

# Проективная геометрия в компьютерном зрении

*Ерухимов Виктор Львович (Itseez), Илья Лысенков (Itseez)*



Kurt Wenner, Disaster

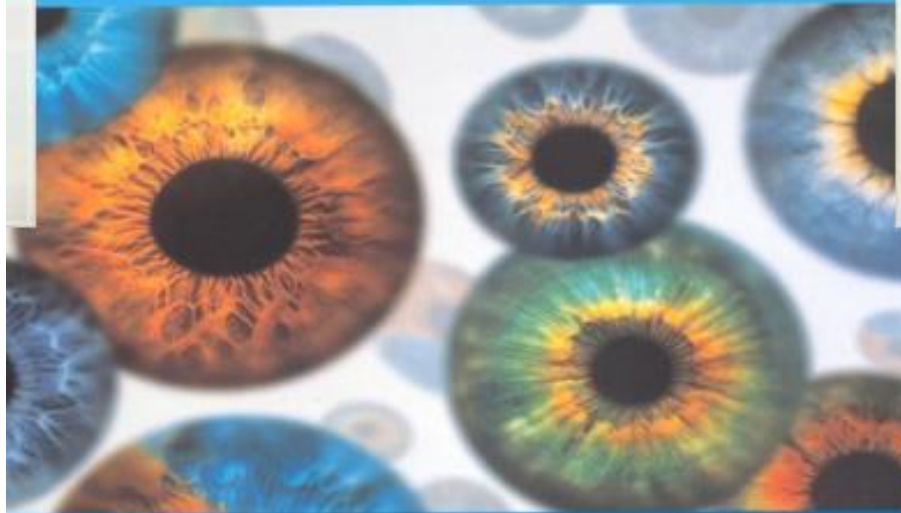
[http://kurtwenner.com/galleries/pavement/pavement\\_3/pages/StreetPaintingGallery3.018.htm](http://kurtwenner.com/galleries/pavement/pavement_3/pages/StreetPaintingGallery3.018.htm)

Copyrighted Material

SECOND EDITION

# Multiple View Geometry

in computer vision



**Richard Hartley and Andrew Zisserman**

Copyrighted Material

CAMBRIDGE



# Зачем нужна геометрия в компьютерном зрении?



©2001 How Stuff Works

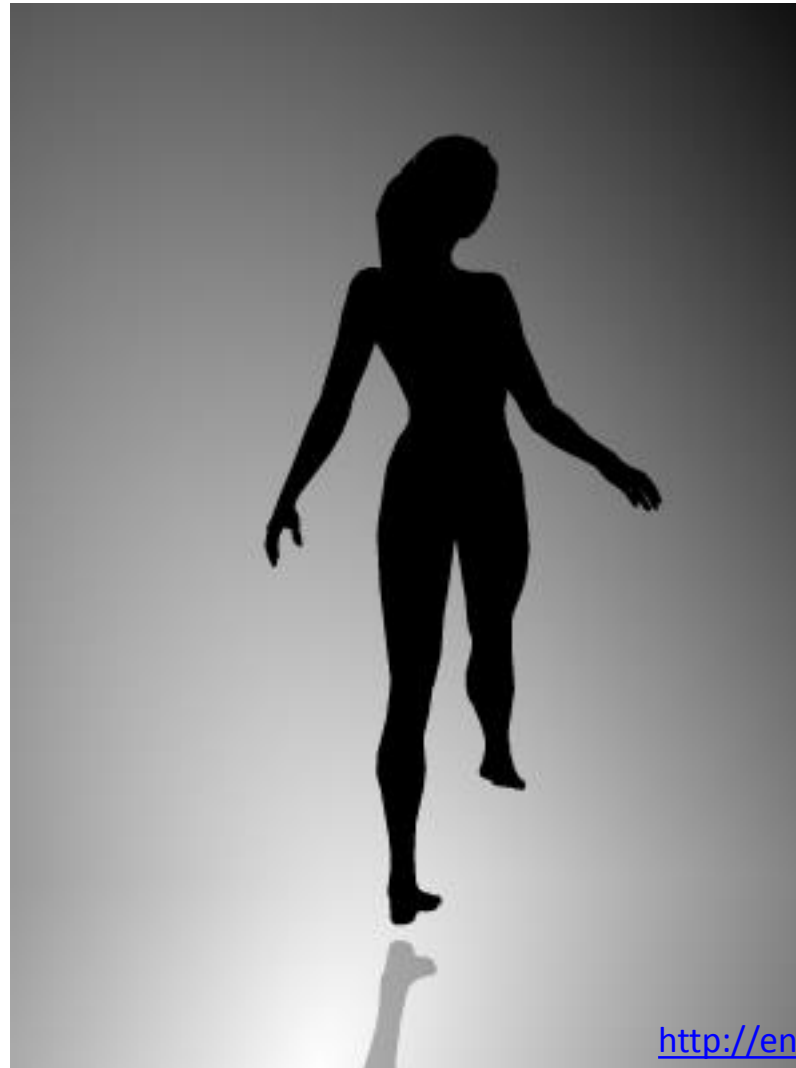


Willow Garage, <http://www.youtube.com/watch?v=gYqfa-YtvW4>  
Garfield, <http://www.imdb.com/media/rm1129879808/tt0356634>

