Seminars in Artificial Intelligence and Robotics

Computer Vision for Intelligent Robotics

Basics and hints on low level vision

DIPARTIMENTO DI INGEGNERIA INFORMATICA
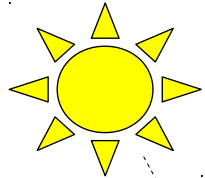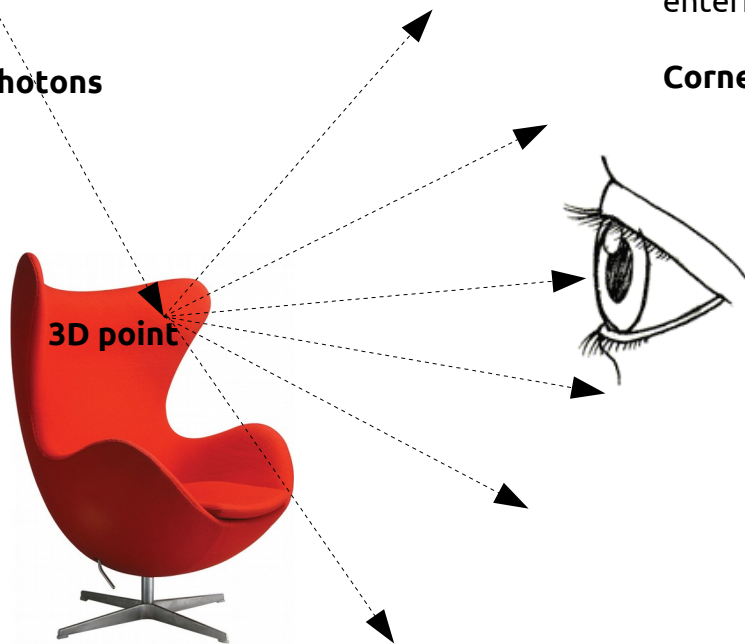AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Alberto Pretto

# Capturing Light - human eye

Light source

Photons

3D point

Retina: Part of the eye that converts images into electrical impulses, i.e. it includes the photoreceptor cells (rods and cones, the latter densely packed in the fovea)

Iris: Pigmented tissue that controls amount of light entering eye by varying size of the **pupil**

Cornea + **Lens**: natural lens of eye

Retina

Macula

Iris

Pupil

Vitreous

Cornea

Lens

Optic Nerve
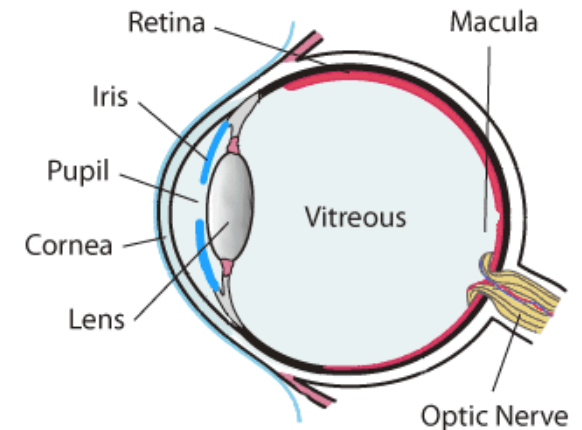
Surface reflectance

Lambertian reflectance: Computer vision algorithms often assume that the color and brightness intensity of a point on a surface does not change with the vantage point
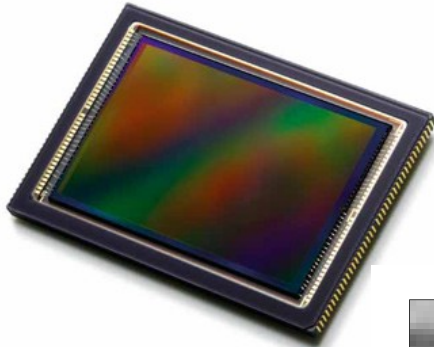

This is **not** true in general! But it is very hard to deal with non-lambertian reflectance...
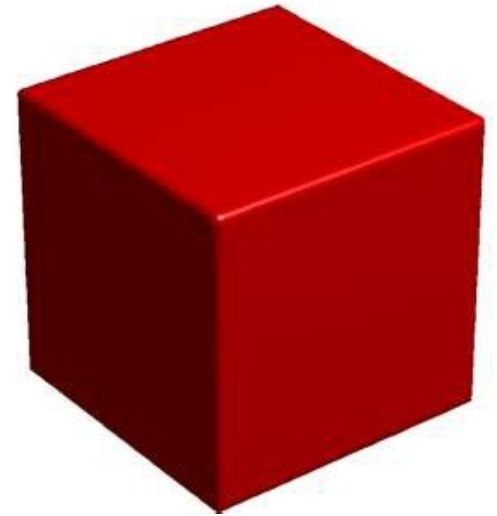
Capturing Light – digital cameras

A continuous scene is framed by a discrete array

A CMOS sensor: array of photoreceptors, each sensor has its own amplifier

Provide a a two-dimensional brightness array: the **image**


```
188 186 188 187 168 130 101  99 110 113 112 107 117 140 153 153 156 158 156 153
189 189 188 181 163 135 109 104 113 113 110 109 117 134 147 152 156 163 160 156
190 190 188 176 159 139 115 106 114 123 114 111 119 130 141 154 165 160 156 151
190 188 188 175 158 139 114 103 113 126 112 113 127 133 137 151 165 156 152 145
191 185 189 177 158 138 110  99 112 119 107 115 137 140 135 144 157 163 158 150
193 183 178 164 148 134 118 112 119 117 118 106 122 139 140 152 154 160 155 147
185 181 178 165 149 135 121 116 124 120 122 109 123 139 141 154 156 159 154 147
175 176 176 163 145 131 120 118 125 123 125 112 124 139 142 155 158 158 155 148
170 170 172 159 137 123 116 114 119 122 126 113 123 137 141 156 158 159 157 150
171 171 173 157 131 119 116 113 114 118 125 113 122 135 140 155 156 160 160 152
174 175 176 156 128 120 121 118 113 112 123 114 122 135 141 155 155 158 159 152
176 174 174 151 123 119 126 121 112 108 122 115 123 137 143 156 155 152 155 150
175 169 168 144 117 117 127 122 109 106 122 116 125 139 145 158 156 147 152 148
179 179 180 155 127 121 118 109 107 113 125 133 130 129 139 153 161 148 155 157
176 183 181 153 122 115 113 106 105 109 123 132 131 131 140 151 157 149 156 159
180     177 147 115 110 111 107 107 105 120 132 133 141 150 154 148 155 157
        140 114 114 118 113 112 107 119 128 130 134 144 153 156 148 152 151
     142 114 114 116 110 108 104 116 125 128 134 148 161 165 159 157 149
  71 138 109 110 114 110 109  97 110 121 127 136 150 160 163 158 156 150
```
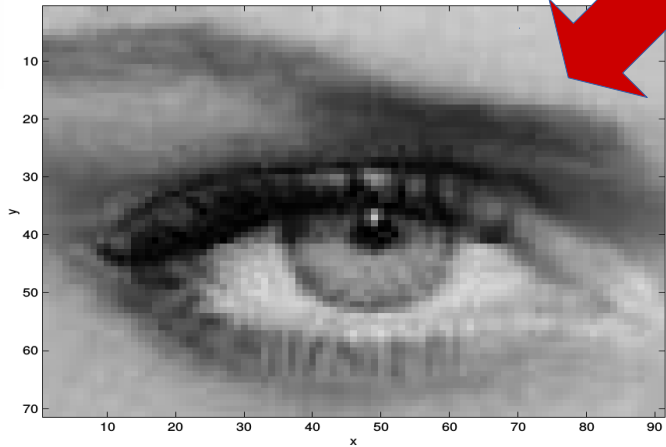

A "**picture**" of the image: a picture is different representation of the image, i.e. a scene that produces on the imaging sensor the same image as the original scene.
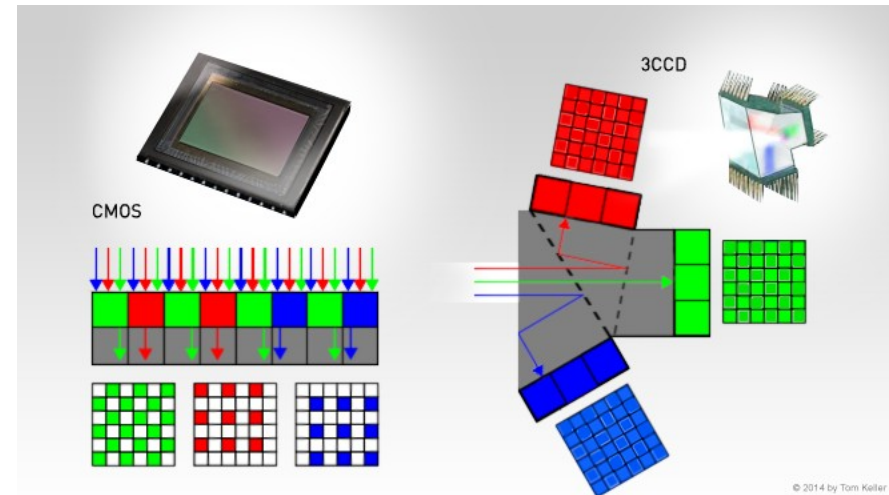
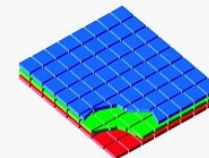# Perceive colors: the RGB model

Currently, 3 approaches:

Bayer pattern   3 sensors
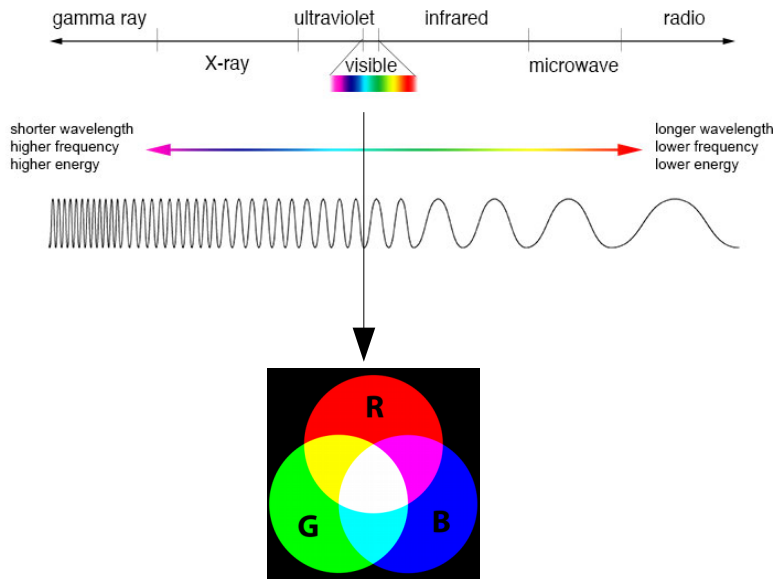
Additive color system: the **RGB model**
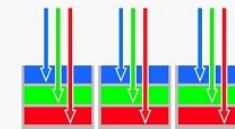
Three layer sensor

A Foveon X3 image sensor features three separate layers of photodetectors embedded in silicon.

