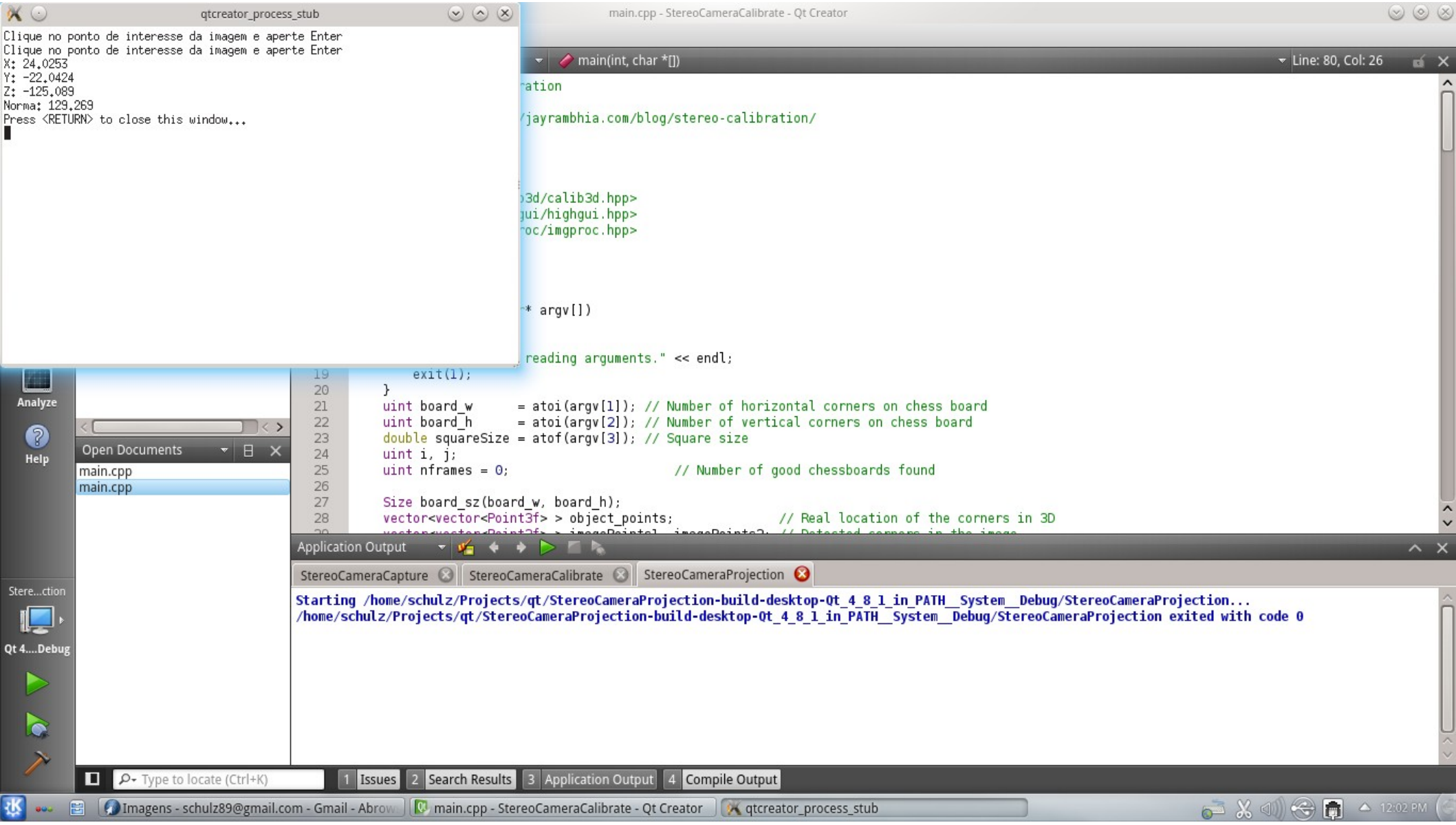


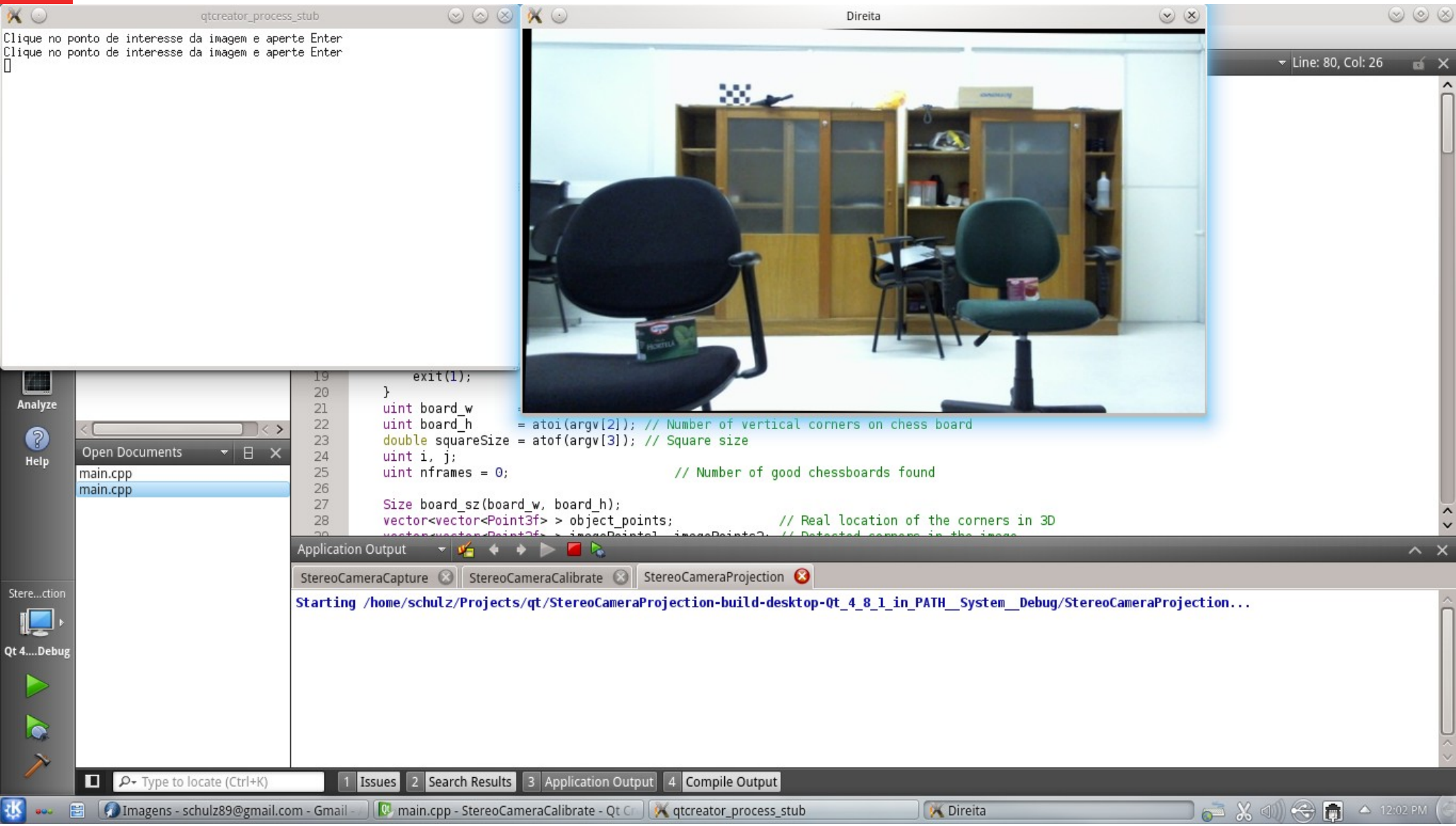
Tamanho informado do tabuleiro: 4x9 (sem as bordas)


qtcreator_process_stub
capture/image2_24.jpg
Corners stored!
capture/image1_25.jpg
capture/image2_25.jpg
Corners stored!
capture/image1_26.jpg
capture/image2_26.jpg
Corners stored!
capture/image1_27.jpg
capture/image2_27.jpg
Corners stored!
capture/image1_28.jpg
capture/image2_28.jpg
Corners stored!
capture/image1_29.jpg
capture/image2_29.jpg
Corners stored!
Starting Calibration
Done Calibration
Starting Rectification
Done Rectification
Applying Undistort
Undistort complete
Disparity
Pair
File
Edit
Welcome
Edit
Design
Debug
Projects
Imagens - schulz89@gmail.com
main.cpp - StereoCameraCalibr
qtcreator_process_stub
Disparity
Pair
12:00 PM

