Computer vision for robotics

Victor Eruhimov

CTO, itseez

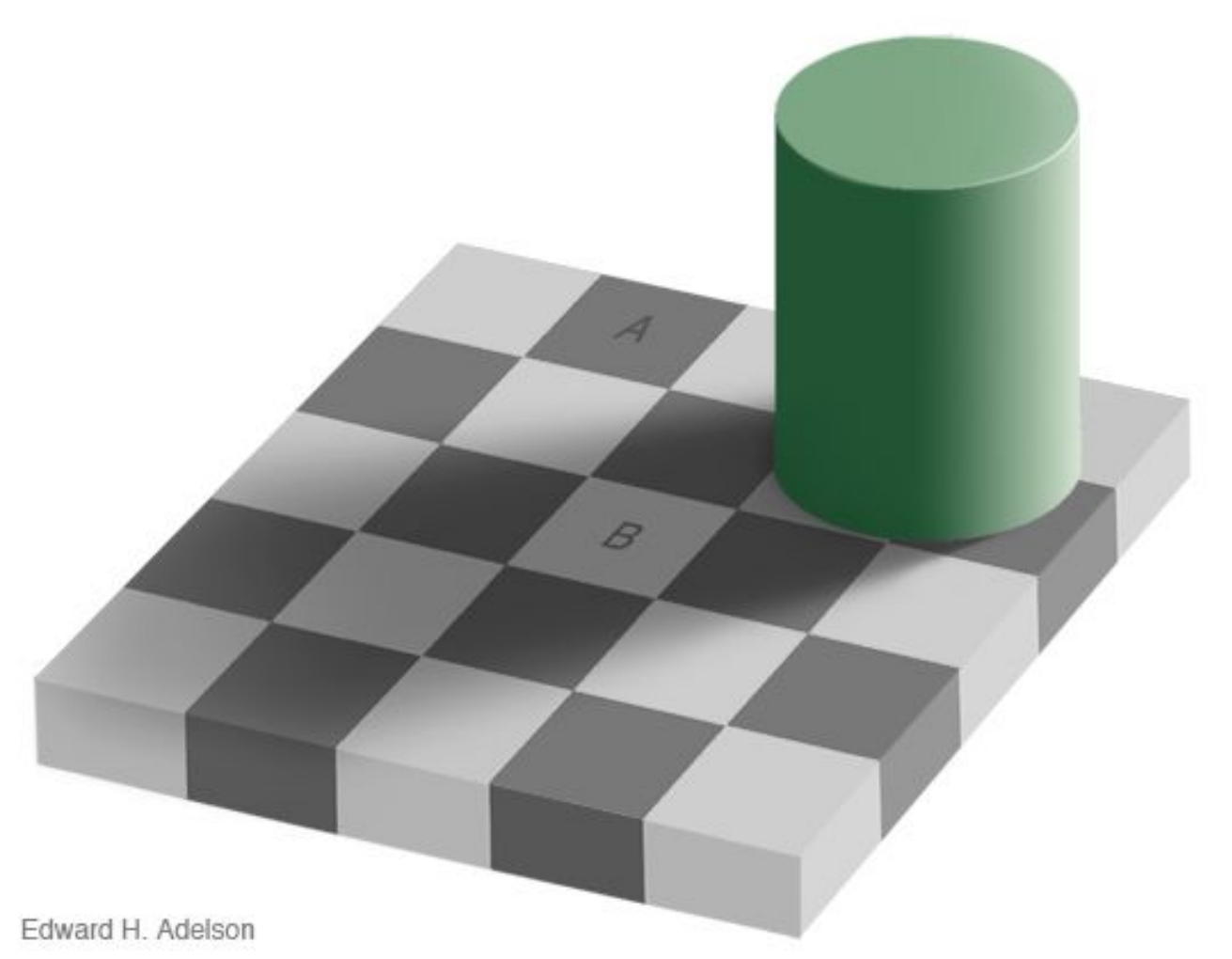
http://www.itseez.com

itseez

Why do we need computer vision?

- Smart video surveillance
- Biometrics
- Automatic Driver Assistance Systems
- Machine vision (Visual inspection)
- Image retrieval (e.g. Google Goggles)
- Movie production
- Robotics

Vision is hard! Even for humans…



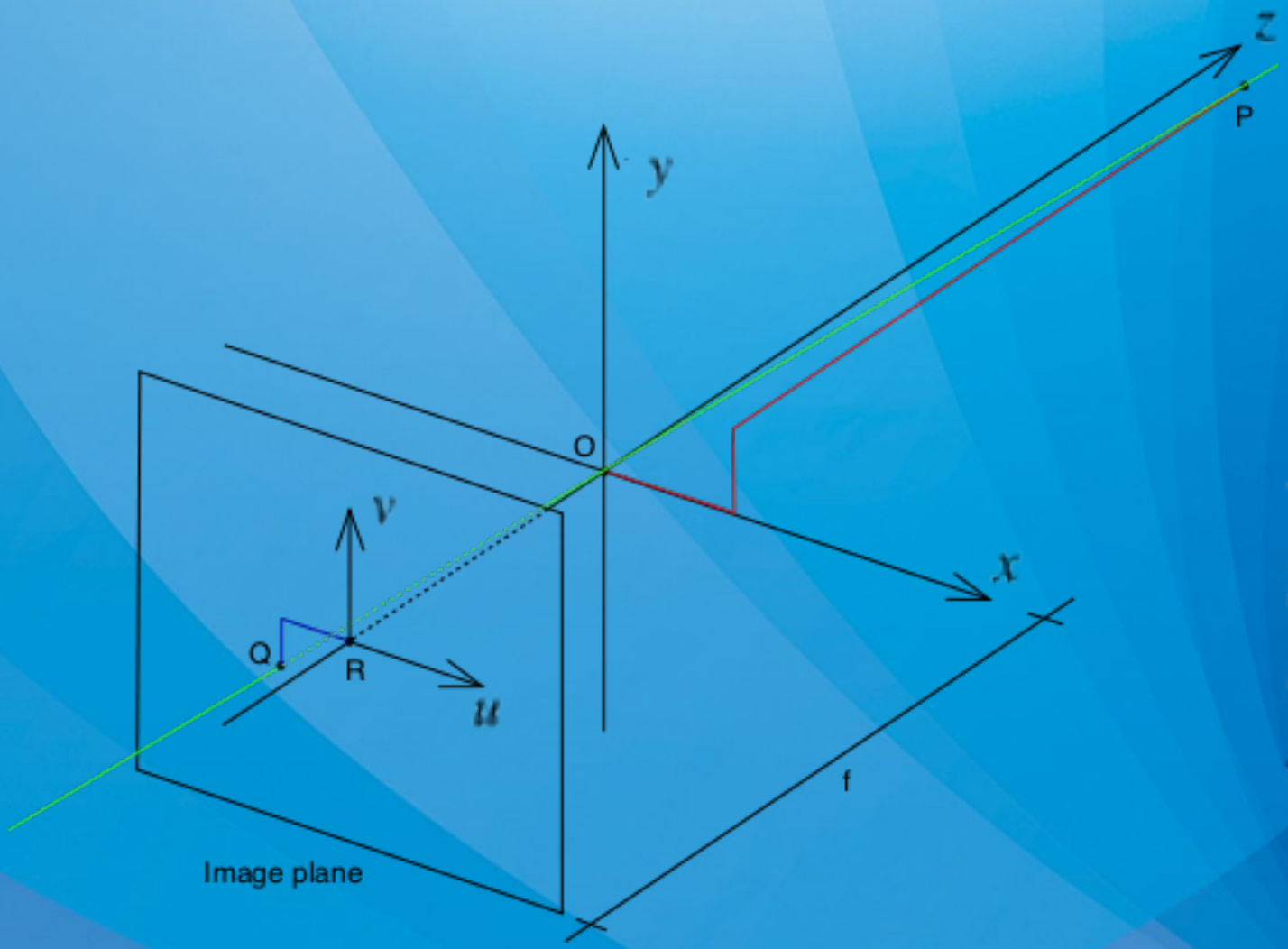
Texai parking

Agenda

- Camera model
- Stereo vision
  - Stereo vision on GPU
- Object detection methods
  - Sliding window
  - Local descriptors
- Applications
  - Textured object detection
  - Outlet detection
  - Visual odometry

itseez

Pinhole camera model



$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$

$$y' = y/z$$

$$u = f_x x' + c_x$$

$$v = f_y y' + c_y$$

Distortion model

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$

$$y' = y/z$$

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

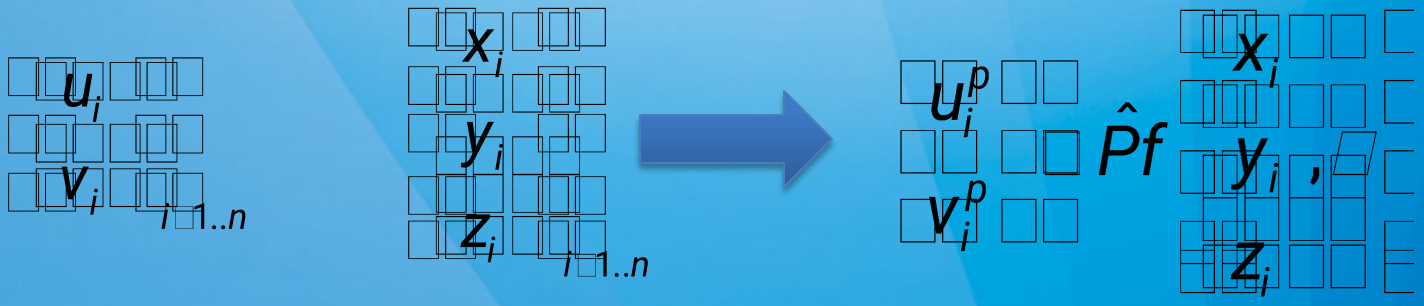
$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y'$$