Since silicon absorbs different colors of light at different depths, each layer captures a different color. Stacked together, they create full-color pixels.

As a result, only Foveon X3 image sensors capture red, green and blue light at every pixel location.

Camera model

Put the sensor (e.g., the CMOS) in front of an object. Does this simple "camera" works?

Sensor (side view)

3D point

3D point
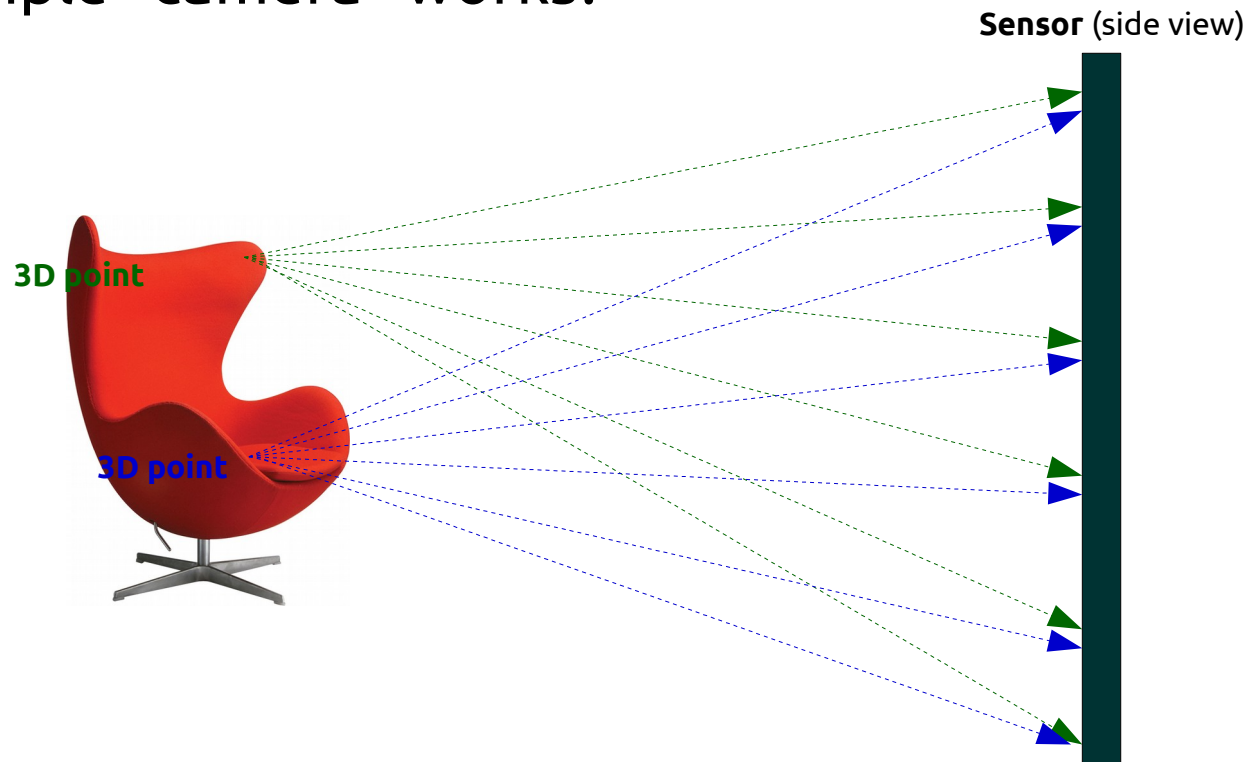
Pinhole camera model (1/2)

Idea: add a barrier to block off most of the rays!

The "size" of the hole is called **aperture**.

Sensor (side view)

3D point

3D point

Pinhole camera model (2/2)