Volvo, <http://bit.ly/162KtRZ>

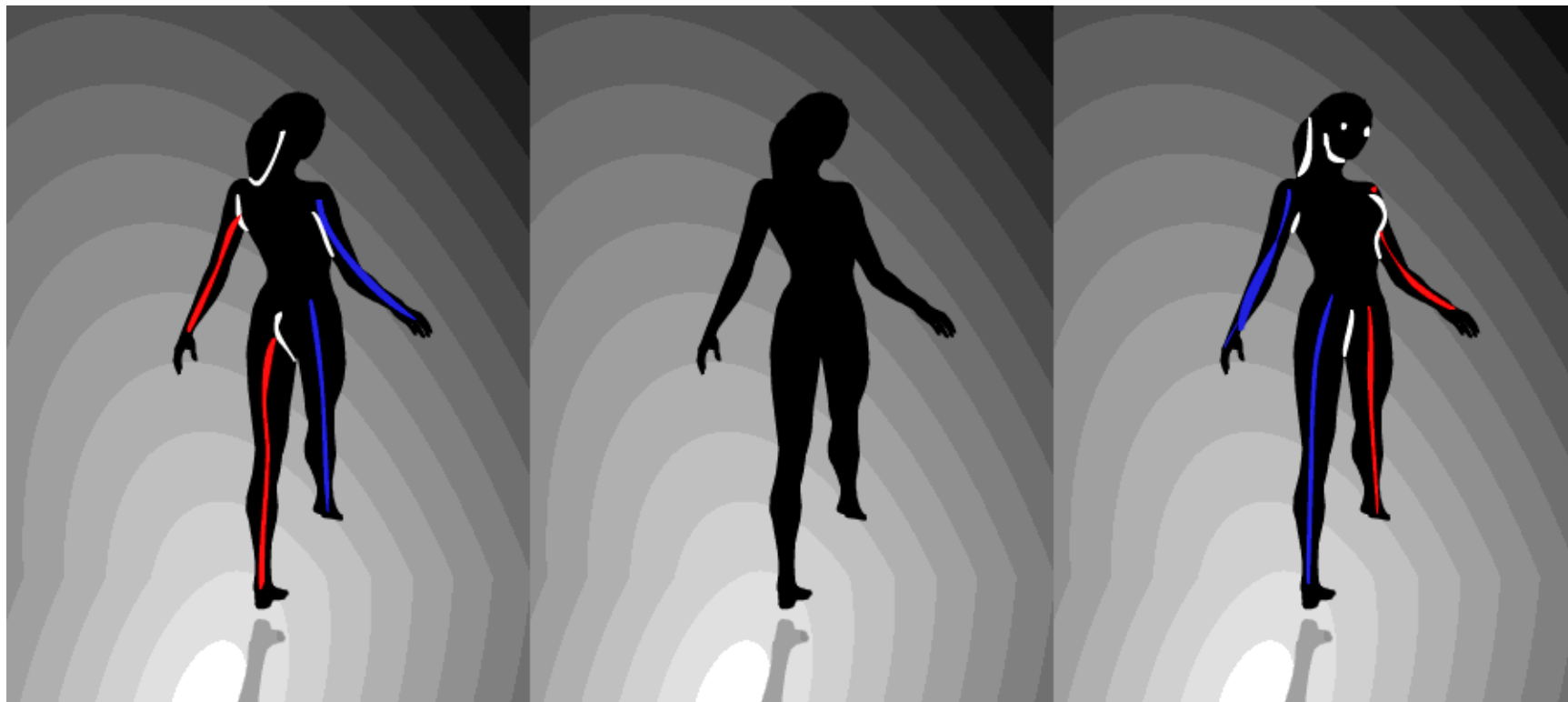
How stuff works, <http://computer.howstuffworks.com>

В какую сторону вращается танцовщица?



[http://en.wikipedia.org/wiki/Spinning\\_Dancer](http://en.wikipedia.org/wiki/Spinning_Dancer)

В какую сторону вращается танцовщица?





# Pinhole camera model

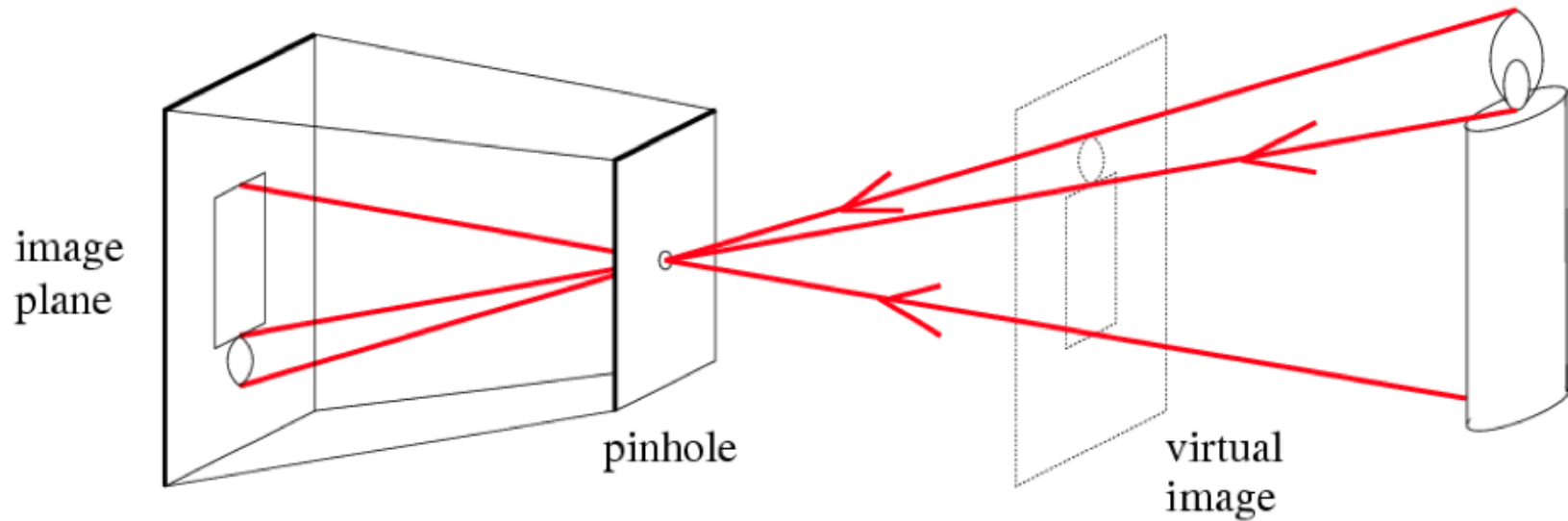
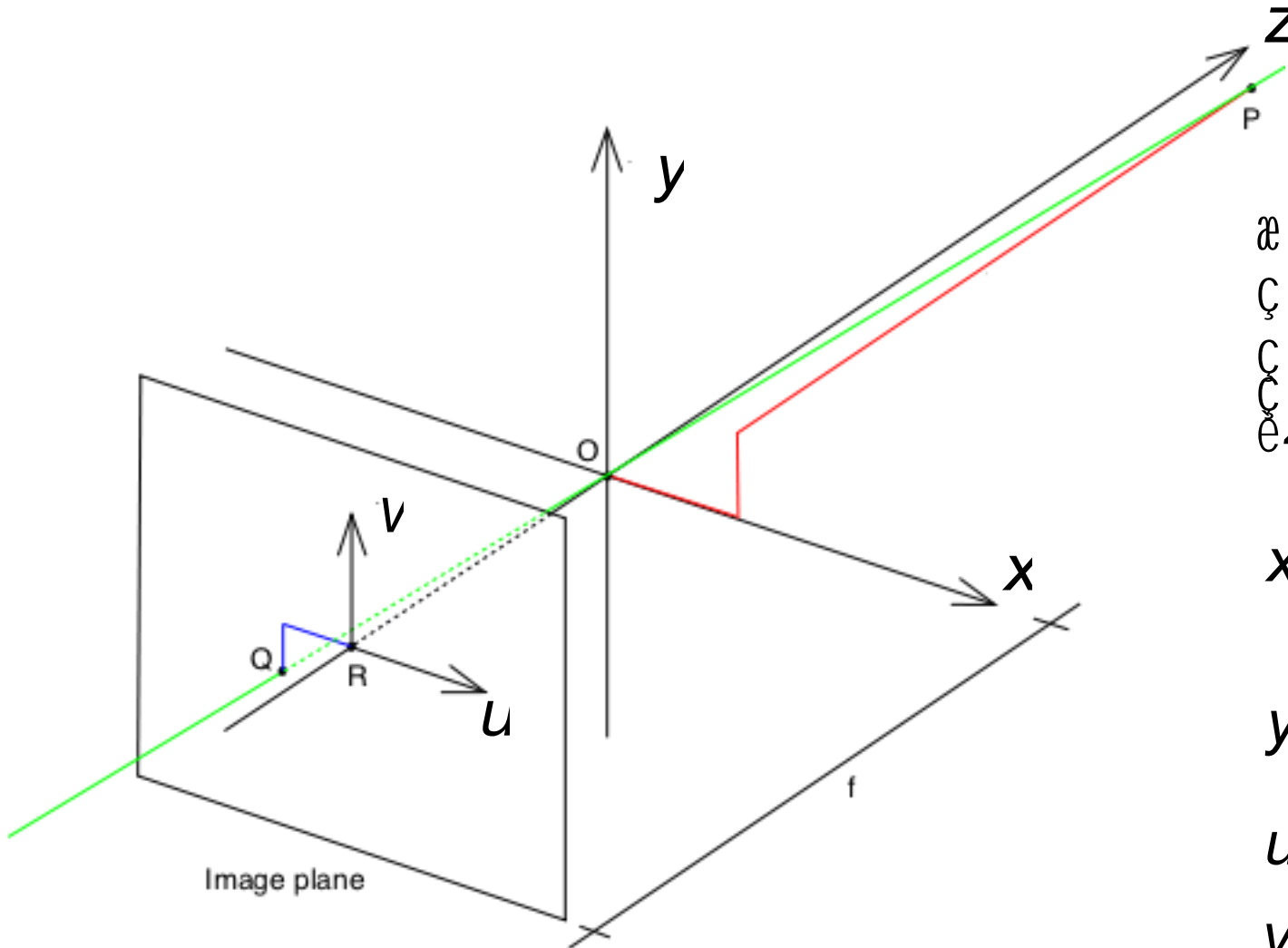