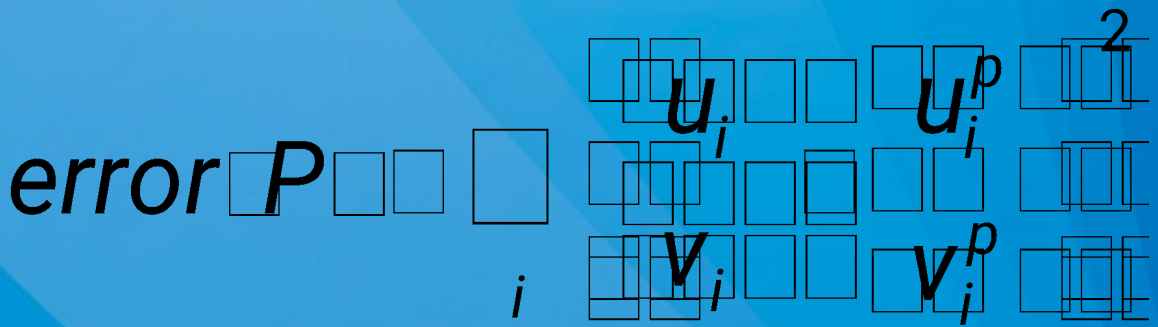
where $r^2 = x'^2 + y'^2$

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

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

Reprojection error



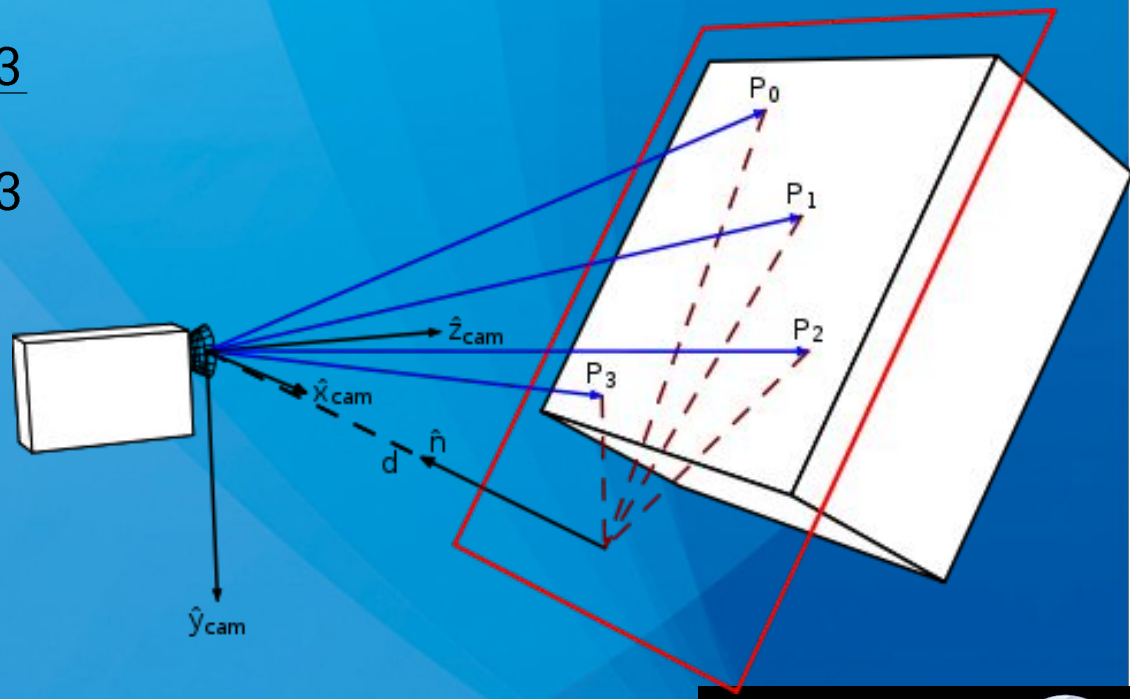


Homography

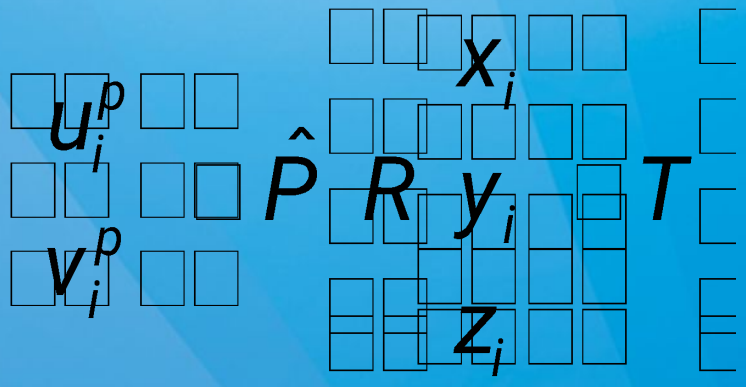
$$\tilde{u} = \frac{h_{11}u + h_{12}v + h_{13}}{h_{31}u + h_{32}v + h_{33}}$$

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

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

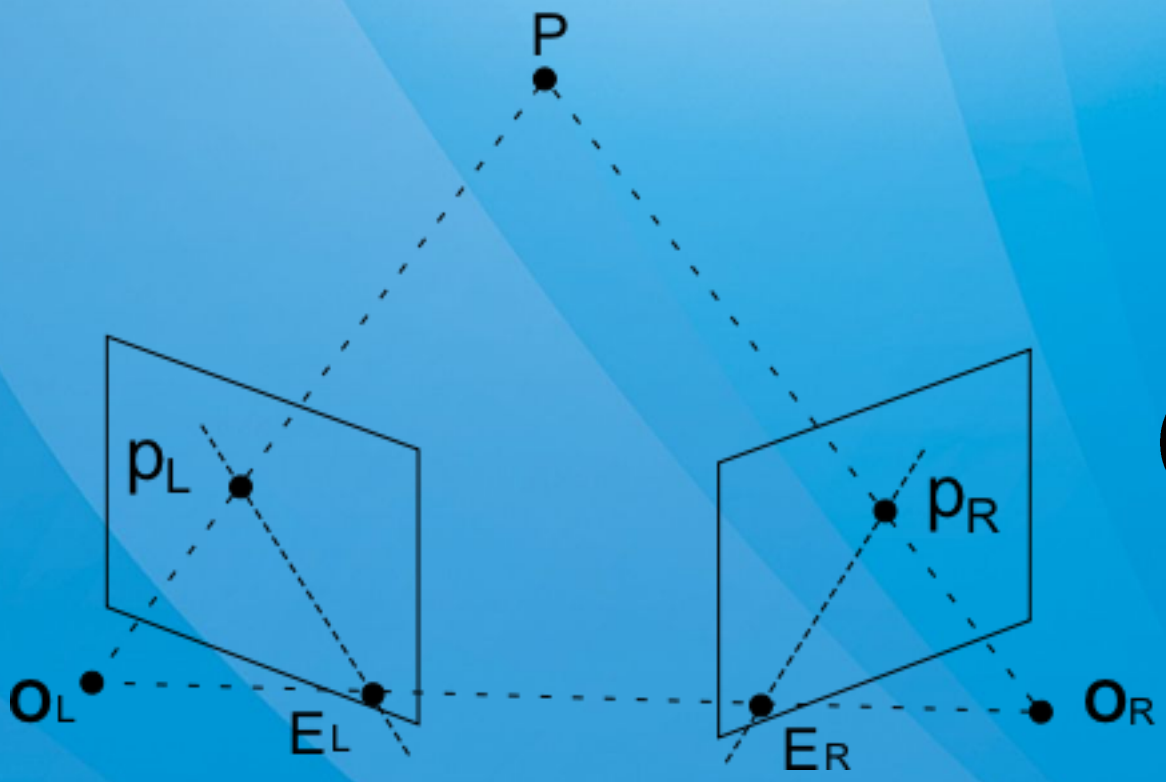


Perspective-n-Points problem



- P4P
- RANSAC (RANdom SAmple Consensus)

