香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
電子及計算機工程學系
DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING

Introduction to Aerial Robotics
Lecture 6

Shaojie Shen

Assistant Professor

Dept. of ECE, HKUST

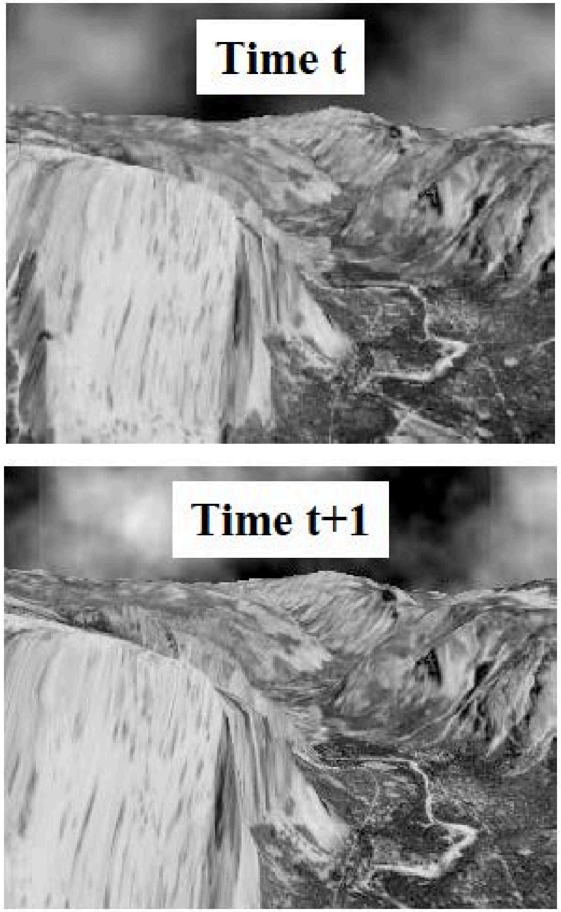
13 October 2015

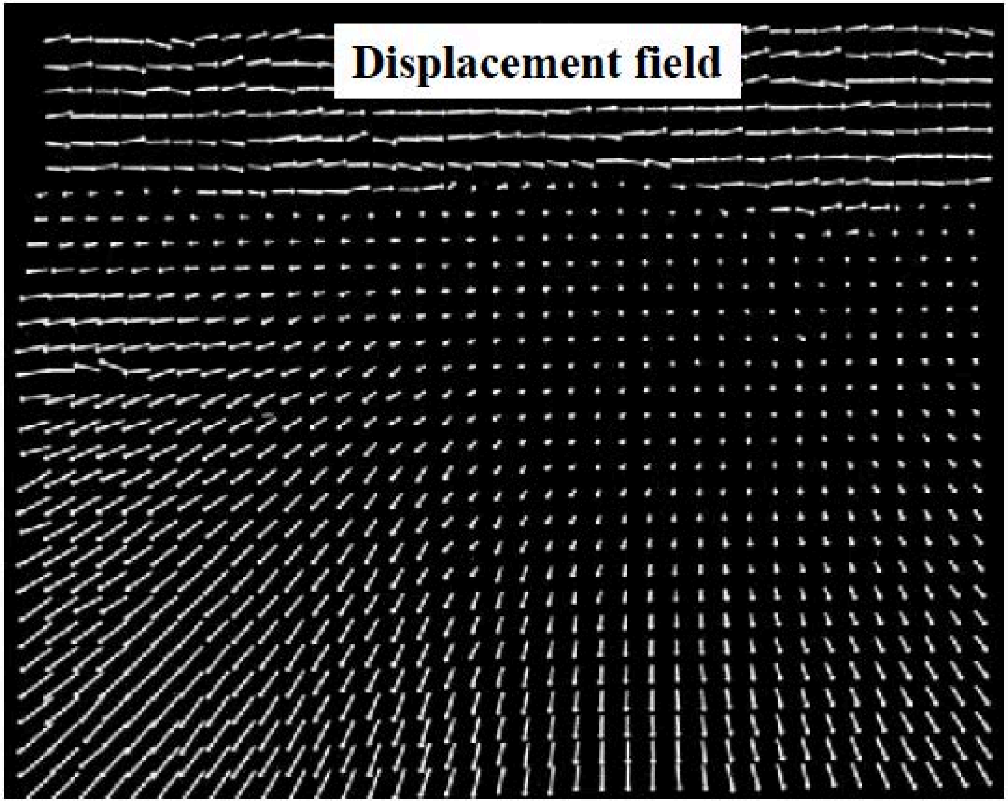
Outline

- Optical Flow
- Dense Stereo

Optical Flow

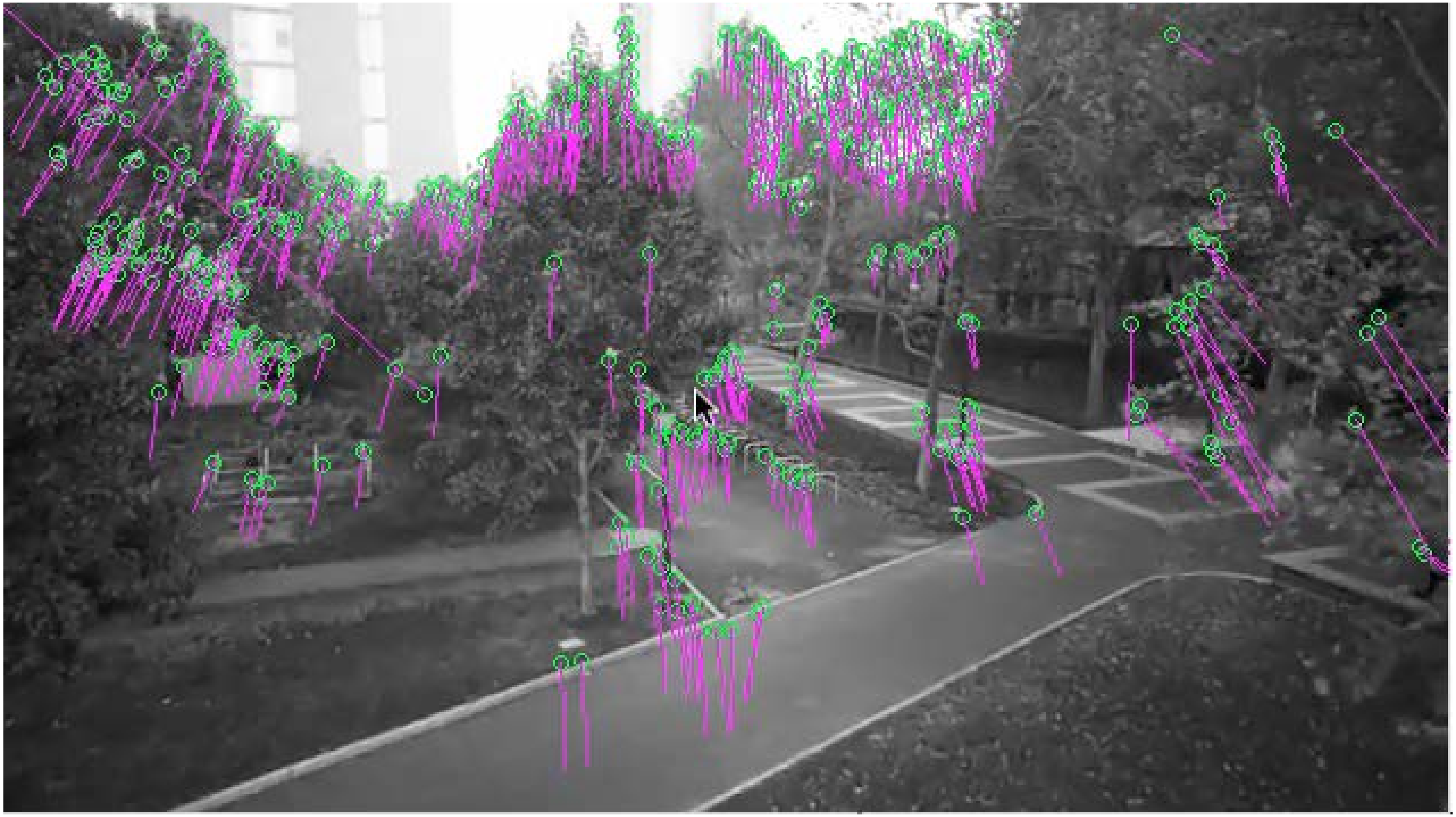
Optical Flow





Adapted from slides from Kostas Daniilidis

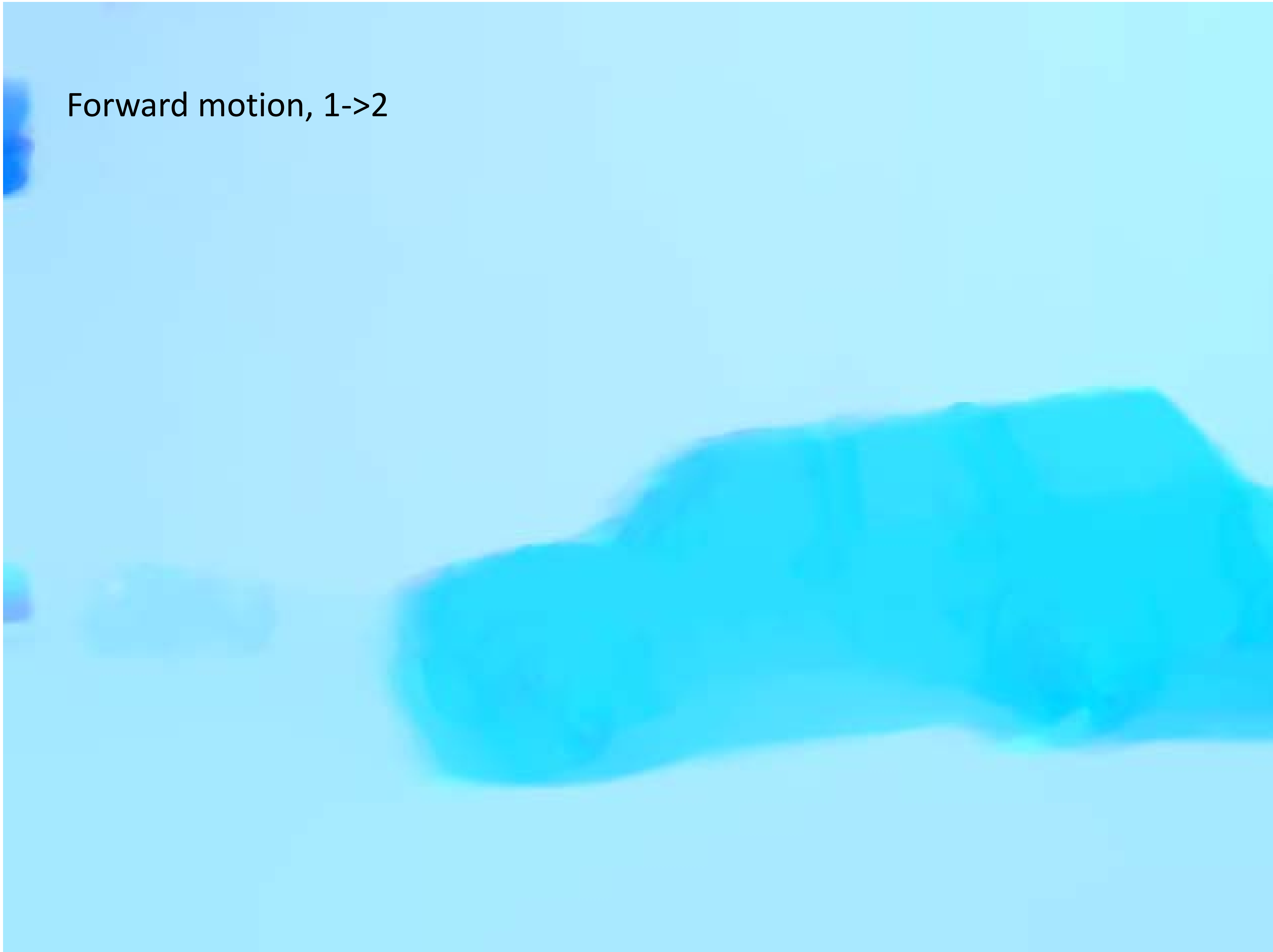
Optical Flow



Adapted from slides from Kostas Daniilidis

Adapted from slides from Kostas Daniilidis

Adapted from slides from Kostas Daniilidis

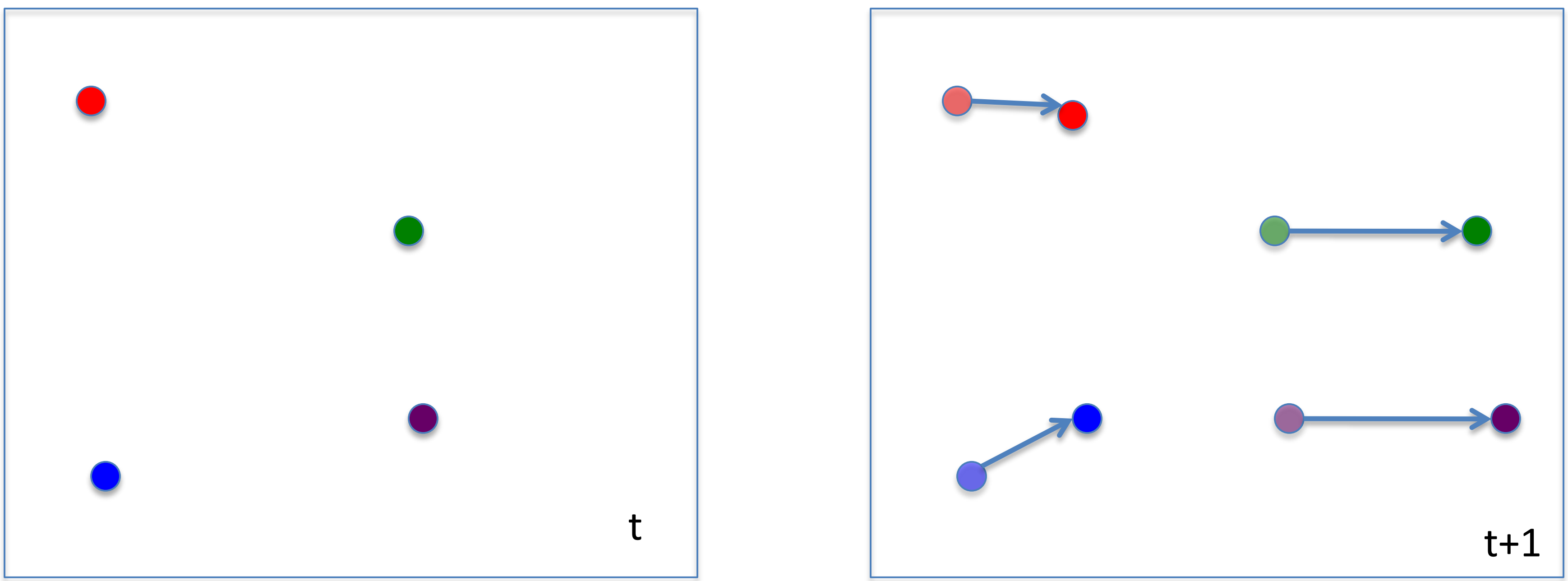


Adapted from slides from Kostas Daniilidis

Adapted from slides from Kostas Daniilidis

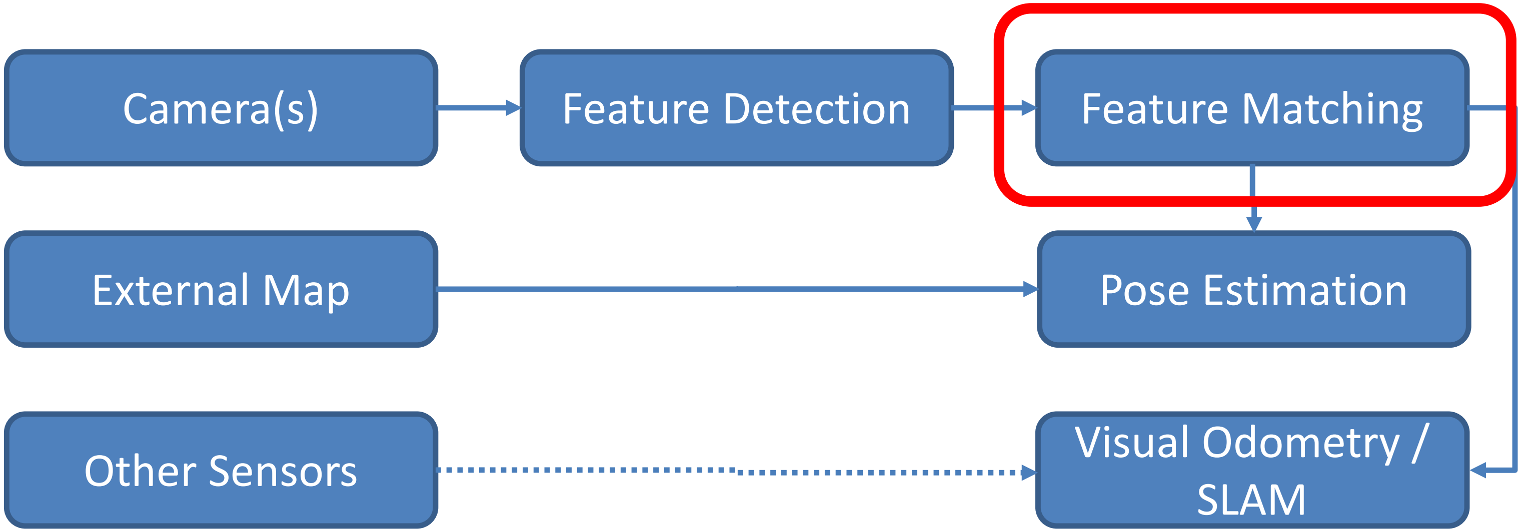
Problem Definition

- Define regions of interests, or points of interests in the first image at time t
- Search for correspondences in the second image at time $t + 1$