Image from J. Sivic's presentation,

[http://www.ens-lyon.fr/LIP/Arenaire/ERVision/camera\\_geometry\\_alignment\\_final.pdf](http://www.ens-lyon.fr/LIP/Arenaire/ERVision/camera_geometry_alignment_final.pdf)

# Pinhole camera model



$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + t$$

$$x' = \frac{X}{Z}$$

$$y' = \frac{Y}{Z}$$

$$u = f_x x' + c_x$$

$$v = f_y y' + c_y$$



# Camera projection matrix

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$P = K[R|T] \quad K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

# Camera distortion



# Distortion model

$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2)$$

$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y'$$

$$\text{where } r^2 = x'^2 + y'^2$$

$$u = f_x * x'' + c_x$$

$$v = f_y * y'' + c_y$$

# Pose estimation

Detected image points:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix}_{i=1..n}$$

Train object points:

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix}_{i=1..n}$$

A class of object transformations:

$$f \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + T$$



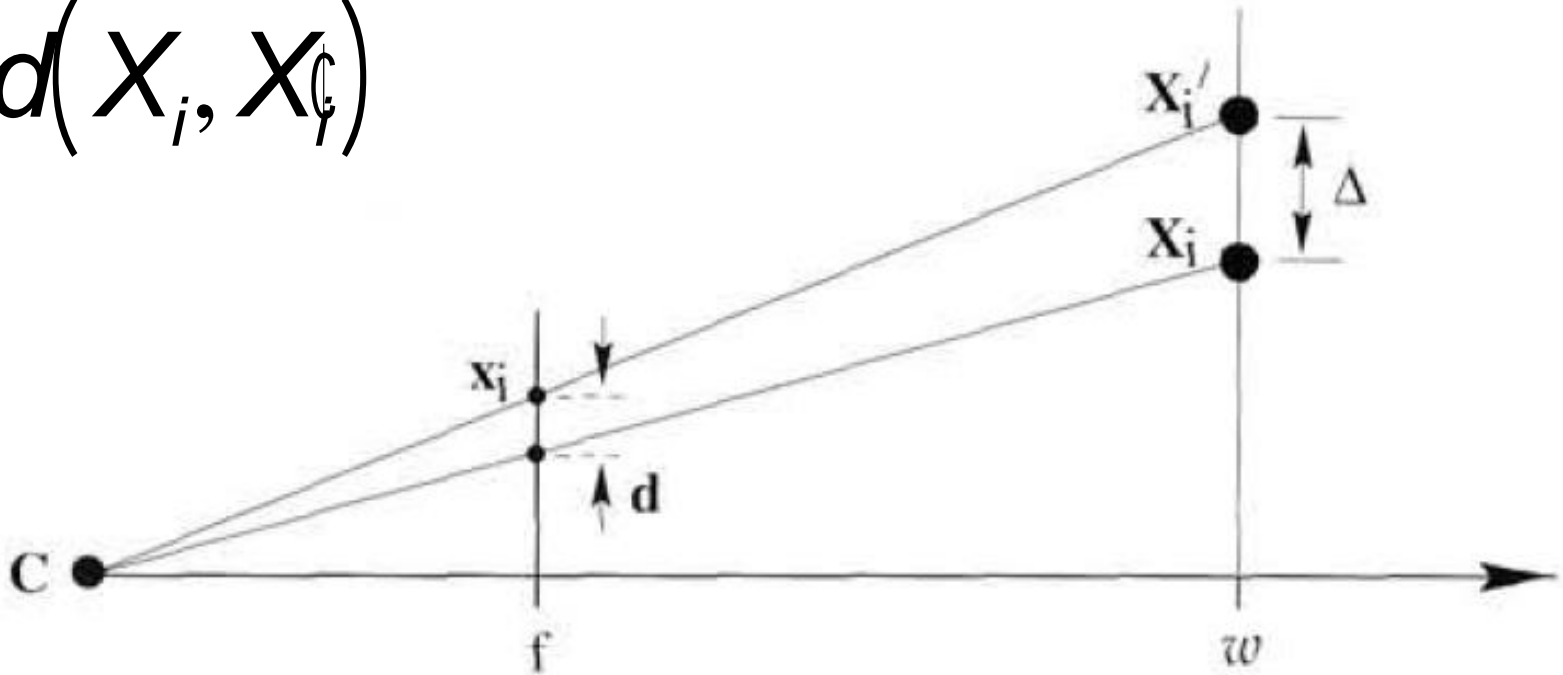
# Reprojection error

$$w_i u_i^p = P \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$error(P) = \sum_i \left( \frac{\|u_i^p\|^2}{\|v_i^p\|^2} - 1 \right)^2$$