Stereo: epipolar geometry



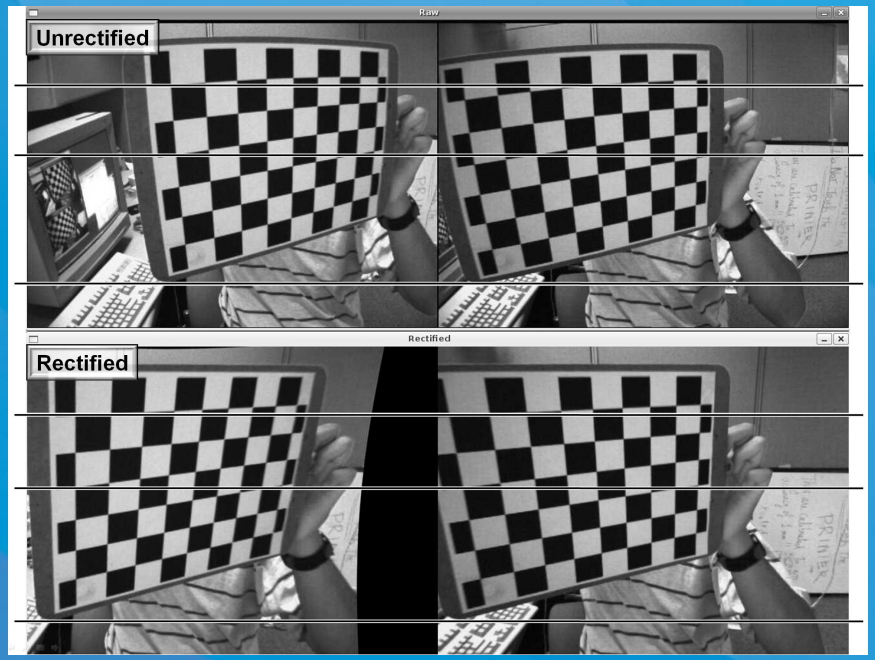
Fundamental matrix constraint

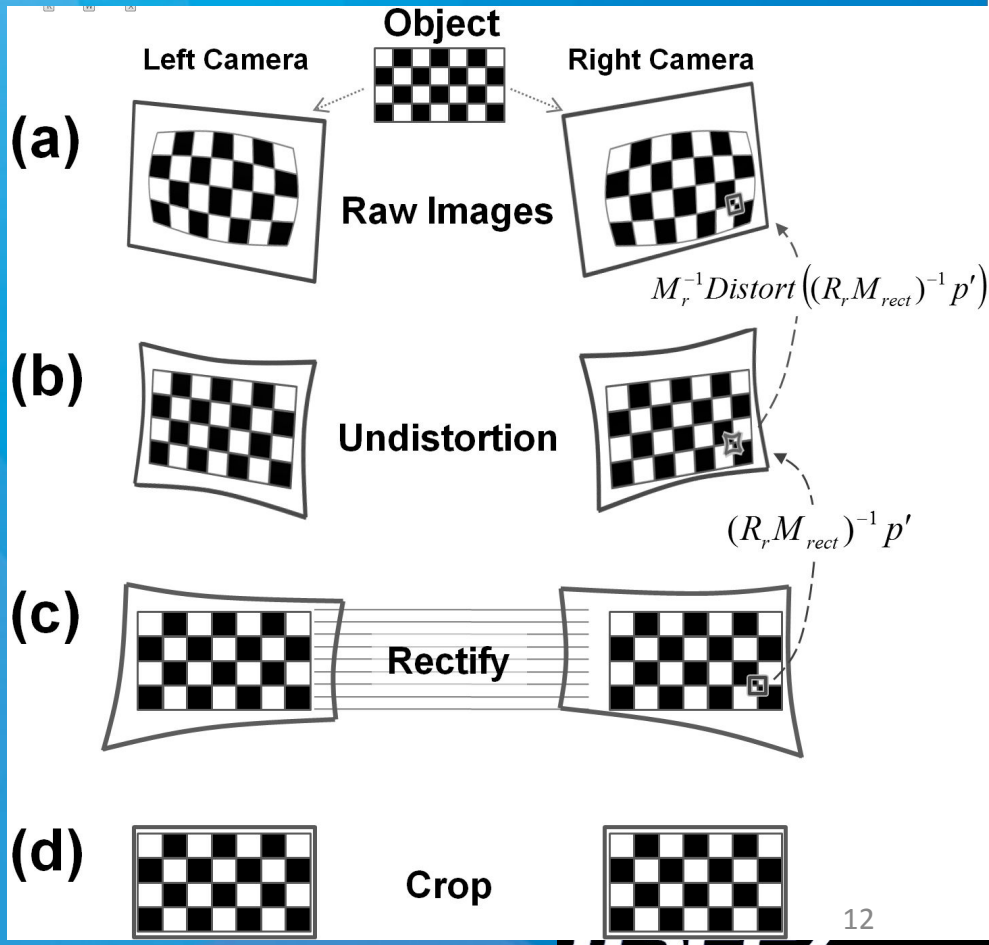
$$\left(x_L, y_L, 1\right) \cdot F \cdot \begin{pmatrix} x_R \\ y_R \\ 1 \end{pmatrix} = 0$$

Stereo Rectification

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

Result: Epipolar alignment of features:



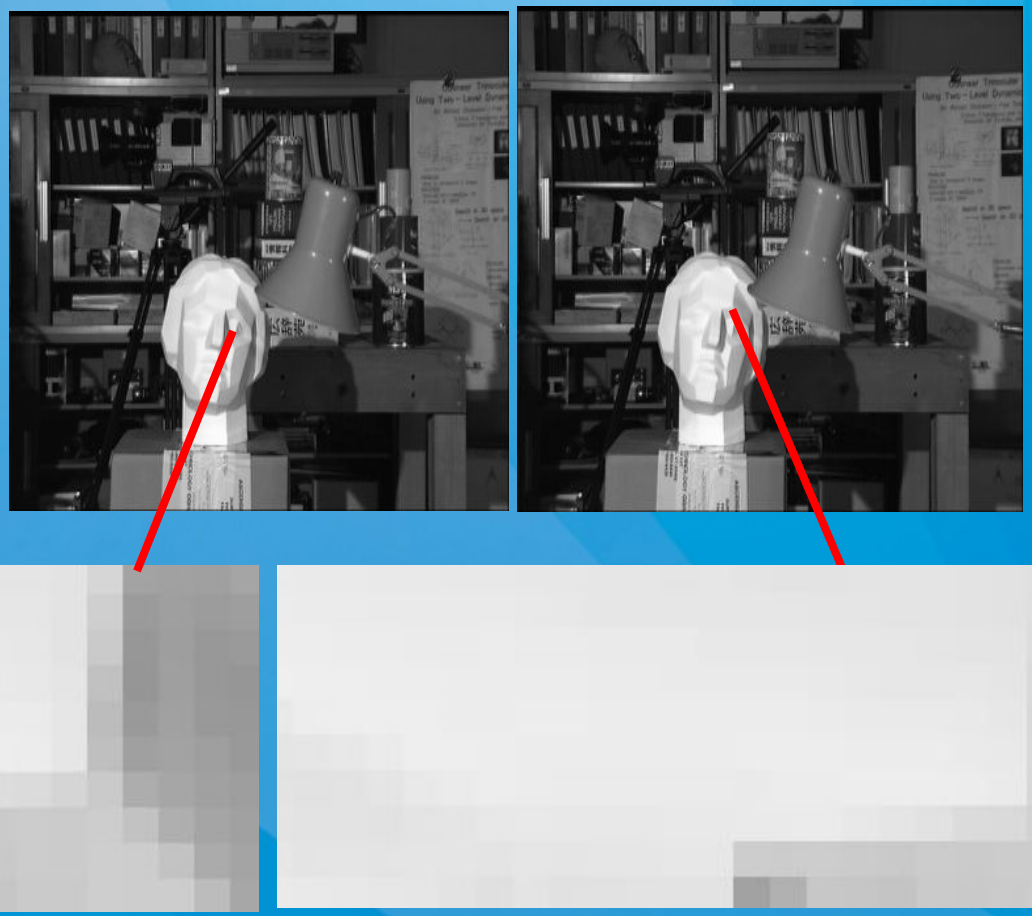


All: Gary Bradski and Adrian Kaehler: Learning OpenCV

Stereo correspondence

- Block matching
- Dynamic programming
- Inter-scanline dependencies
  - Segmentation
  - Belief propagation

Stereo correspondence block matching

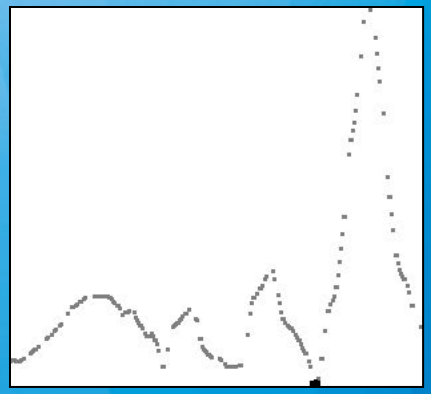


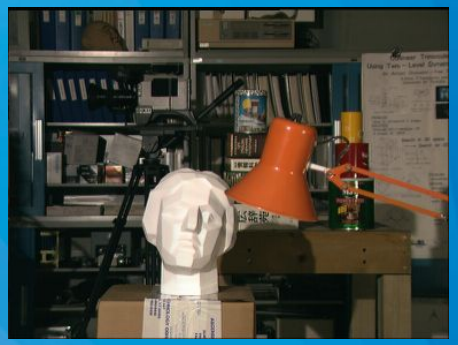
For each block in left image:

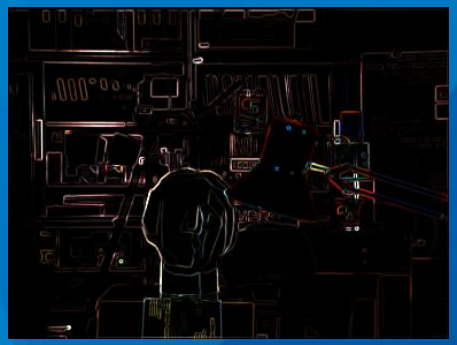
- Search for the corresponding block in the right image such that SSD or SAD between pixel intensities is minimum

Pre- and post processing

- Low texture filtering
- SSD/SAD minimum ambiguity removal
- Using gradients instead of intensities
- Speckle filtering