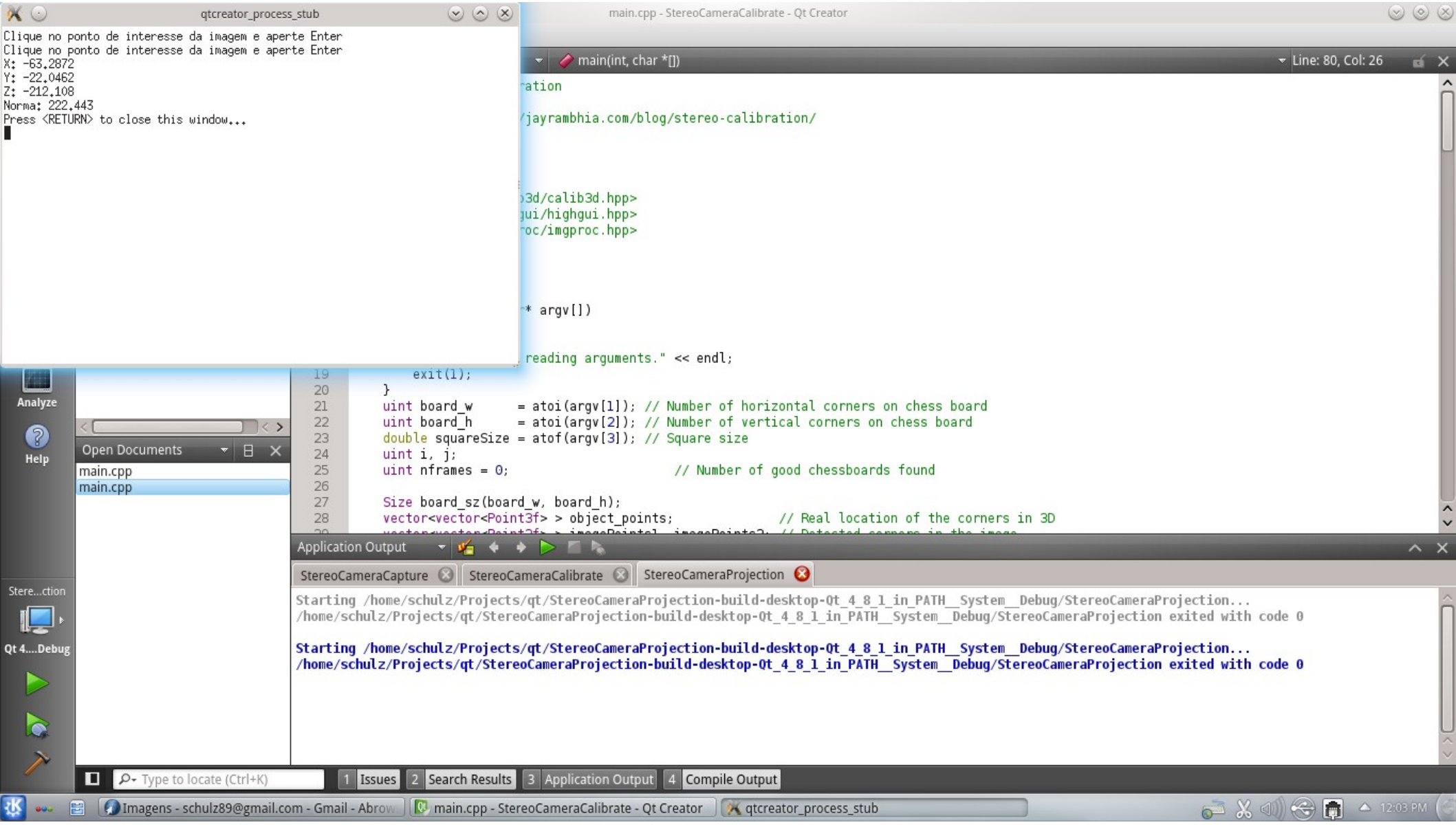


Caixa de chá a aprox. 1,3m

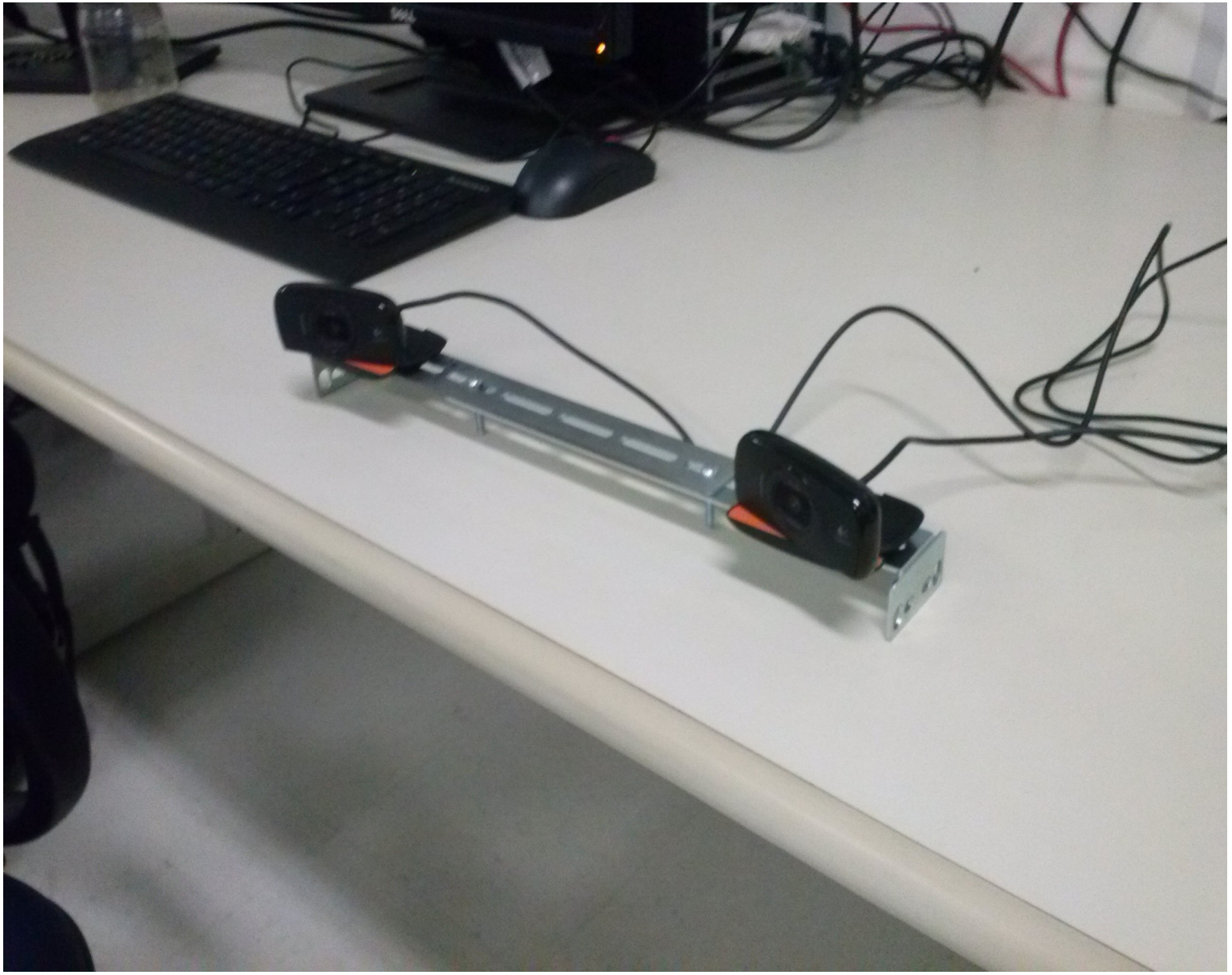

qtcreator_process_stub
Clique no ponto de interesse da imagem e aperte Enter
Clique no ponto de interesse da imagem e aperte Enter
X: 24.0253
Y: -22.0424
Z: -125.089
Norma: 129.269
Press <RETURN> to close this window...
main.cpp - StereoCameraCalibrate - Qt Creator
main(int, char *[])
Line: 80, Col: 26
ration
/jayrambhia.com/blog/stereo-calibration/
3d/calib3d.hpp>
gui/highgui.hpp>
roc/imgproc.hpp>
* argv[])
reading arguments." << endl;
exit(1);
}
uint board_w = atoi(argv[1]); // Number of horizontal corners on chess board
uint board_h = atoi(argv[2]); // Number of vertical corners on chess board
double squareSize = atof(argv[3]); // Square size
uint i, j;
uint nframes = 0; // Number of good chessboards found
Size board_sz(board_w, board_h);
vector<vector<Point3f> > object_points; // Real location of the corners in 3D
Analyze
Help
Open Documents
main.cpp
main.cpp
Stere...ction
Qt 4....Debug
Application Output
StereoCameraCapture
StereoCameraCalibrate
StereoCameraProjection
Starting /home/schulz/Projects/qt/StereoCameraProjection-build-desktop-Qt_4_8_1_in_PATH__System__Debug/StereoCameraProjection...
/home/schulz/Projects/qt/StereoCameraProjection-build-desktop-Qt_4_8_1_in_PATH__System__Debug/StereoCameraProjection exited with code 0
Type to locate (Ctrl+K)
1 Issues
2 Search Results
3 Application Output
4 Compile Output
Imagens - schulz89@gmail.com - Gmail - Abrow
main.cpp - StereoCameraCalibrate - Qt Creator
qtcreator_process_stub
12:02 PM



Caixa de chá a aprox. 2,2m


qtcreator_process_stub
Clique no ponto de interesse da imagem e aperte Enter
Clique no ponto de interesse da imagem e aperte Enter
X: -63.2872
Y: -22.0462
Z: -212.108
Norma: 222.443
Press <RETURN> to close this window...
main.cpp - StereoCameraCalibrate - Qt Creator
main(int, char *[])
Line: 80, Col: 26
ration
/jayrambhia.com/blog/stereo-calibration/
3d/calib3d.hpp>
gui/highgui.hpp>
roc/imgproc.hpp>
* argv[])
reading arguments." << endl;
exit(1);
}
uint board_w = atoi(argv[1]); // Number of horizontal corners on chess board
uint board_h = atoi(argv[2]); // Number of vertical corners on chess board
double squareSize = atof(argv[3]); // Square size
uint i, j;
uint nframes = 0; // Number of good chessboards found
Size board_sz(board_w, board_h);
vector<vector<Point3f> > object_points; // Real location of the corners in 3D
Analyze
Help
Open Documents
main.cpp
main.cpp
Application Output
StereoCameraCapture
StereoCameraCalibrate
StereoCameraProjection
Starting /home/schulz/Projects/qt/StereoCameraProjection-build-desktop-Qt_4_8_1_in_PATH__System__Debug/StereoCameraProjection...
/home/schulz/Projects/qt/StereoCameraProjection-build-desktop-Qt_4_8_1_in_PATH__System__Debug/StereoCameraProjection exited with code 0
Starting /home/schulz/Projects/qt/StereoCameraProjection-build-desktop-Qt_4_8_1_in_PATH__System__Debug/StereoCameraProjection...
/home/schulz/Projects/qt/StereoCameraProjection-build-desktop-Qt_4_8_1_in_PATH__System__Debug/StereoCameraProjection exited with code 0
Stere...ction
Qt 4....Debug
Type to locate (Ctrl+K)
1 Issues
2 Search Results
3 Application Output
4 Compile Output
Imagens - schulz89@gmail.com - Gmail - Abrow
main.cpp - StereoCameraCalibrate - Qt Creator
qtcreator_process_stub
12:03 PM