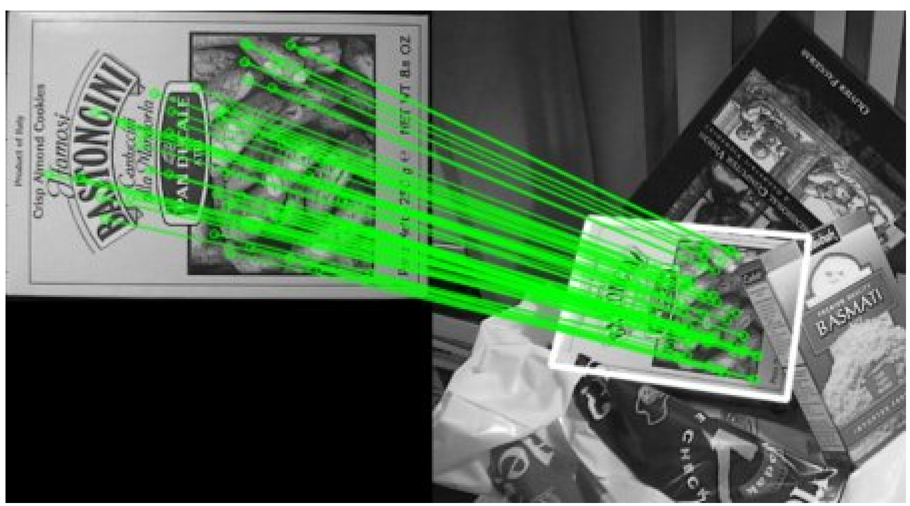
Adapted from slides from Kostas Daniilidis

Vision-based State Estimation Pipeline



Discrete Matching Approach

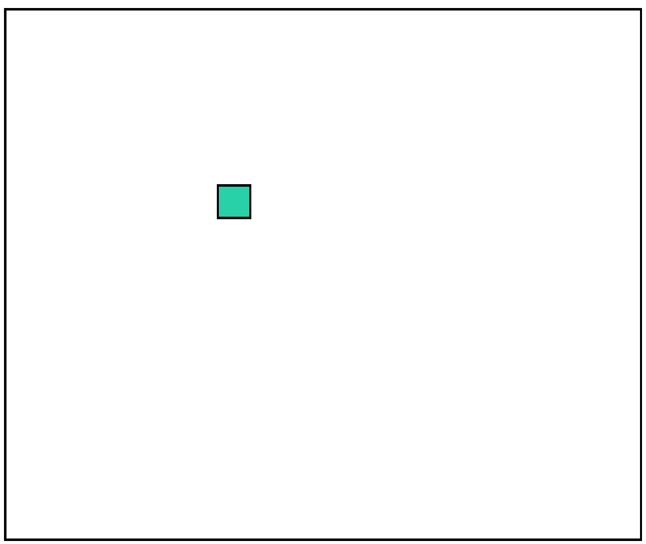
- Detect corners features in both images
- Use image patch as feature description
  - Could be extended to color, texture, SIFT/HOG descriptor
- Find correspondences using descriptor matching



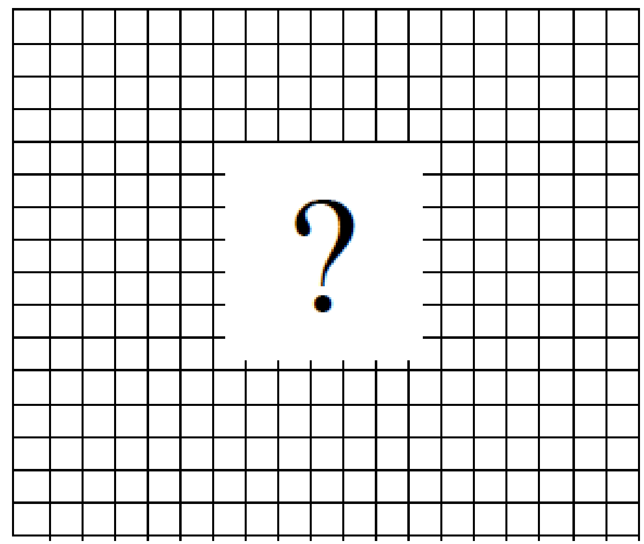
Differential Approach: KLT Tracker

- Detect corners features in first image
- Use image patch as feature description
  - Could be extended to color and texture descriptors
- Use Lucas-Kanade algorithm to compute displacement of the pixels in the patch
  - Motion model could be translation (2-DoF), affine (6-DoF), or more general 3D models
- Subpixel accuracy
- Do not need repeated detection

Given image patch in one image

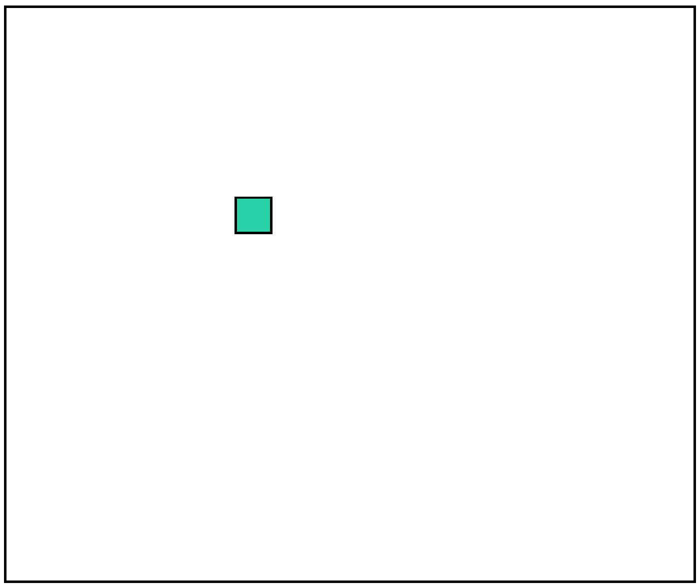


We don't want to search everywhere in the second image for a match.

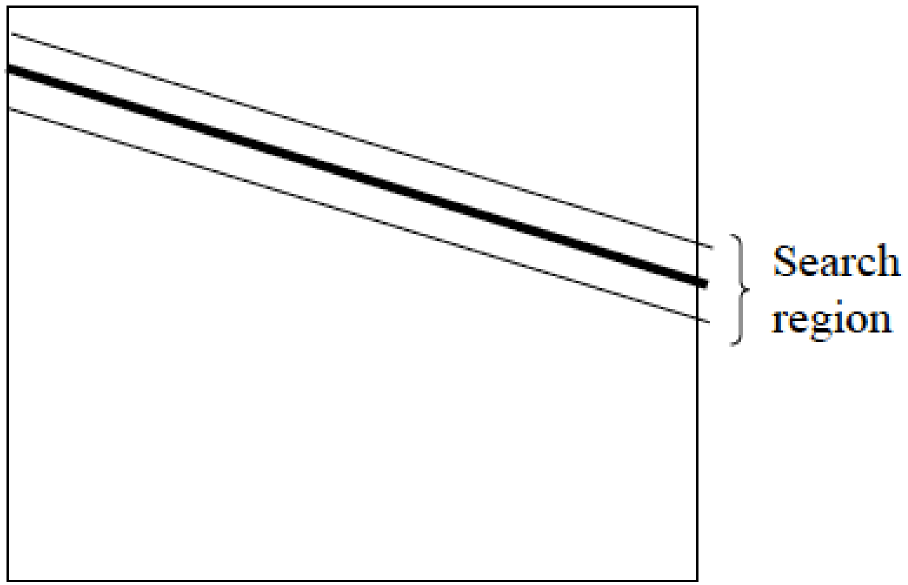


Adapted from slides from Kostas Daniilidis

Given image patch in one image



We don't want to search everywhere in the second image for a match.

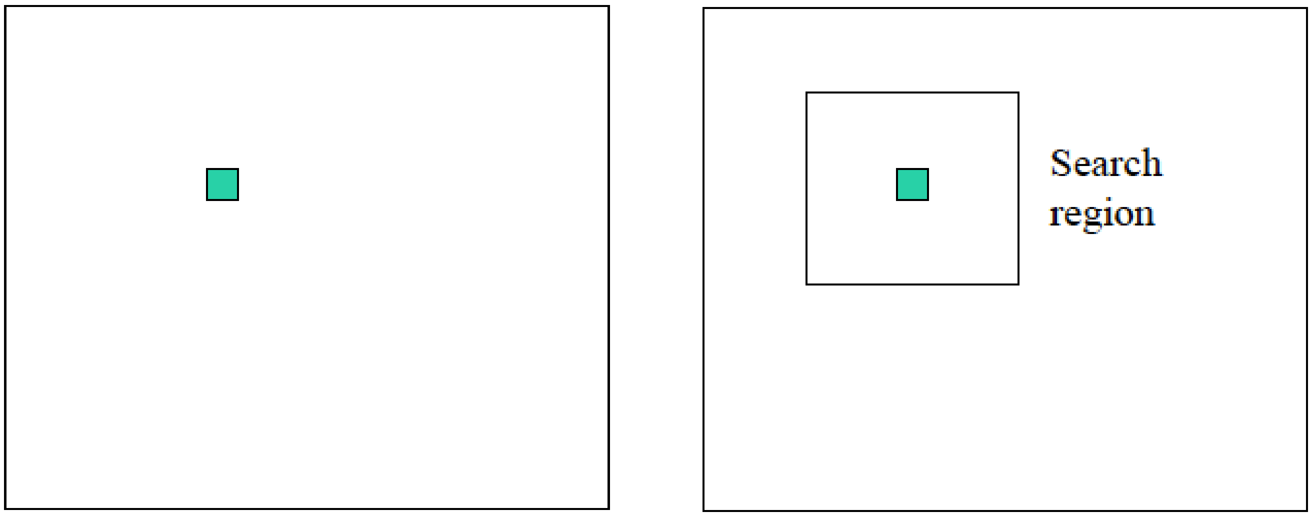


With stereo, we had an epipolar line constraint.

Adapted from slides from Kostas Daniilidis

Given image patch in one image

We don't want to search everywhere in the second image for a match.

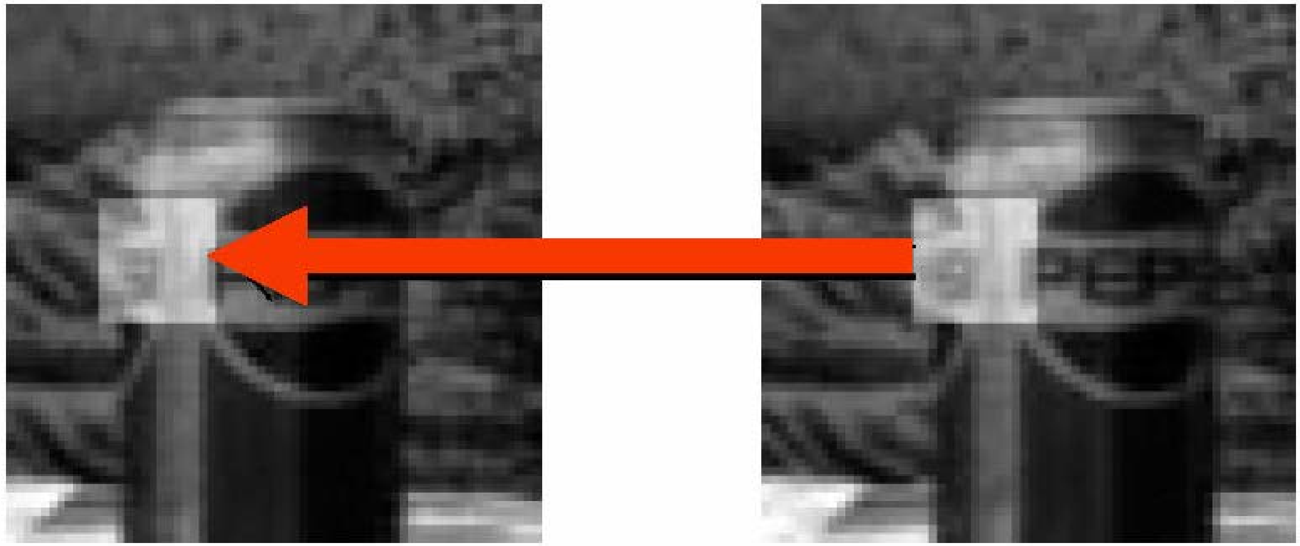


We don't know the relative R,T here, so no epipolar constraint.
But… motion is known to be "small", so can still bound the search region.

Adapted from slides from Kostas Daniilidis

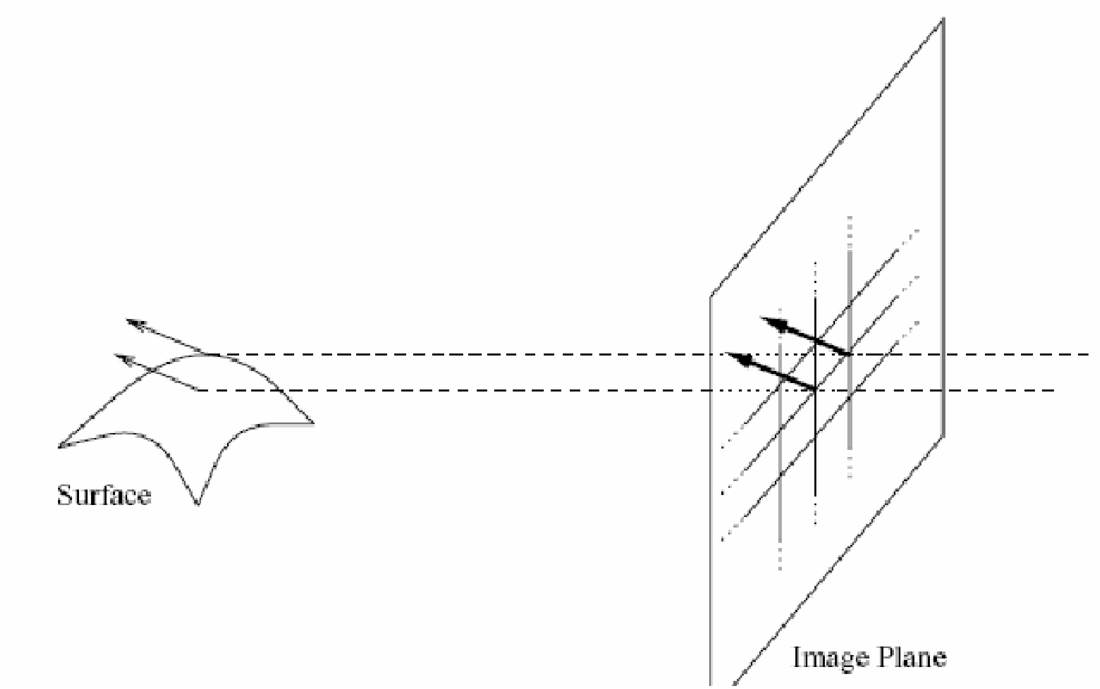
Optical Flow Assumption: Brightness Constancy

- Image measurements (brightness) in a small region remains the same even though their location may change



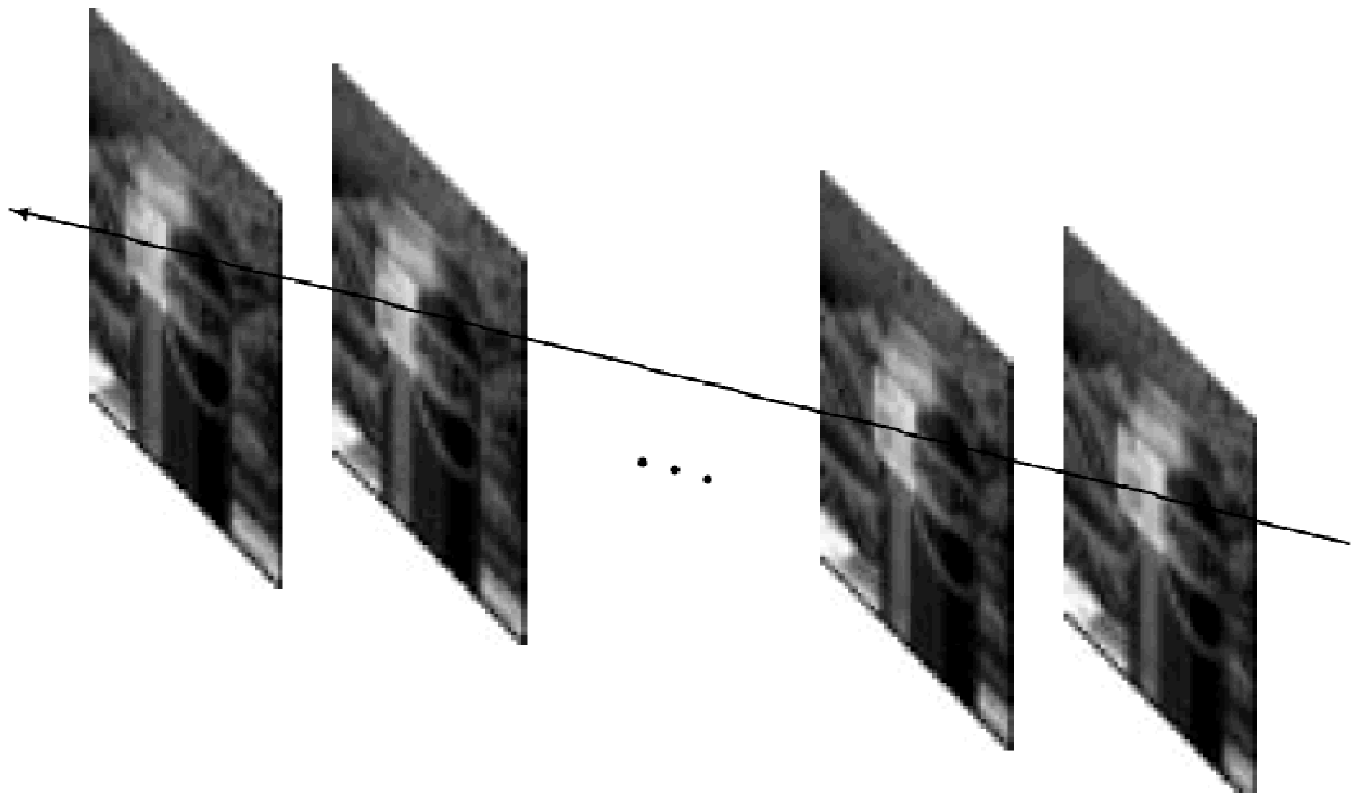
Optical Flow Assumption: Spatial Coherence