Focal length
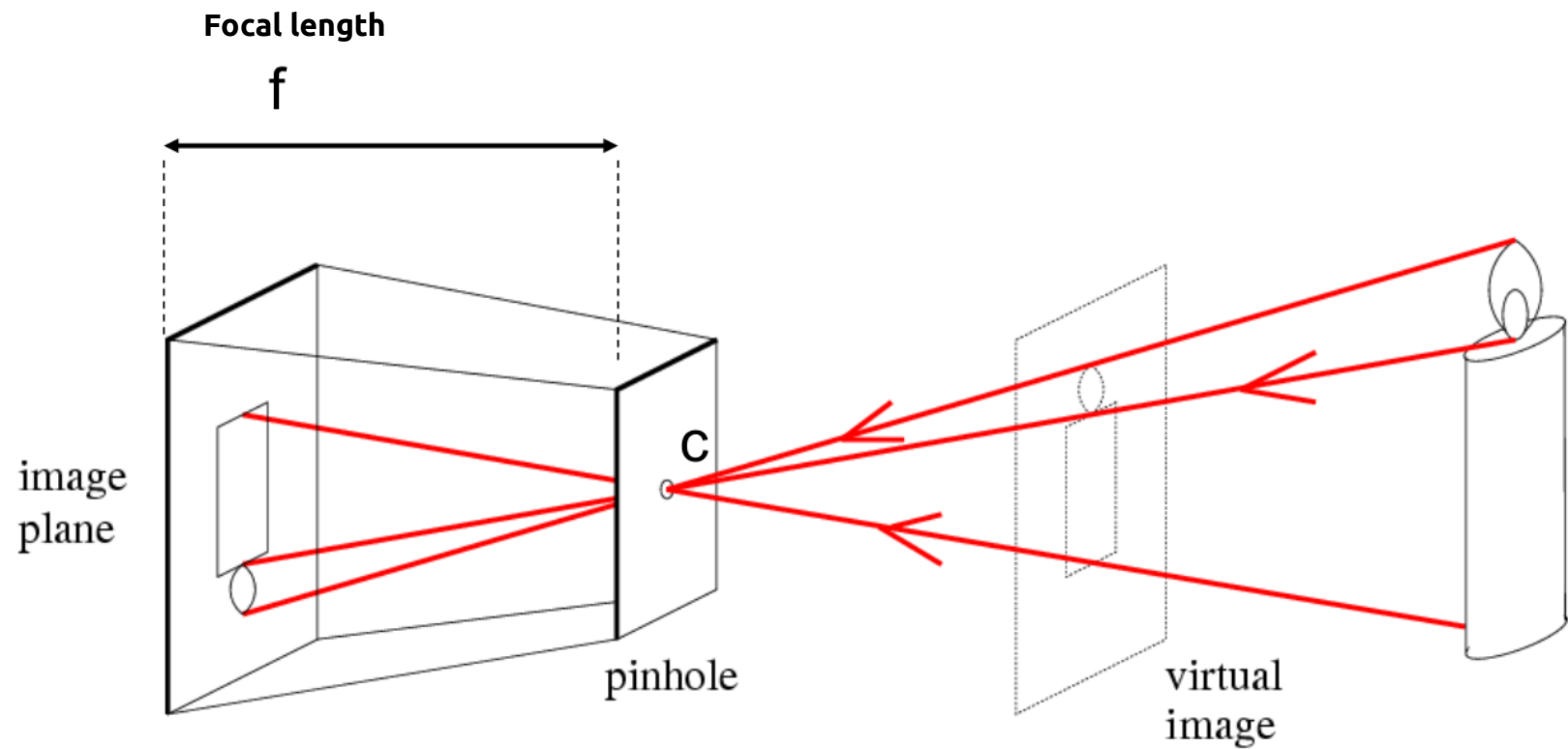
f

image plane

C

pinhole

virtual image
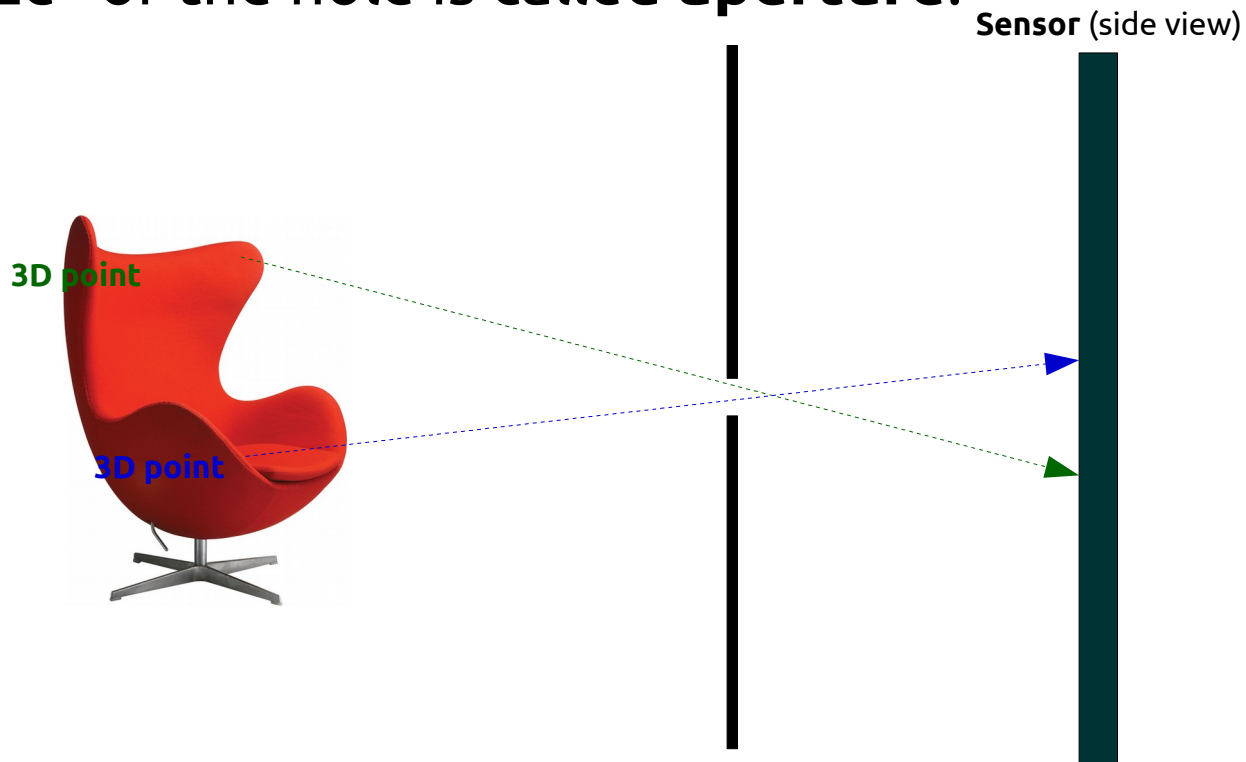
Adding a lens to the pinhole camera

The lens redirect the rays in order to focuses light onto the sensor (**thin lens model**)

- Any ray that passes through the center of the lens will not change its direction
- Any ray that enters parallel to the axis on one side of the lens proceeds towards the focal point **f** on the other side.

Sensor (side view)

3D point

3D point

Focal length f

There is a specific distance at which objects are "in focus". Other points project to a "circle of confusion" in the image
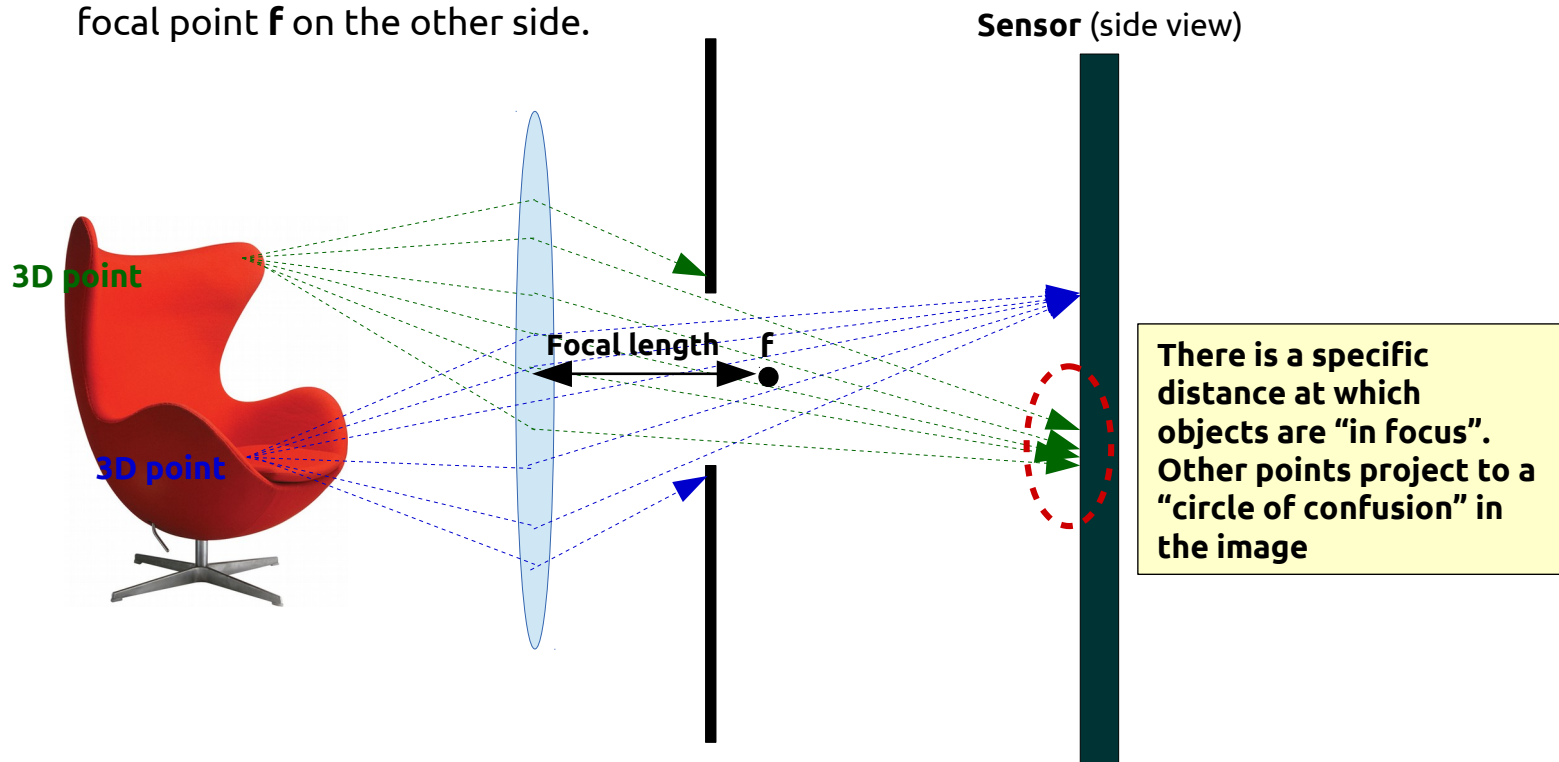
Depth of field

DOF: distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image
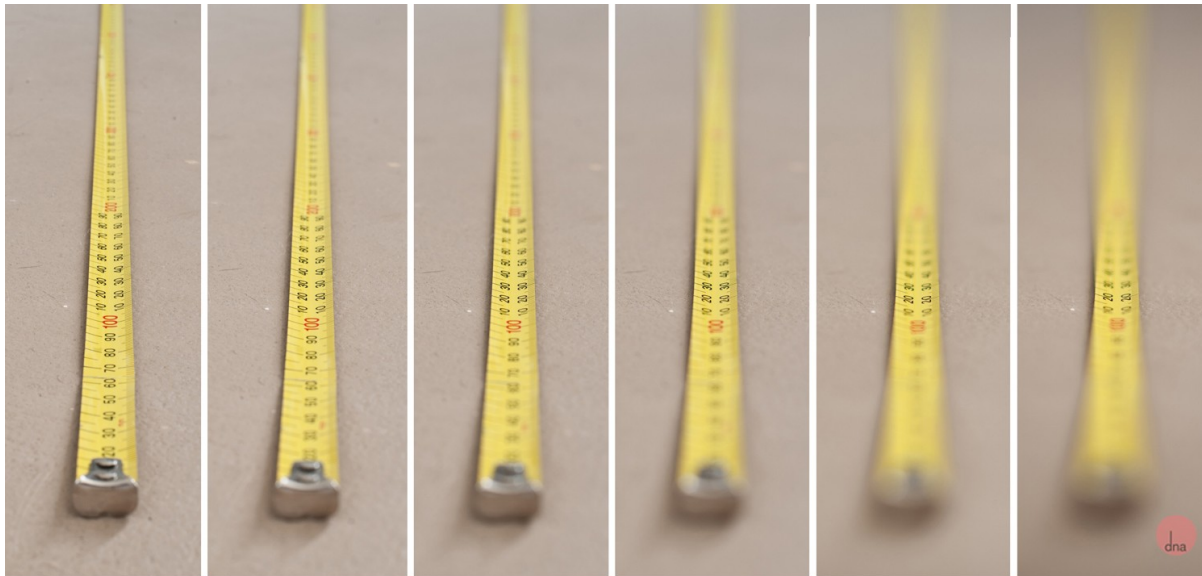
Decreasing the aperture we get a larger depth of field... but we get a less bright image (i.e., less light collide with the sensor)

From 3D to 2D

A 3D scene is "projected" on the 2D the image plane

Dimensionality reduction → **the depth (i.e., the z cooridnate) of each point is lost!**

The standard coordinate system of the pinhole camera system seen from the X axis.
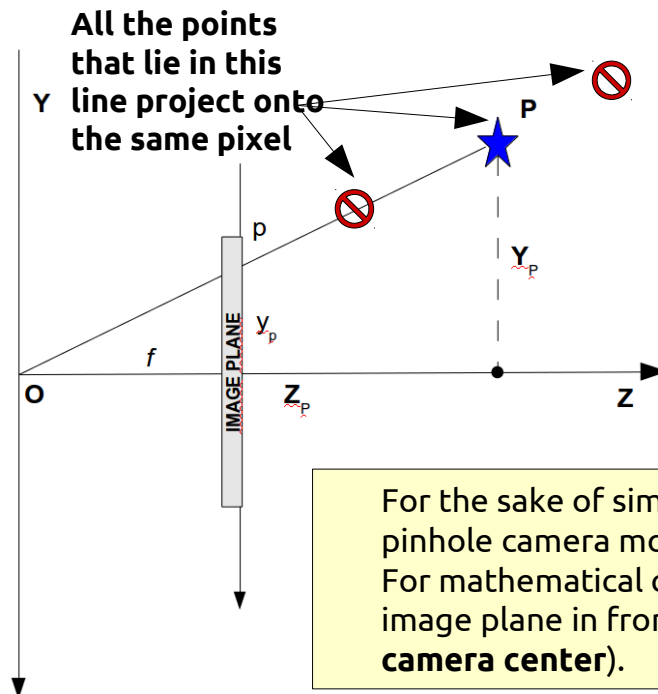
$$x_p = \frac{X_P f}{Z_P} \quad , \quad y_p = \frac{Y_P f}{Z_P}$$

For the sake of simplicity we will use the pinhole camera model.
For mathematical convenience, we put the image plane in front of the focal point (i.e., the **camera center**).