Câmeras com eixos principais não paralelos



qtcreator_process_stub

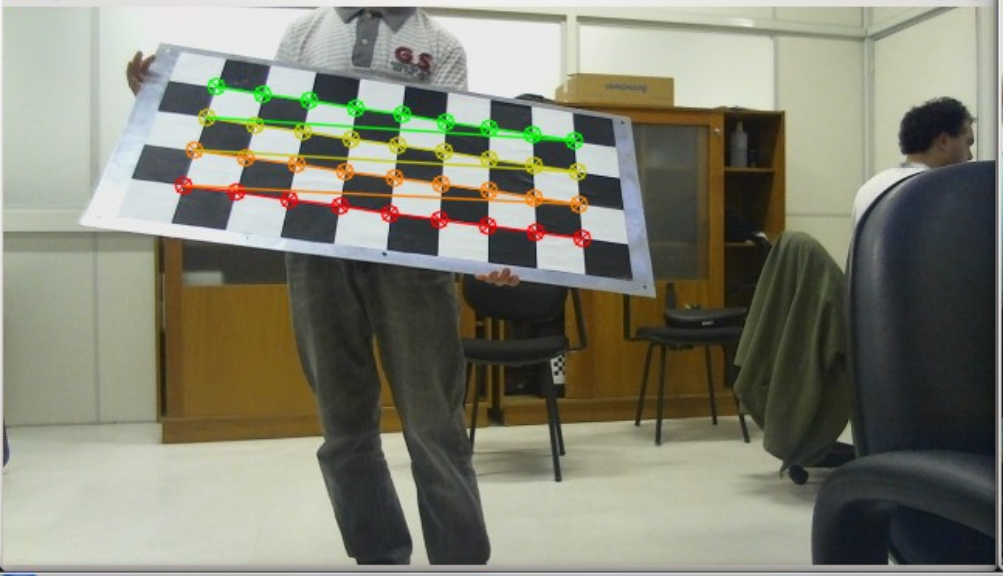
```
Reading imglist.txt...
capture/image1_0.jpg
capture/image2_0.jpg
Corners stored!
capture/image1_1.jpg
capture/image2_1.jpg
Corners stored!
capture/image1_2.jpg
capture/image2_2.jpg
Corners stored!
capture/image1_3.jpg
capture/image2_3.jpg
Corners stored!
capture/image1_4.jpg
capture/image2_4.jpg
Corners stored!
capture/image1_5.jpg
capture/image2_5.jpg
Corners stored!
capture/image1_6.jpg
capture/image2_6.jpg
Corners stored!
```

main.cpp - StereoCameraCalibrate - Qt Creator

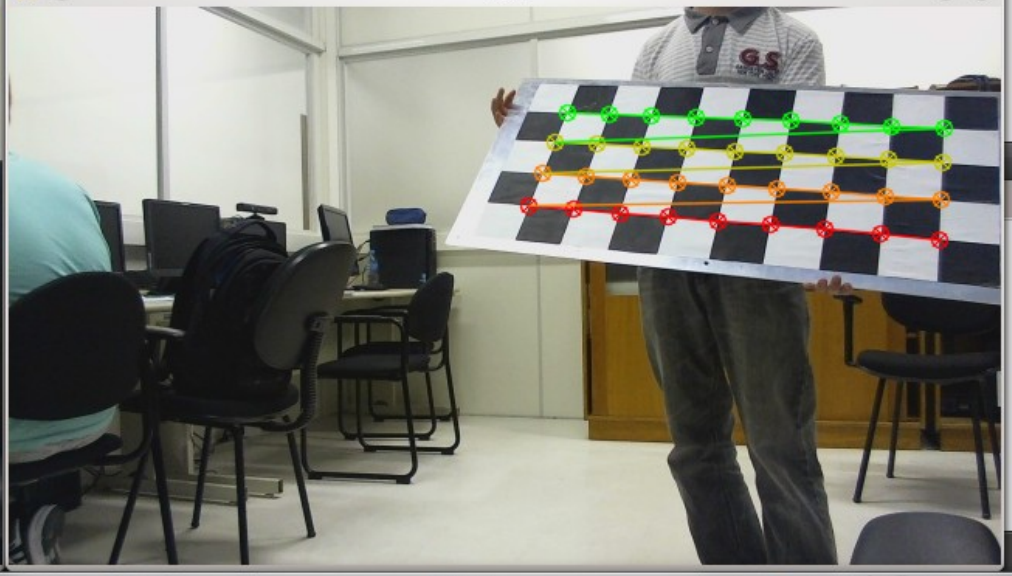
main(int, char *[]) Line: 80, Col: 25

```
rnerSubPix(gray1, corners1, Size(11, 11), Size(-1, -1), TermCriteria(CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 30, 0.1));
awChessboardCorners(img1, board_sz, corners1, found1);
rnerSubPix(gray2, corners2, Size(11, 11), Size(-1, -1), TermCriteria(CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 30, 0.1));
awChessboardCorners(img2, board_sz, corners2, found2);

agePoints1.push_back(corners1);
agePoints2.push_back(corners2);
ject_points.push_back(obj);
ut << "Corners stored!" << endl;
rames++;


ut << "Error finding corners! Continuing..." << endl;

("Left",img1);
("Right",img2);
(0);
```

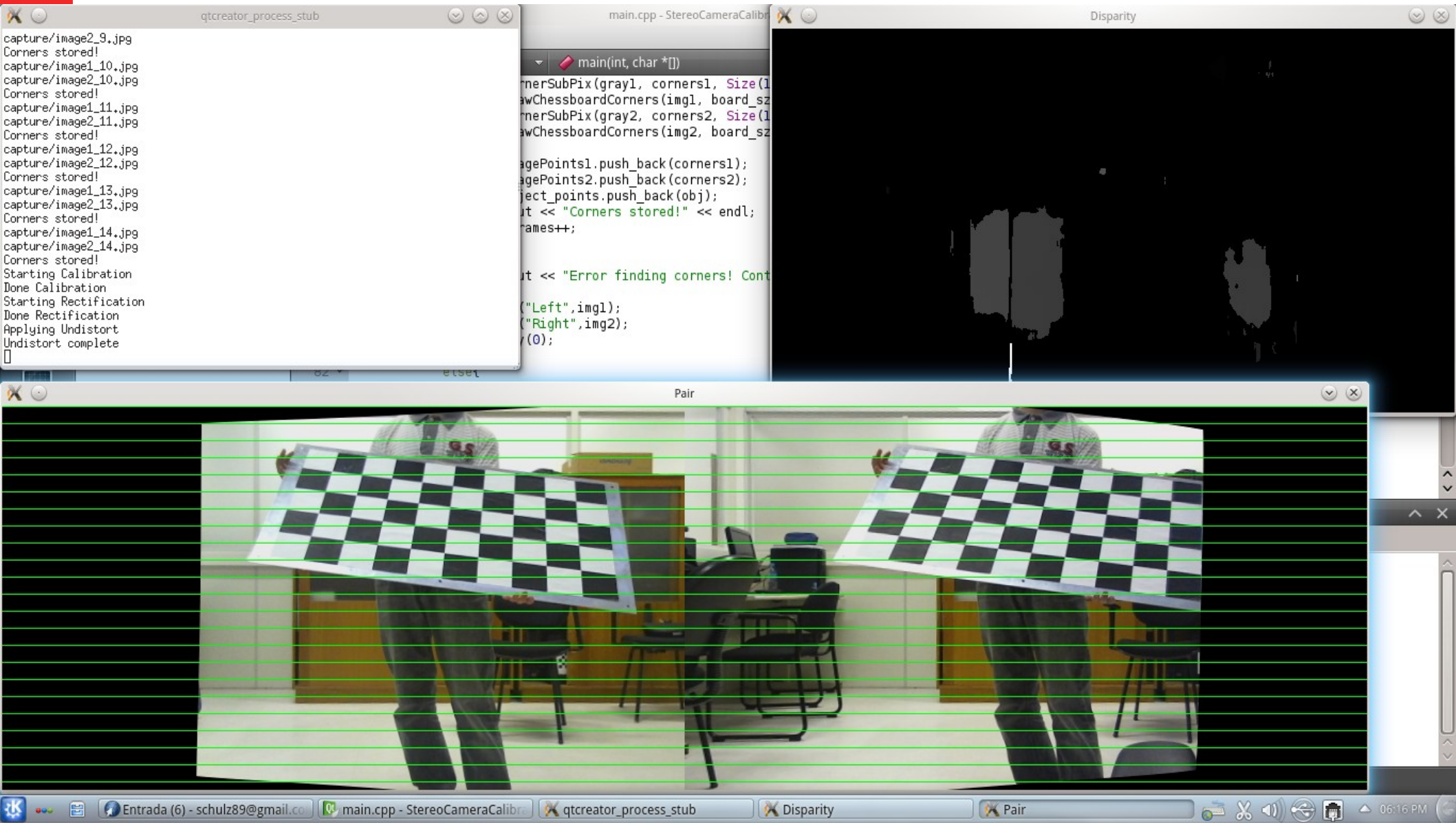
Left



Right



Entrada (6) - schulz89@gmail.co | main.cpp - StereoCameraCalibra | qtcreator_process_stub | Left | Right | 06:15 PM


qtcreator_process_stub
capture/image2_9.jpg
Corners stored!
capture/image1_10.jpg
capture/image2_10.jpg
Corners stored!
capture/image1_11.jpg
capture/image2_11.jpg
Corners stored!
capture/image1_12.jpg
capture/image2_12.jpg
Corners stored!
capture/image1_13.jpg
capture/image2_13.jpg
Corners stored!
capture/image1_14.jpg
capture/image2_14.jpg
Corners stored!
Starting Calibration
Done Calibration
Starting Rectification
Done Rectification
Applying Undistort
Undistort complete
main.cpp - StereoCameraCalibr
main(int, char *[])
rnerSubPix(gray1, corners1, Size(1
awChessboardCorners(img1, board_sz
rnerSubPix(gray2, corners2, Size(1
awChessboardCorners(img2, board_sz
agePoints1.push_back(corners1);
agePoints2.push_back(corners2);
ject_points.push_back(obj);
ut << "Corners stored!" << endl;
rames++;
ut << "Error finding corners! Cont
("Left",img1);
("Right",img2);
(0);
Disparity
Pair
Entrada (6) - schulz89@gmail.co
main.cpp - StereoCameraCalibr
qtcreator_process_stub
Disparity
Pair
06:16 PM

OpenCV stereoCalibration

CM1 - Camera Matrix for left camera

CM2 - Camera Matrix for right camera

D1 - Vector of distortion coefficients for left camera

D2 - Vector of distortion coefficients for right camera

R - Rotation matrix between the left and the right camera coordinate systems

T - Translation vector between the left and the right coordinate systems of the cameras