- Neighboring points in the scene typically belong to the same surface and have similar motions



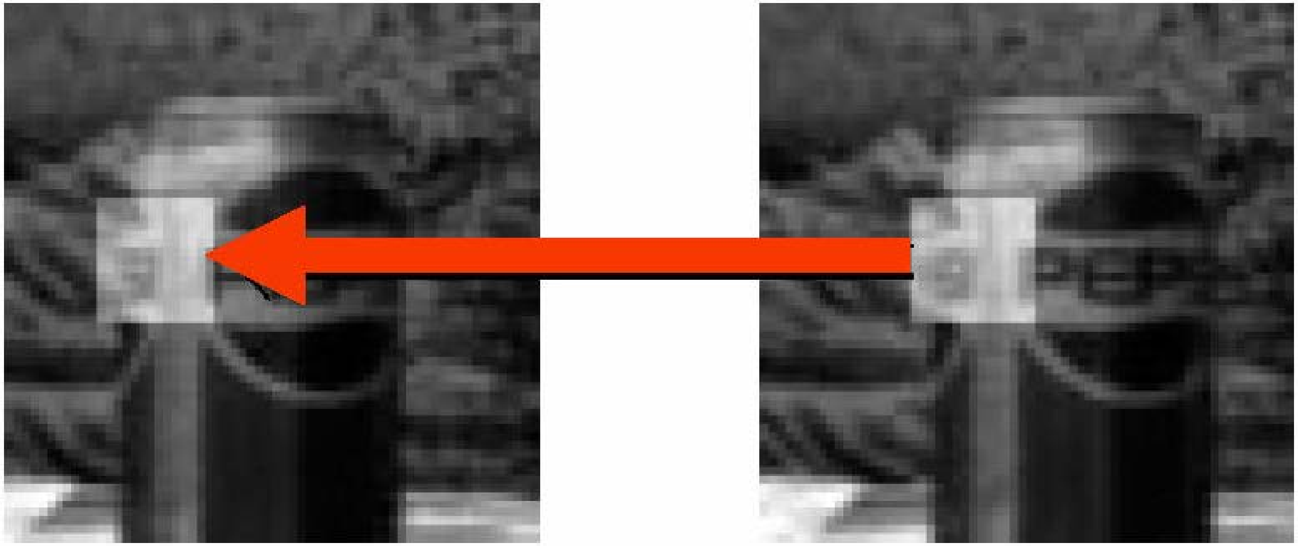
Optical Flow Assumption: Temporal Persistence

- Image motion of a surface patch changes smoothly over time



Lucas-Kanade (KLT) Tracking

- Intensity constancy constraint: $J(x+d) = I(x)$
  - $J(x) = I(x, t+1)$
  - $I(x) = I(x, t)$



Lucas-Kanade (KLT) Tracking

- Define Sum of Squared Difference (SSD) error as:
  - $\epsilon = \int_W [\, J(x+d) - I(x)\,]^2 \omega(x) dx$
  - $\omega(x)$ is the smoothing term
  - Minimize ϵ with respect to $d \in R^{2\times 1}$
- 4 steps for solving this problem:
  - Set $\frac{\partial \epsilon}{\partial d}$ to 0
  - Linearization by Taylor expansion on $J(x+d)$ with respect to d
  - Solve the resulting linearized system
  - Iterative refinement

Step 1: Set Derivative to 0

Differentiate SSD with respect to d and set to 0:

$$\frac{1}{2}\frac{\partial \epsilon}{\partial d} = \int_W [J(x+d) - I(x)]\; g\; w\; dx = 0$$

$$g = \left(\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y}\right)^T$$

Adapted from slides from Kostas Daniilidis

Step 1: Set Derivative to 0

Differentiate SSD with respect to d and set to 0:

$$\frac{1}{2}\frac{\partial \epsilon}{\partial d} = \int_W [J(x+d) - I(x)]\, g\ w\ dx = 0$$

$$g = \left(\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y}\right)^T$$

Step 2: Linearization

$$\boxed{[J(x+d) - I(x)]}$$

Assume small motion, Taylor expansion of $J(x+d)$ is:

$$\boxed{J(x+d) = J(x) + g^T\ d}$$

$$g = (\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y})^T$$

Adapted from slides from Kostas Daniilidis

Step 2: Linearization

Combining previous equations:

$$\frac{1}{2}\frac{\partial \epsilon}{\partial d} = \int_W [J(x+d) - I(x)]\, g\, w\, dx = 0$$

$$J(x+d) = J(x) + g^T\, d$$

$$\downarrow$$

$$\boxed{\int_W g\ (g^T d) w\ dx = \int_W [I(x) - J(x)] g\ w\ dx}$$

Adapted from slides from Kostas Daniilidis

Step 3: Solve Linear System

$$\boxed{\int_W g\,(g^T d) w\,dx = \int_W [I(x) - J(x)] g\,w\,dx}$$

$$\int_W g\,(g^T) w\,dx = \underbrace{\sum_{i,j} \begin{bmatrix} g_x(i,j)g_x(i,j) & g_y(i,j)g_x(i,j) \\ g_x(i,j)g_y(i,j) & g_y(i,j)g_y(i,j) \end{bmatrix}}_{\text{A: second moment matrix}}$$

A: second moment matrix

Adapted from slides from Kostas Daniilidis

Step 3: Solve Linear System

$$\boxed{\int_W g\,(g^T d) w\,dx = \int_W [I(x) - J(x)] g\,w\,dx}$$

$$\int_W g\,(g^T) w\,dx = \underbrace{\sum_{i,j} \begin{bmatrix} g_x(i,j)[I(i,j) - J(i,j)] \\ g_y(i,j)[I(i,j) - J(i,j)] \end{bmatrix}}_{\text{Error vector b}}$$

Adapted from slides from Kostas Daniilidis

Step 3: Solve Linear System

$$\boxed{\int_W g\,(g^T d)w\;dx = \int_W [I(x) - J(x)]g\;w\;dx}$$

$$A = \sum_{i,j} \begin{bmatrix} g_x(i,j)g_x(i,j) & g_y(i,j)g_x(i,j) \\ g_x(i,j)g_y(i,j) & g_y(i,j)g_y(i,j) \end{bmatrix}$$

$$b = \sum_{i,j} \begin{bmatrix} g_x(i,j)[I(i,j) - J(i,j)] \\ g_y(i,j)[I(i,j) - J(i,j)] \end{bmatrix}$$

$$A\,d = b$$

$$d = A^{-1}b$$

Adapted from slides from Kostas Daniilidis

Step 3: Solve Linear System

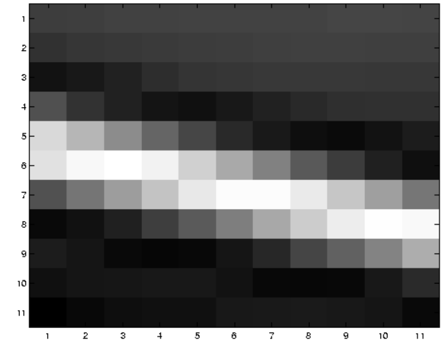
$$\boxed{\int_W g\ (g^T d) w\ dx = \int_W [I(x) - J(x)] g\ w\ dx}$$

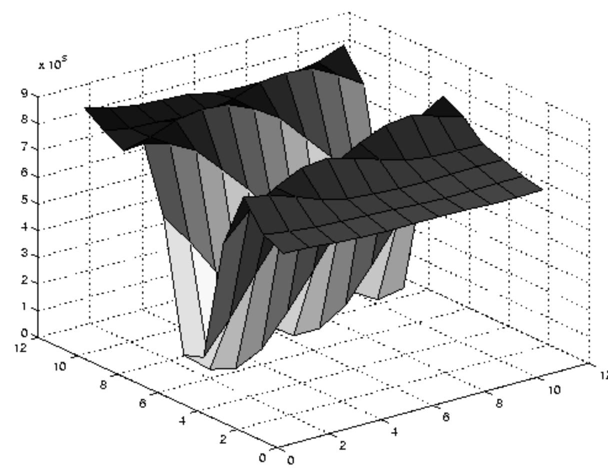
$$A = \sum_{i,j} \left[\begin{array}{cc} g_x(i,j) g_x(i,j) & g_y(i,j) g_x(i,j) \\ g_x(i,j) g_y(i,j) & g_y(i,j) g_y(i,j) \end{array} \right] \qquad b = \sum_{i,j} \left[\begin{array}{c} g_x(i,j)[I(i,j) - J(i,j)] \\ g_y(i,j)[I(i,j) - J(i,j)] \end{array} \right]$$

$$d = A^{-1} b$$

- What if A is not full rank? Recall the structure of A:
  - Same as the one for corner detection
  - Eigenvalues and eigenvectors of A tells whether we are tracking a corner

Edge



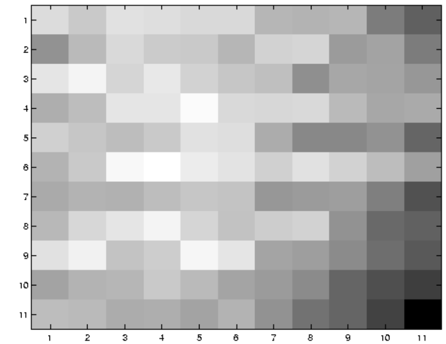


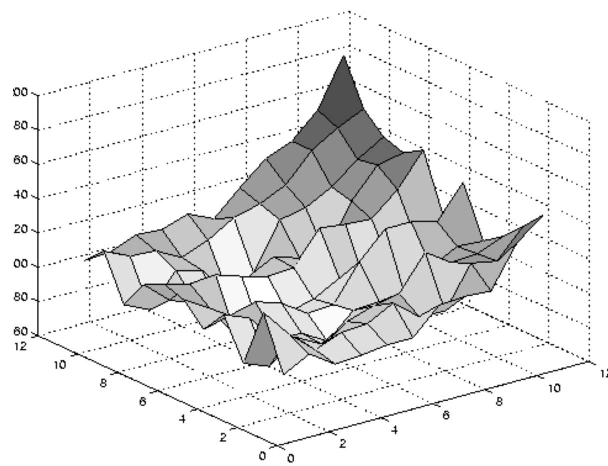
– large gradients, all the same direction

– large λ_1, small λ_2

Adapted from slides from Kostas Daniilidis

Low Texture Region



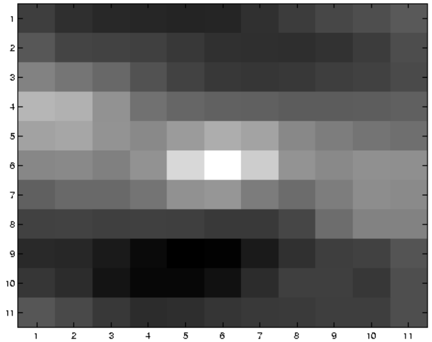


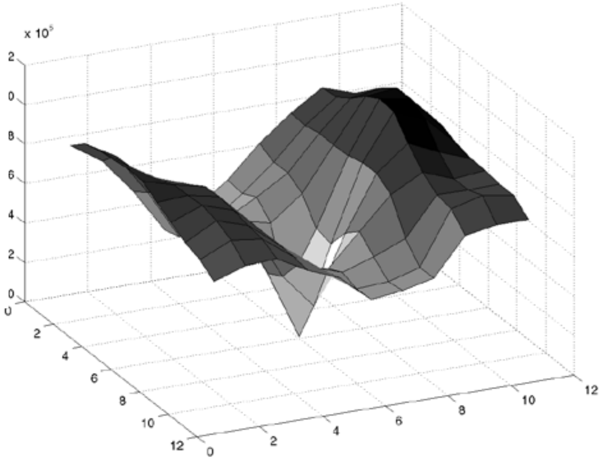
– gradients have small magnitude

– small λ_1, small λ_2

Adapted from slides from Kostas Daniilidis

High Texture Region





– gradients are different, large magnitudes
– large λ_1, large λ_2

Adapted from slides from Kostas Daniilidis

Step 4: Iterative Refinement

- Iterative refinement
  - Estimate velocity at pixels of interests using one iteration of Lucas-Kanade algorithm
  - Transform pixels using the estimated flow field
  - Refine estimate by repeating the process

Step 4: Iterative Refinement

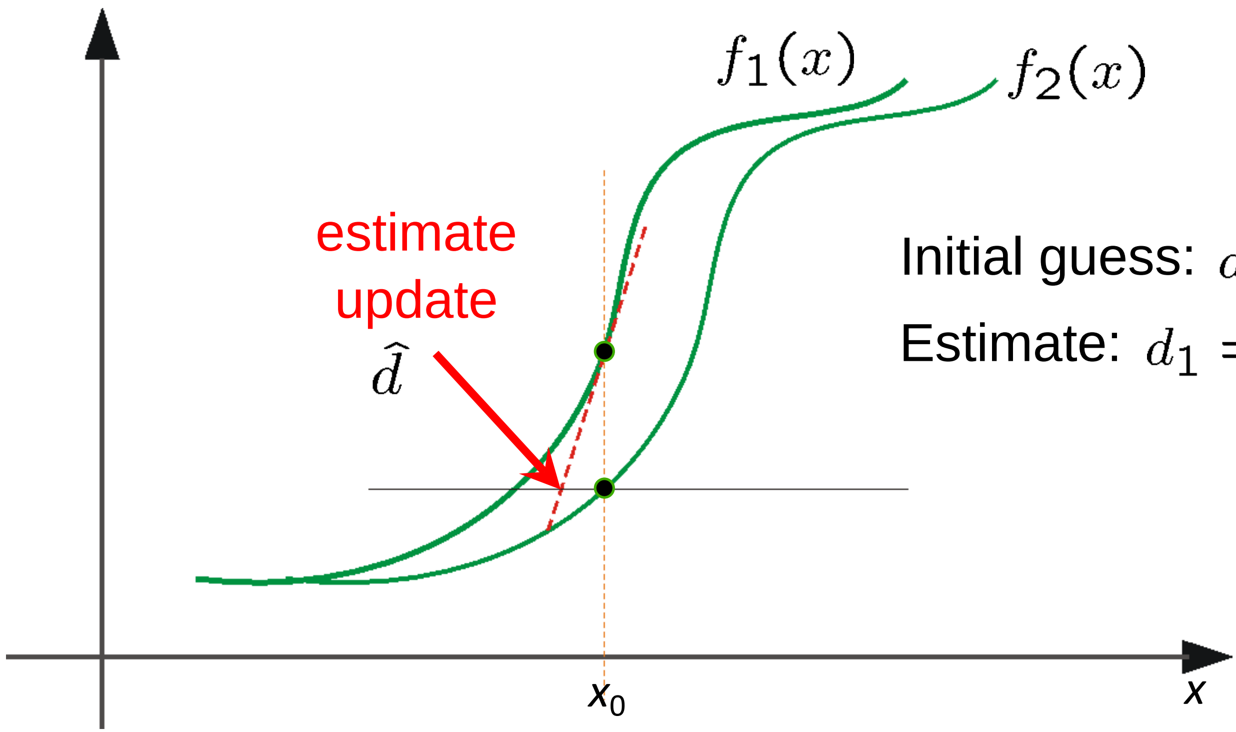
$$\boxed{\int_W g\ (g^T d) w\ dx = \int_W [I(x) - J(x)] g\ w\ dx}$$

$$A = \sum_{i,j} \begin{bmatrix} g_x(i,j) g_x(i,j) & g_y(i,j) g_x(i,j) \\ g_x(i,j) g_y(i,j) & g_y(i,j) g_y(i,j) \end{bmatrix} \qquad b = \sum_{i,j} \begin{bmatrix} g_x(i,j)[I(i,j) - J(i,j)] \\ g_y(i,j)[I(i,j) - J(i,j)] \end{bmatrix}$$

$$d = A^{-1} b$$

- Iterate:
  - Update $J_{i+1}(x) \rightarrow J_i(x+d)$
  - Recompute d between $J_{i+1}(x)$ and $I(x)$