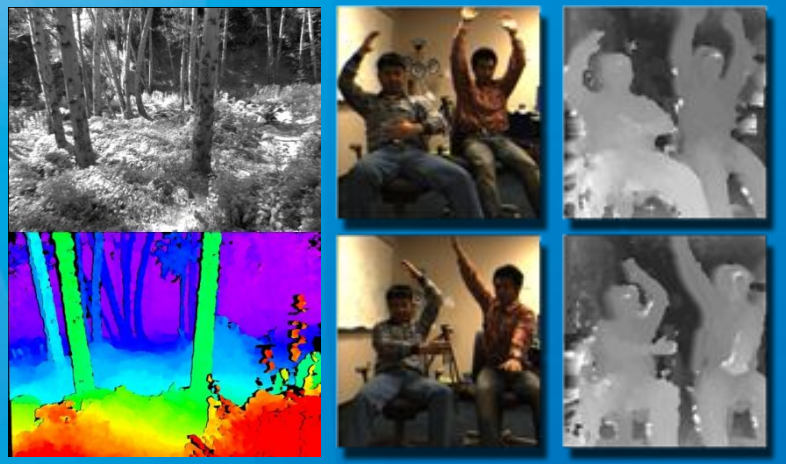


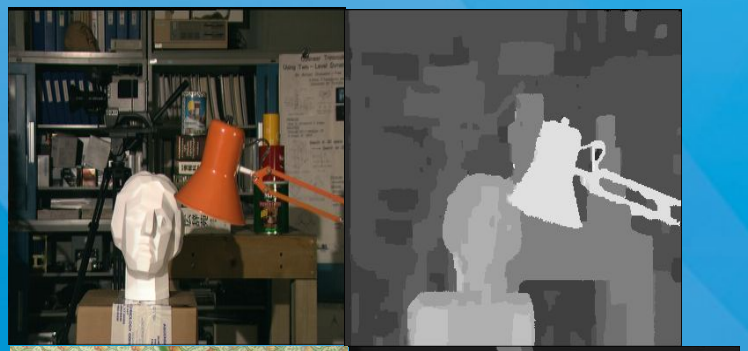


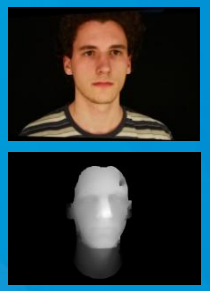


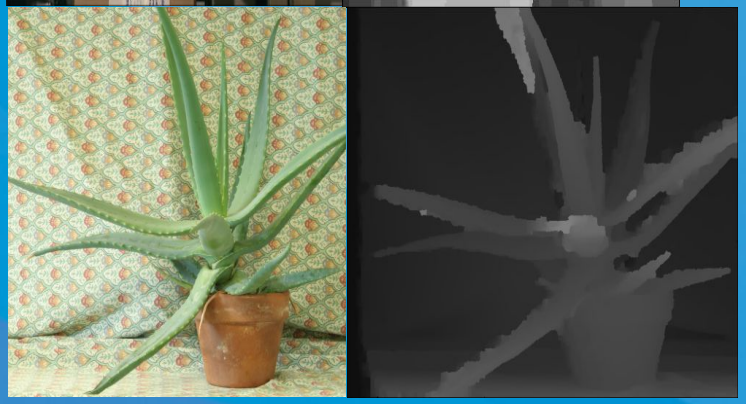
itseez

Stereo Matching



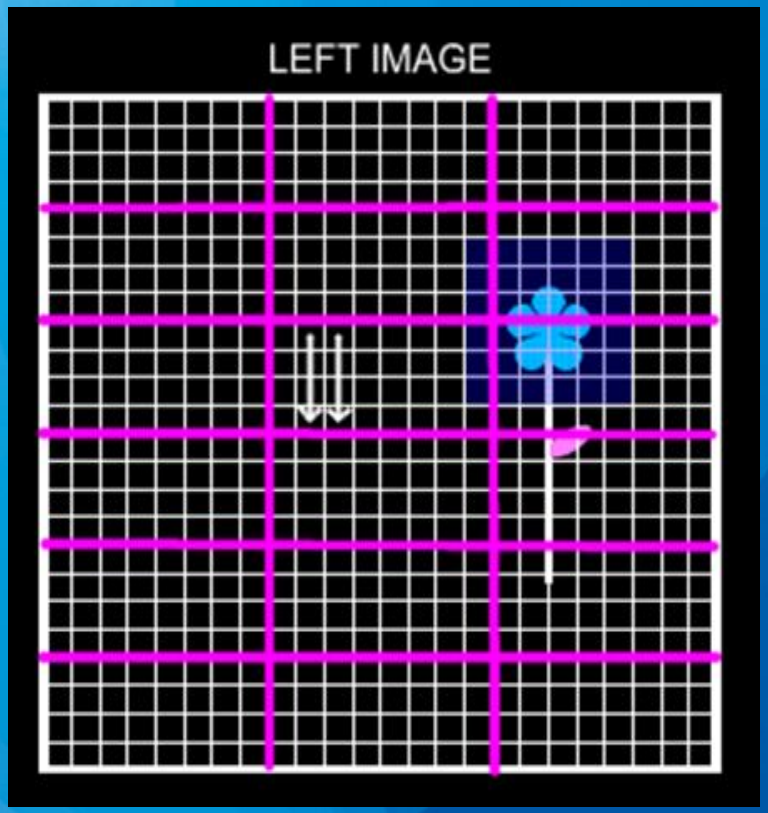




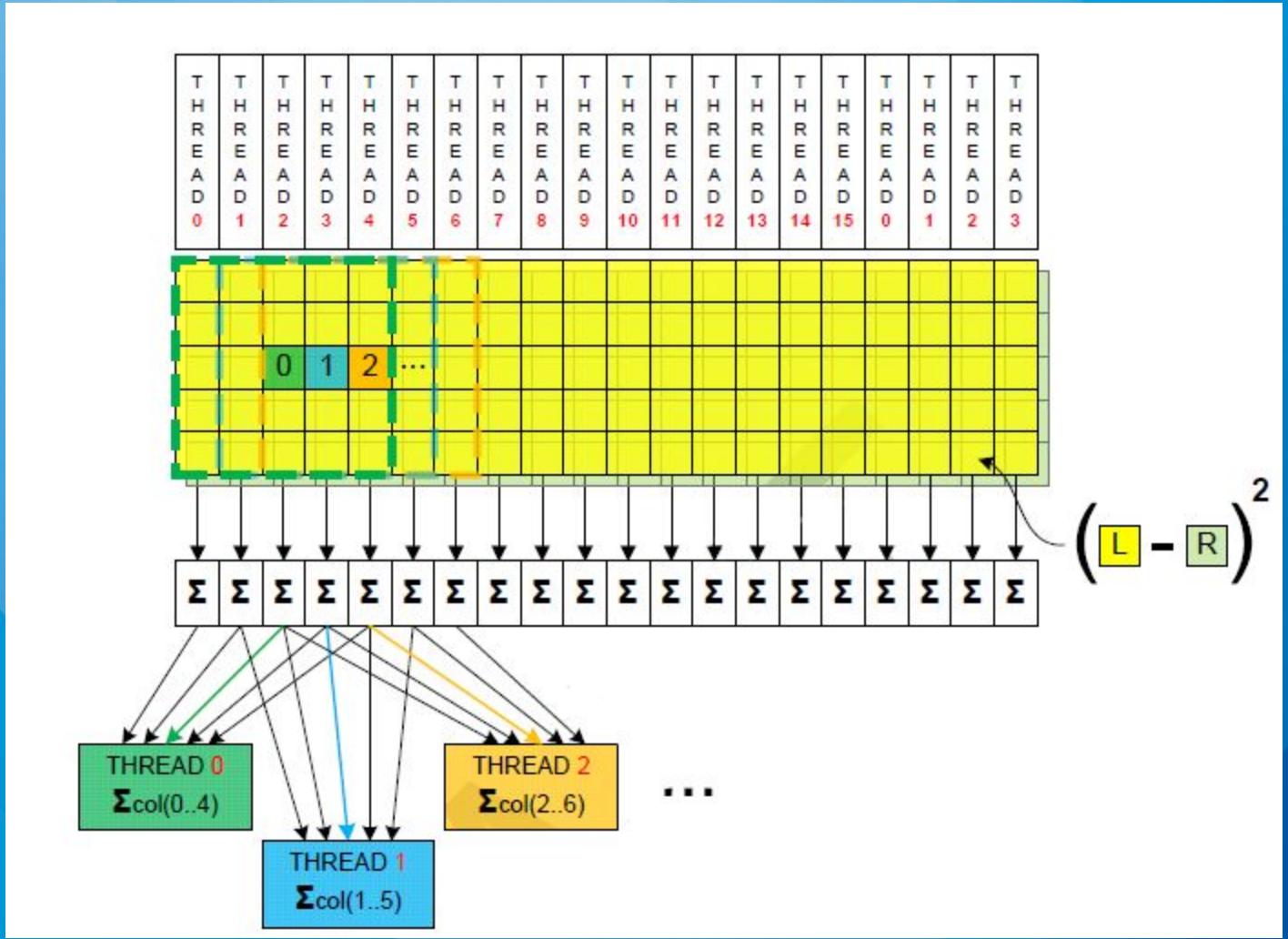


Parallel implementation of block matching

- The outer cycle iterates through disparity values
- We compute SSD and compare it with the current minimum for each pixel in a tile
- Different tiles reuse the results of each other



Parallelization scheme



Optimization concepts

- Not using texture – saving registers
- 1 thread per 8 pixels processing – using cache
- Reducing the amount of arithmetic operations
- Non-parallelizable functions (speckle filtering) are done on CPU

Performance summary

- CPU (i5 750 2.66GHz), GPU (Fermi card 448 cores)
- Block matching on CPU+2xGPU is 10 times faster than CPU implementation with SSE optimization, enabling real-time processing of HD images!

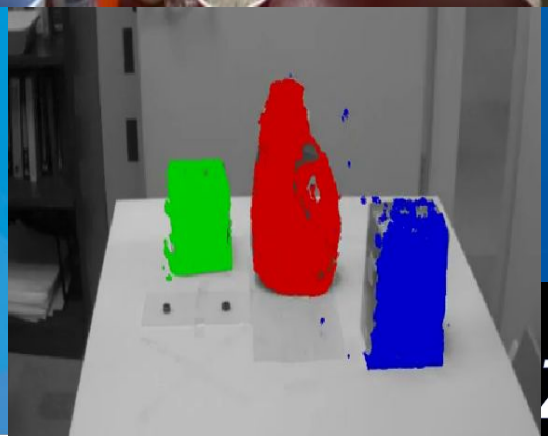
Full-HD stereo in realtime

http://www.youtube.com/watch?v=ThE7sRAtaWU

itseez

Applications of stereo vision

- Machine vision
- Automatic Driver Assistance
- Movie production
- Robotics
  - Object recognition
  - Visual odometry / SLAM

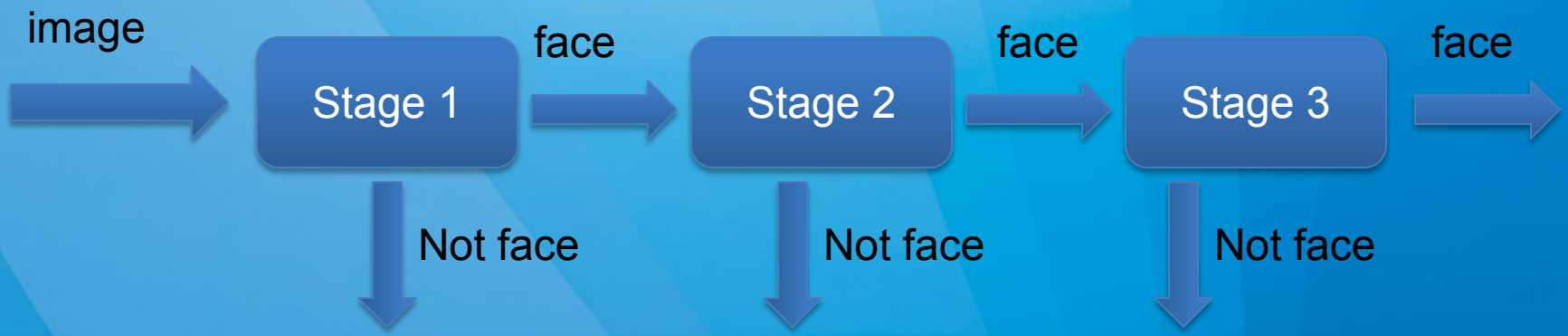


Object detection

itseez

Sliding window approach

Cascade classifier



Real-time in year 2000!