E - Essential matrix

F - Fundamental matrix

Recuperação de imagens

• • •

> 5000 images

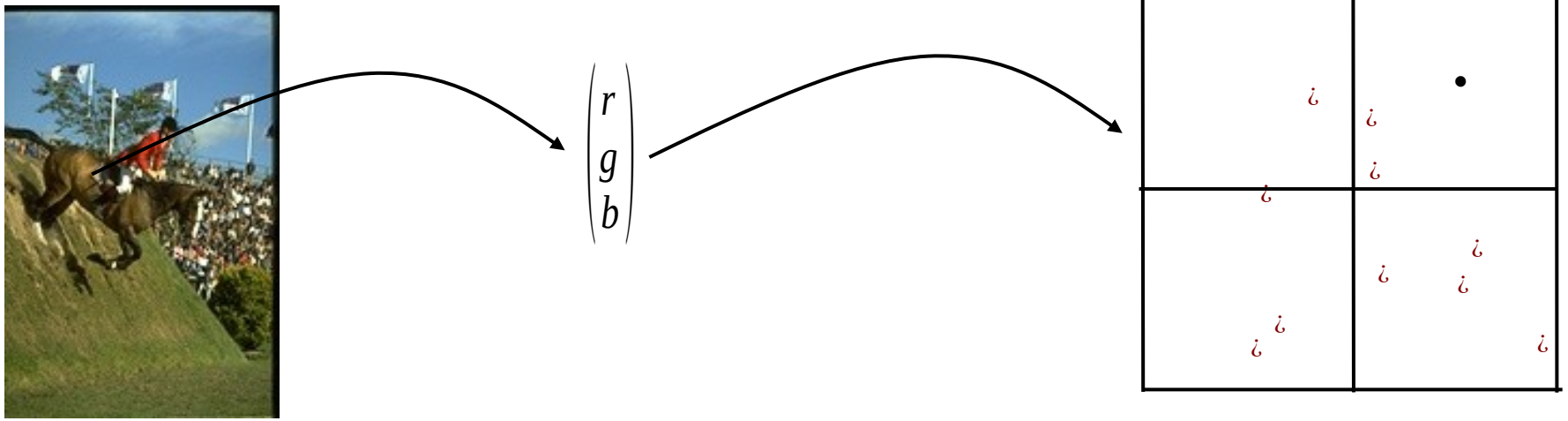
change in viewing angle

Primeiras bordagens em reconhecimento

Color histogram [Swain & Balalrd, 1991]

Histograma descreve distribuição de vetores de cor

Cada pixel um vetor de cor



=> não é robusto à oclusão, não é invariante, não é distintivo

Primeiras bordagens em reconhecimento

Autoimagens (eigenimages) [Turk & Pentland, 1991]

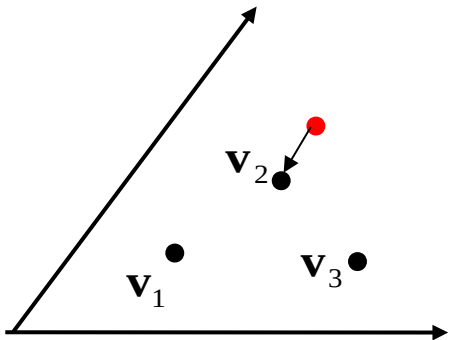
Projeta imagens de faces para um espaço de características, *face space*

Eigenfaces são os eigenvectors (*principal components*) da matriz de covariância de um conjunto de faces

Cada imagem é um ponto ou vetor em um espaço hiperdimensional

Cada face é uma soma ponderada das *eigenfaces* → comparam-se os pesos

=> não é robusto à oclusão, não é invariante



Primeiras bordagens em reconhecimento

Autoimagens (eigenimages) [Turk & Pentland, 1991]

Imagens de treinamento

Sete eigenfaces

Primeiras bordagens em reconhecimento

Invariantes geométricos [Rothwell, Zisserman, Mundy and Forsyth, 1992]

Invariantes obtidos por curvas algébricas planas,i.e., linhas e cônicas:

5 linhas coplanares fornecem dois invariantes independentes

1 cônica e 3 linhas permitem calcular 3 índices

1 par de cônicas fornece dois índices invariantes

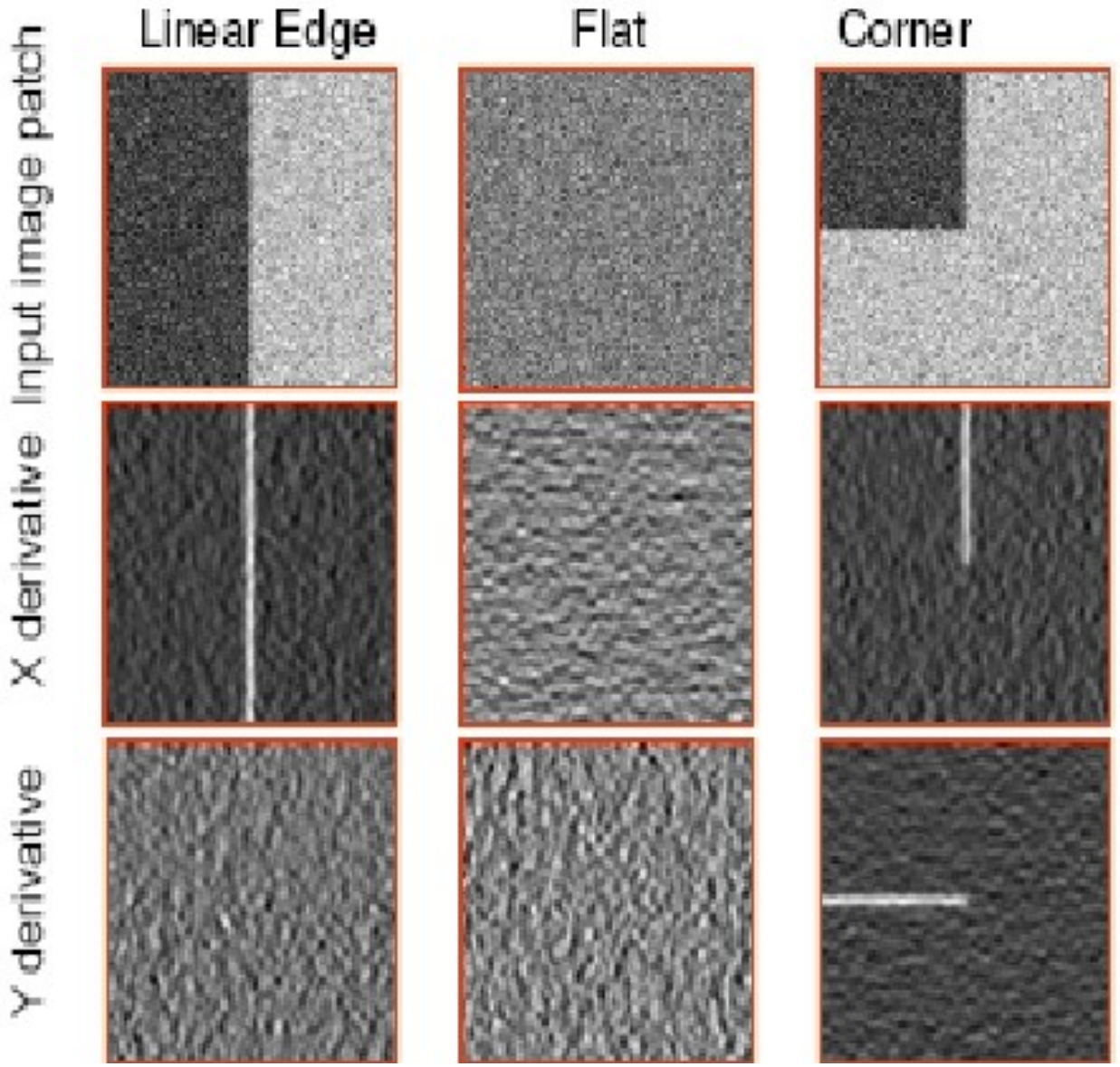
Extrai bordas com Canny

Distância entre dois pontos é invariante à rotação

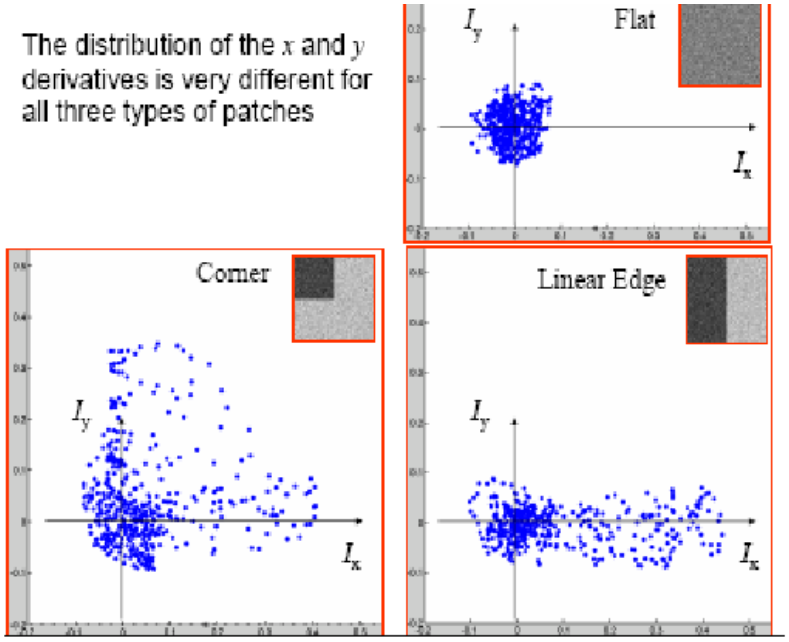
Cross-ratio é invariante à homografia planar