Step 4: Iterative Refinement

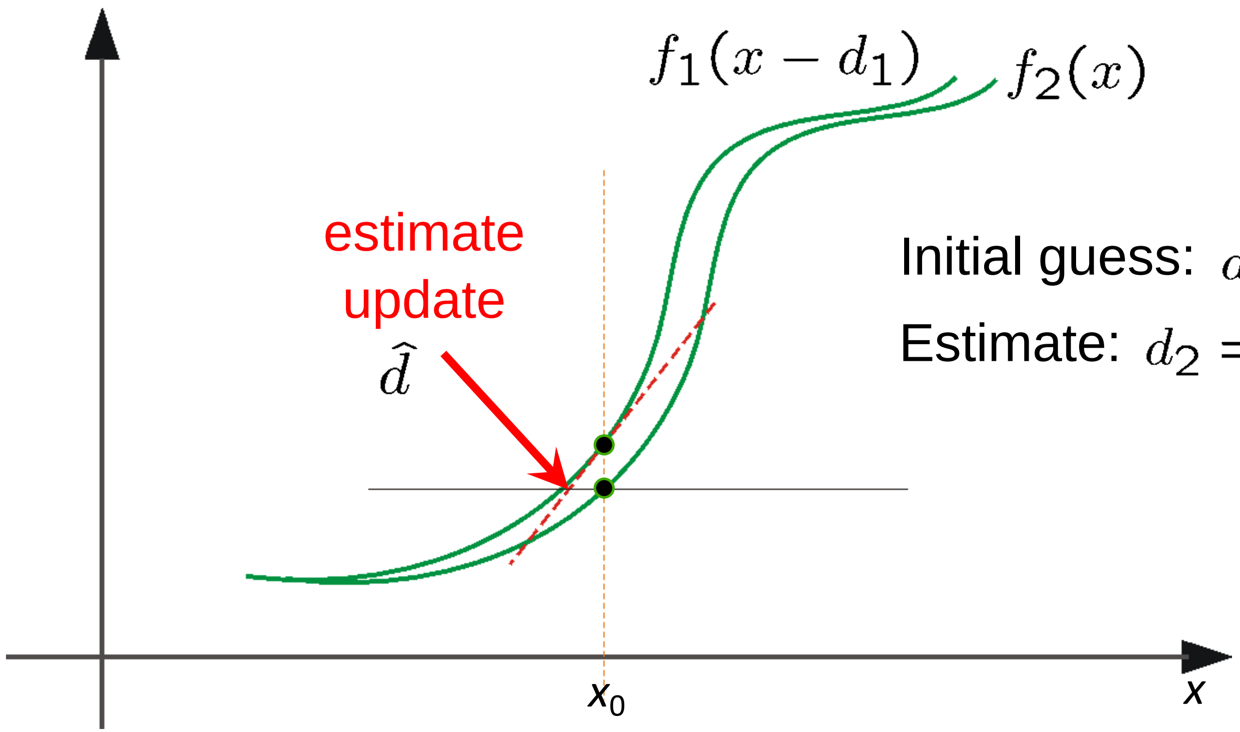


Initial guess: $d_0 = 0$

Estimate: $d_1 = d_0 + \hat{d}$

Adapted from slides from Richard Szeliski

Step 4: Iterative Refinement

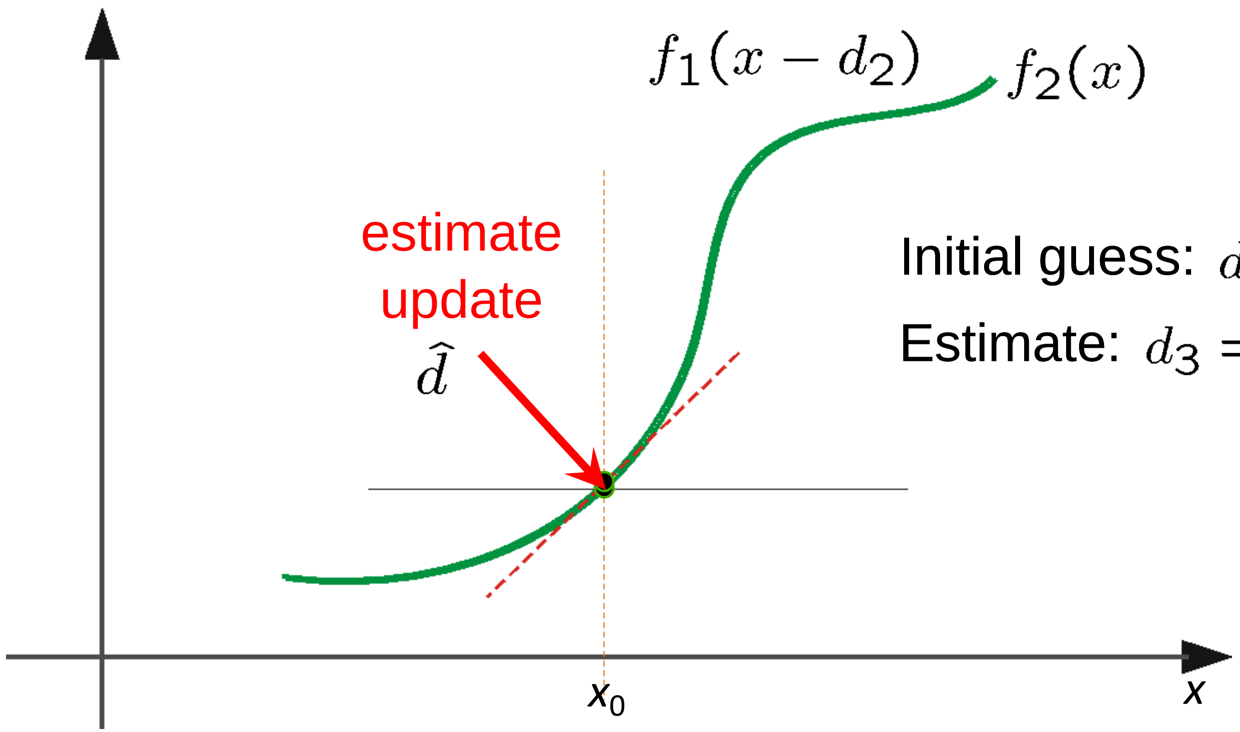


Initial guess: d_1

Estimate: $d_2 = d_1 + \hat{d}$

Adapted from slides from Richard Szeliski

Step 4: Iterative Refinement

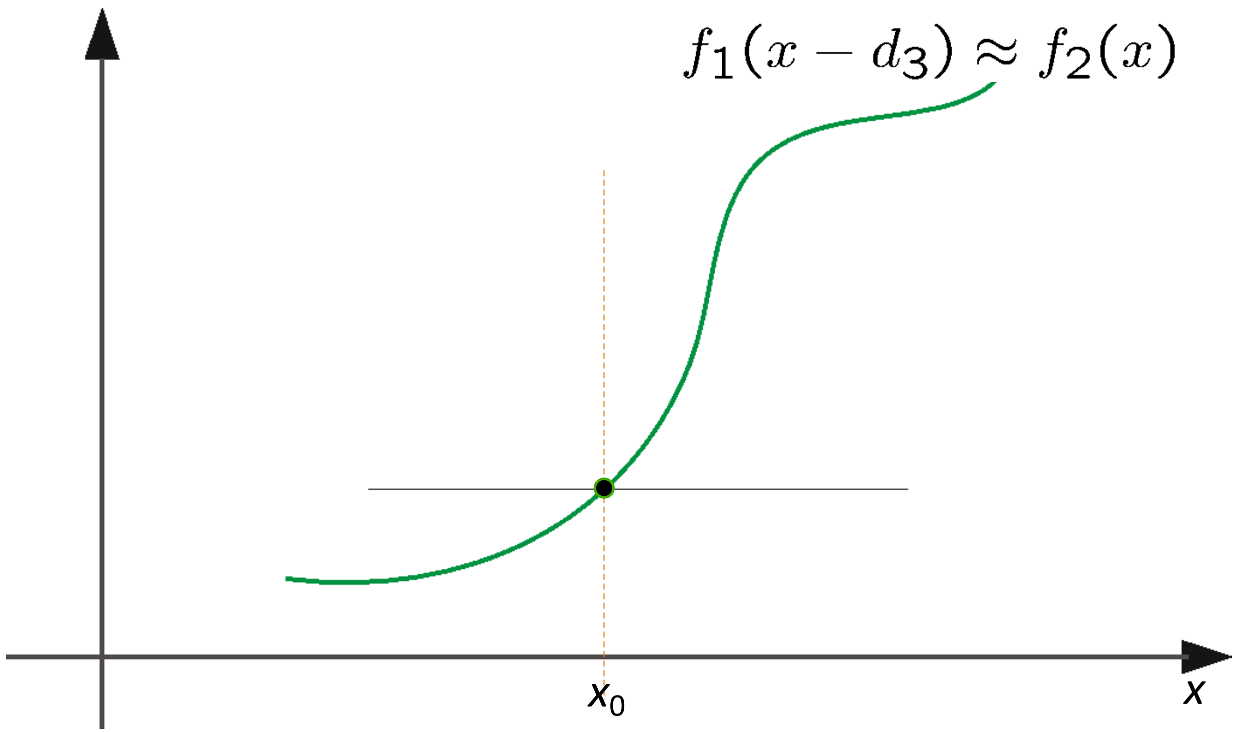


Initial guess: d_2

Estimate: $d_3 = d_2 + \hat{d}$

Adapted from slides from Richard Szeliski

Step 4: Iterative Refinement



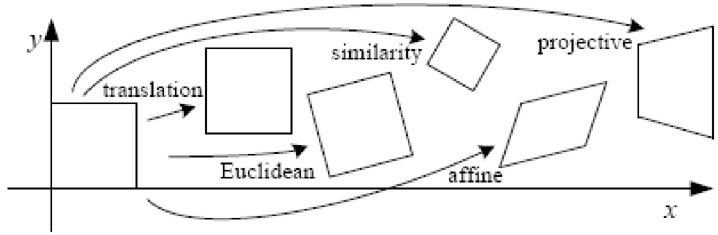
Adapted from slides from Richard Szeliski

Motion Models

- 2D Models:
  - Affine
  - Quadratic
  - Planar projective transform (Homography)
- 3D Models:
  - Instantaneous camera motion models
  - Homography+epipole
  - Plane+Parallax

Adapted from slides from Richard Szeliski

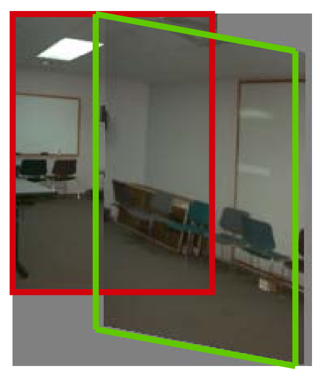
Motion Models



Translation

2 unknowns

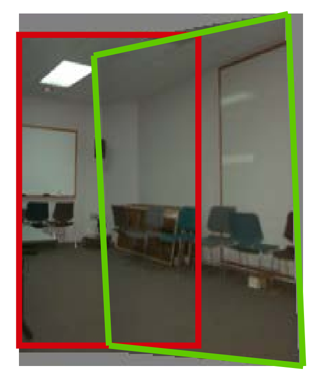
Affine



6 unknowns:

$$x' = Ax + d$$

3D rotation

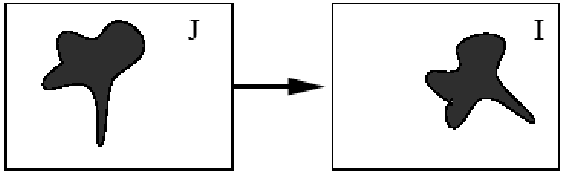


3 unknowns

Adapted from slides from Richard Szeliski

Compute Affine Motion

- Intensity constancy constraint: $J(Ax + d) = I(x)$



- Define Sum of Squared Difference, SSD, error as:

$$\epsilon = \int_W [J(Ax + d) - I(x)]^2 w(x) dx \tag{1}$$

Let $A = I + D$, min. ϵ with respect to $D \in R^{2\times 2}$, and $d \in R^{2\times 1}$.

- Three steps for solving this problem:
  - Set $\frac{\partial \epsilon}{\partial D}$, $\frac{\partial \epsilon}{\partial d}$ to 0;
  - Taylor expansion on J(Ax+d) respect to x;
  - Solve for A (D) and d, iterate with Newton Raphson.

Adapted from slides from Kostas Daniilidis

$$\epsilon = \int_W [J(Ax+d) - I(x)]^2 w(x) dx$$

- Differentiate ϵ with respect to D and d,

$$\frac{1}{2}\frac{\partial \epsilon}{\partial D} = \int_W [J(Ax+d) - I(x)]\; g\; x^T\; w\; dx = 0 \tag{2}$$

$$\frac{1}{2}\frac{\partial \epsilon}{\partial d} = \int_W [J(Ax+d) - I(x)]\; g\; w\; dx = 0 \tag{3}$$

where $g = (\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y})^T$.

- Assume small motion, $Ax + d = x + (Dx + d) = x + u$,

  – Taylor expansion of J(Ax+d) is

$$J(Ax+d) = J(x) + g^T\; u \tag{4}$$

Adapted from slides from Kostas Daniilidis

$$\epsilon = \int_W [J(Ax+d) - I(x)]^2 w(x) dx$$

- From previous slide, we have:

$$\int_W \; [J(Ax+d) - I(x)]\; g\; x^T\; w\; dx = 0$$
$$\int_W \; [J(Ax+d) - I(x)]\; g\; w\; dx = 0$$
$$J\;\; (Ax+d) = J(x) + g^T\; u$$

where $g = (\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y})^T$.

- Combining them, we have:

$$\int_W g\; x^T\; (g^T u) w\; dx \;=\; \int_W [I(x) - J(x)] g\; x^T\; dx \tag{5}$$
$$\int_W g\; (g^T u) w\; dx \;=\; \int_W [I(x) - J(x)] g\; w\; dx \tag{6}$$

- Can rewrite (5) and (6) as a linear system of 6 equations and 6 unknowns.

Adapted from slides from Kostas Daniilidis

$$\epsilon = \int_W [J(Ax+d) - I(x)]^2 w(x) dx$$

- Tz = a:

$$T = \int_W \begin{bmatrix} g_x^2 x^2 & g_x g_y xy & g_x^2 xy & g_x g_y x^2 & g_x^2 x & g_x g_y x \\ g_x g_y xy & g_y^2 y^2 & g_x g_y y^2 & g_y^2 xy & g_x g_y y & g_y^2 y \\ g_x^2 xy & g_x g_y y^2 & g_x^2 y^2 & g_x g_y xy & g_x^2 y & g_x g_y y \\ g_x g_y x^2 & g_y^2 xy & g_x g_y xy & g_y^2 x^2 & g_x g_y x & g_y^2 x \\ g_x^2 x & g_x g_y y & g_x^2 y & g_x g_y x & g_x^2 & g_x g_y \\ g_x g_y x & g_y^2 y & g_x g_y y & g_y^2 x & g_x g_y & g_y^2 \end{bmatrix} w \, dx \qquad (7)$$

and

$$z = [D(1,1), D(2,2), D(1,2), D(2,1), d(1), d(2)]^T \qquad (8)$$

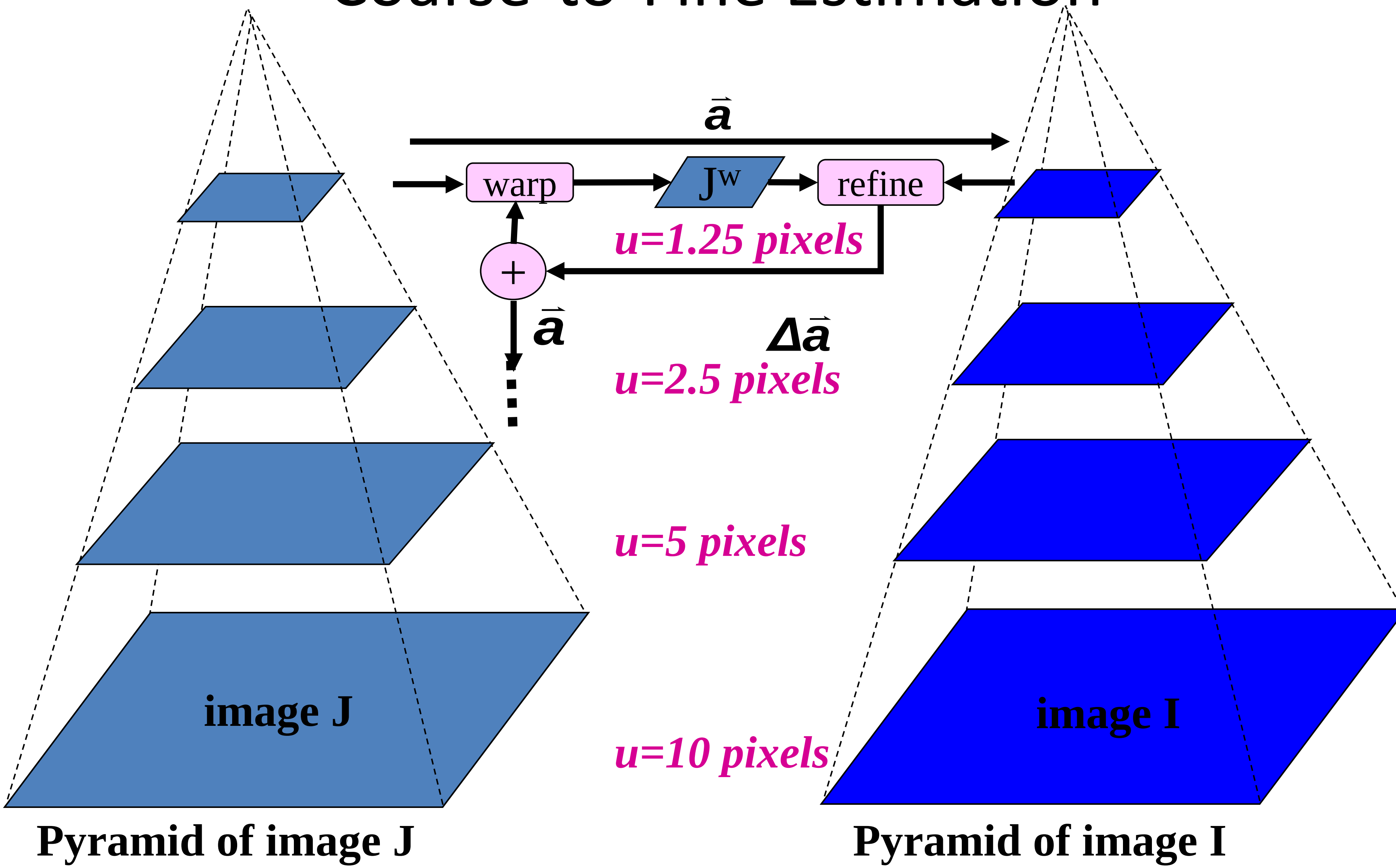
$$a = \int_W (I(x) - J(x)) \begin{bmatrix} g_x x \\ g_y y \\ g_x y \\ g_y x \\ g_x \\ g_y \end{bmatrix} dx \qquad (9)$$