# 3D geometric error

$$d(X_i, X_i')$$



# Perspective-n-Points problem

$$\min_{R, T} \text{error}(K, R, T)$$

# Direct Linear Transformation

$$\begin{bmatrix} u_i^p w_i^0 \\ \vdots \\ v_i^p w_i^0 \\ \vdots \\ w_i^0 \\ \vdots \\ 1 \end{bmatrix} = P \begin{bmatrix} X_i^0 \\ \vdots \\ Y_i^0 \\ \vdots \\ Z_i^0 \\ \vdots \\ 1 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} u_i^p \\ \vdots \\ v_i^p \\ \vdots \\ 1 \end{bmatrix} - P \begin{bmatrix} X_i^0 \\ \vdots \\ Y_i^0 \\ \vdots \\ Z_i^0 \\ \vdots \\ 1 \end{bmatrix} = 0$$



# Other methods

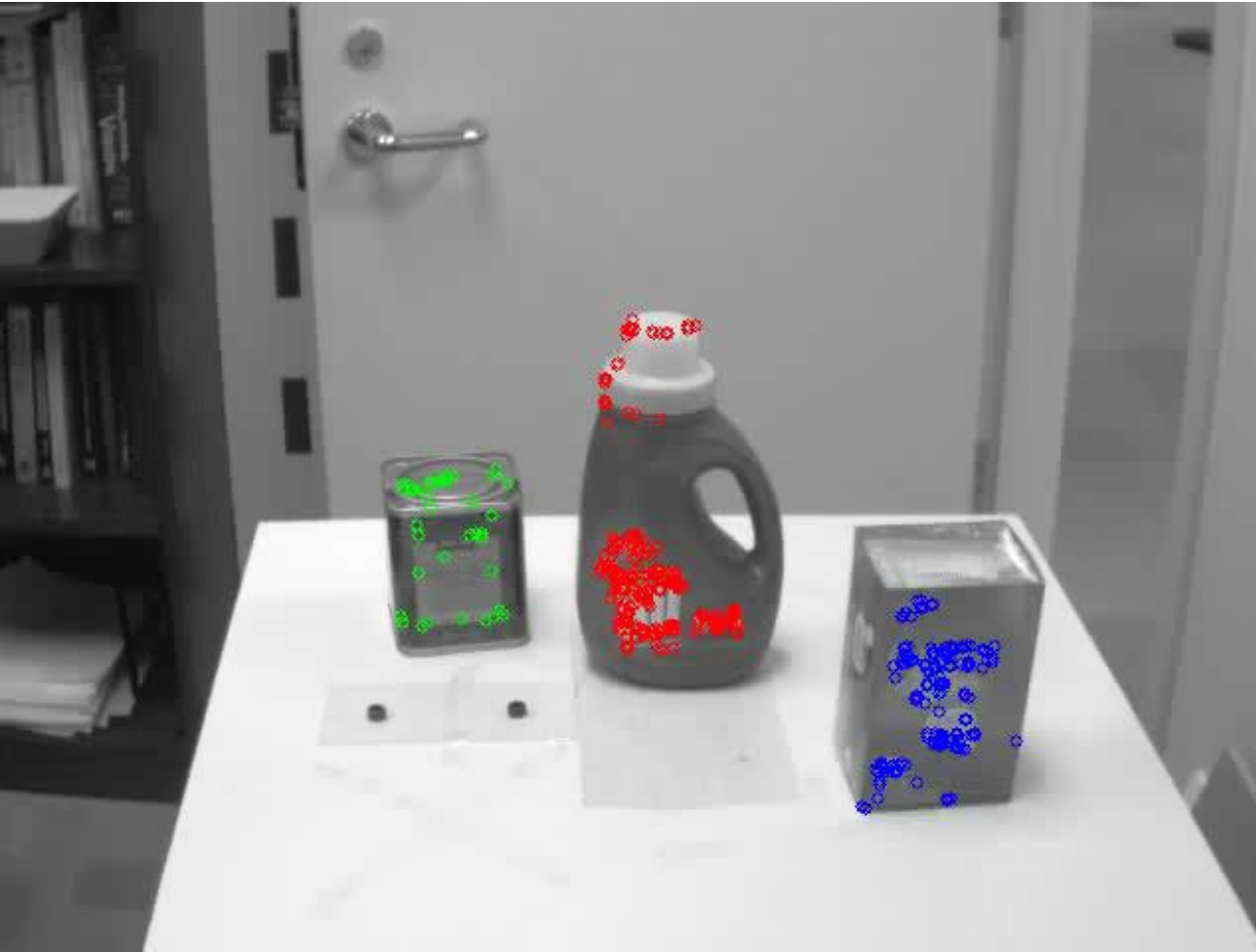
- $O(N)$  closed-form methods
- Levenberg-Marquardt
- P3P, P4P
- RANSAC

# Random Sample Consensus

- Do  $n$  iterations until  $\#inliers > inlierThreshold$ 
  - Draw  $k$  matches randomly
  - Find the transformation
  - Calculate inliers count
  - Remember the best solution