=> local e invariante

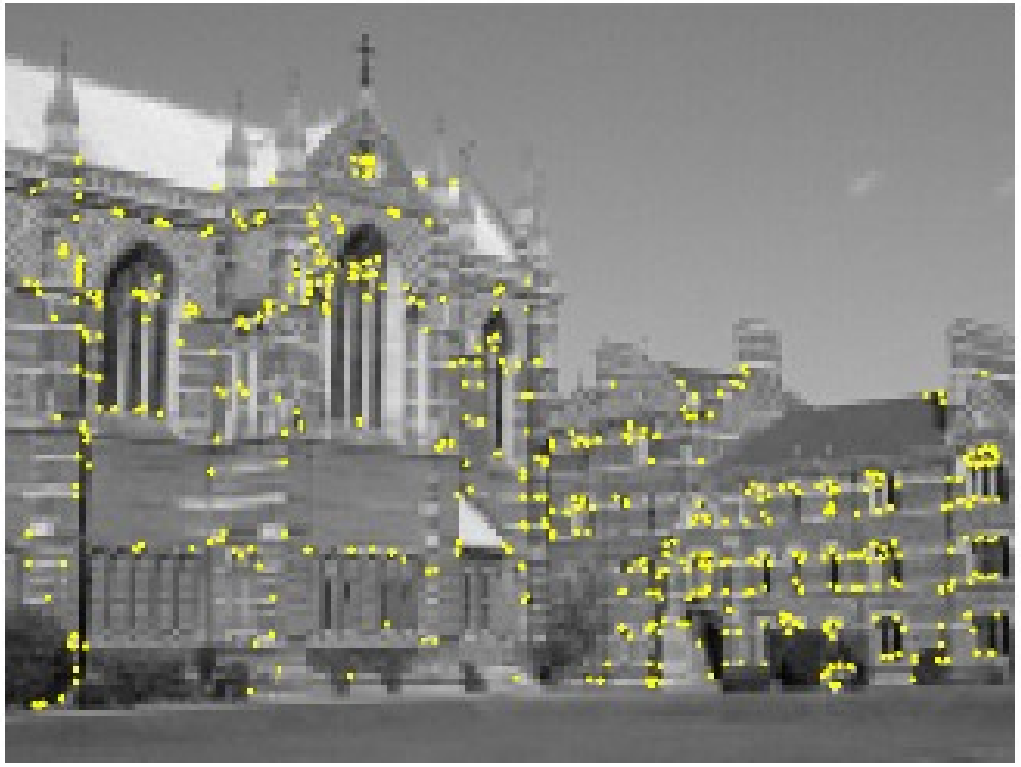
Bordas x Cantos

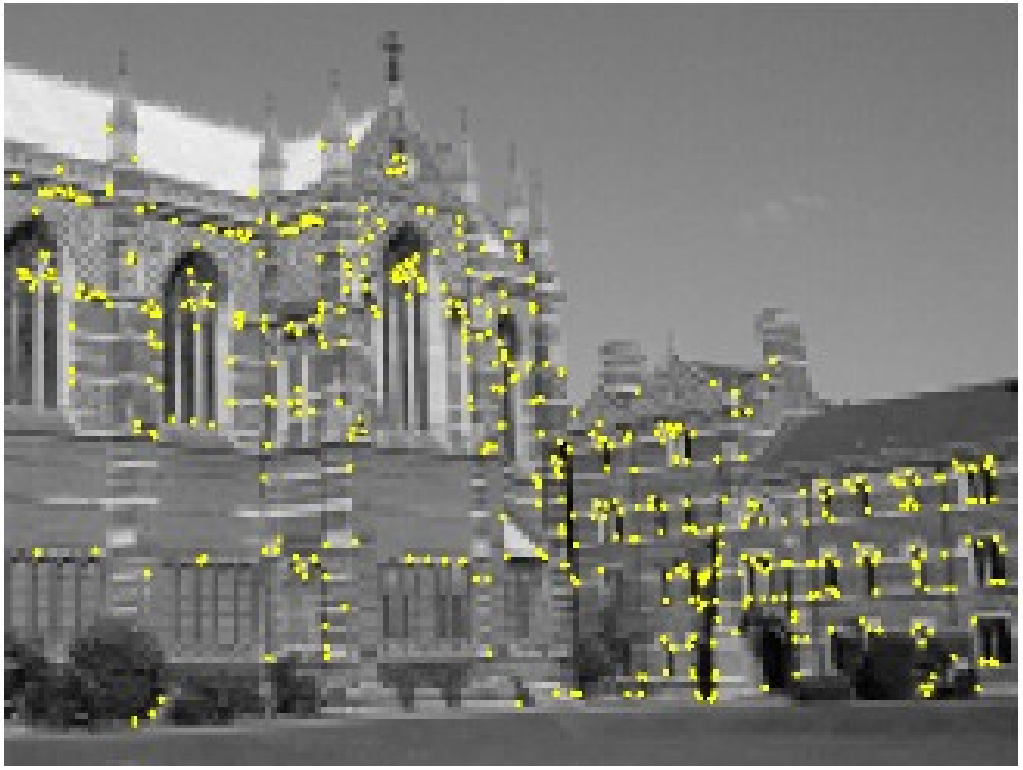


The distribution of the x and y derivatives is very different for all three types of patches



Pontos de interesse





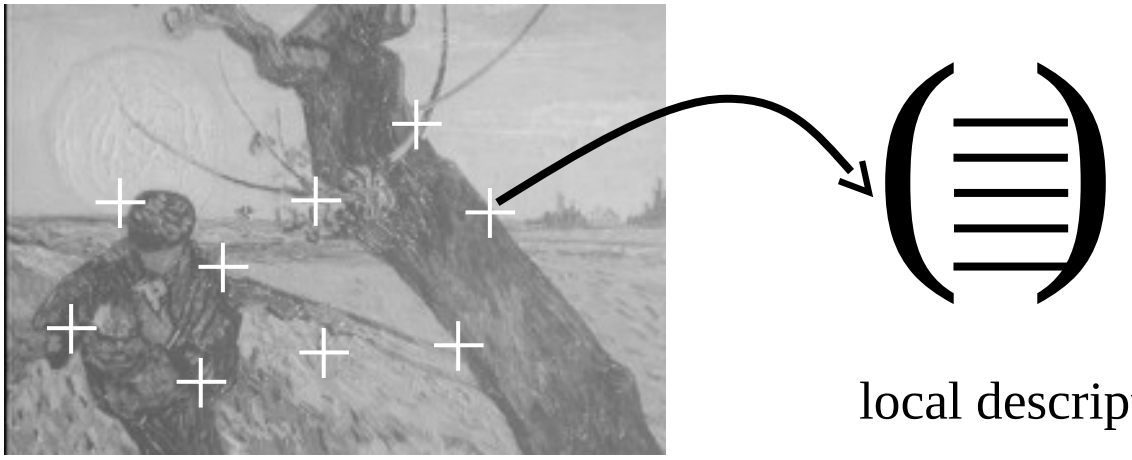
Interest points extracted with Harris (~ 500 points)

Harris detector

Baseado na ideia de auto-correlação

Diferença relevante em todas direções => ponto de interesse

Abordagem proposta por Schmid and Mohr, 1997



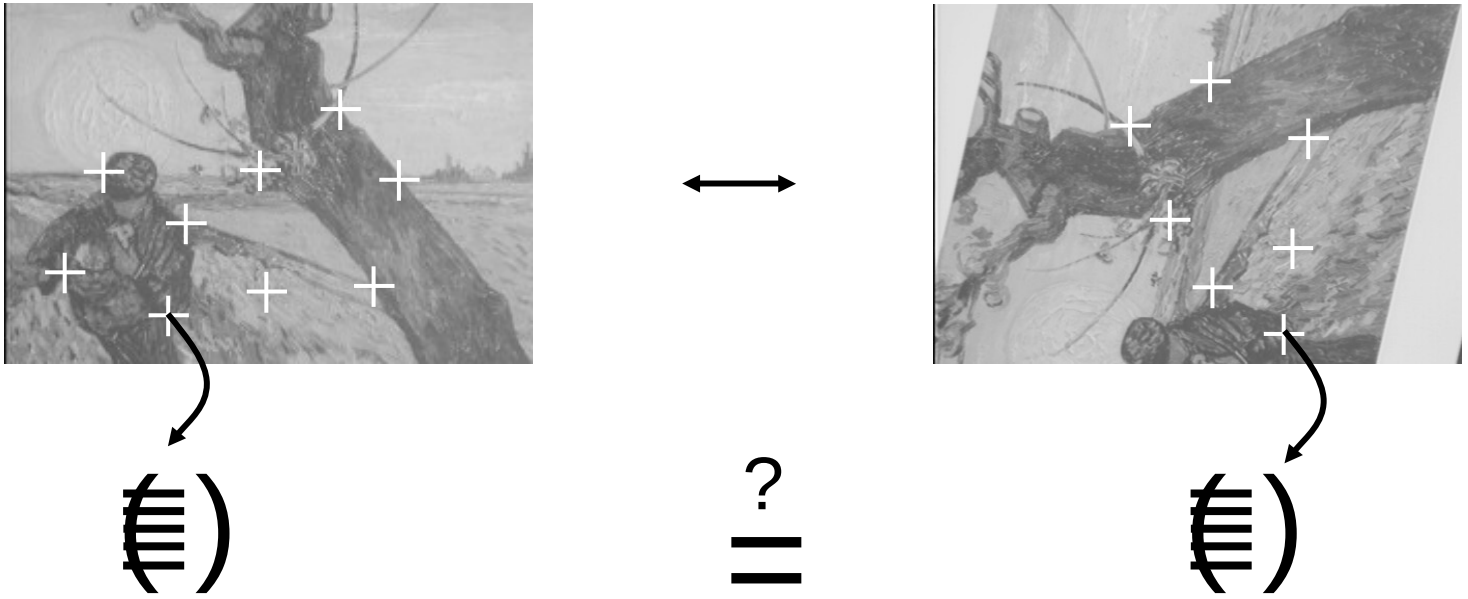
1) Extração de pontos de interesse (*characteristic locations*)

2) Cálculo de descritores locais

3) Determinação de correspondências

4) Seleção de imagens similares

Determinando correspondência



Comparação dos vetores com distância de Mahalanobis:

$$dist_M(\mathbf{p},\mathbf{q}) = \sqrt{(\mathbf{p}-\mathbf{q})^T \Lambda^{-1} (\mathbf{p}-\mathbf{q})}$$

Descritores locais

Derivadas na escala de cinza

$$v(x,y)=\begin{pmatrix} I(x,y)*G(\sigma) \\ I(x,y)*G_x(\sigma) \\ I(x,y)*G_y(\sigma) \\ I(x,y)*G_{xx}(\sigma) \\ I(x,y)*G_{xy}(\sigma) \\ I(x,y)*G_{yy}(\sigma) \\ \vdots \end{pmatrix}$$

$$I(x,y)*G(\sigma)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(x',y')I(x-x',y-y')dx'dy'$$

$$G((x',y')^t,\sigma)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x',y')^{t^2}}{2\sigma^2}\right)$$