Adapted from slides from Kostas Daniilidis

Limits of the KLT Tracker

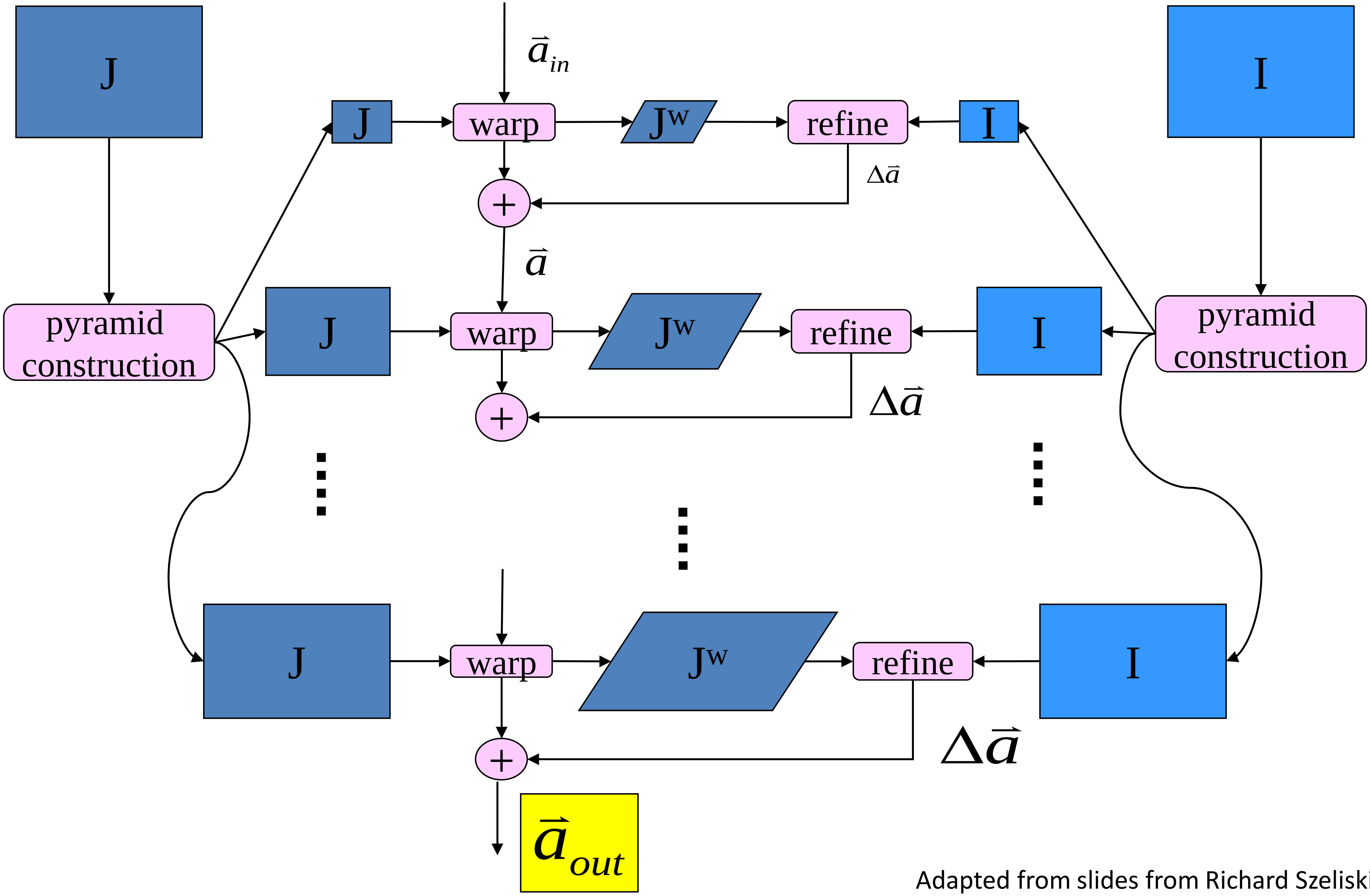
- Fails when intensity structure in window is poor
- Fails when the displacement is large (typical operating range is motion of 1 pixel
  - Linearization of brightness is suitable only for small displacement
- Brightness is not strictly constant in images
  - Actually less problematic than it appears, since we can filter images to make them look similar

Coarse-to-Fine Estimation



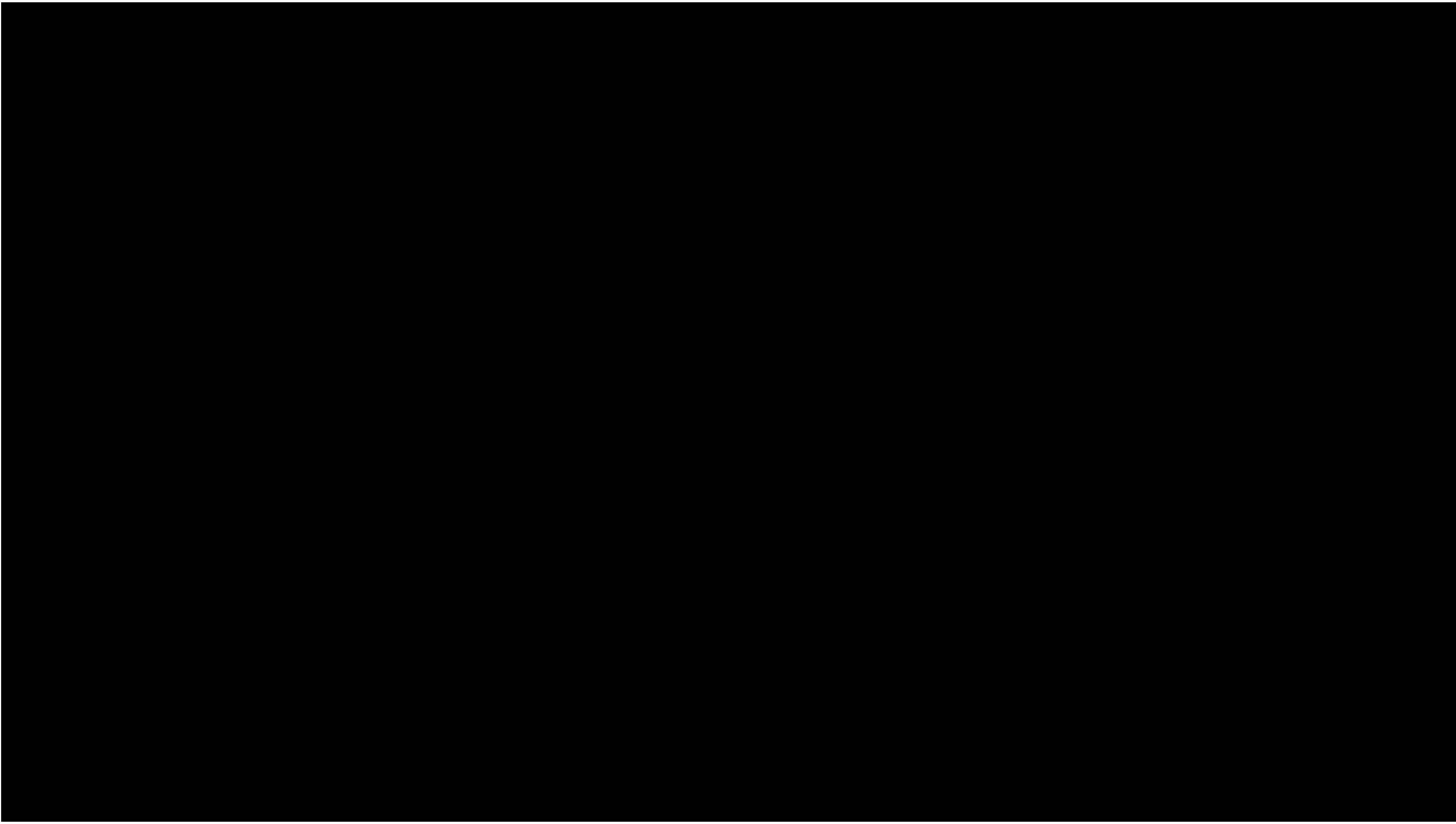
Adapted from slides from Richard Szeliski

Coarse-to-Fine Estimation



Adapted from slides from Richard Szeliski

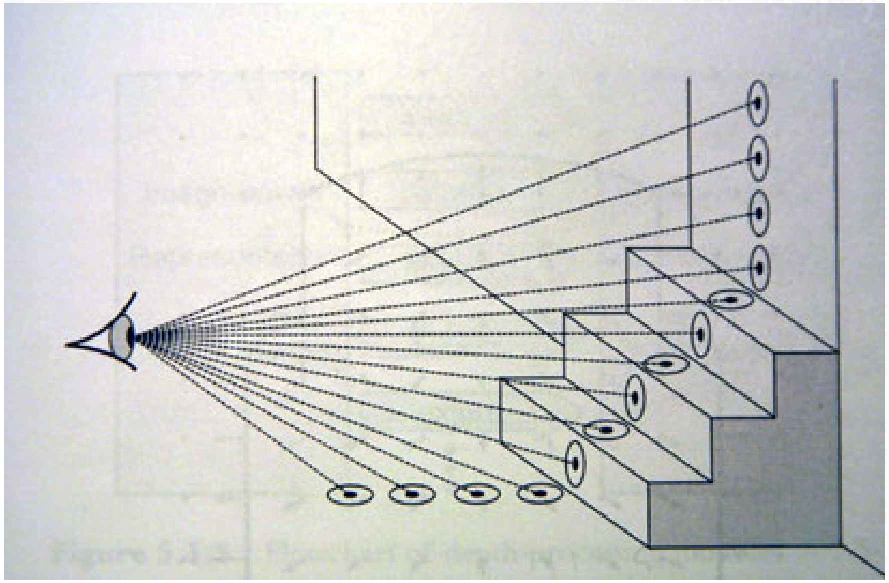
Optical Flow-based Velocity Estimator

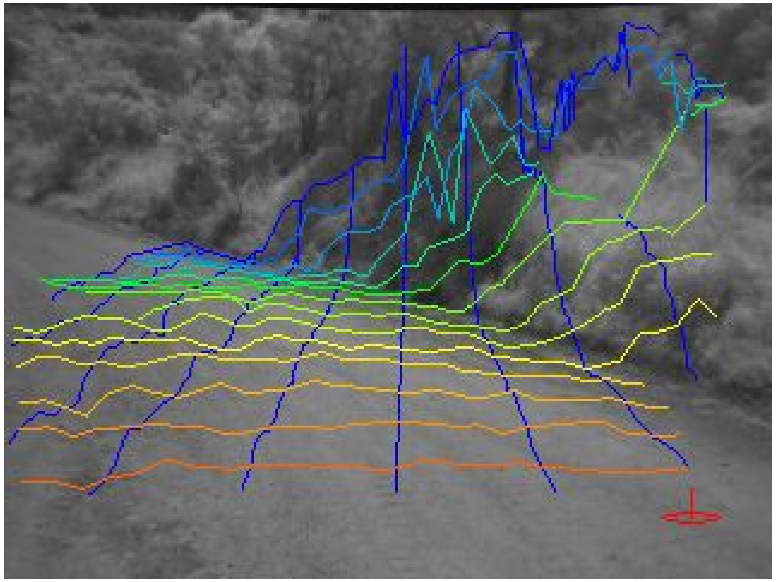


Dense Stereo

3D Shape perception

1) **Depth**: *the distance of the surface from the observer*
2) **Surface orientation**: *the slant and tilt of the surface with respect to observers' sight*

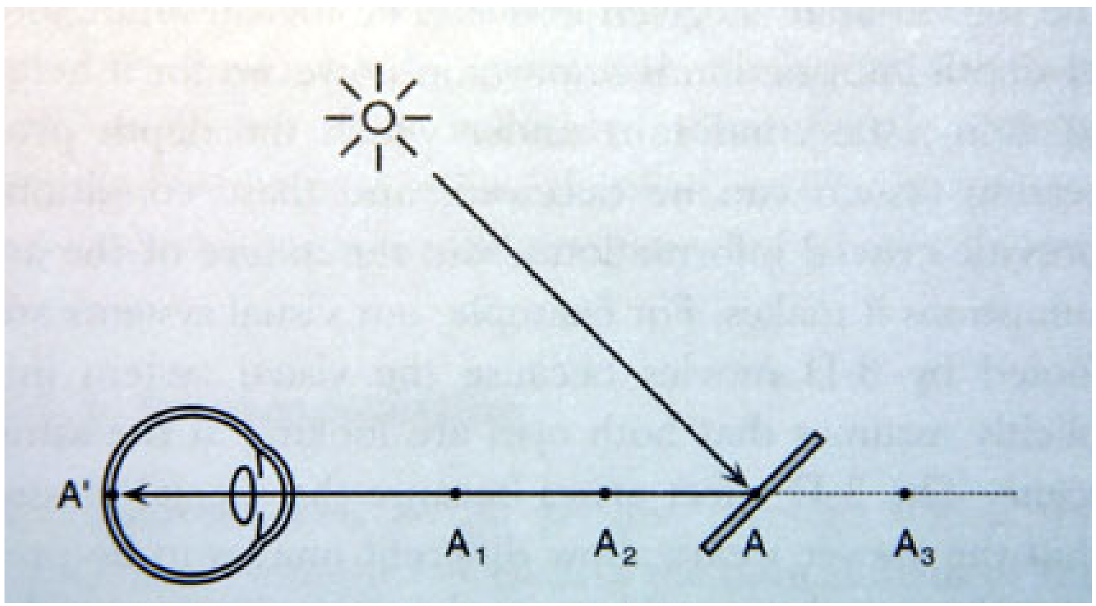




Adapted from slides from Kostas Daniilidis

Depth ambiguity

Inverse problem: multiple solution exists

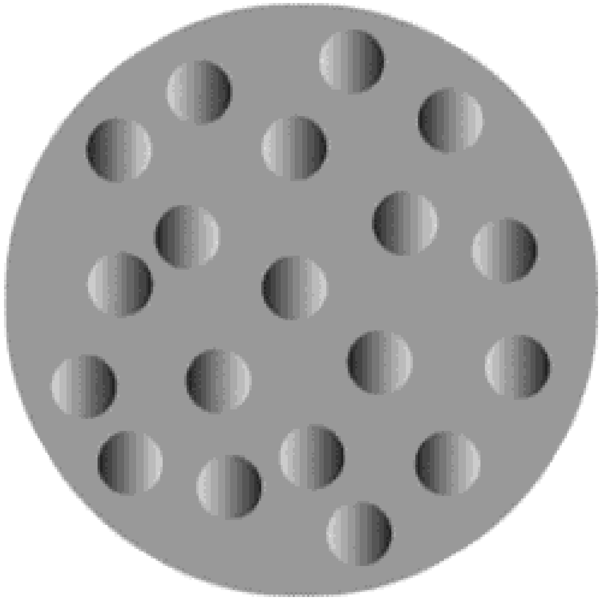


Adapted from slides from Kostas Daniilidis

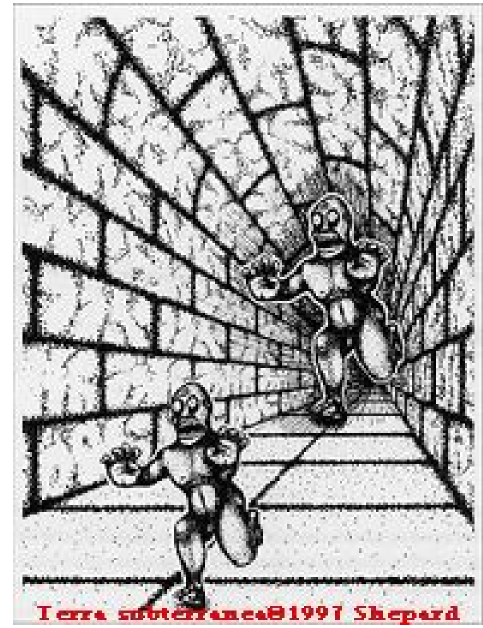
Pictorial cues for 3D shape

Perspective projection gives us the relative position to horizon, therefore we can deduce its physical size.