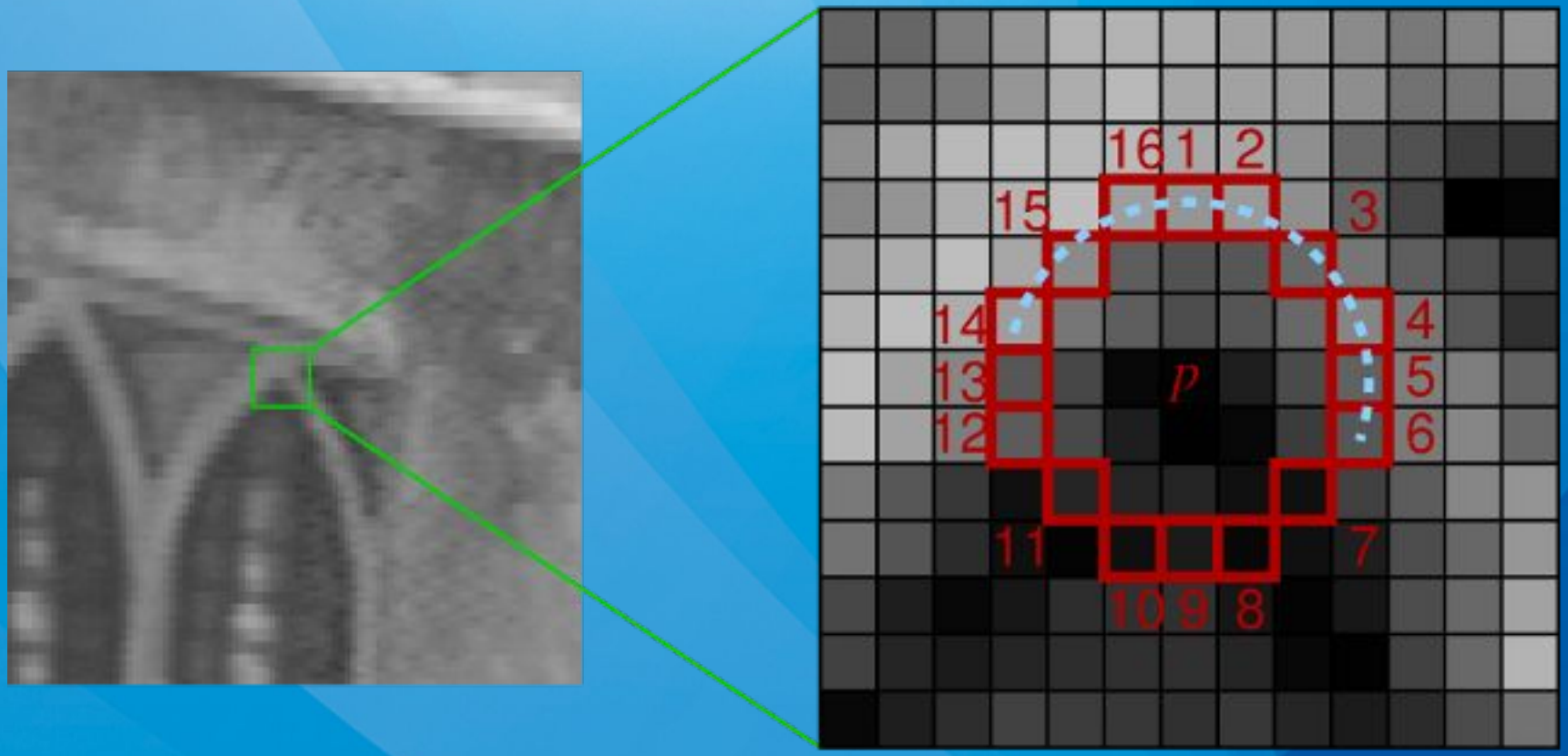
Face detection

itseez

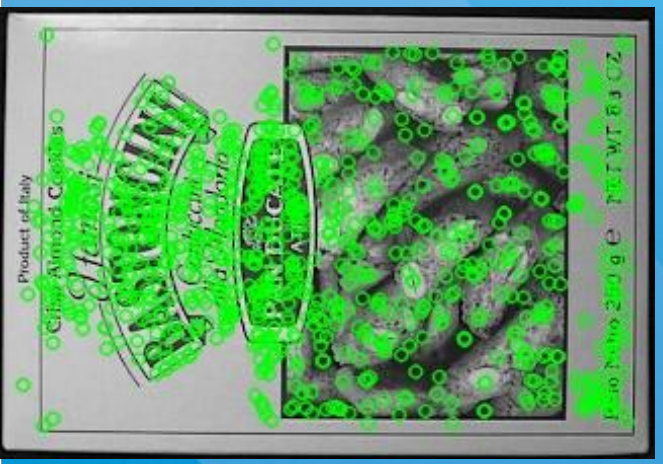
Object detection with local descriptors

- Detect keypoints
- Calculate local descriptors for each point
- Match descriptors for different images
- Validate matches with a geometry model