From 3D to 2D (cont.)

Use homogeneous coordinates!

$$x_p = \frac{X_P f}{Z_P} \quad , \quad y_p = \frac{Y_P f}{Z_P}$$
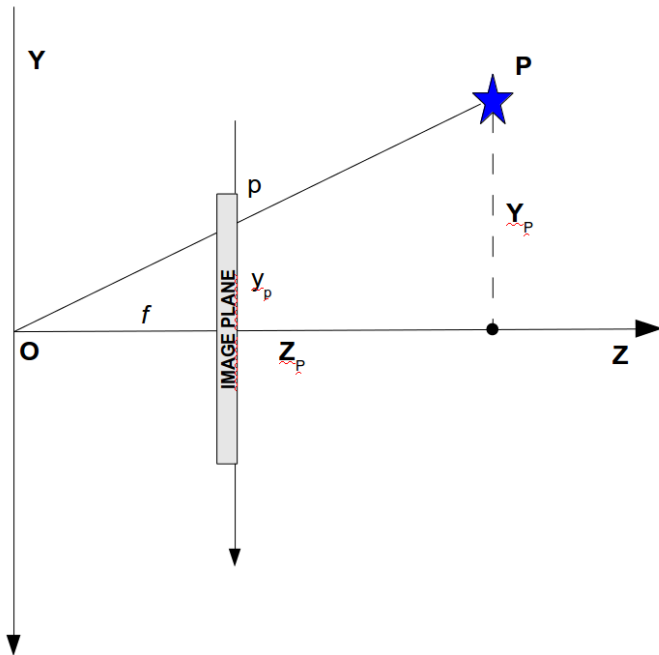

Converting to *homogeneous* coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous image
coordinates

homogeneous scene
coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix}$$

Map to pixels (1/2)

We need to convert an image coordinates from meters to pixels.

Remember that the top left pixel has (0,0) coordinates (i.e., in pixel space, the origin is not the center of the image)

$$u = u_c + \frac{x}{w_p} \quad , \quad v = v_c + \frac{y}{h_p}$$

u_c and v_c are the coordinates of the principal point in pixels and w_p and h_p are the pixel width and height, respectively.

Map to pixels (2/2)

Using homogeneous coordinates:

Perspective
projection matrix

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_c & 0 \\ 0 & \alpha_v & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix}$$

where $\quad \alpha_u = \dfrac{f}{w_p} \text{ and } \alpha_v = \dfrac{f}{h_p}$

$$\tilde{\mathbf{u}} = \mathbf{A}\tilde{\mathbf{P}}$$

The perspective projection camera contains the **intrinsic parameters** of the camera (i.e., focal length, image sensor format, and principal point). These parameters are not are usually not known in advance, due the inaccuracies in camera assembly. The camera calibration process aims to estimate the intrinsic parameters.

Note: here we are not taking into account the skew factor.

From world to camera

In general, coordinates of a 3D point are not specified in current camera frame (remember that the drone with its camera is moving, while the ground is fixed): 3 points are often specified in a more convenient fixed frame we call here *world frame*.
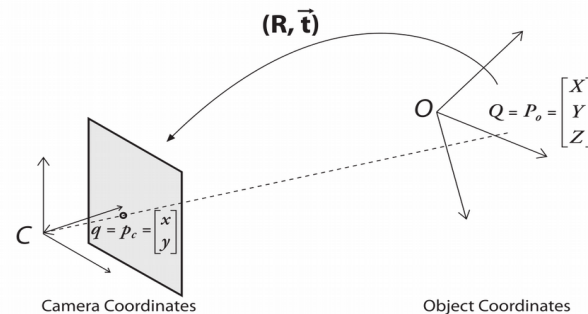
Before projecting a point onto the image plane, we need to change its coordinates from the world frame to the current camera coordinate system.

3D point in camera coordinates

3D point in world coordinates

3x3 rotation matrix

3x1 translation vector

$$\tilde{\mathbf{P}} = \mathbf{T}\tilde{\mathbf{P}}' \quad , \quad \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix}$$

Putting all together

From world frame to pixels:

$$\tilde{\mathbf{u}} = \mathbf{A}\mathbf{T}\tilde{\mathbf{P}}' = \mathbf{C}\tilde{\mathbf{P}}'$$

Sometimes this equation is rewritten in an alternative way:

$$\tilde{\mathbf{u}} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\tilde{\mathbf{P}}'$$

Where **[R|t]** is a 3x4 matrix composed by the rotation matrix followed by the translation vector, and K is a 3 × 3 matrix holding the intrinsic parameters:

$$\mathbf{K} = \begin{bmatrix} \alpha_u & 0 & u_c \\ 0 & \alpha_v & v_c \\ 0 & 0 & 1 \end{bmatrix}$$
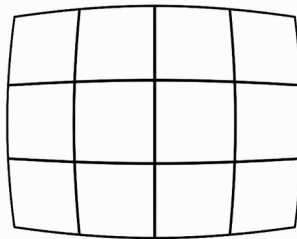
Hints on radial distortion

In many cases, like in the wide-angles camera applications, the lens distortion should be taken into account in the perspective projection: distortion is modeled by the nonlinear intrinsic parameters.
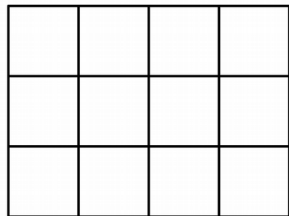
m43photo.blogspot.com
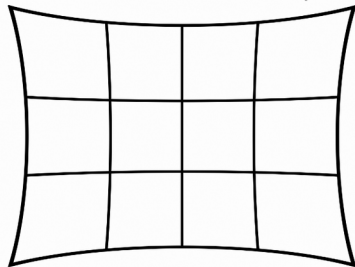
Pincushion distortion

Barrel distortion

Rectilinear

Hints camera calibration (1/2)

The camera calibration process aims to estimate the camera intrinsic parameters and the radial distortion parameters.
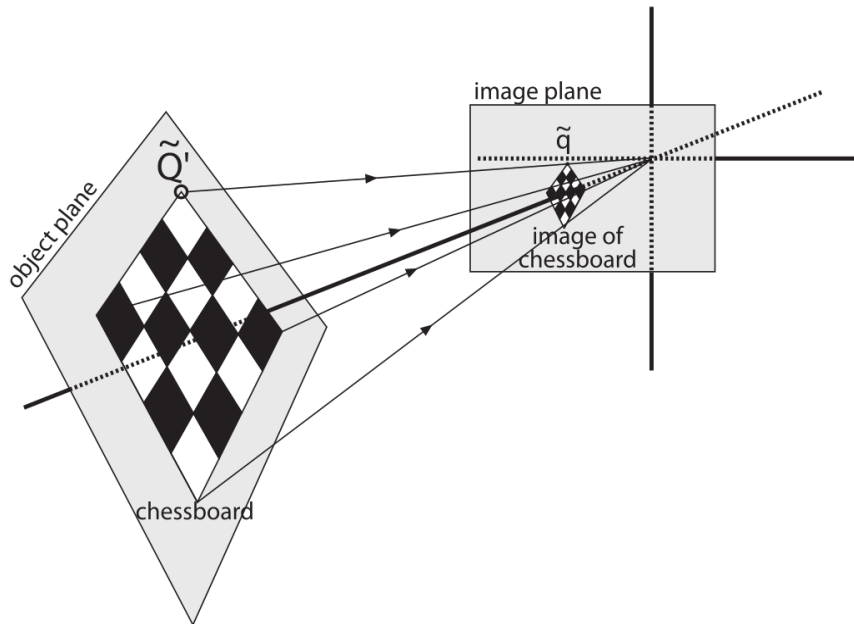
Basic idea of the most popular calibration technique [Zhang00], [Bouguet]:

-Use a known planar pattern (e.g., a cesser board)

-Collect a sequence of images of the pattern in several positions, and extract all corners

-Find the parameters set that minimize the squared distances in the image space, using conventional least square methods (e.g., Levenberg Marquardt)

Hints camera calibration (2/2)
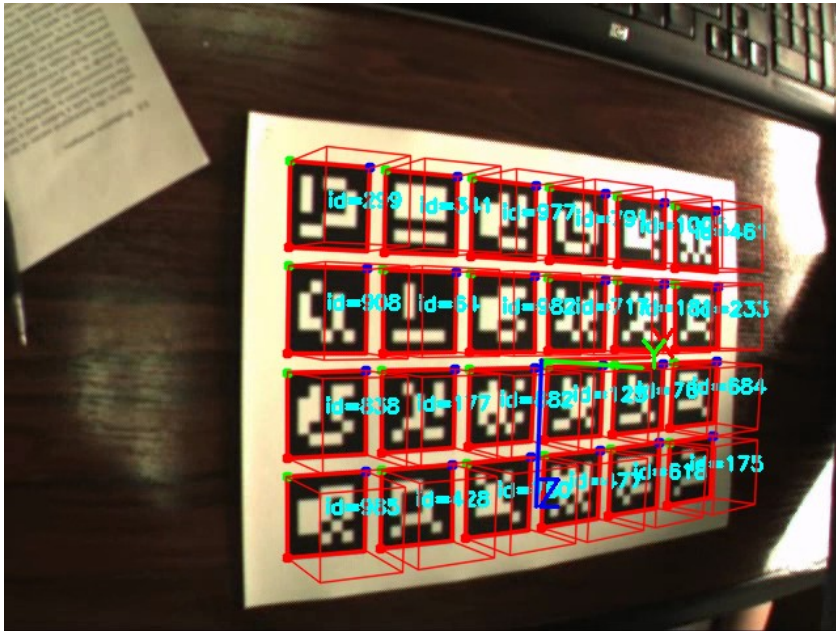
An Augmented Reality (AR) board is a better calibration pattern compared with a checkerboard [Aruco]. An AR board is a marker composed by several markers arranged in a grid. Boards present two main advantages. First, since there have more than one markers, it is less likely to lose them all at the same time. Second, the more markers are detected, the more points are available.