The number of iterations required  $\sim \frac{\# matches^k}{\# inliers}$

# Object detection example

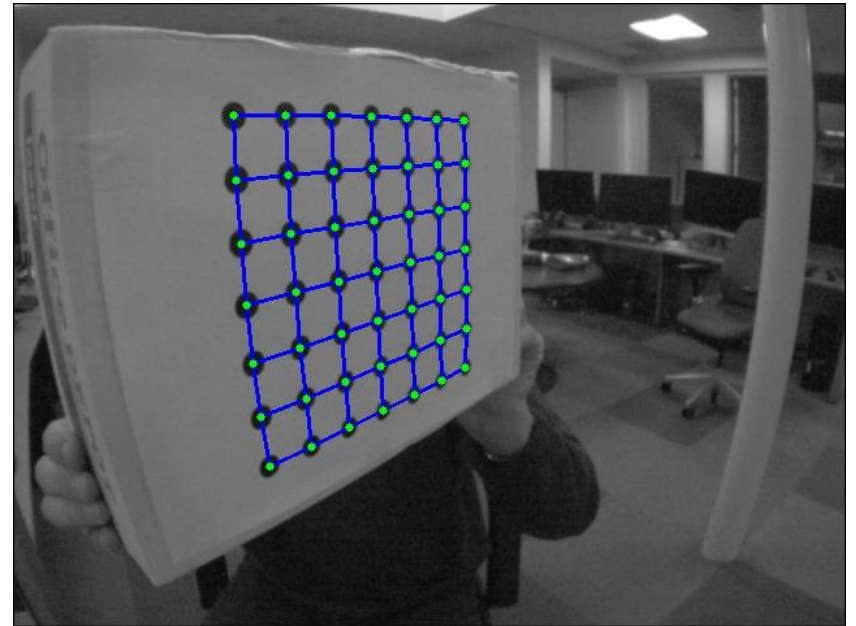


**Iryna Gordon and David G. Lowe**, "What and where: 3D object recognition with accurate pose," in *Toward Category-Level Object Recognition*, eds. J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, (Springer-Verlag, 2006), pp. 67-82.

**Manuel Martinez Torres, Alvaro Collet Romea, and Siddhartha Srinivasa**, MOPED: A Scalable and Low Latency Object Recognition and Pose Estimation System, Proceedings of ICRA 2010, May, 2010.

# Camera calibration

- Estimate camera parameters given images of a known template



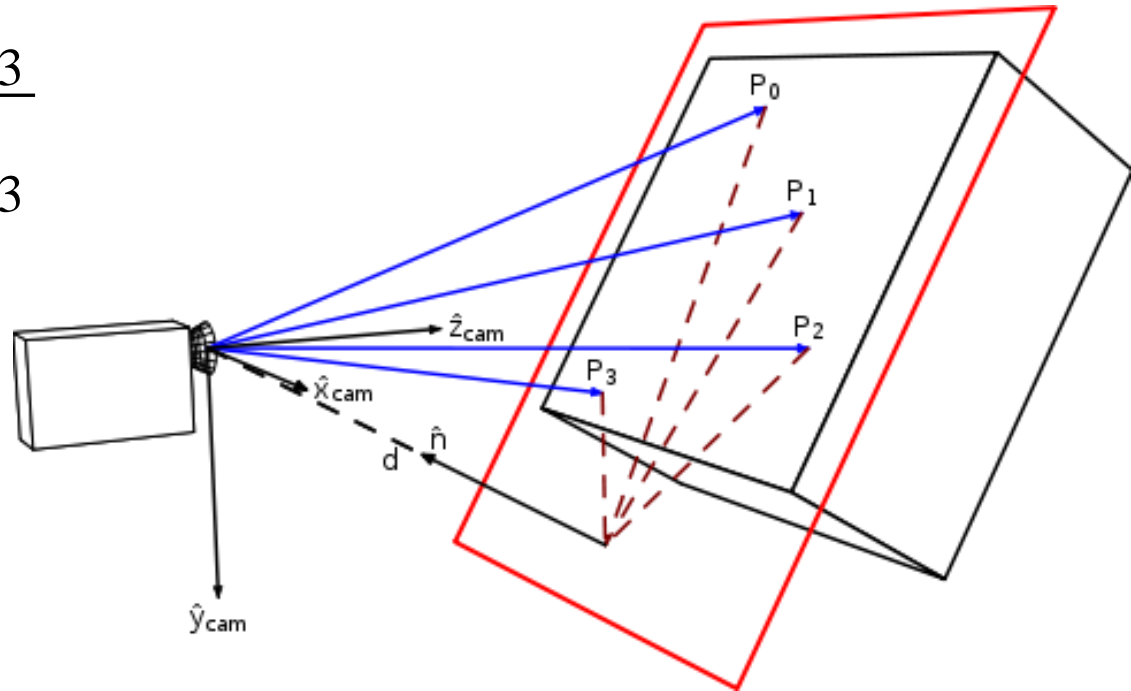


# Homography

$$\tilde{u} = \frac{h_{11}u + h_{12}v + h_{13}}{h_{31}u + h_{32}v + h_{33}}$$

$$\tilde{v} = \frac{h_{21}u + h_{22}v + h_{23}}{h_{31}u + h_{32}v + h_{33}}$$

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ 1 \end{bmatrix} = H \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