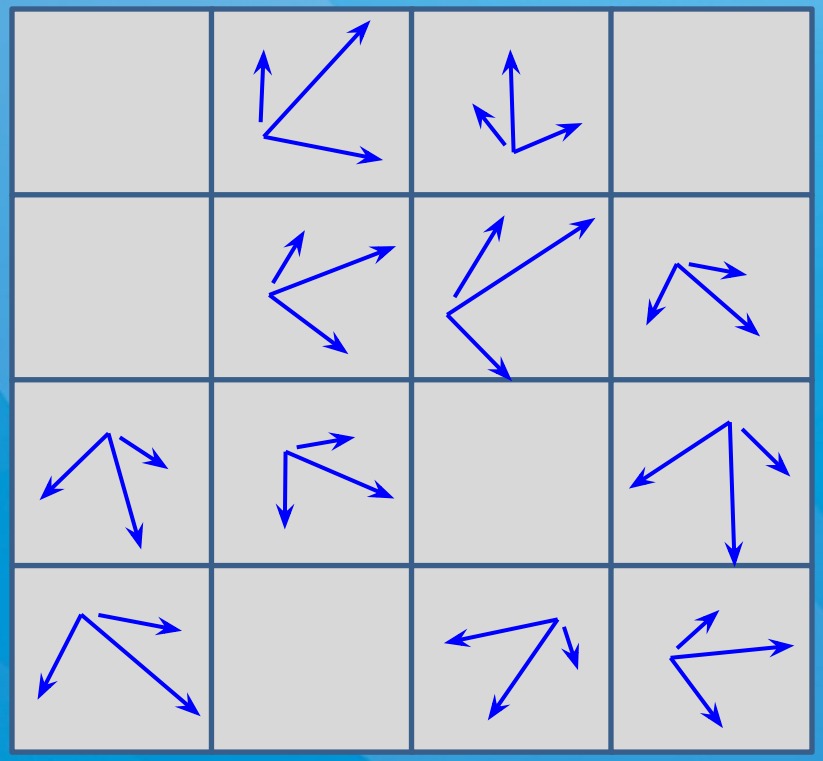
FAST feature detector



Keypoints example

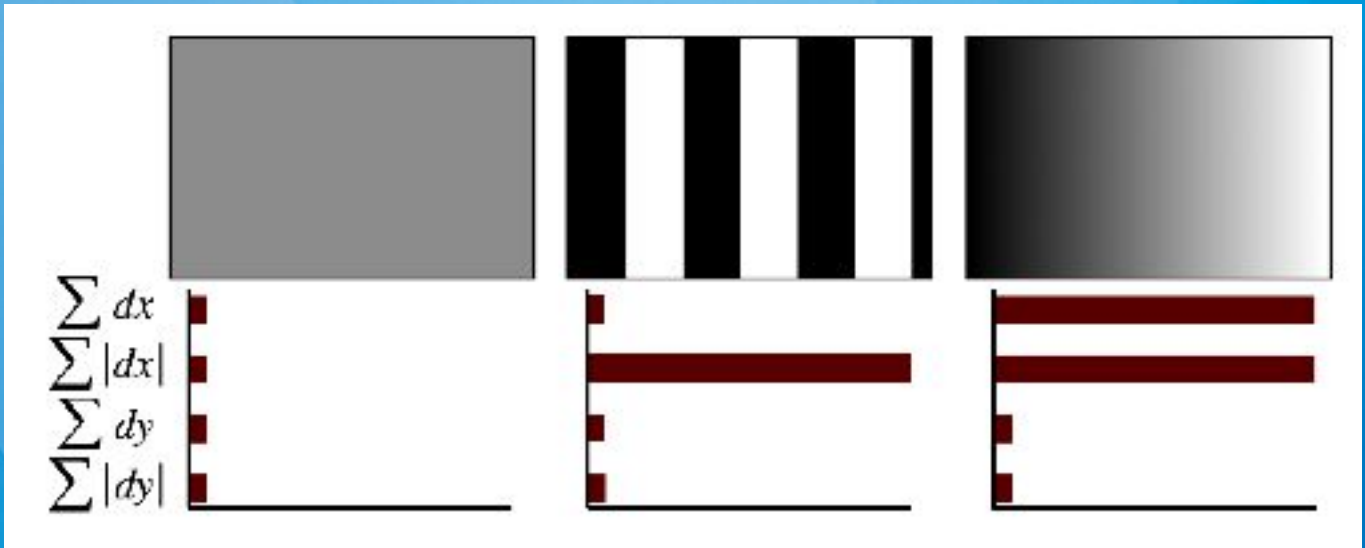


SIFT descriptor



David Lowe, 2004

SURF descriptor

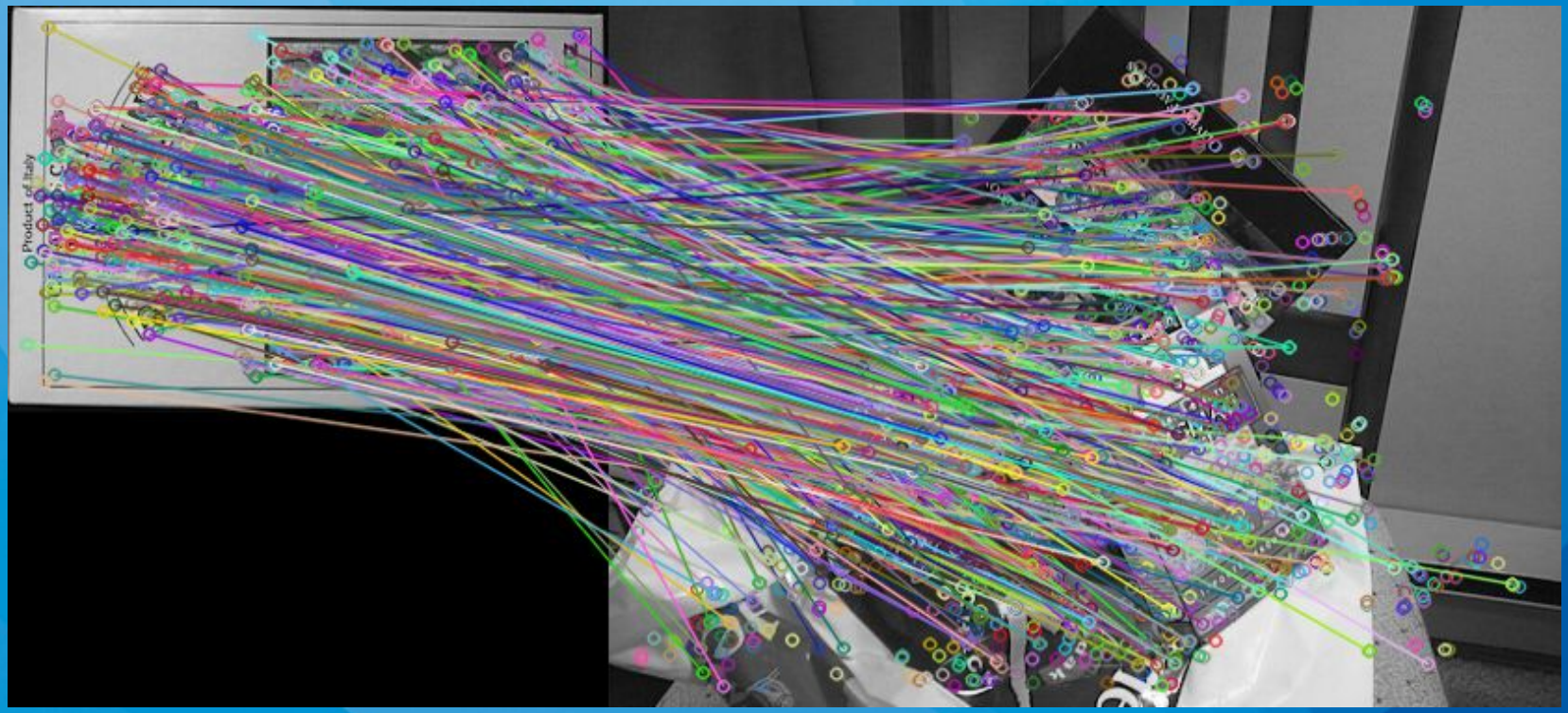


- 4x4 square regions inside a square window 20*s
- 4 values per square region

More descriptors

- One way descriptor
- C-descriptor, FERNS, BRIEF
- HoG
- Daisy

Matching descriptors example



itseez

Ways to improve matching

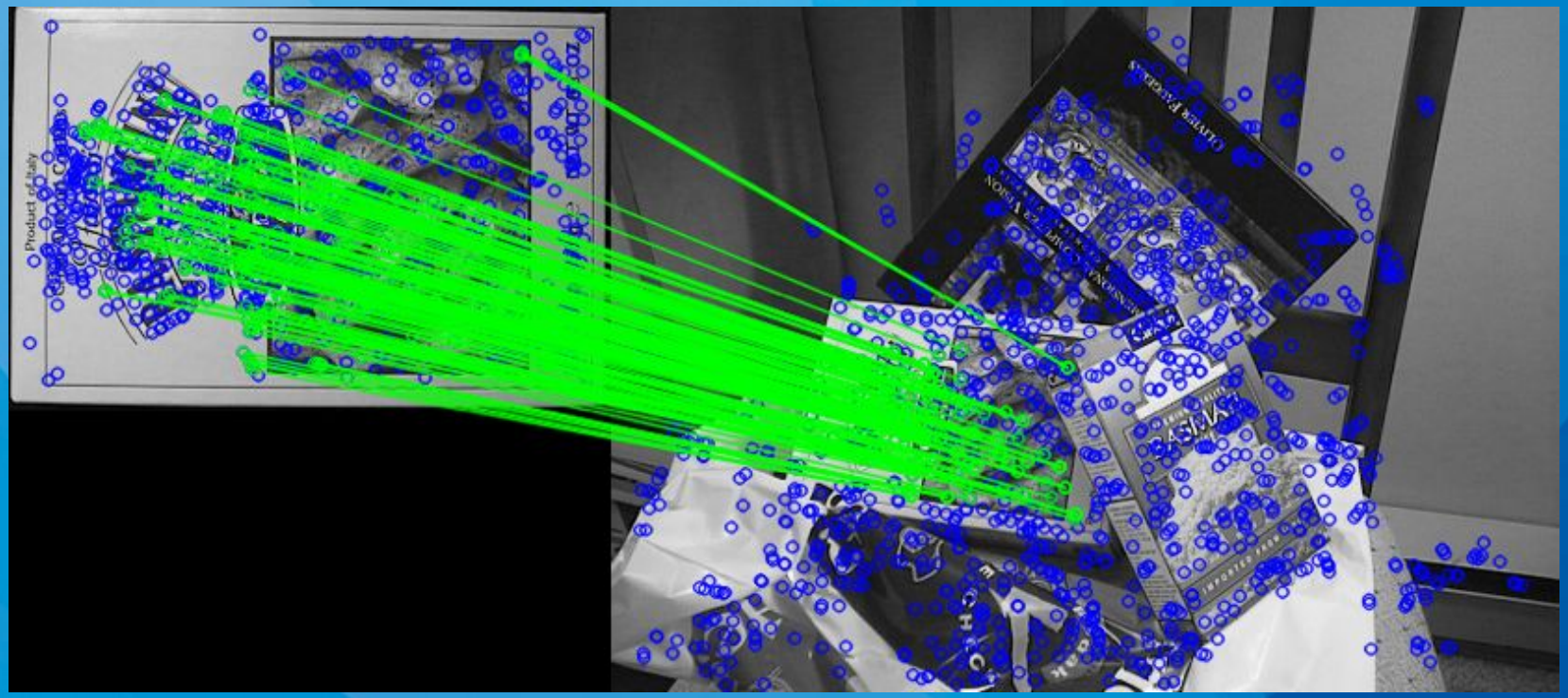
- Increase the inliers to outliers ratio
  - Distance threshold
  - Distance ratio threshold (second to first NN distance)
  - Backward-forward matching
  - Windowed matching
- Increase the amount of inliers
  - One to many matching

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 ~ $10 * \left(\frac{\#matches}{\#inliers}\right)^{k}$

Geometry validation



itseez

Scaling up

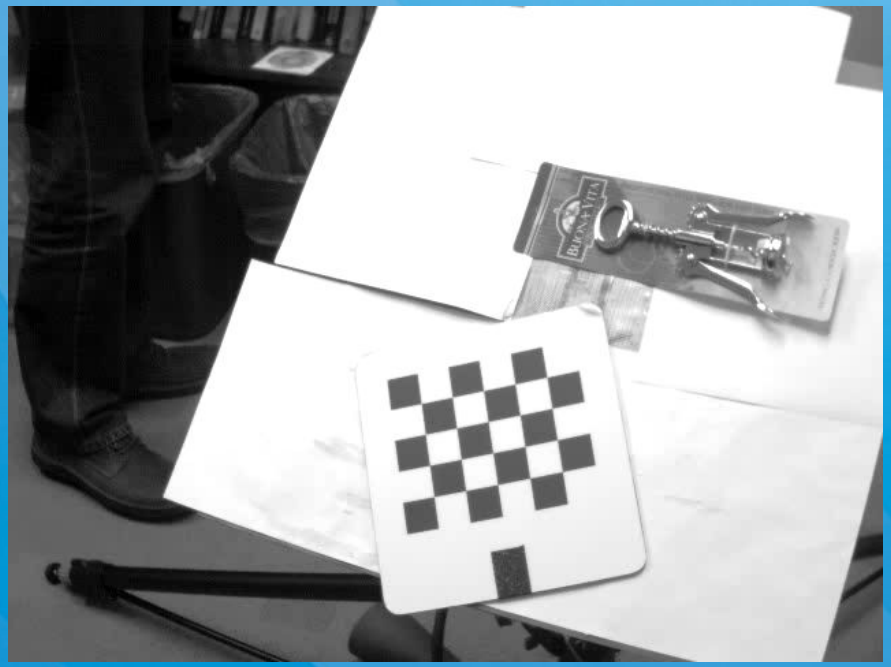
- FLANN (Fast Library for Approximate Nearest Neighbors)
  - In OpenCV thanks to Marius Muja
- Bag of Words
  - In OpenCV thanks to Ken Chatfield
- Vocabulary trees
  - Is going to be in OpenCV thanks to Patrick Mihelich

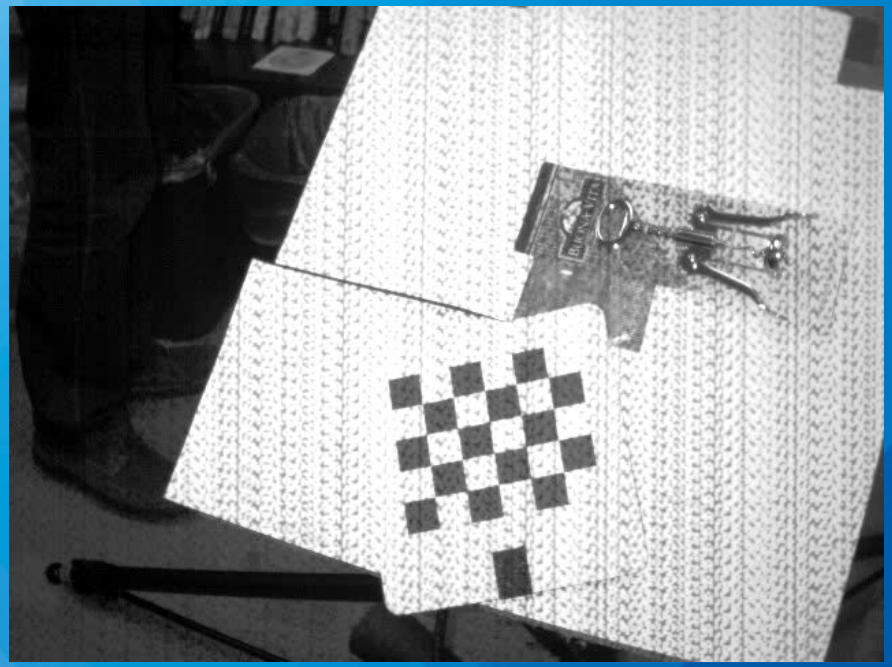
Projects

- Textured object detection
- PR2 robot automatic plugin
- Visual odometry / SLAM