References on calibration:

[Zhang00] Z. Zhang, "A flexible new technique for camera calibration," IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000): 1330–1334
[Bouguet] http://www.vision.caltech.edu/bouguetj/calib_doc/
[Aruco] http://www.uco.es/investiga/grupos/ava/node/26

Epipolar constraint (1/3)

When using more than one camera:
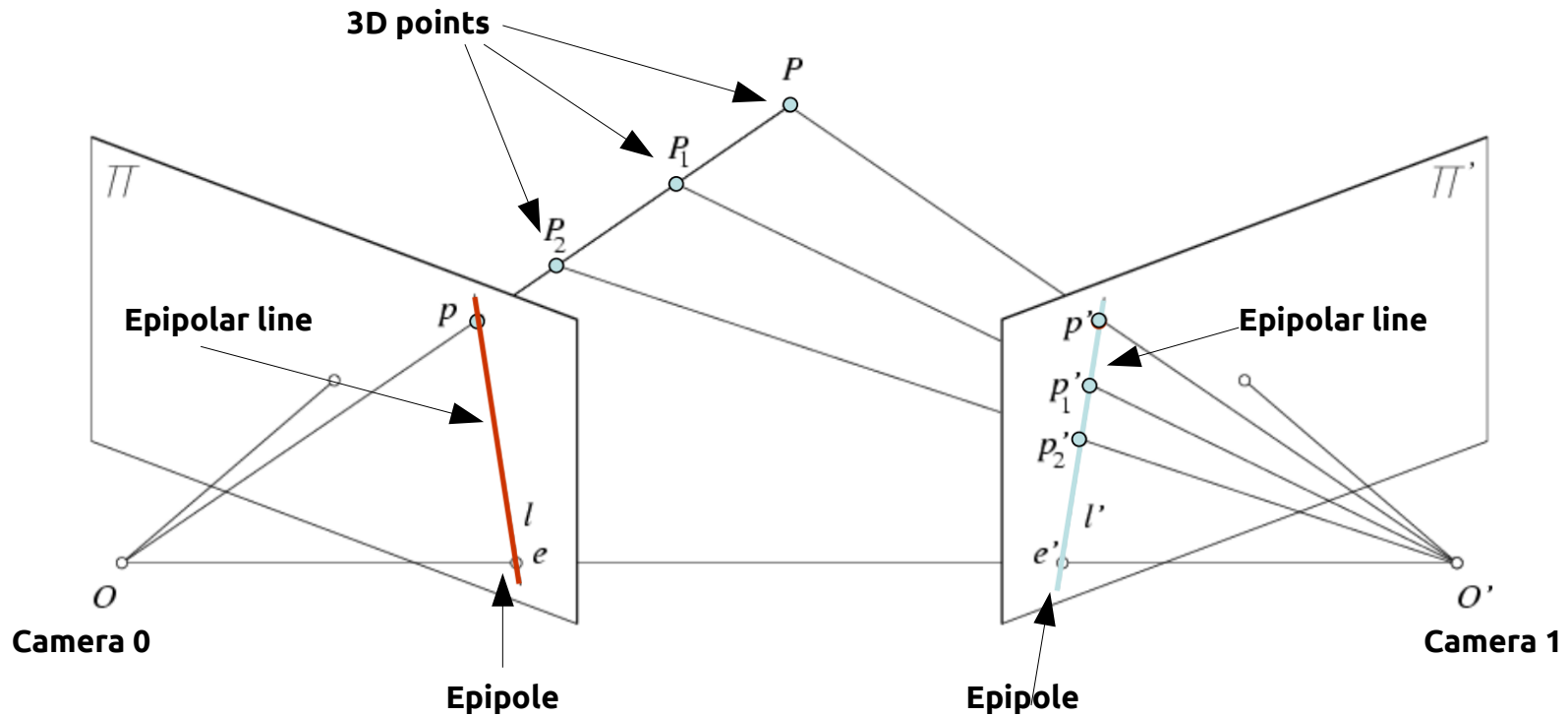
Mapping from p' to l:
$$l = t \times Rp' = [t]_\times R \cdot p' = E \cdot p'$$

E is a 3x3, 5 DOF matrix

Epipolar constraint ➡ $$p^T E p' = 0$$

Epipolar constraint (3/3)

Reduces the correspondence problem to a 1D search in the second image along an epipolar line

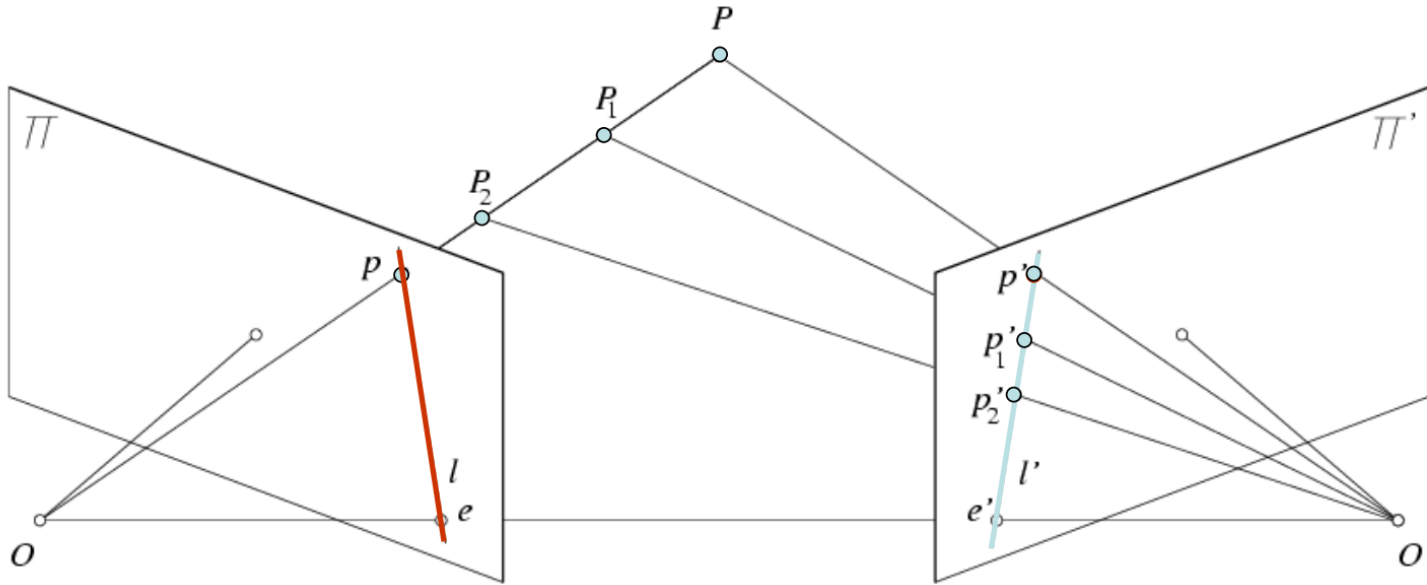# Image filtering

Compute function of local neighborhood at each position

Really important:

- Enhance images (de-noise, resize, increase contrast, etc...)
- Extract information from images (texture, edges, distinctive points, etc...)
- Detect patterns

Image filters in spatial domain: modify the pixels in an image based on some function of a local neighborhood of the pixels

- Filter is a mathematical operation of a grid of numbers
- Smoothing, sharpening, measuring texture

Linear filtering

Linear case is simplest and most useful

— Replace each pixel with a linear combination of its neighbors.

The prescription for the linear combination is sometimes called the "convolution kernel" (even if linear filtering uses correlation).

For symmetrical kernel, there's no difference between correlation and convolution.

10	5	3
4	5	1
1	1	7

$\otimes$

0	0	0
0	0.5	0
0	1.0	0.5

kernel

$=$

	7	

Example: box (mean) filter

Simple method for reducing noise in an image

Convolution kernel

$$g[\cdot,\cdot]$$

$$\frac{1}{9}
\begin{array}{|c|c|c|}
\hline
1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\hline
\end{array}$$

Filtered image

Input image

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Example: box (mean) filter

$$f[\cdot,\cdot]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[\cdot,\cdot]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Gaussian Smoothing (1/2)

A better way to reduce noise and details from an image is to employ a Gaussian filter → **it removes the "high-frequency" components from the image** (low-pass filter)

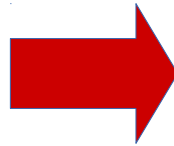Weight contributions of neighboring pixels by nearness:
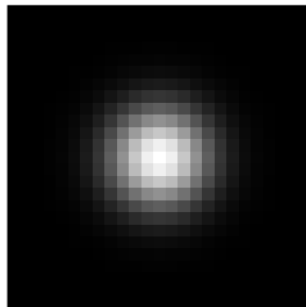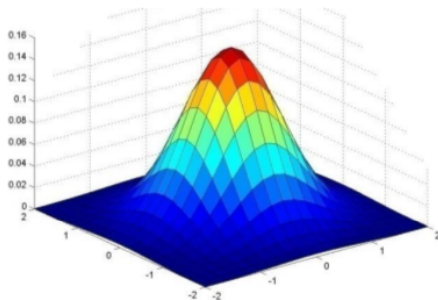
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$
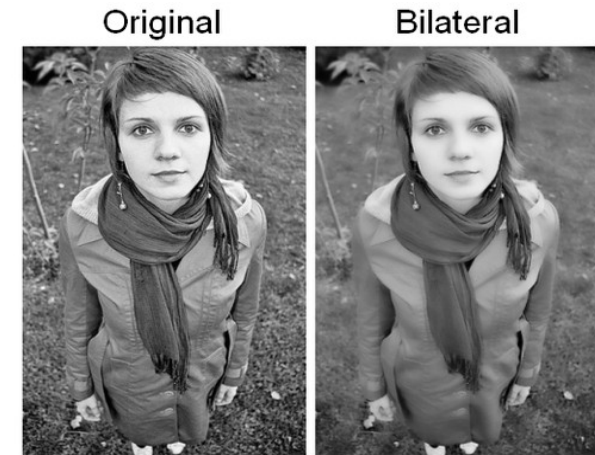
Gaussian Smoothing (2/2)

Bilateral Filter (1/2)

Non-linear, edge-preserving and noise-reducing filter

Is basically a modified Gaussian filter that <u>takes into account also the difference in the intensity domain</u>.