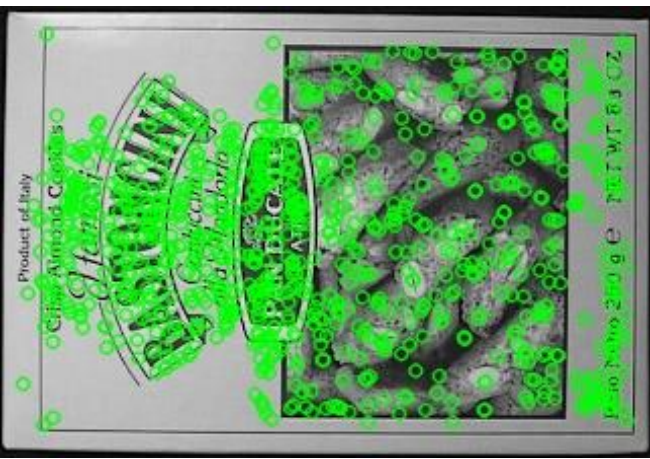


# Нахождение гомографии

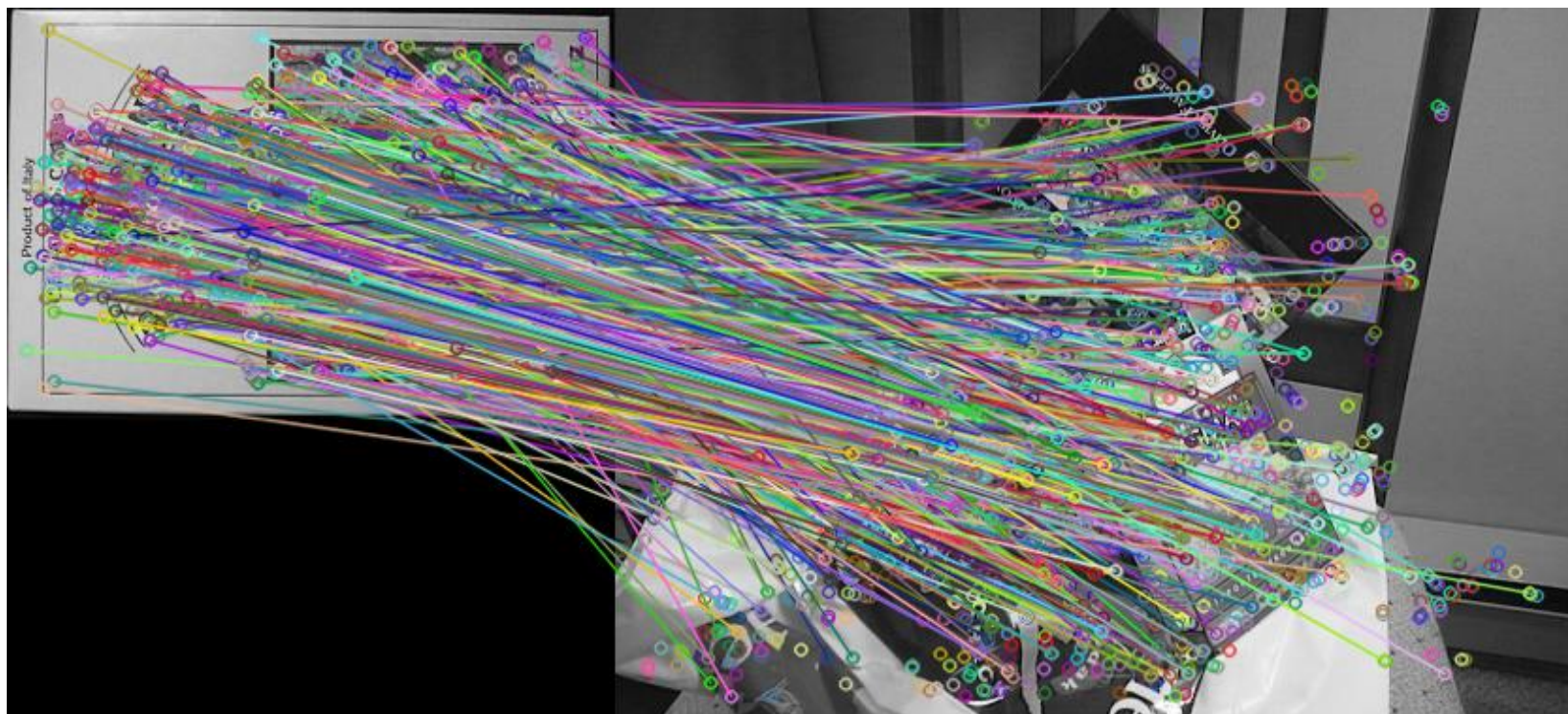
- Direct Linear Transformation

$$\begin{array}{c}
 \begin{array}{cc}
 \begin{array}{c} \tilde{u}_i \\ \vdots \\ \tilde{v}_i \\ \vdots \\ 1 \\ \vdots \\ 0 \end{array} & = H \begin{array}{c} u_i \\ \vdots \\ v_i \\ \vdots \\ 1 \\ \vdots \\ 0 \end{array}
 \end{array}
 \end{array}
 \Rightarrow
 \begin{array}{c}
 \begin{array}{cc}
 \begin{array}{c} \tilde{u}_i \\ \vdots \\ \tilde{v}_i \\ \vdots \\ 1 \\ \vdots \\ 0 \end{array} & - H \begin{array}{c} u_i \\ \vdots \\ v_i \\ \vdots \\ 1 \\ \vdots \\ 0 \end{array} = 0
 \end{array}
 \end{array}$$

# Keypoints example



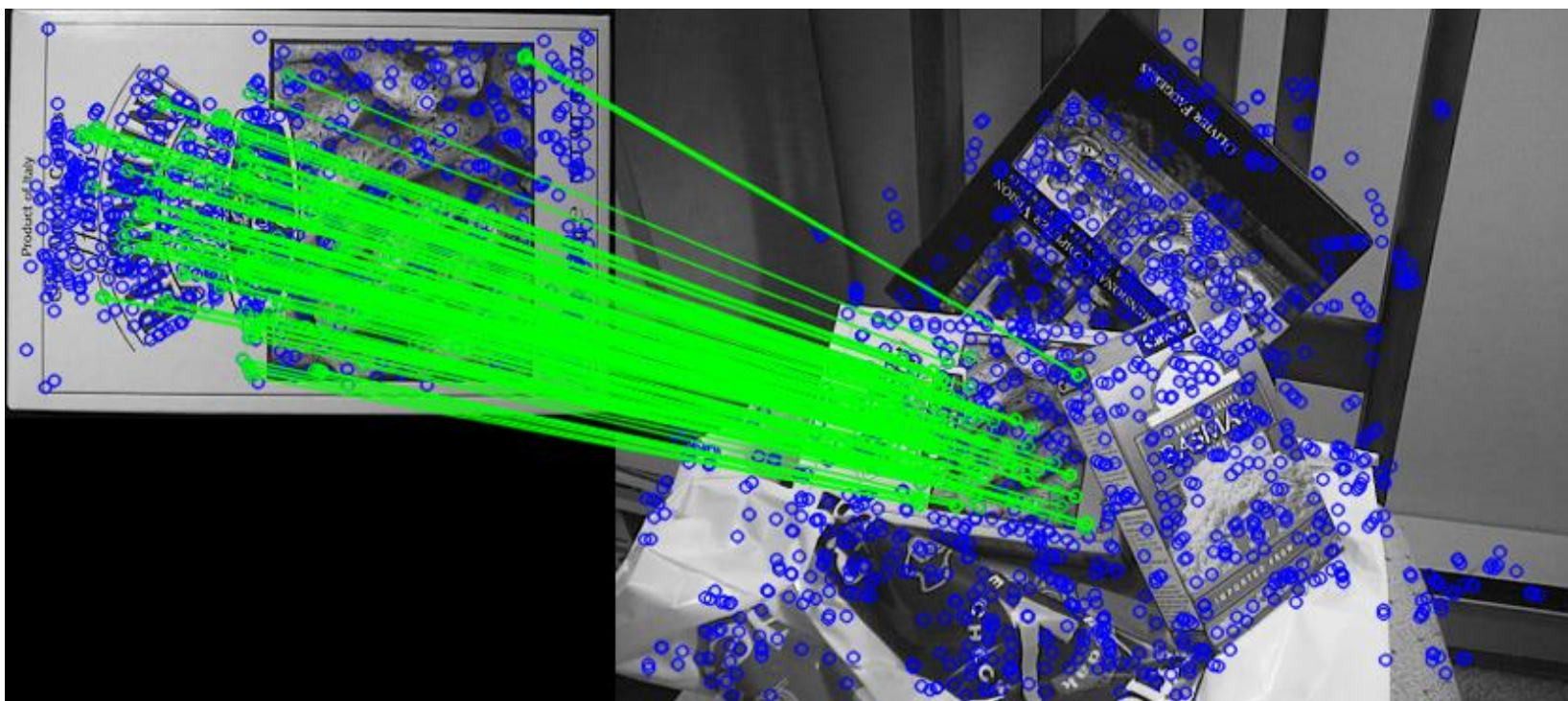
# Matching descriptors example



OpenCV features2d tutorial,  
[http://docs.opencv.org/doc/tutorials/features2d/table\\_of\\_content\\_features2d/table\\_of\\_content\\_features2d.html](http://docs.opencv.org/doc/tutorials/features2d/table_of_content_features2d/table_of_content_features2d.html)