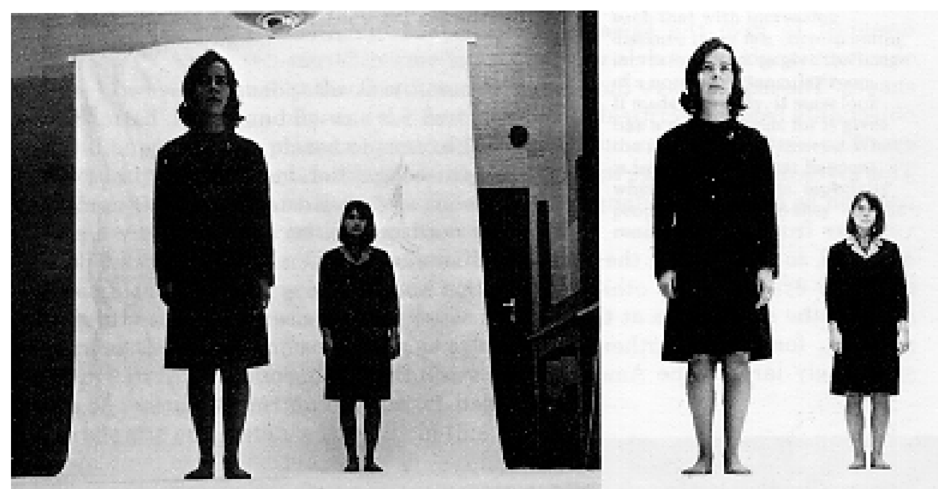
Shading also reveal shape using illumination model



Adapted from slides from Kostas Daniilidis

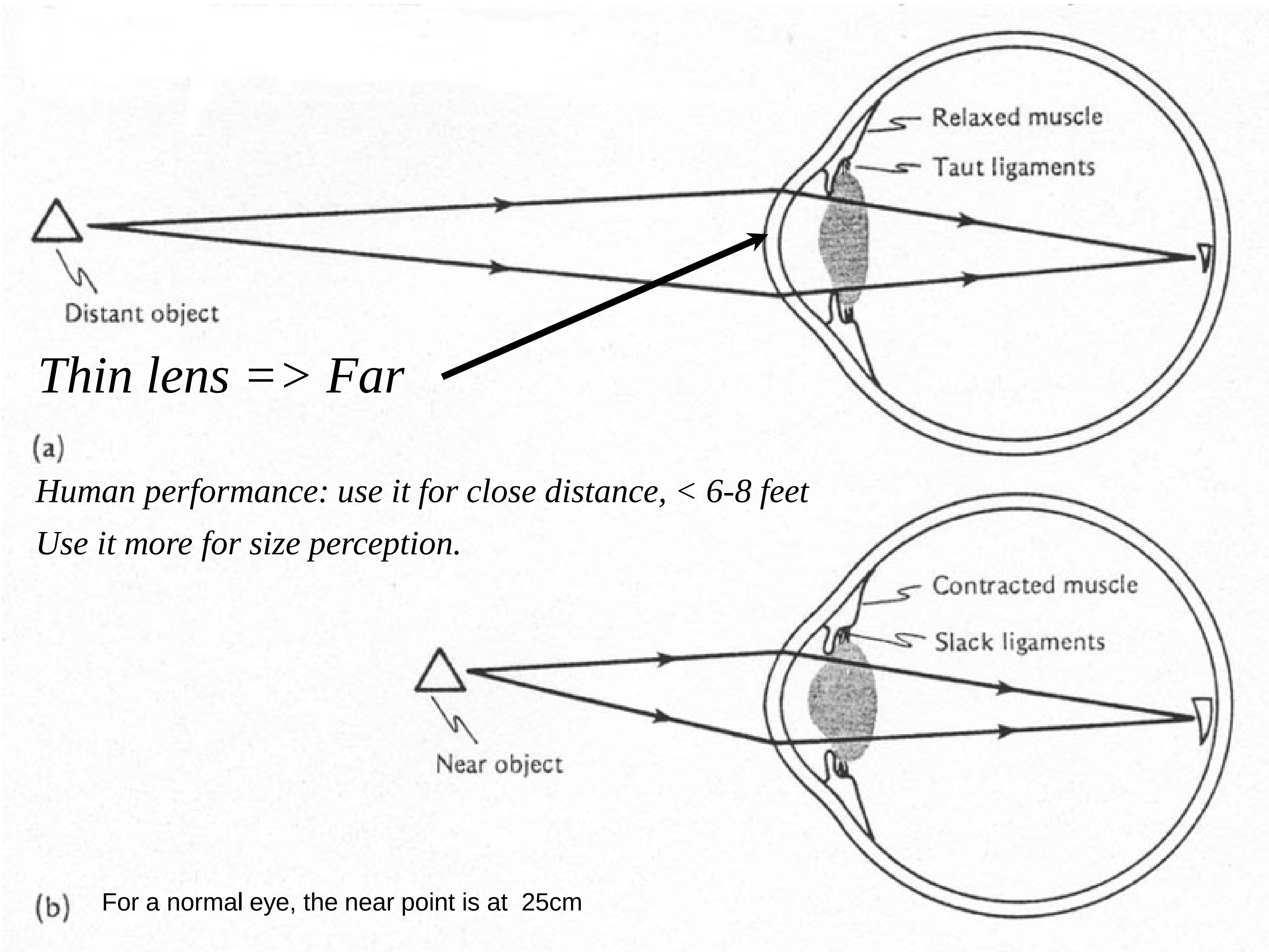


How?



Adapted from slides from Kostas Daniilidis

Shape from Focus, Accommodation



Human performance: use it for close distance, < 6-8 feet

Use it more for size perception.

For a normal eye, the near point is at 25cm

Adapted from slides from Kostas Daniilidis

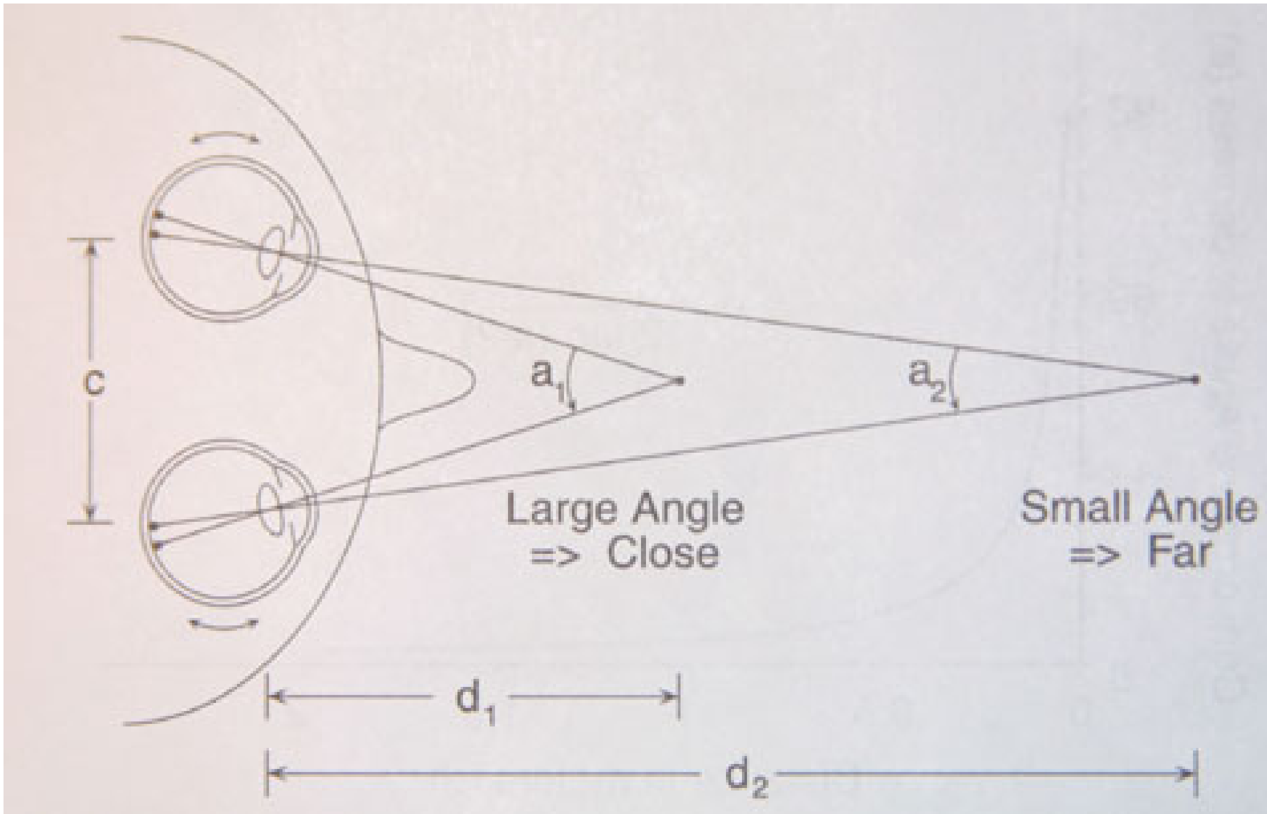
1) Because of its restricted range (6-8 feet), accommodation is rarely a crucial source of depth in humans.

2) In the chameleon, it is of paramount importance, for it controls this organism’s ability to feed itself. A chameleon catches its prey by slicking its sticky tongue out just the right distance to catch an insect.

3) When chameleons were outfitted with prisms and spectacles that manipulated the accommodation and convergence of their eyes, the distance they flicked their tongues was changed.

Adapted from slides from Kostas Daniilidis

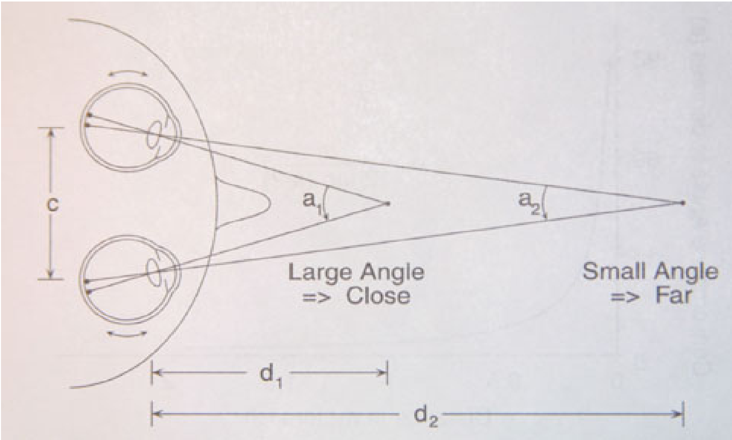
Convergence



Adapted from slides from Kostas Daniilidis

Convergence

- The eyes fixate a given point in external space when both of them, are aimed directly at the point so that light coming from it falls on the centers of both foveae simultaneously.
- The crucial fact about the convergence that provides information about fixation depth is that the angle formed by the two lines of sight varies systematically with distance between the observer and the fixated point.

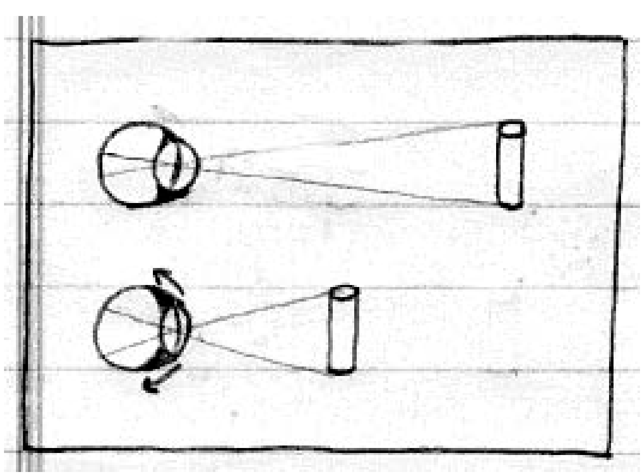


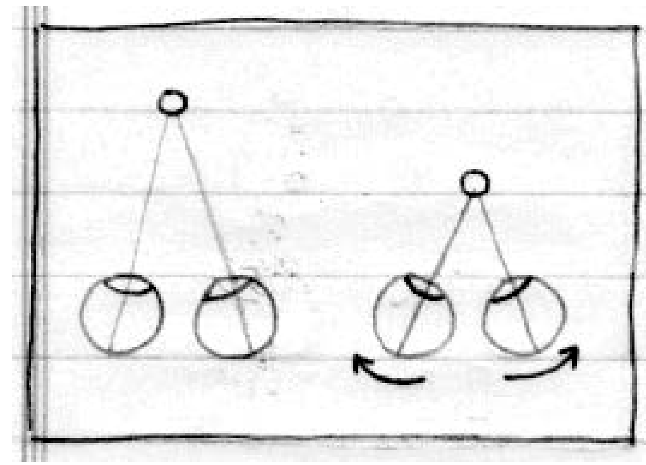
$$\boxed{d = \frac{c}{2tan(a/2)}}$$

Adapted from slides from Kostas Daniilidis

Accommodation and Convergence

•Accommodation and convergence normally change in lock steps. For human, they are important sources of depth information at close distance.

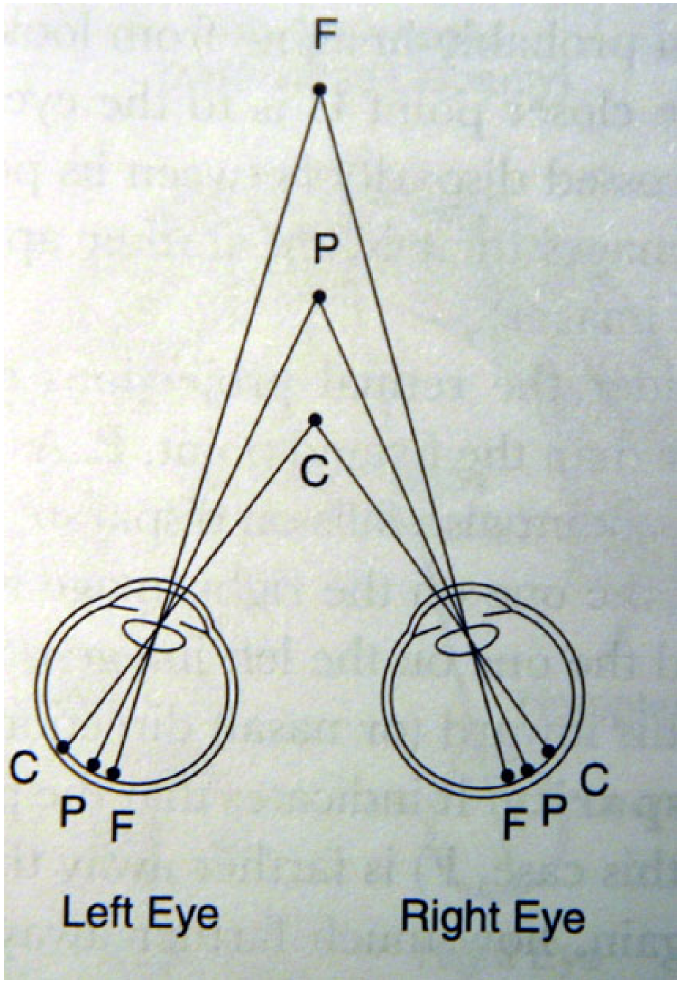




Human performance: upto a few meters

Adapted from slides from Kostas Daniilidis

Stereoscopic : binocular disparity



Sign and magnitude of disparity

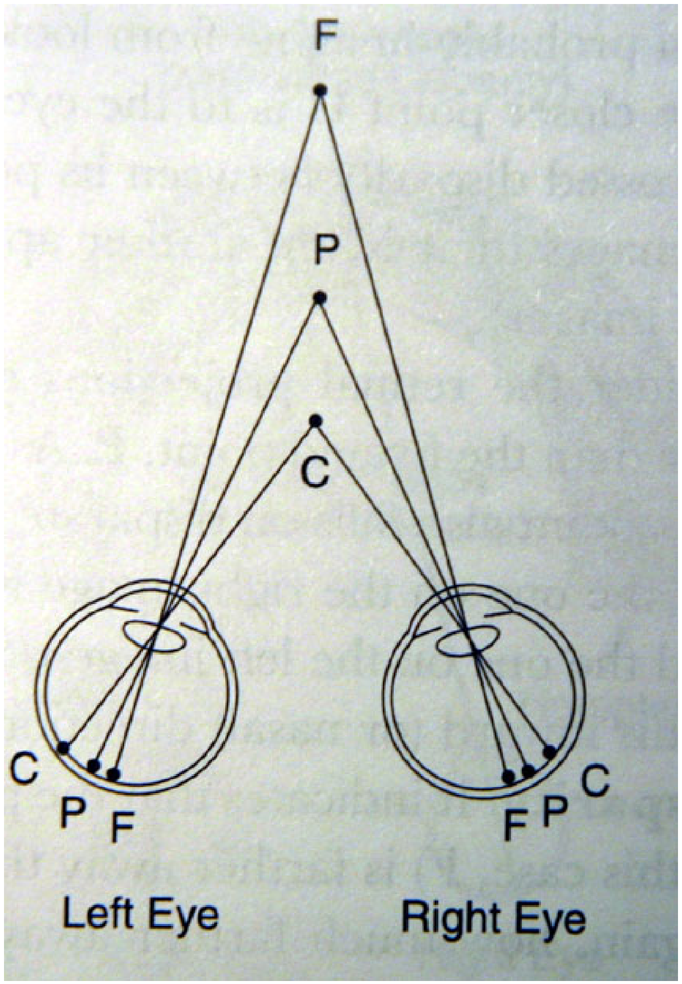
P: converging point

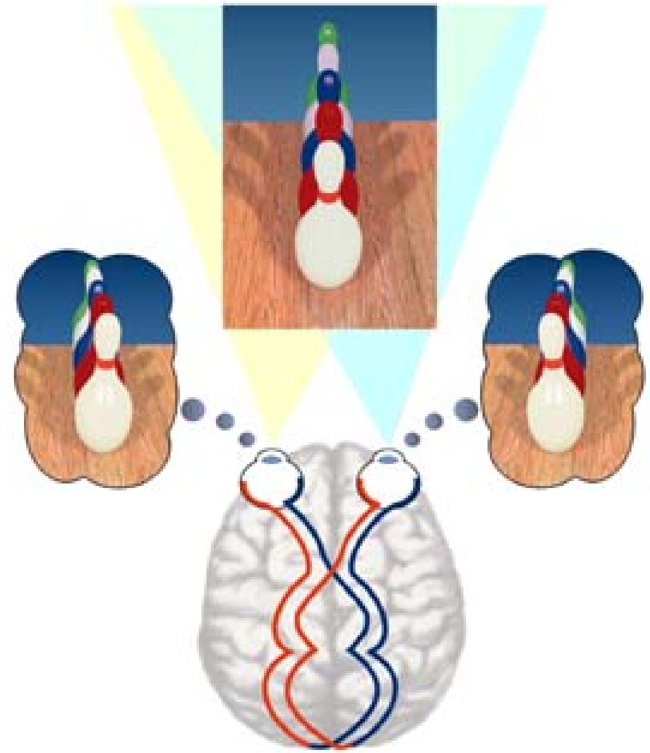
C: object nearer projects to the outside of the P, disparity = +