Descritores locais

Invariância à rotação: invariantes diferenciais

[Koen87]

$$
\begin{aligned}
& L \\
& L_i L_i \\
& L_i L_{ij} L_j \\
& L_{ii} \\
& L_{ij} L_{ij} \\
& \varepsilon_{ij}\left(L_{jkl} L_i L_k L_l - L_{jkk} L_i L_l L_l\right) \\
& L_{iij} L_j L_k L_k - L_{ijk} L_i L_j L_l \\
-& \varepsilon_{ij} L_{jkl} L_i L_k L_l \\
& L_{ijk} L_i L_j L_k
\end{aligned}
=
\begin{aligned}
& L \\
& L_x L_x + L_y L_y \\
& L_{xx} L_x L_x + 2 L_{xy} L_x L_y + L_{yy} L_{yy} \\
& L_{xx} + L_{yy} \\
& L_{xx} L_{xx} + 2 L_{xy} L_{xy} + L_{yy} L_{yy} \\
& \dots \\
& \dots \\
& \dots \\
& \dots
\end{aligned}
$$

onde ε_{ij} é o tensor antisimétrico epsilon

Descritores locais

Robustez à variação em iluminação

$$\frac{L_i L_{ij} L_j}{(L_i L_i)^{3/2}}$$

$$\frac{L_{ii}}{(L_i L_i)^{1/2}}$$

$$\frac{L_{ij} L_{ji}}{L_i L_i}$$

$$\frac{\varepsilon_{ij}(L_{jkl} L_i L_k L_l - L_{jkk} L_i L_l L_l)}{(L_i L_i)^2}$$

$$\frac{L_{iij} L_j L_k L_k - L_{ijk} L_i L_j L_k)}{(L_i L_i)^2}$$

$$\frac{-\varepsilon_{ij} L_{jkl} L_i L_k L_l}{(L_i L_i)^2}$$

$$\frac{L_{ijk} L_i L_j L_k}{(L_i L_i)^2}$$

Em caso de transformações afins:

$$I_1(\mathbf{x}) = aI_2(\mathbf{x}) + b$$

Ou normalização de pedaço da imagem com média e variância

Scale Invariant Feature Transform (SIFT)

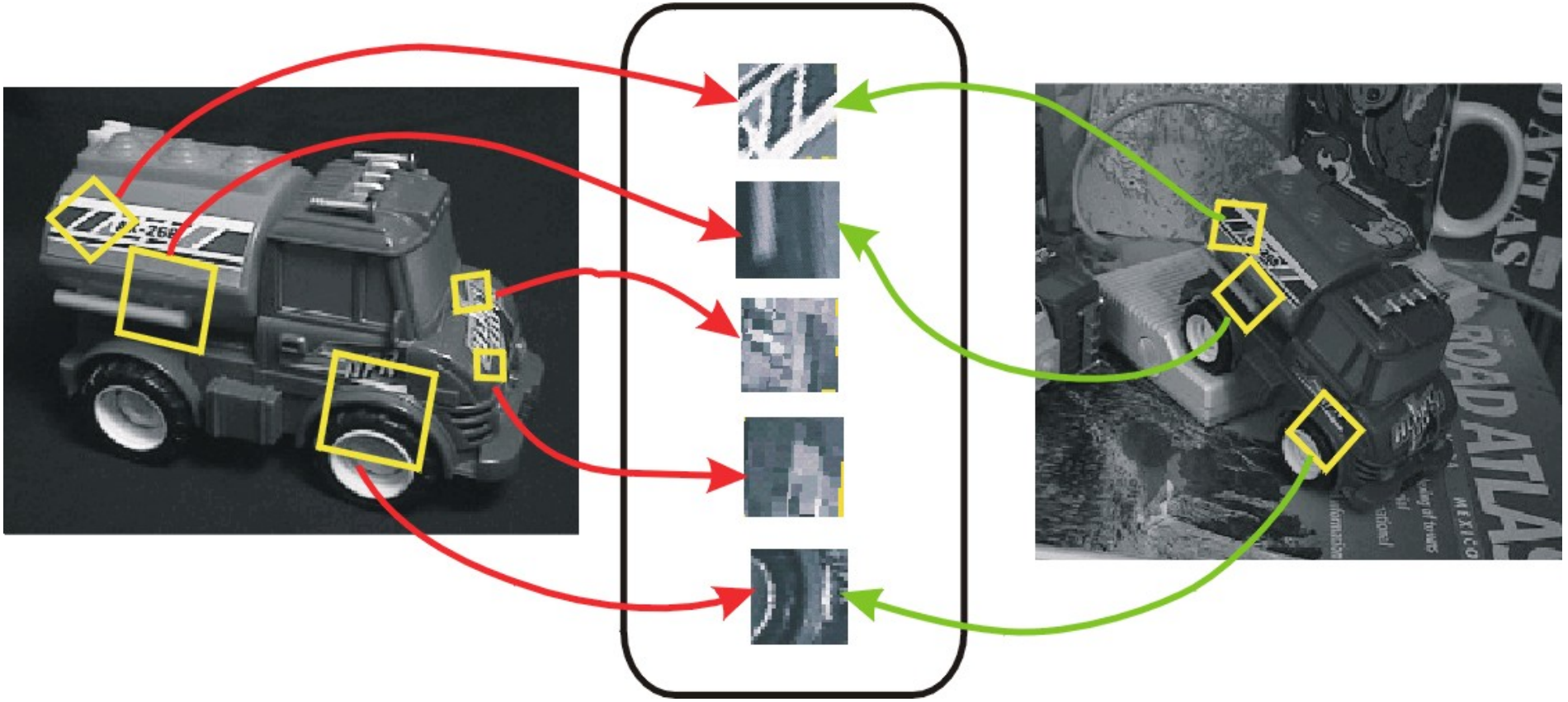
Detector e descritor de pontos de interesse proposto por Lowe em 2003

Características locais

Invariância

Invariant Local Features

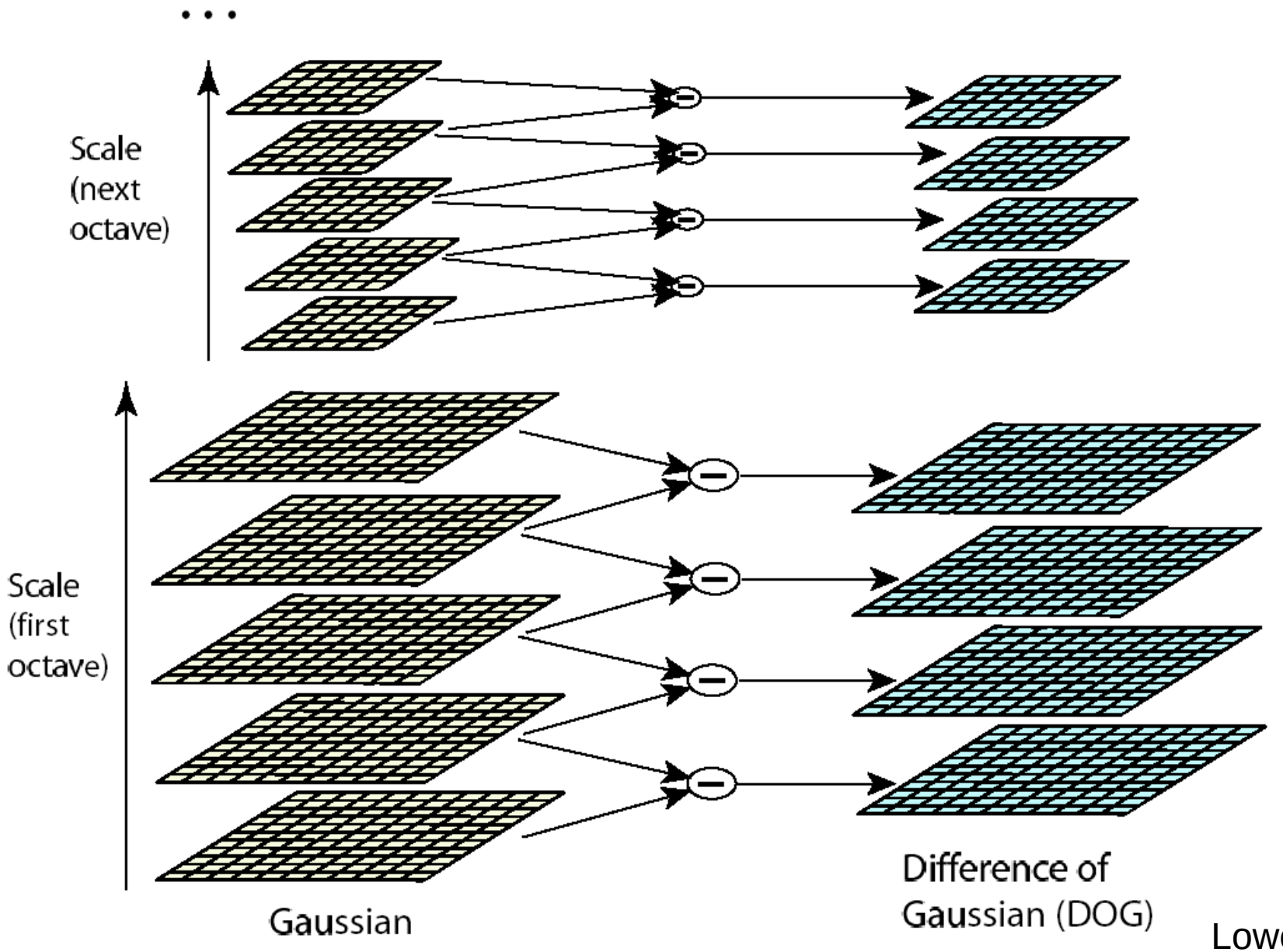
Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