[Tomasi98] Carlo Tomasi, Roberto Manduchi, "Bilateral Filtering for Gray and Color Images", Proceedings of the ICCV 1998

Bilateral Filter (2/2)

Gaussian Blur

$$I_{\mathbf{p}}^{\mathrm{b}} \quad = \quad \sum_{\mathbf{q}\in\mathcal{S}} \boxed{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)}\, I_{\mathbf{q}}$$

space

Bilateral filter

$$I_{\mathbf{p}}^{\mathrm{bf}} \quad = \quad \boxed{\frac{1}{W_{\mathbf{p}}^{\mathrm{bf}}}} \sum_{\mathbf{q}\in\mathcal{S}} \boxed{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)}\, \boxed{G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}\, I_{\mathbf{q}}$$

space   **intensity**

normalization

Edge detection

Edges are very simple but informative part of the image

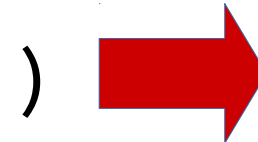Replace image with a binary "edge map" that highlights all the borders in the image

Sobel Operator

The Sobel operator is used to compute the 2D spatial derivatives of an image: higher gradient measurement emphasizes regions of high spatial frequency that correspond to edges.

abs($\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$) →

abs($\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$) →

Sobel Operator

Typically it is used to find the approximate absolute gradient magnitude at each point → **edges**

abs(Sobel x)   abs(Sobel y)

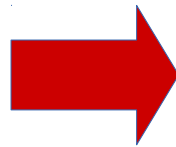Binary version
(threshold = 0.4*max)

0.5*(+   ) =

Hints on the Canny edge detector

Edge localized using an operator very similar to the derivative of a Gaussian

Non-maximum suppression - remove edges orthogonal to a maxima

Hysteresis thresholding - Improved recovery of long image contours
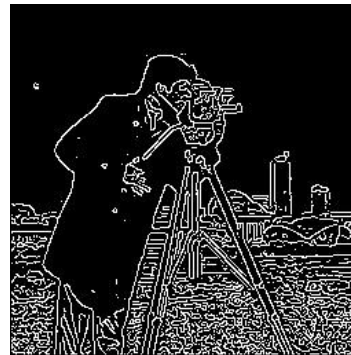
[Canny86] J Canny, "A computational approach to edge detection", Pattern Analysis and Machine Intelligence, IEEE Transactions on, 679-698

LSD - Line Segments Detector
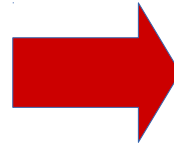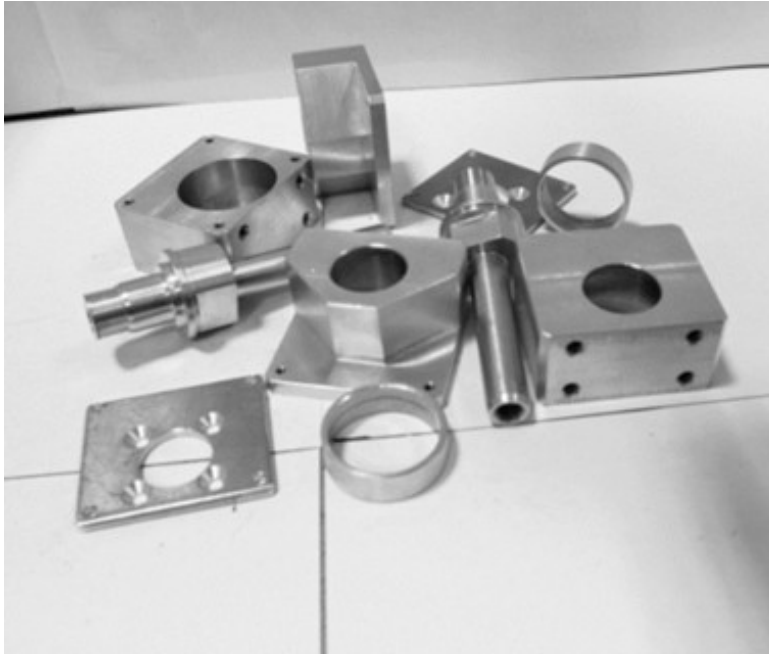
[VonGioi2010] Von Gioi, R. Grompone, et al. LSD: A fast line segment detector with a false detection control, IEEE Transactions on Pattern Analysis and Machine Intelligence 32.4 (2010): 722-732.

Feature Detection

A feature (also called keypoint) can be defined as a meaningful, detectable parts of the image
- Corners, blobs, stable regions

Used to match points in different images

Feature should be repeatable and distinctive
- **Distinctive** : features should be easily matched between them
- **Robust** to noise, blur, discretization, compression
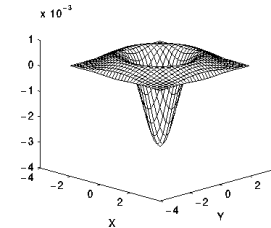- **Repeatable**

SIFT – Scale Invariant Features

Scale-invariant feature transform (SIFT) is a very popular method to **detect** and to **describe** visual features (i.e., provide also a *signature* for each feature).
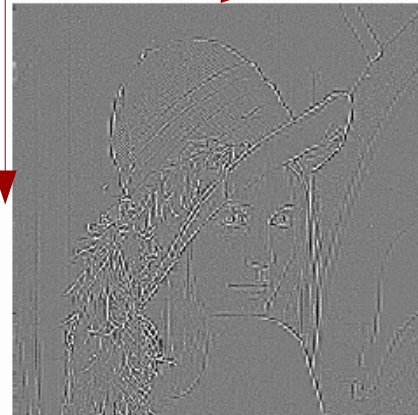
Enable very robust **features matching** also in presence of changing in scale (i.e., depth) and rotation around the optical axis (the x-axis).

[SIFT] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

Detect features using a Laplacian of Gaussian fiter
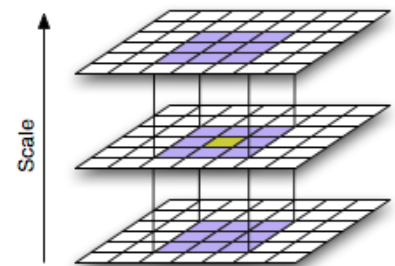
In *space* ...

.. but also in *scale*

original image

Gaussian pyramid

Laplacian pyramid

Scale
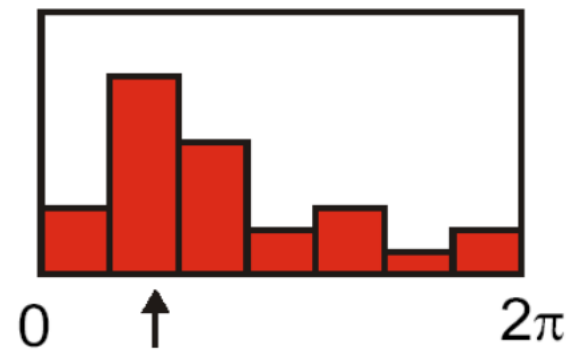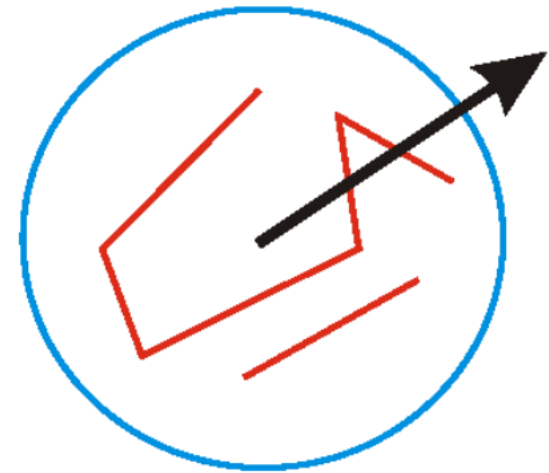
Assign an orientation to each detected point:

Compute gradient magnitude and gradient orientation of the image at each scale.