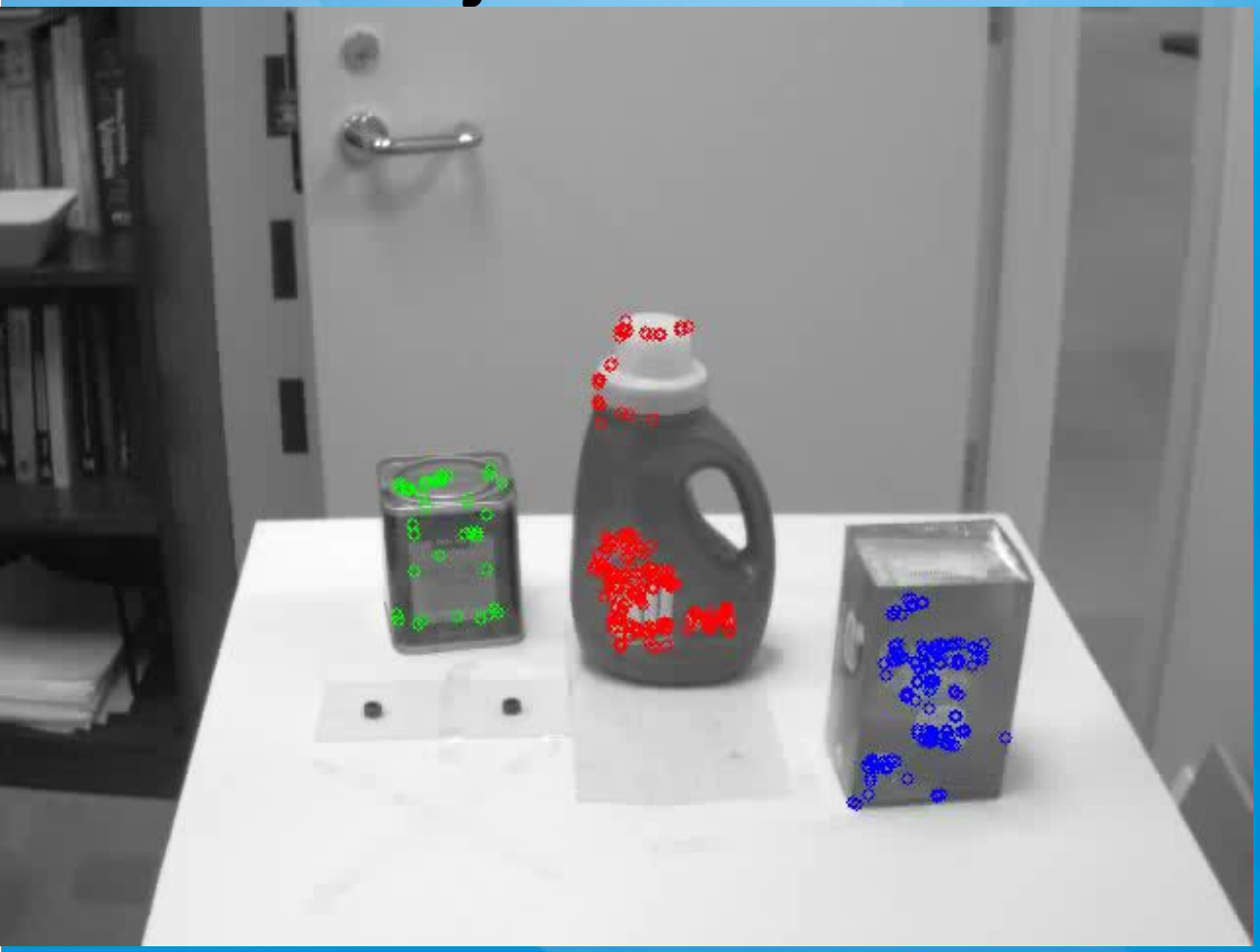
itseez

Textured object detection





Object detection example



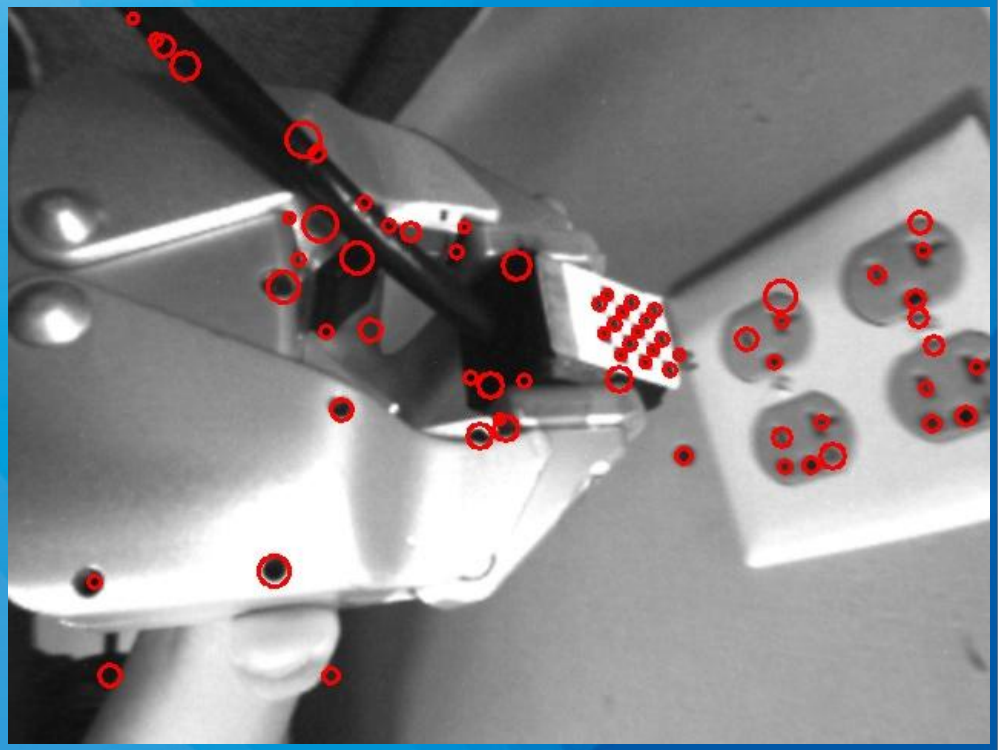
Iryna Gordon and David G. Lowe, "What and where: 3D object recognition with accurate pose," in *Toward Category-Level Object Recognition, eds. J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, (Springer-Verlag, 2006), pp. 67-82.*

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.

itseez

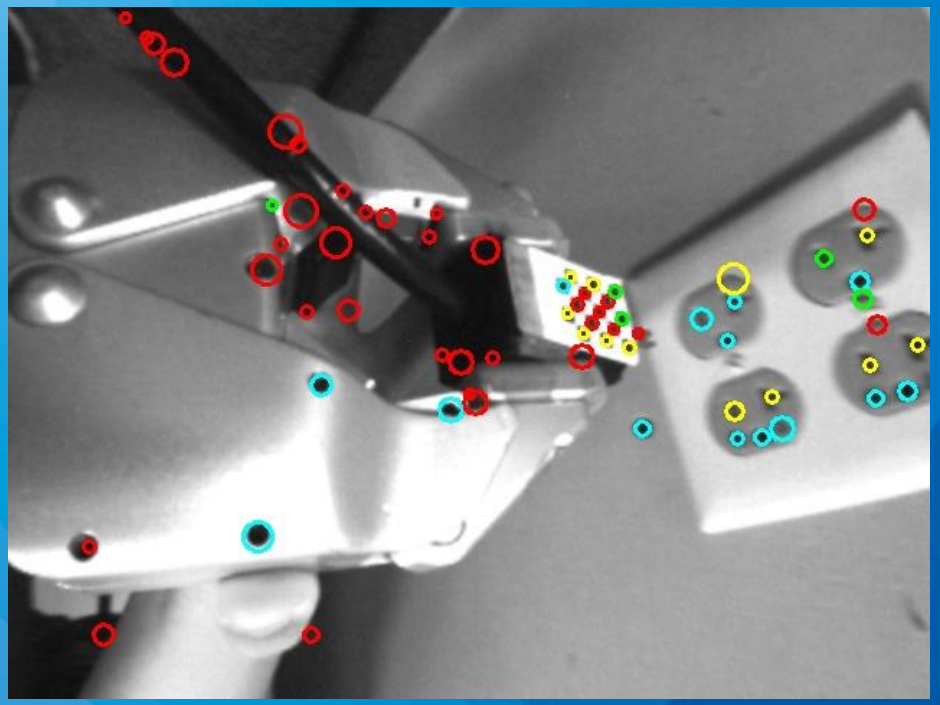
Keypoint detection

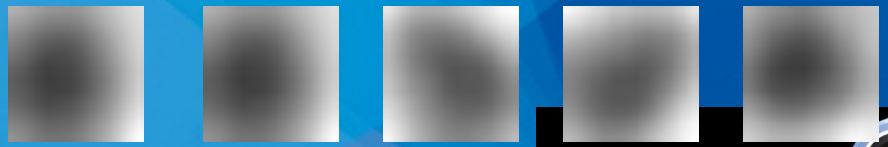
- We are looking for small dark regions
- This operation takes only ~10ms on 640x480 image
- The rest of the algorithm works only with keypoint regions



Classification with one way descriptor

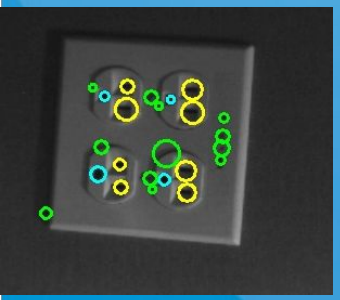
- Introduced by Hinterstoisser et al (Technical U of Munich, Ecole Polytechnique) at CVPR 2009
- A test patch is compared to samples of affine-transformed training patches with Euclidean distance
- The closest patch together with a pose guess are reconstructed

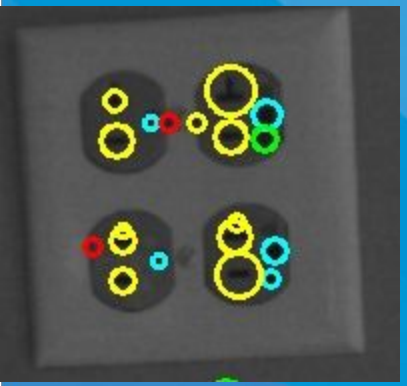


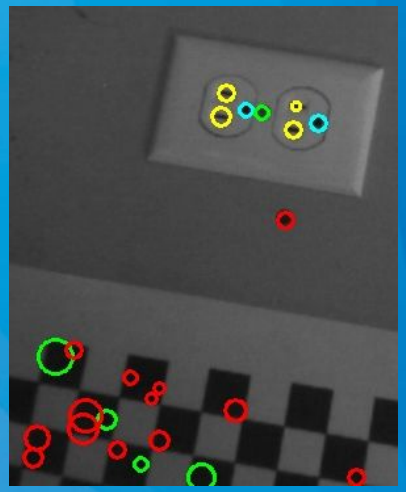


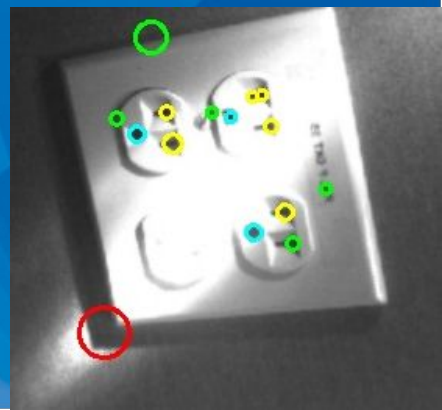
Keypoint classification examples

- One way descriptor does the most of the outlet detection job for us. Few holes are misclassified