SIFT Features

Lowe 2003

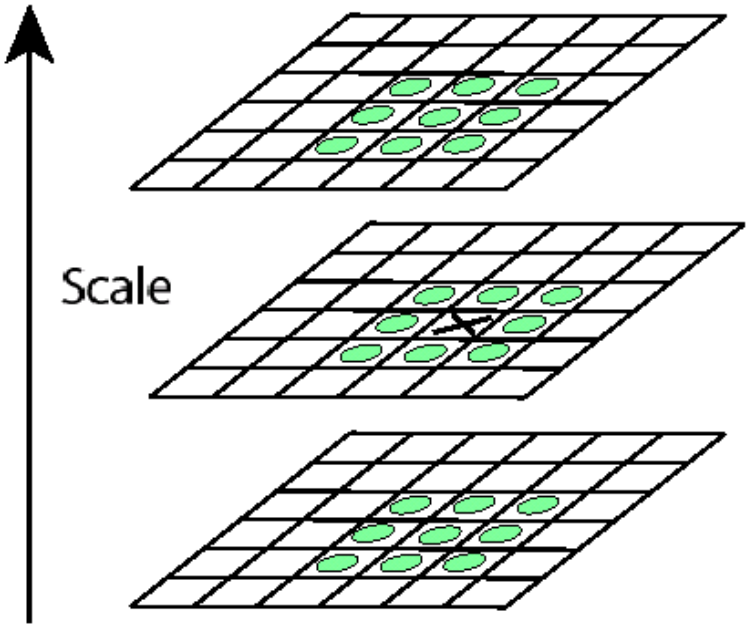
Scale space processed one octave at a time



Lowe 2003

Key point localization

- Detect maxima and minima of difference-of-Gaussian in scale space
- Fit a quadratic to surrounding values for sub-pixel and sub-scale interpolation (Brown & Lowe, 2002)
- Taylor expansion around point:



- Offset of extremum (use finite differences for derivatives):

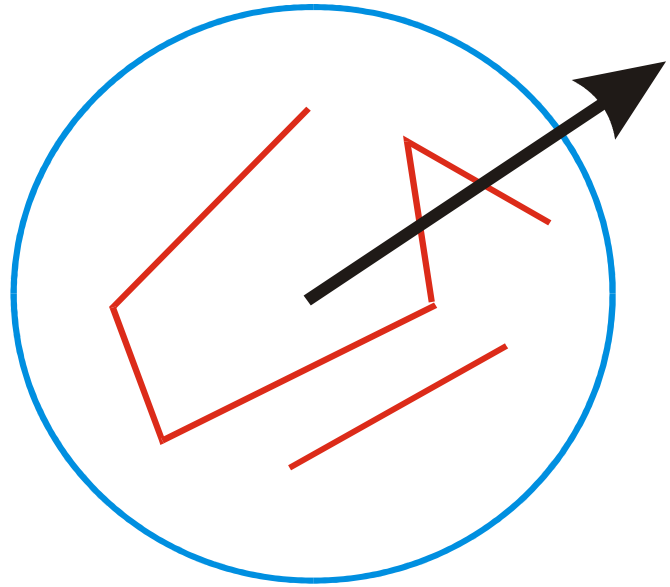
$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^\mathbf{T}\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

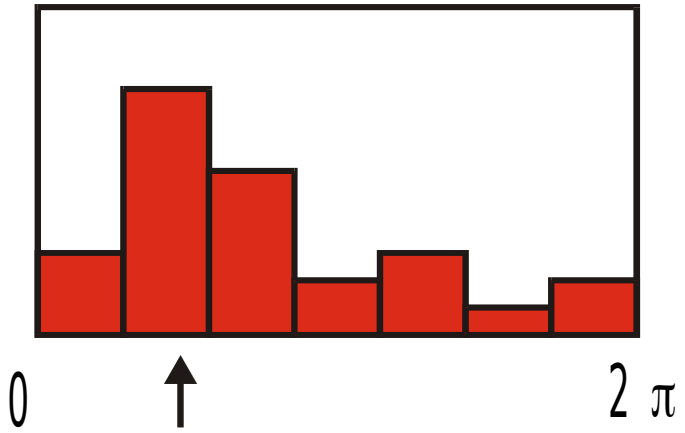
$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1}\frac{\partial D}{\partial \mathbf{x}}$$

Lowe 2003

Select canonical orientation

- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)

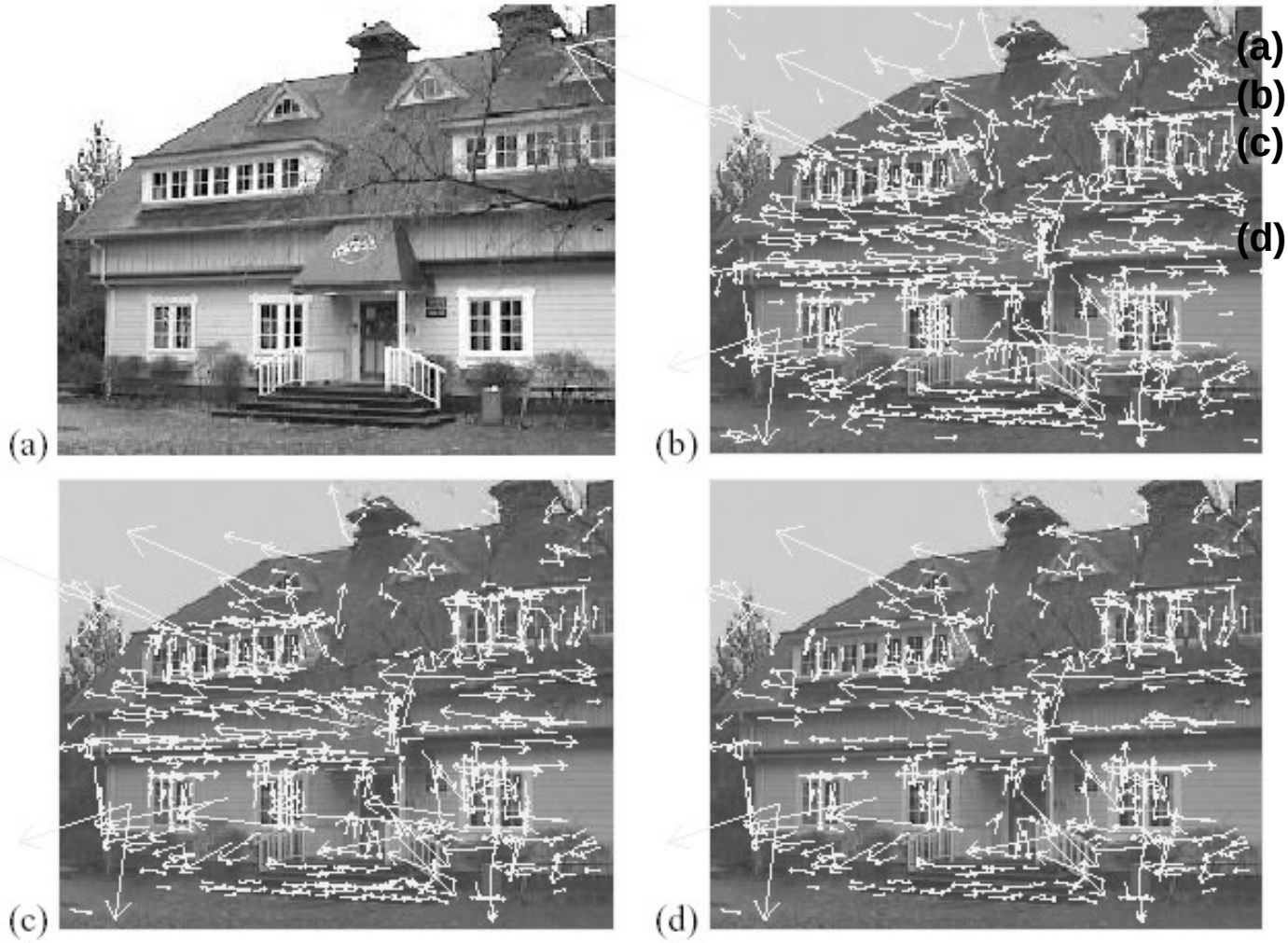




Lowe 2003

Example of keypoint detection

Threshold on value at DOG peak and on ratio of principle curvatures (Harris approach)