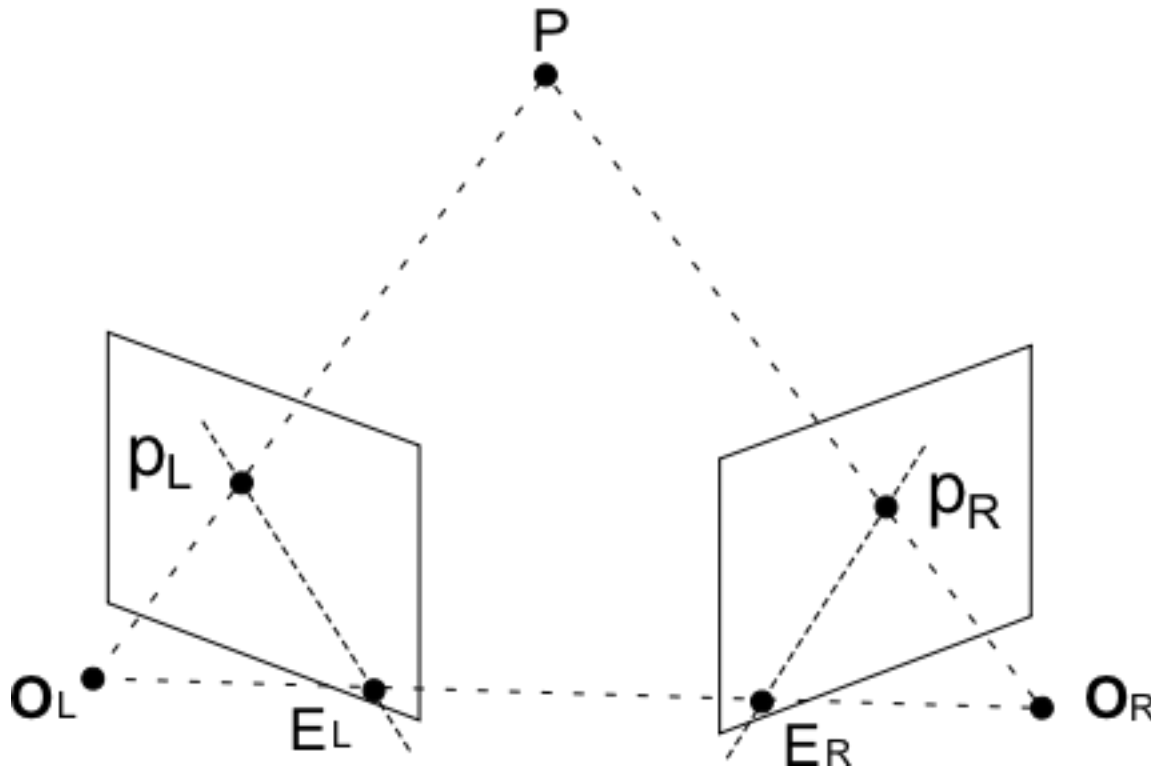
# Geometry validation



OpenCV features2d tutorial,

[http://docs.opencv.org/doc/tutorials/features2d/table\\_of\\_content\\_features2d/table\\_of\\_content\\_features2d.html](http://docs.opencv.org/doc/tutorials/features2d/table_of_content_features2d/table_of_content_features2d.html)

# Stereo: epipolar geometry



Fundamental/essential matrix constraint

$$(u_1, v_1, 1) \times F \times \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = 0$$

$$(x_1, y_1, 1) \times E \times \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = 0$$