Ground hole

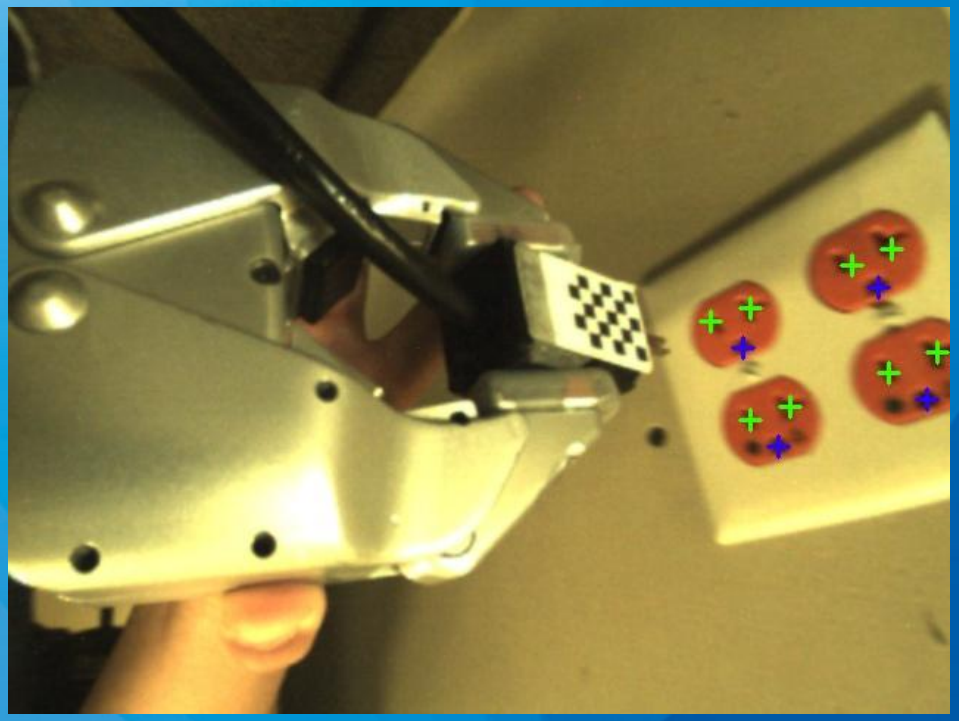
Power hole

Non-hole keypoint from outlet image

Background keypoint

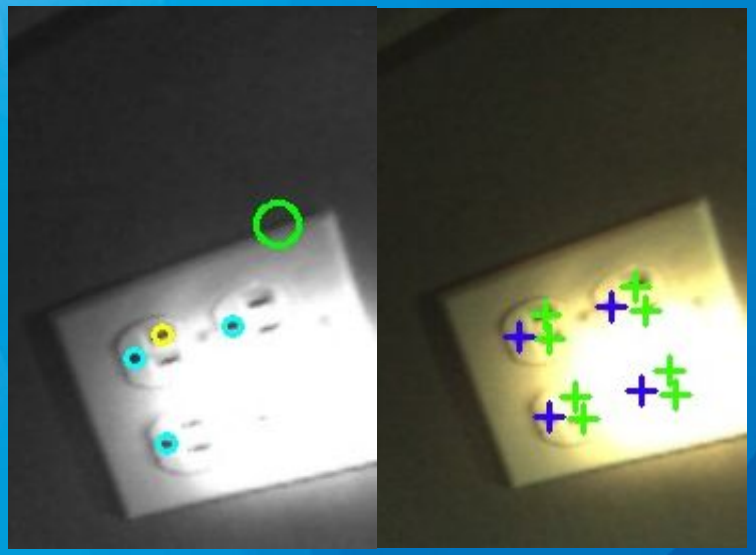
Object detection

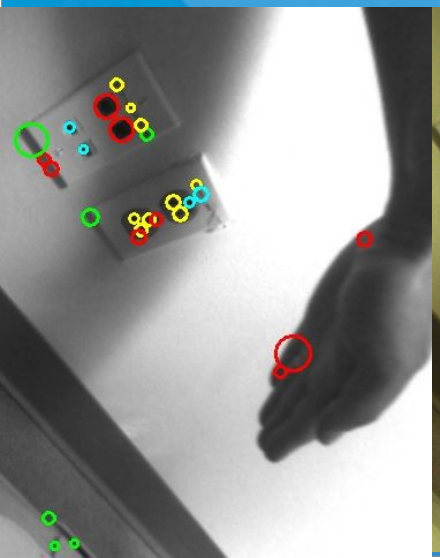
- Object pose is reconstructed by geometry validation (using geomertic hashing)

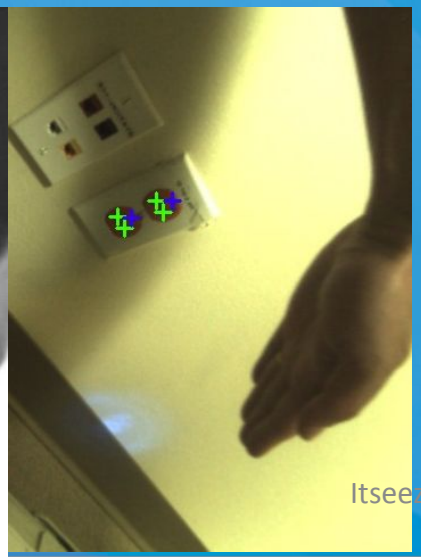


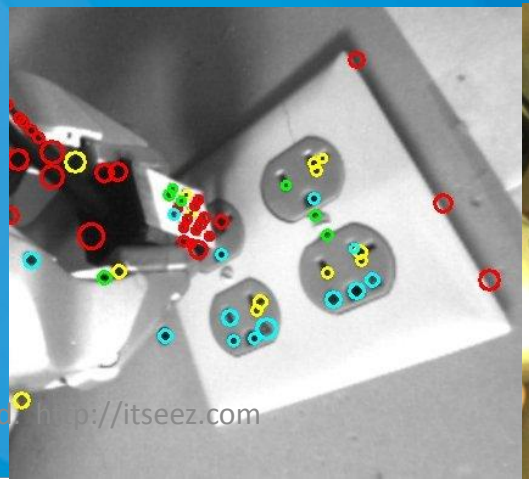
Outlet detection: challenging cases

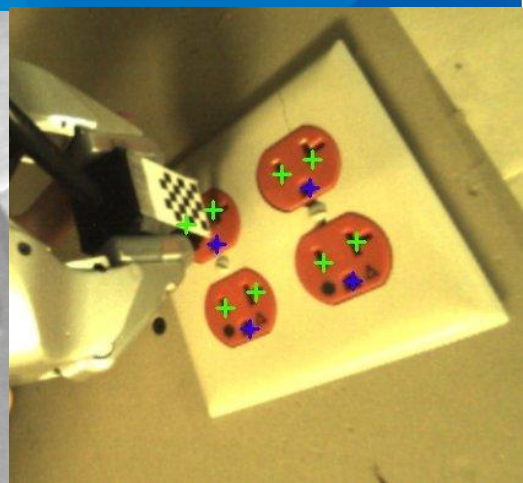
- ✔Shadows
- ✔Severe lighting conditions
- ✔Partial occlusions



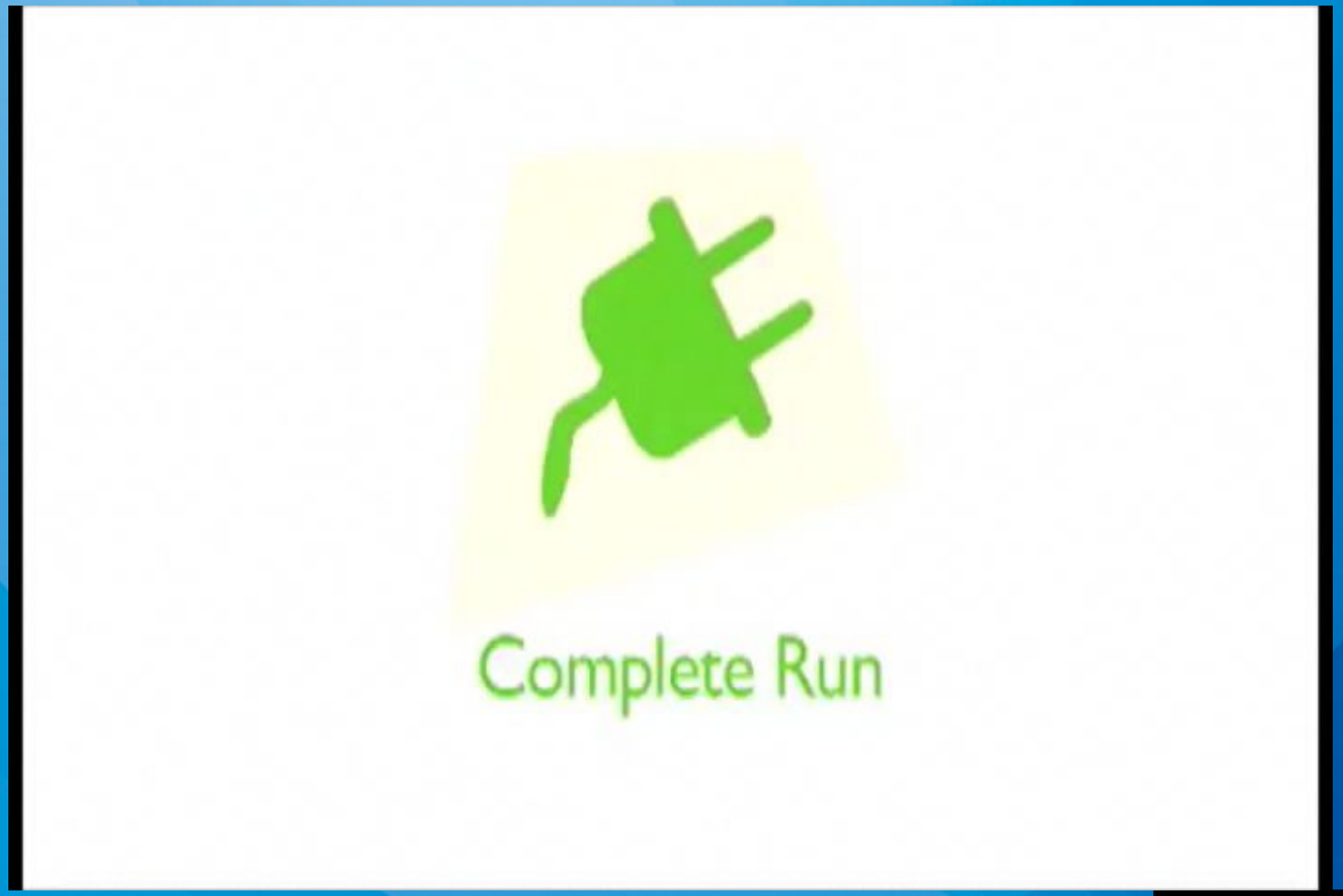








PR2 plugin (outlet and plug detection)



http://www.youtube.com/watch?v=GWcepdggXsU

itseez

Visual odometry

itseez

Visual odometry (II)

itseez

More fun

itseez