For each point, create a histogram of local (i.e., inside a patch surrounding the point) **quantized** gradient directions at selected scale
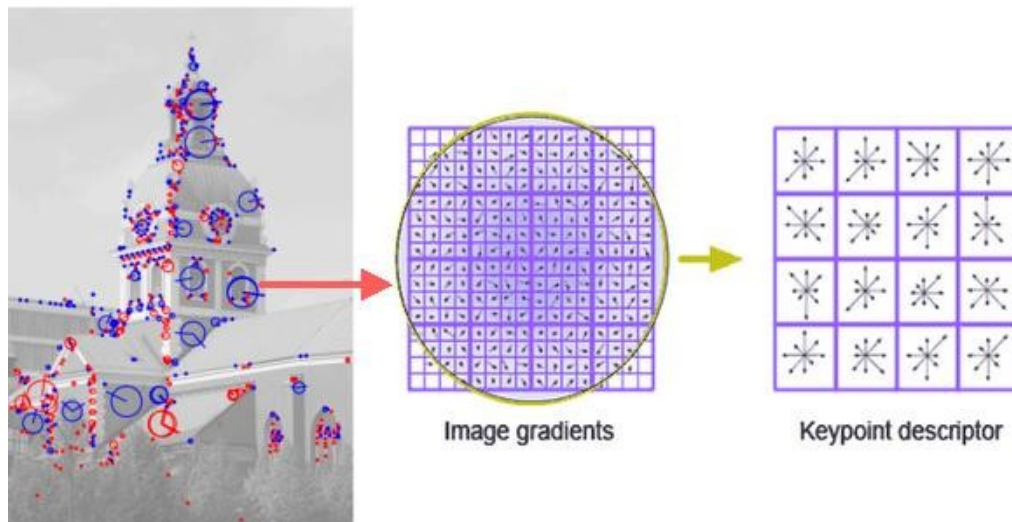
0 2π

Hints on SIFT (3/3)

For each point, define a patch that is rotated to the estimated orientation orientation.
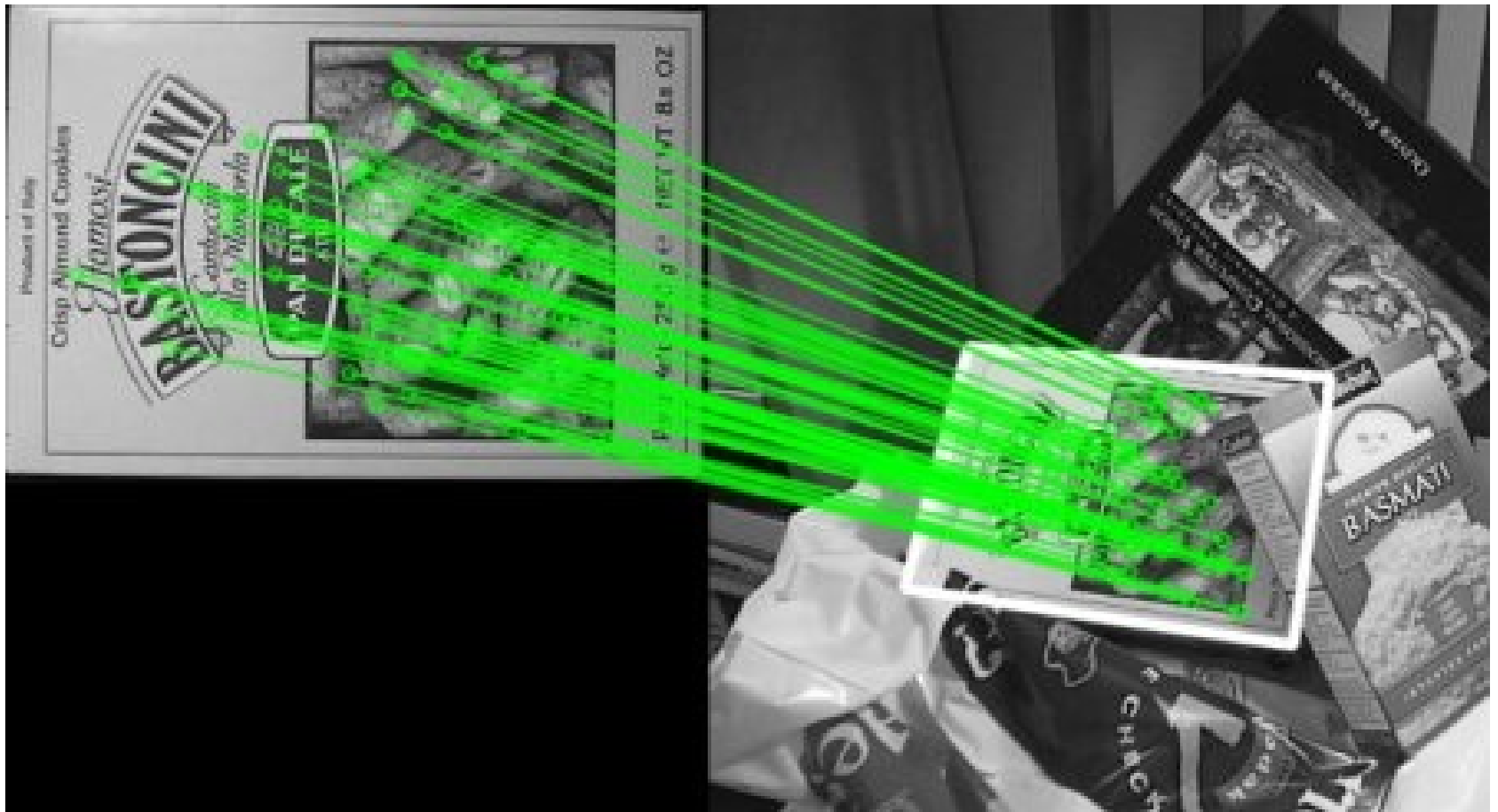
Compute gradient magnitude and orientation at each point in the patch.

Create a normalized orientation histogram over the 4 X 4 subregions of the window → the SIFT descriptor

Image gradients   Keypoint descriptor

Feaures matching with SIFT

Simply use a nearest neighbor search

Other Features

FAST detector

E. Rosten, T. Drummond "Machine Learning for High-speed Corner Detection", ECCV 2006

SURF - Speeded Up Robust Feature

Herbert Bay, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features", ECCV 2006

BRISK - Binary Robust Invariant Scalable Keypoints

Stefan Leutenegger, Margarita Chli and Roland Siegwart: BRISK: Binary Robust Invariant Scalable Keypoints. ICCV 2011: 2548-2555.

ORB - oriented BRIEF

Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: ORB: An efficient alternative to SIFT or SURF. ICCV 2011: 2564-2571.

…

Optical flow

Optical flow: two-dimensional apparent motion field of two consecutive images in an image sequence.

Whenever a camera records a scene over time, the resulting (gray levels) image sequence can be considered as a <u>function</u> I (x, y, t) of the image pixel position x = (x, y) and time t.
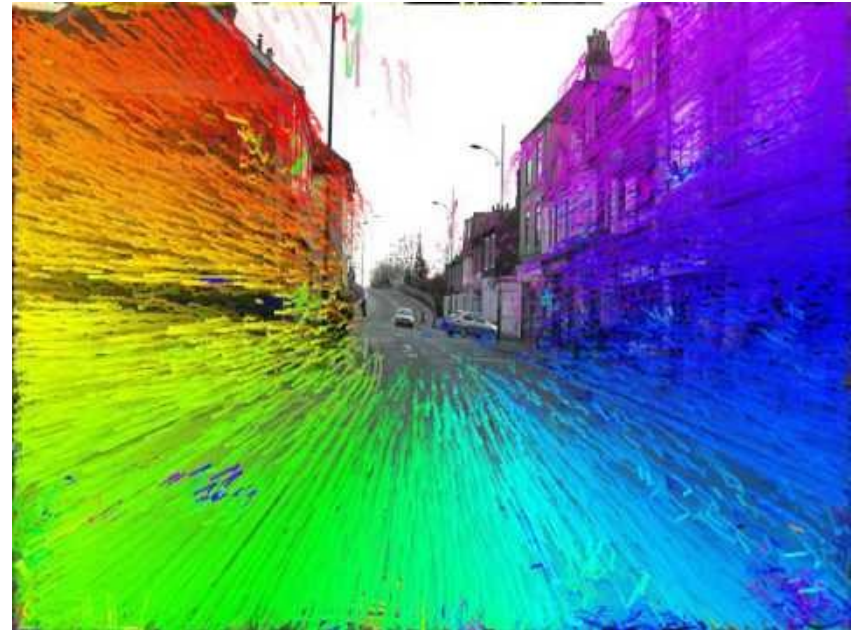
Brightness Constancy Equation

$$I(x, y, t-1) = I(x+u(x,y), y+v(x,y), t)$$

Linearizing right hand side using Taylor expansion:

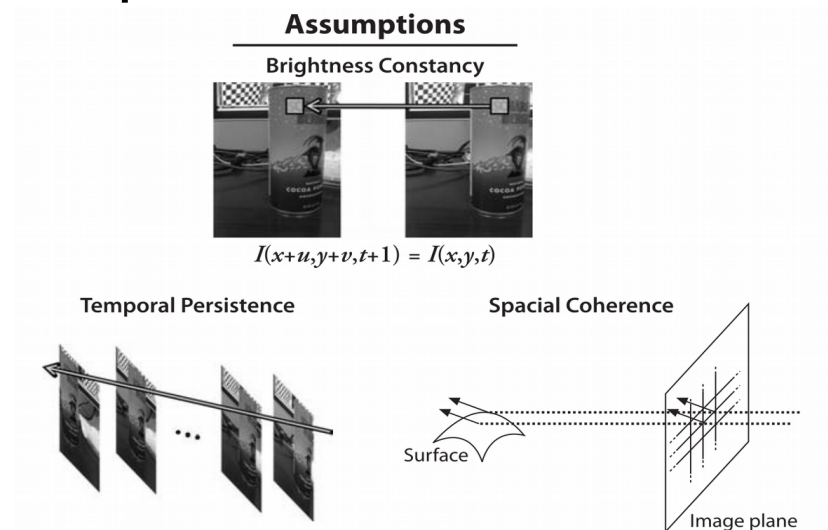$$I(x+u, y+v, t) \approx I(x, y, t-1) + I_x \cdot u(x,y) + I_y \cdot v(x,y) + I_t$$

$$I(x+u, y+v, t) - I(x, y, t-1) = I_x \cdot u(x,y) + I_y \cdot v(x,y) + I_t$$

$$\Rightarrow \quad I_x \cdot u + I_y \cdot v + I_t \approx 0$$

Hints on L-K Method (1/2)

Lucas–Kanade is a *local* method: selected features are tracked over time and their movement is converted into velocity vectors.