# Нахождение фундаментальной матрицы

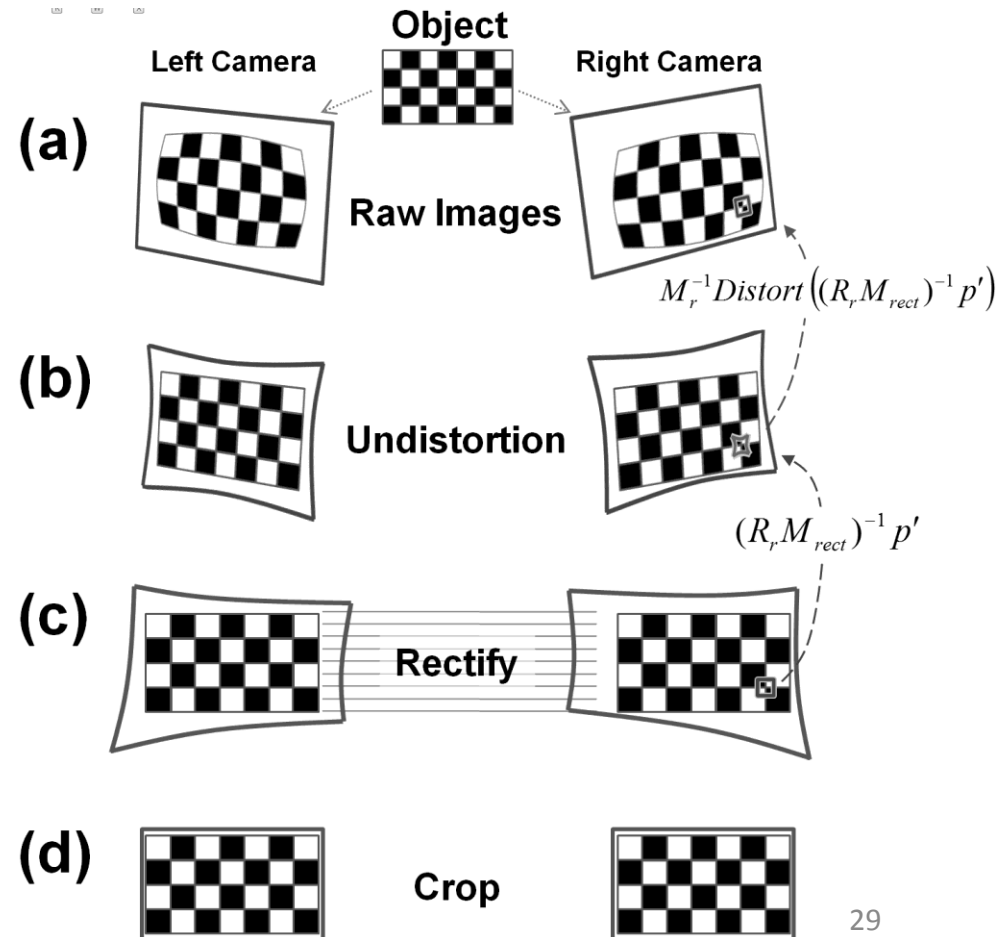
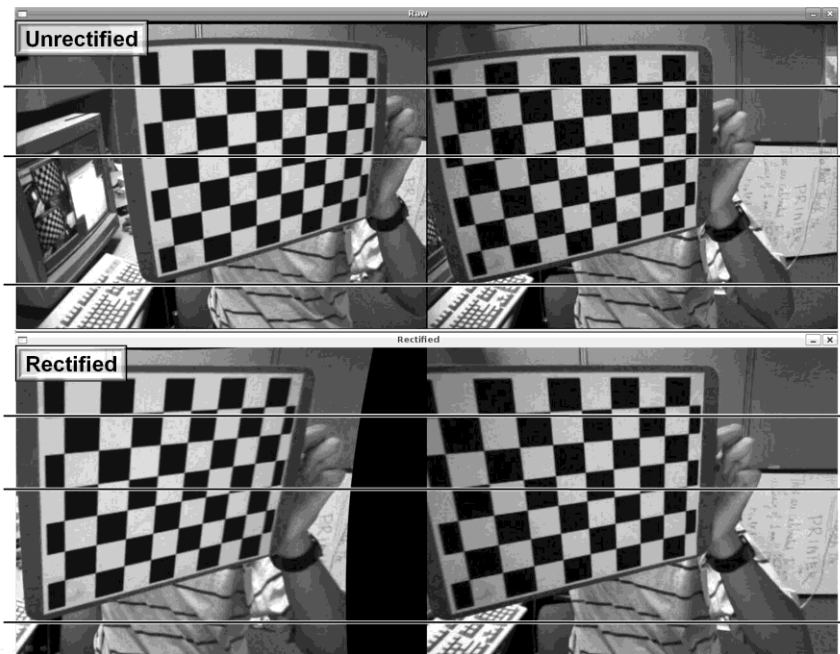
- По 8 соответствиям
  - Опциональная нормализация для устойчивости
  - Решение системы уравнений  $q_i'^T F q_i = 0$
  - Замена  $F$  на ближайшую сингулярную
- По 7 соответствиям
  - Общее решение имеет вид  $F = aF_1 + (1-a)F_2$
  - Используем условие  $\det(F) = 0$  и получаем уравнение 3-й степени относительно  $F$

# The Fundamental Matrix Song



# Stereo Rectification

- Algorithm steps are shown at right:
- Goal:
  - Each row of the image contains the same world points
  - “Epipolar constraint”



# Stereo correspondence block matching

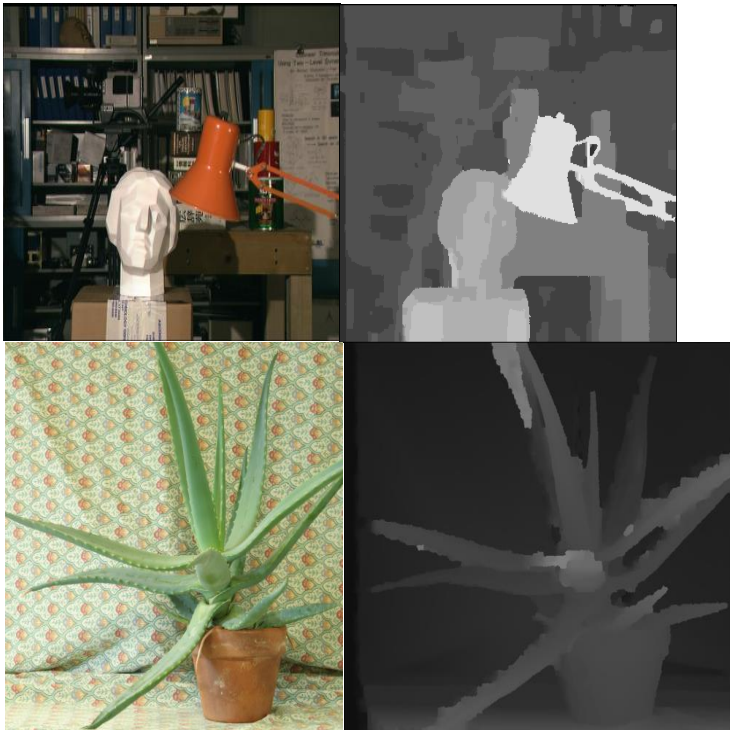
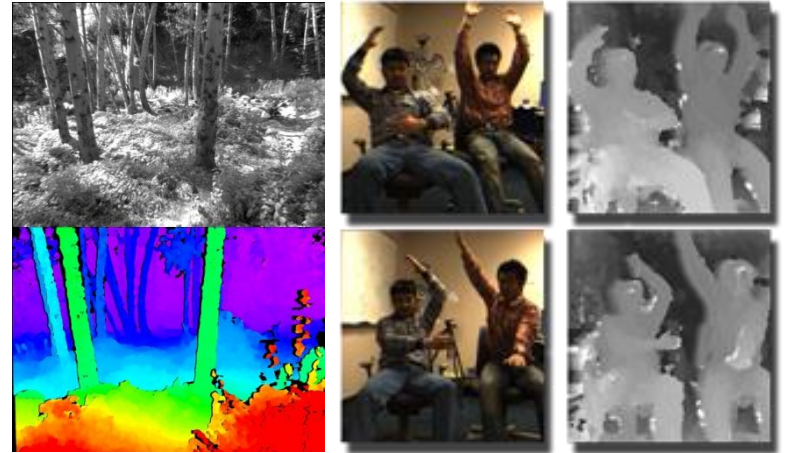


For each block in left image:

Search for the corresponding block in the right image such that SSD or SAD between pixel intensities is minimum



# Stereo Matching



$$Z = f_x \frac{T}{D}$$

# OpenCV functions: single view

- SVD
- projectPoints
- undistort
- solvePnP
- calibrateCamera
- findChessboardCorners/findCirclesGrid

# OpenCV functions: two views

- findHomography
- findFundamentalMat
- stereoCalibrate
- stereoRectify/reprojectImageTo3D/initUndistortRectifyMap