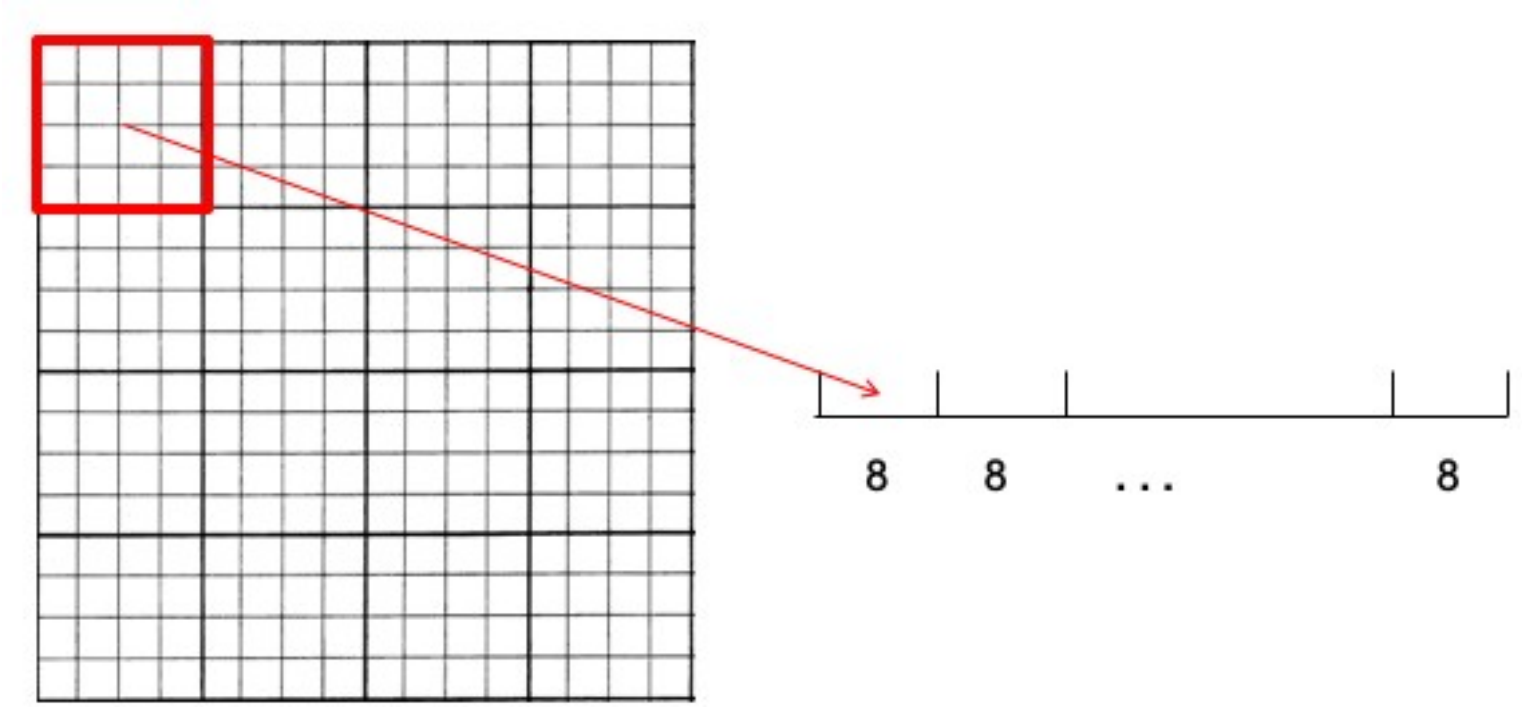


(a) 233x189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures

Lowe 2003

SIFT vector formation

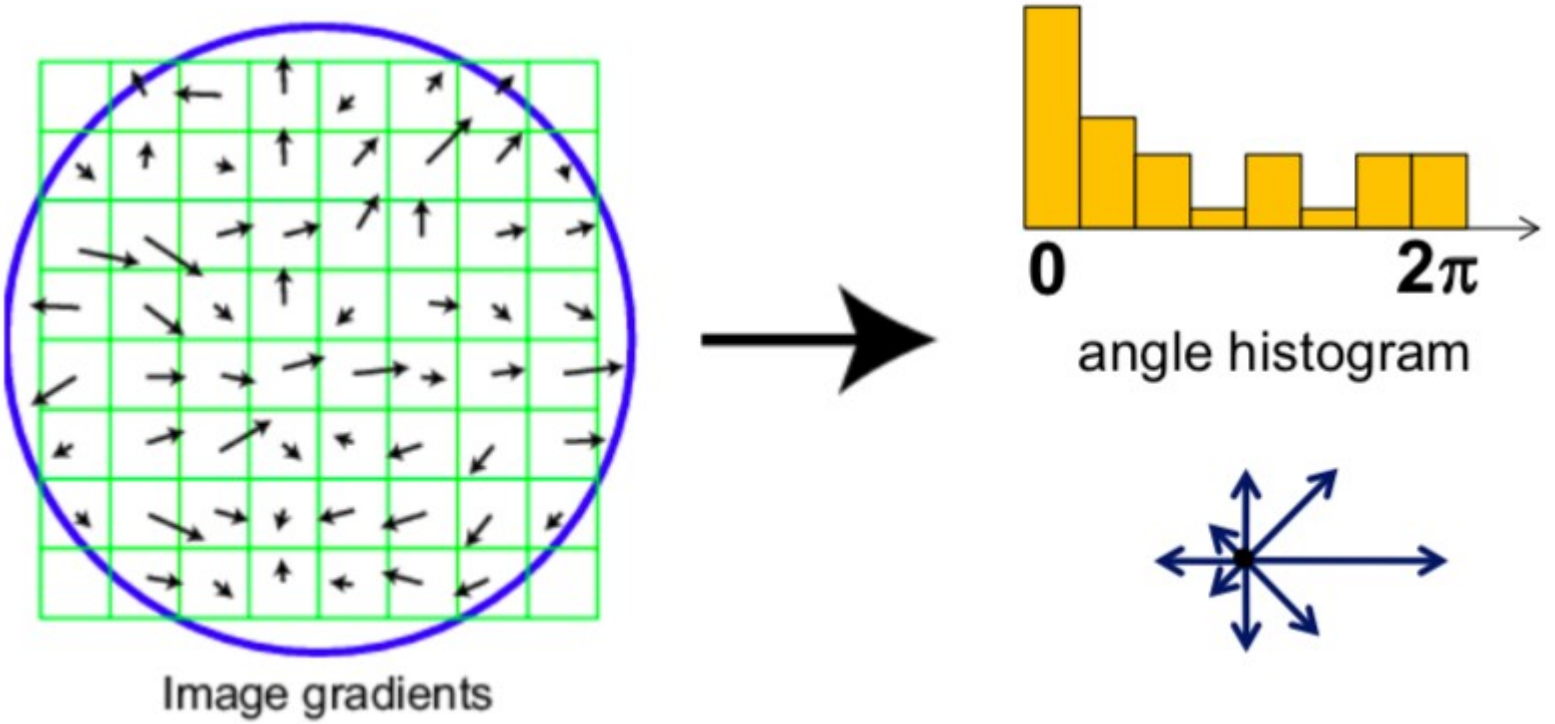
- Thresholded image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms
- 8 orientations x 4x4 histogram array = 128 dimensions



Lowe 2003

SIFT vector formation

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations



Lowe 2003

SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

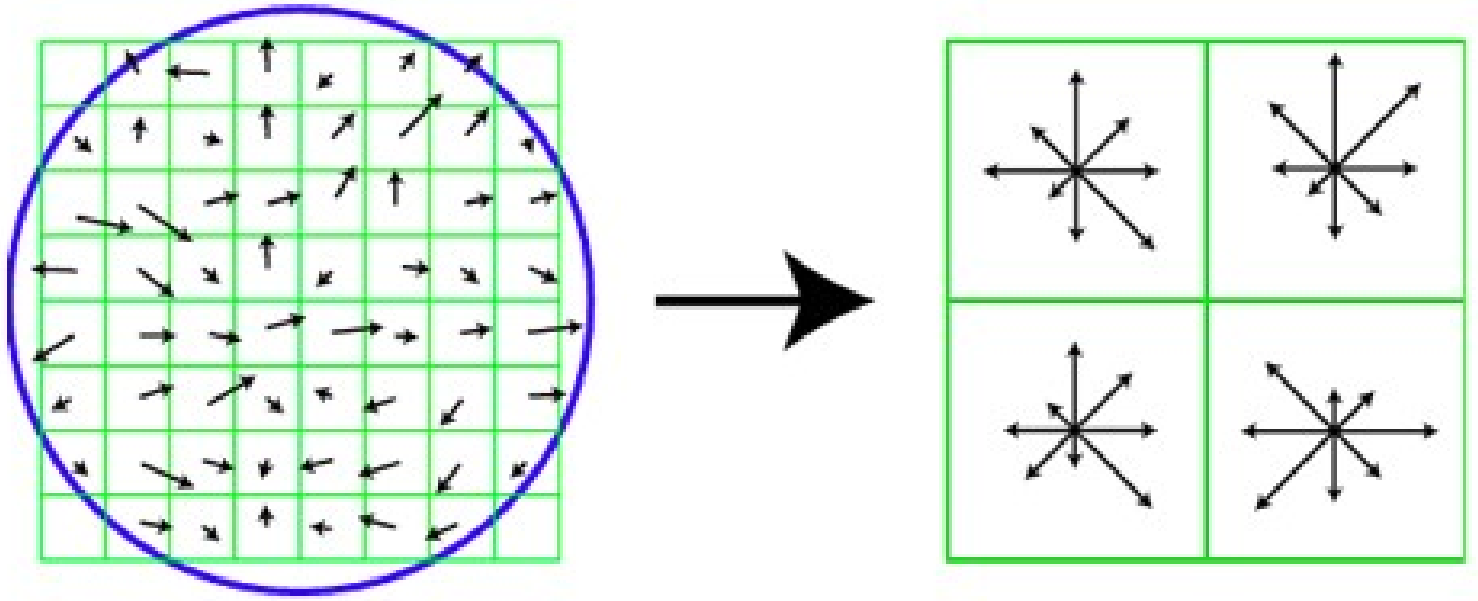
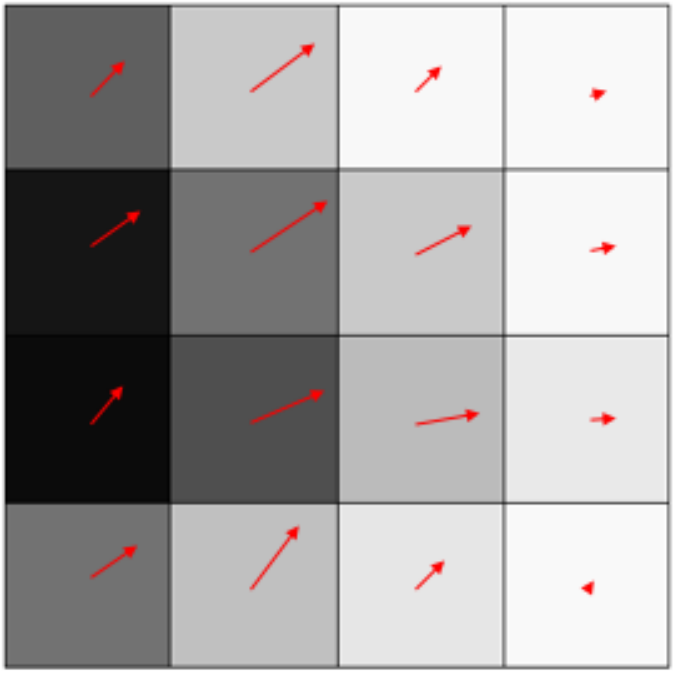


Image gradients

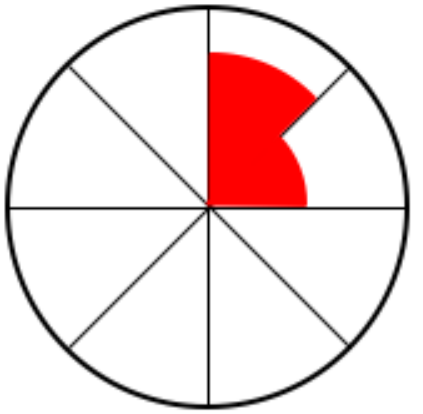
Keypoint descriptor

Lowe 2003

Histogram of gradients



Orientations in each of the 16 pixels of the cell



The orientations all ended up in two bins: 11 in one bin, 5 in the other. (rough count)

5 11 0 0 0 0 0 0 0