F: object farther projects to the inside of the P, disparity = -

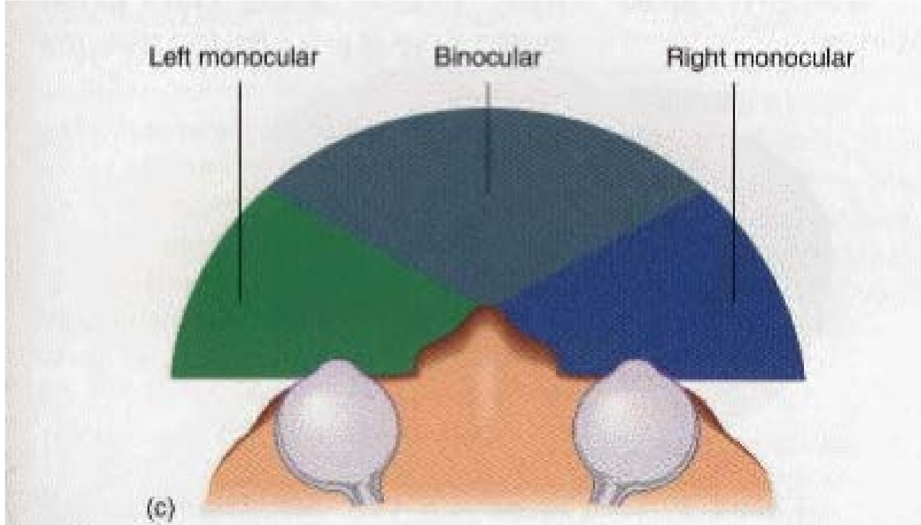
Adapted from slides from Kostas Daniilidis

Stereo Vision



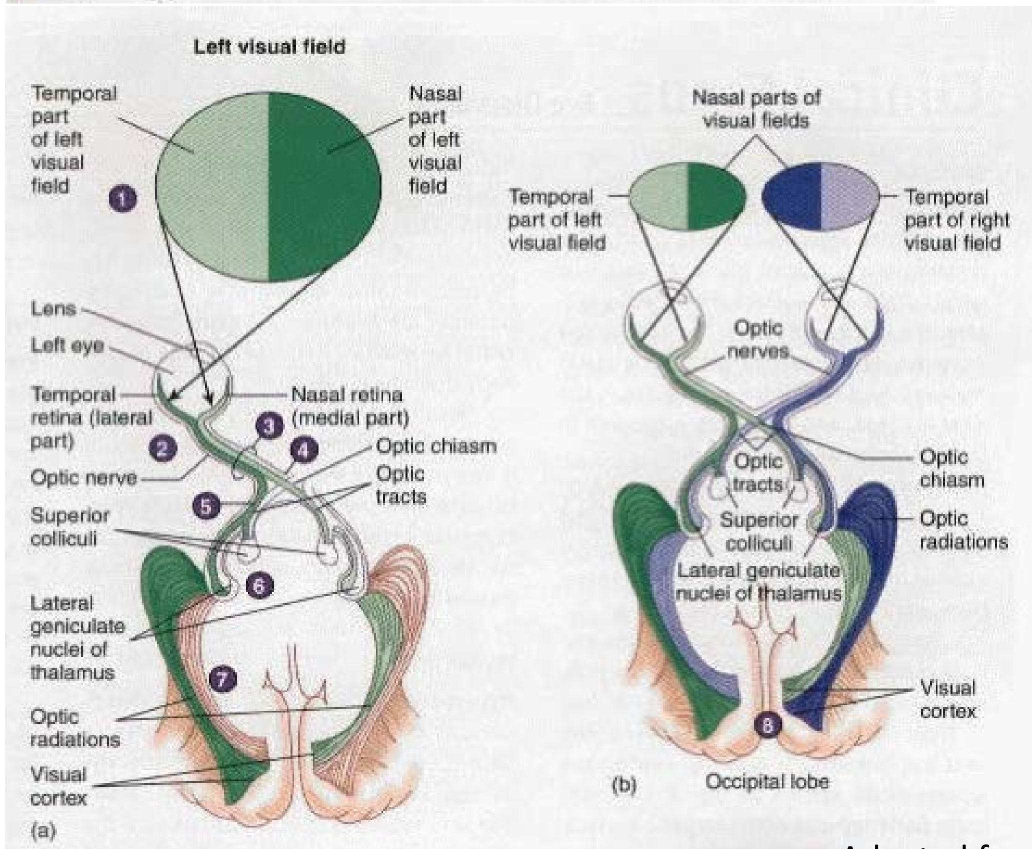


Adapted from slides from Kostas Daniilidis



Our visual angle is 104d, and it is facing forward.

What happened to rabbit’s vision?



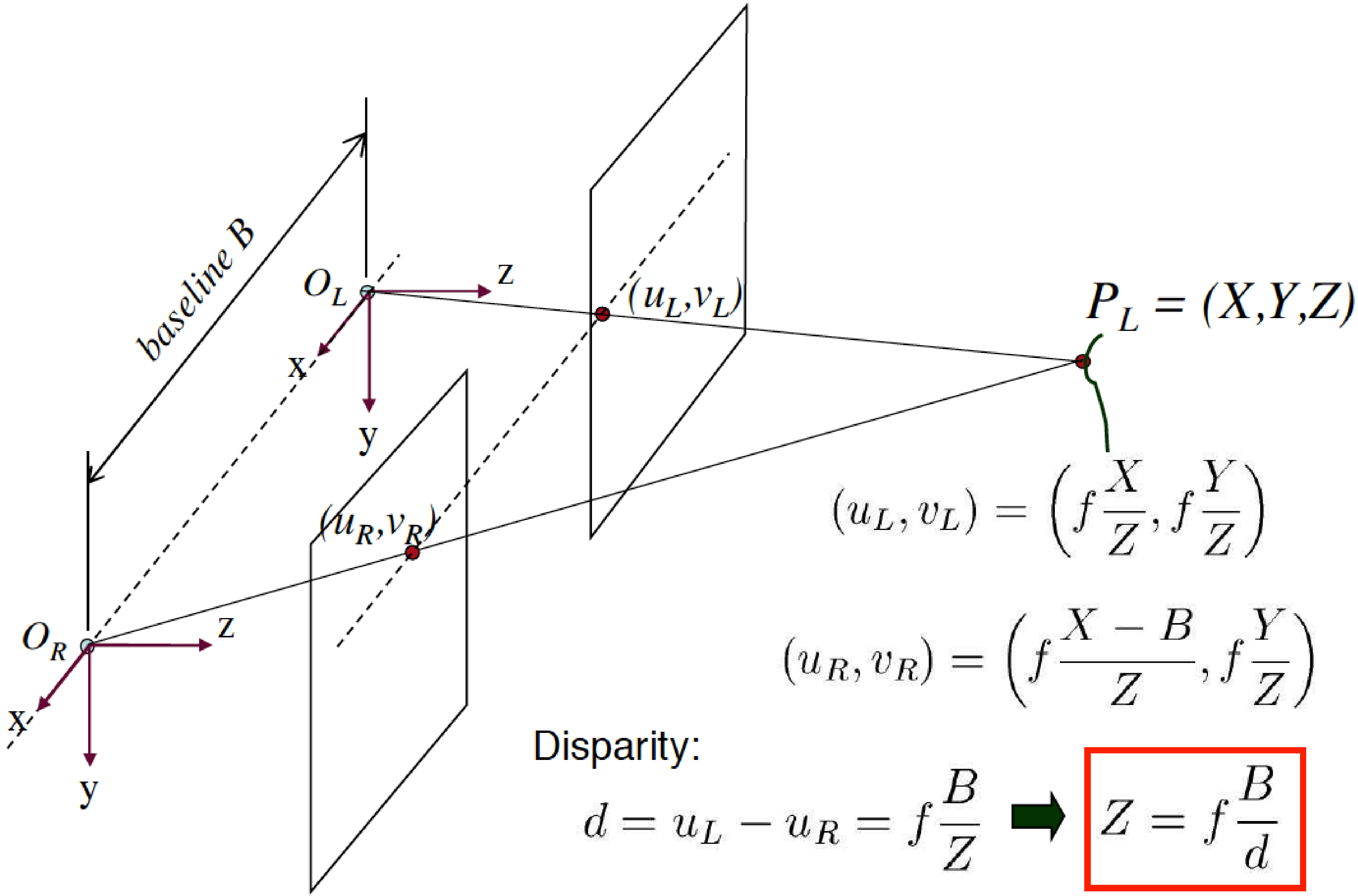
Adapted from slides from Kostas Daniilidis

Computational Stereo Vision Outline:

- Basic geometrical setup of the stereo vision
- Computational model of stereo vision, Correspondence problem.

Adapted from slides from Kostas Daniilidis

Basic Stereo Derivations



$$(u_L, v_L) = \left(f\frac{X}{Z}, f\frac{Y}{Z}\right)$$

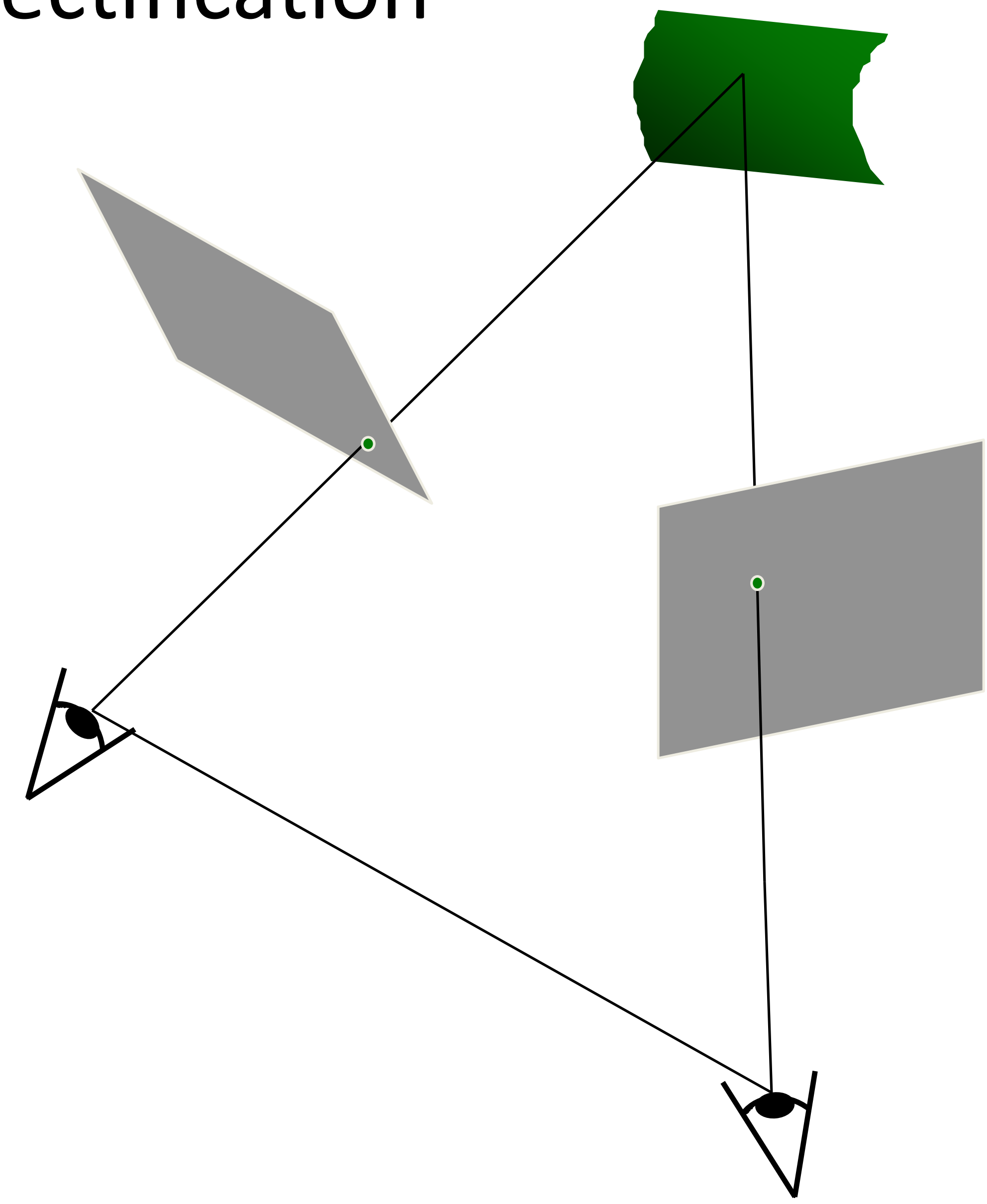
$$(u_R, v_R) = \left(f\frac{X - B}{Z}, f\frac{Y}{Z}\right)$$

Disparity:

$$d = u_L - u_R = f\frac{B}{Z} \Rightarrow \boxed{Z = f\frac{B}{d}}$$

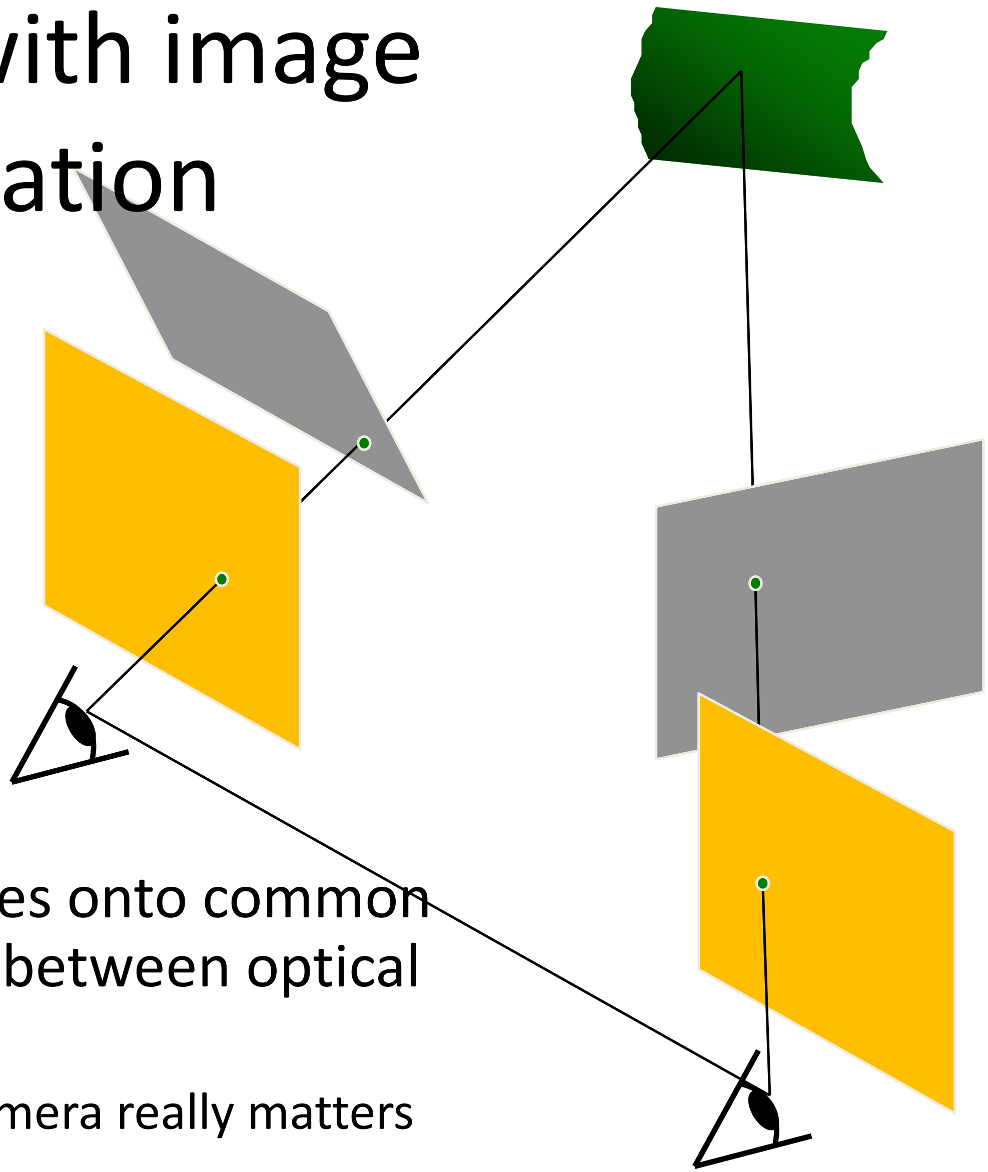
Adapted from slides from Kostas Daniilidis