The **Lucas-Kanade** (**LK**) algorithm is usually used to perform **sparse** optical-flow.

[LK] B. D. Lucas and T. Kanade (1981), An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pages 121--130

Hints on L-K Method (2/2)

L-K method minimize the sum of quadratic deviations of the brightness constancy equation in a neighborhood *N* of each feature x

$$\min_{u,v}\left\{ \sum_{\mathbf{x}'\in\mathcal{N}(\mathbf{x})} \left(I_t(\mathbf{x}') + I_x(\mathbf{x}')u + I_y(\mathbf{x}')v\right)^2 \right\}.$$

In order to deal with large displacement optical flow, image pyramids are usually used by solving for low frequency structures in low resolution images first and refining the search on higher resolved images.
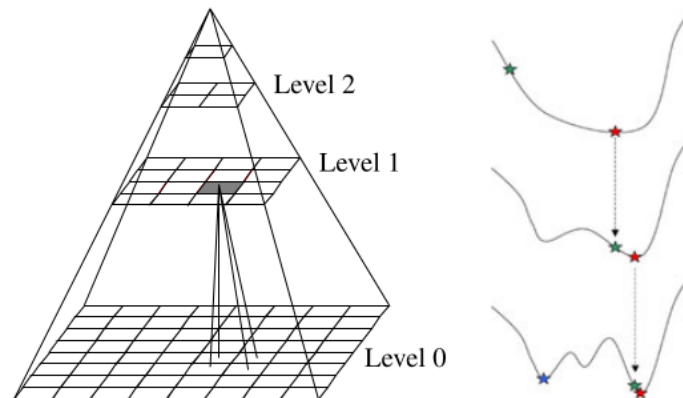
Pyramid L-K example

The real frame rate is 4X

Dense optical flow (1/2)

Variational method [Horn and Schunck, 1981]: variational approaches → penalizing the derivative of the optical flow field.

$$\min_{u(\mathbf{x}), v(\mathbf{x})} \left\{ \int_{\Omega} \left(|\nabla u(\mathbf{x})|^2 + |\nabla v(\mathbf{x})|^2 \right) \mathrm{d}\Omega + \lambda \int_{\Omega} \left(I_t + I_x u(\mathbf{x}) + I_y v(\mathbf{x}) \right)^2 \mathrm{d}\Omega \right\}.$$

Regularization term

Variational optical flow approaches compute the optical flow field for all pixels within the image.

[Horn and Schunck, 1981] B.K.P. Horn and B.G. Schunck, "Determining optical flow." Artificial Intelligence, vol 17, pp 185–203, 1981.

Dense optical flow (2/2)

The became only recently more popular as processor speed has increased.

- Smoothing and fixed point iterations [Brox et al. '04]
- Primal-dual optimization [Chambolle, Pock, '10]
- …

[Brox et al. '04] Brox, Thomas, et al. "High accuracy optical flow estimation based on a theory for warping." Computer Vision-ECCV 2004. Springer Berlin Heidelberg, 2004.

[Chambolle, Pock, '10] Chambolle, A. & Pock, T. "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging" J Math Imaging Vis (2011) 40: 120.

Stereo vs Optical Flow

Stereo matching can be considered a special case of optical flow
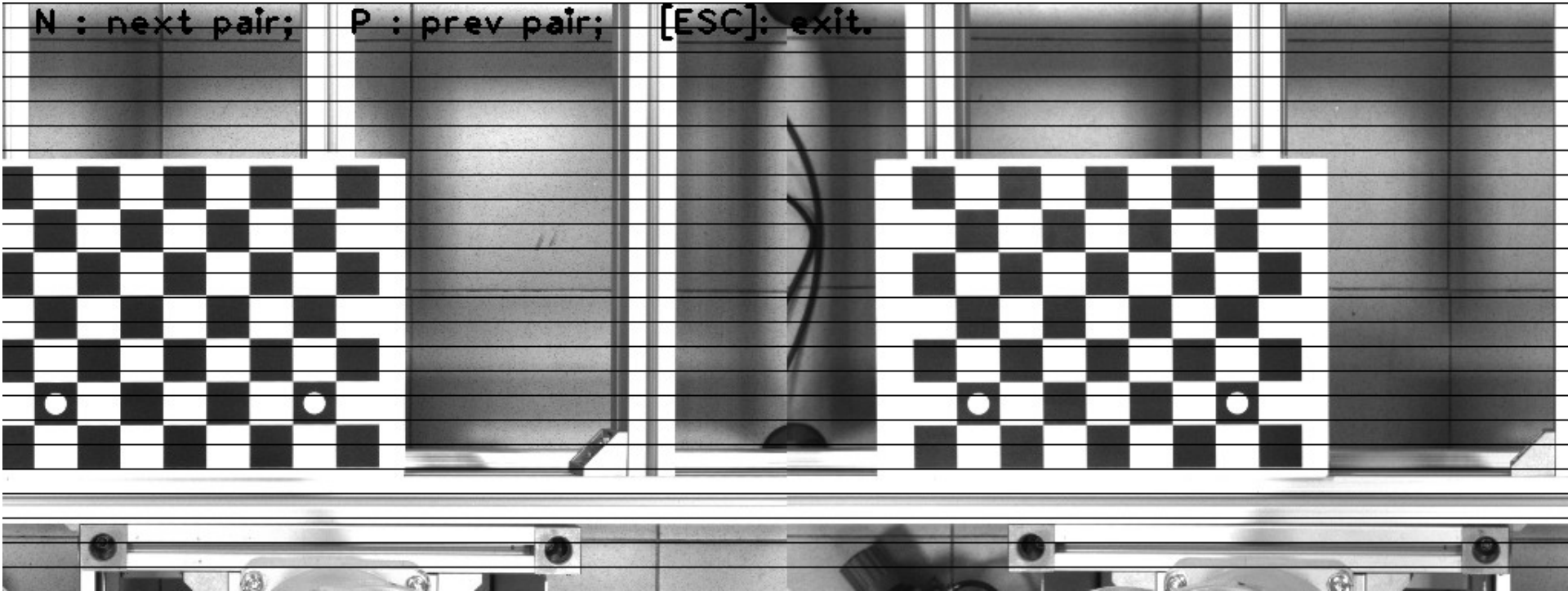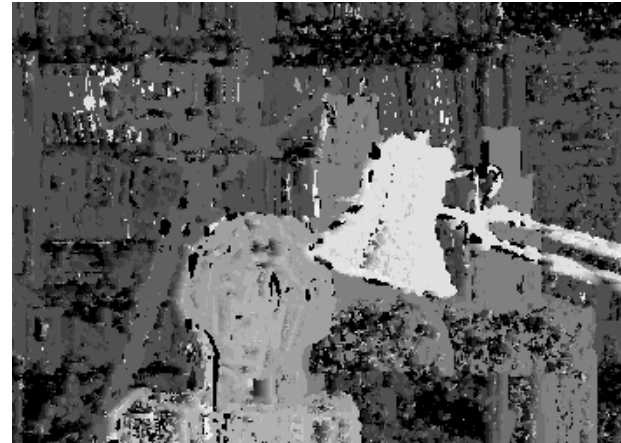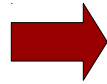

(a)

(b)

(c)

(d)

Stereo Rectification

Dense stereo matching

Dense stereo using a block matching algorithm: pixelwise cost calculation is generally ambiguous!

Hints on Semi-global matchnig (1/2)

As in variational optical flow, add regularization terms:

sum of all pixel matching costs for the disparity map D

Penalty term for small changes

$$E(D) = \sum_{\mathbf{p}} (C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 \, \mathrm{T}[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1]$$
$$+ \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 \, \mathrm{T}[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1])$$

Penalty term for big changes

NP-complete problem!

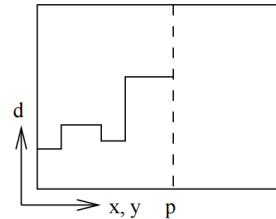– Aggregate costs for a number of directions

[Hirschmuller 2010] H Hirschmuller "Stereo processing by semiglobal matching and mutual information", Pattern Analysis and Machine Intelligence, IEEE Transactions on 30 (2), 328-341
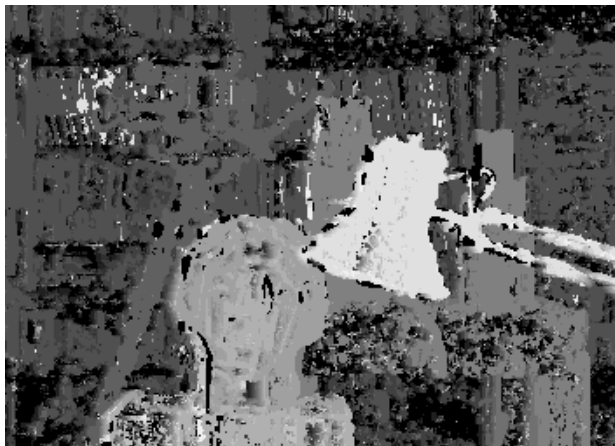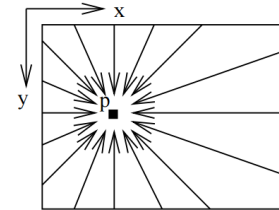
Hints on Semi-global matchnig (2/2)

$$L'_{\mathbf{r}}(\mathbf{p}, d) = C(\mathbf{p}, d) + \min(L'_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d),$$
$$L'_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d - 1) + P_1,$$
$$L'_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d + 1) + P_1,$$
$$\min_i L'_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + P_2).$$

$$S(\mathbf{p}, d) = \sum_{\mathbf{r}} L_{\mathbf{r}}(\mathbf{p}, d)$$

Minimum Cost Path $L_r(p, d)$

16 Paths from all Directions r

Using SGM

Seminars in Artificial Intelligence and Robotics

Computer Vision for Intelligent Robotics

Basics and hints on low level vision

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Alberto Pretto