Stereo Image Rectification



Adapted from slides from Kostas Daniilidis

We can always achieve this geometry with image rectification

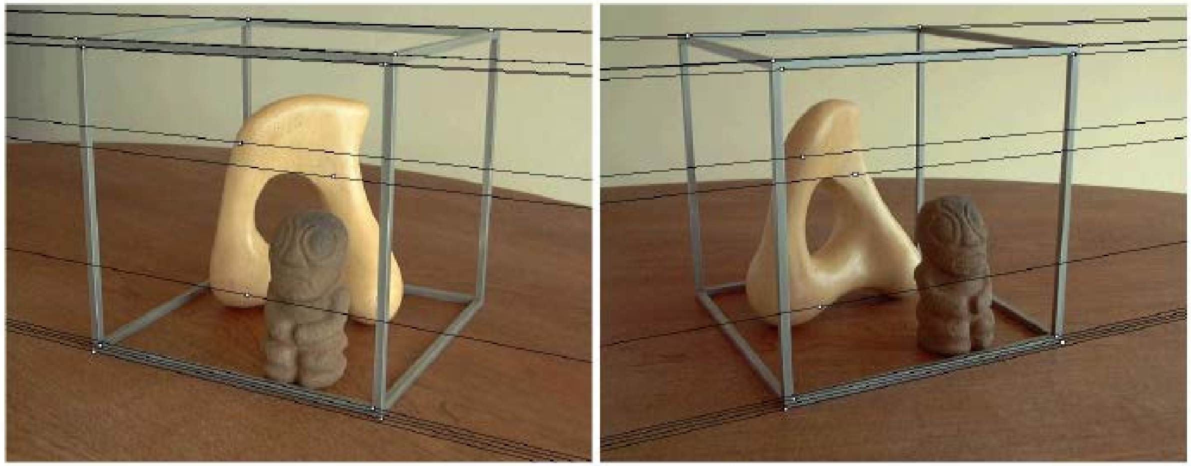


- Image Reprojection
  - reproject image planes onto common plane parallel to line between optical centers
- Notice, only focal point of camera really matters

(Seitz)

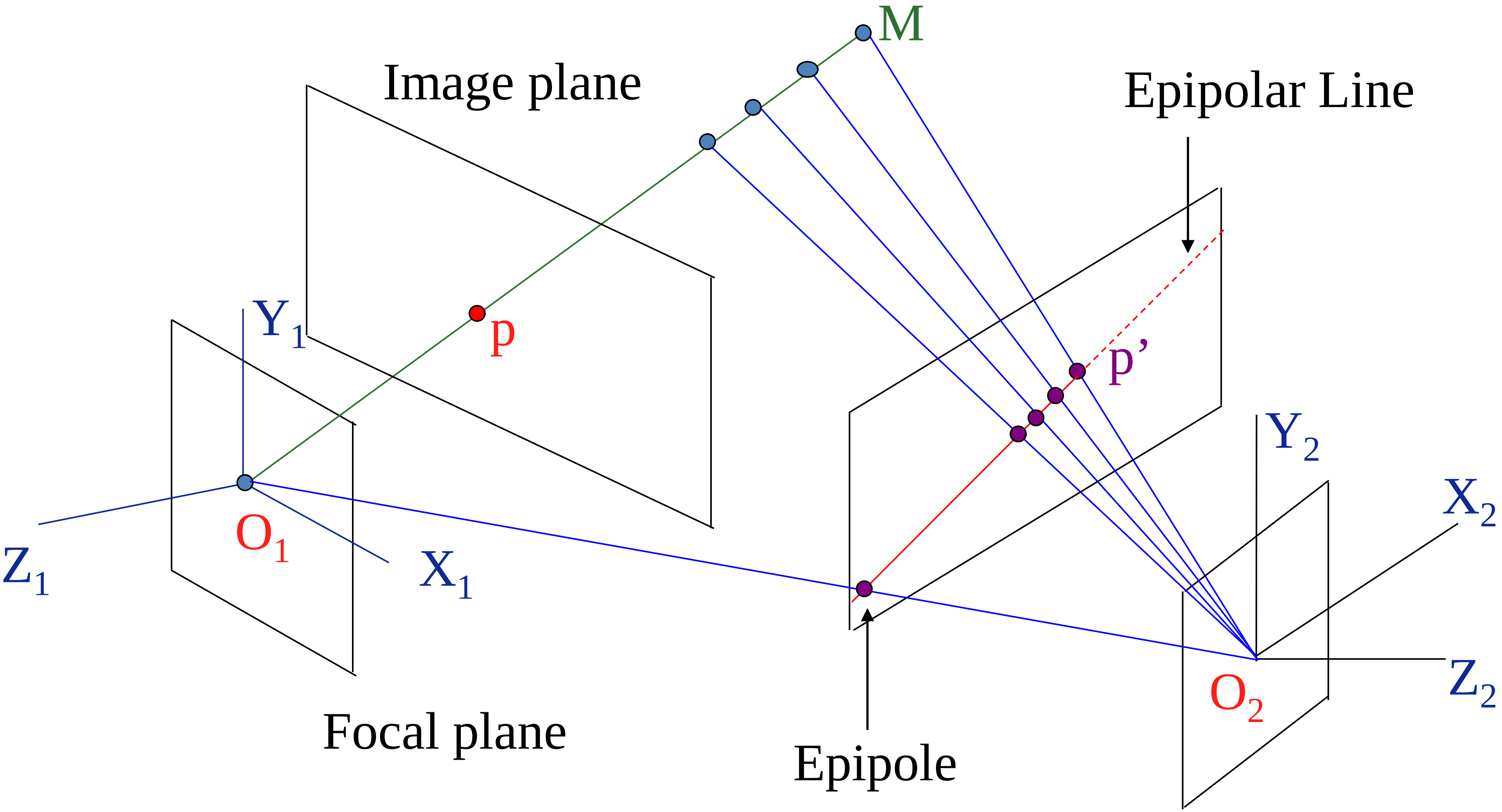
Adapted from slides from Kostas Daniilidis

Stereo Rectification



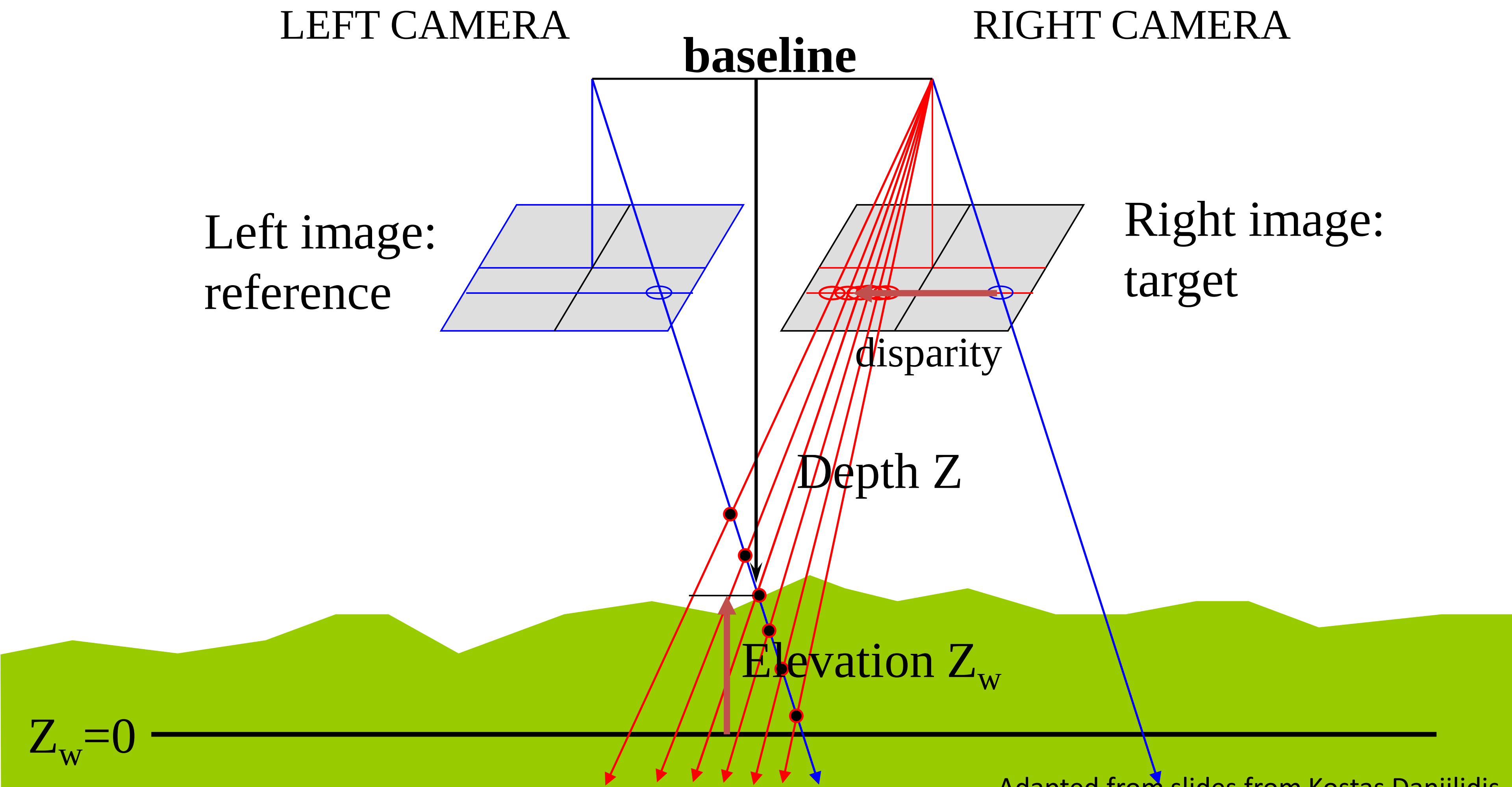
Adapted from slides from Kostas Daniilidis

Stereo Constraints



Adapted from slides from Kostas Daniilidis

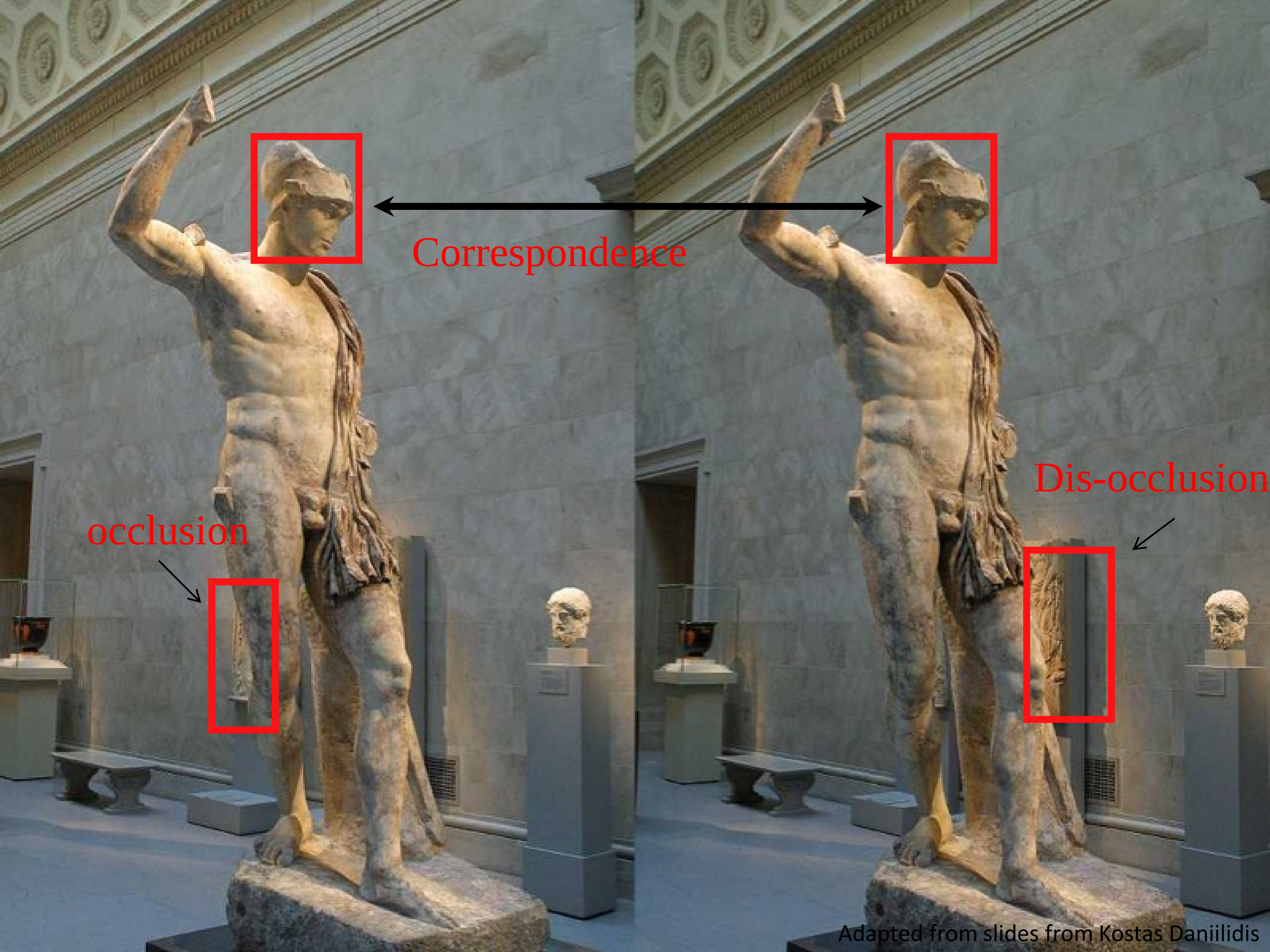
A Simple Stereo System

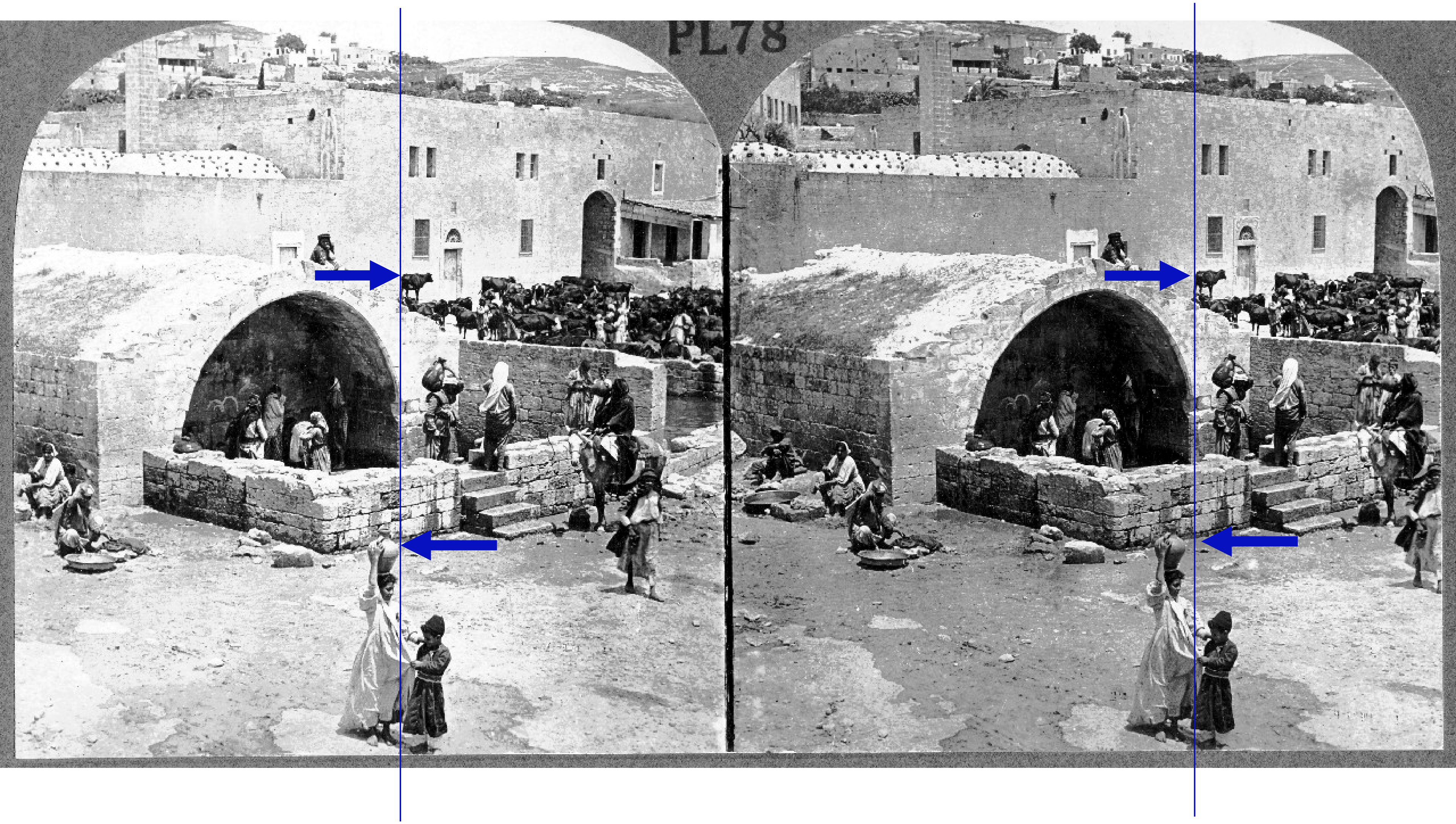


Adapted from slides from Kostas Daniilidis

Adapted from slides from Kostas Daniilidis

Adapted from slides from Kostas Daniilidis

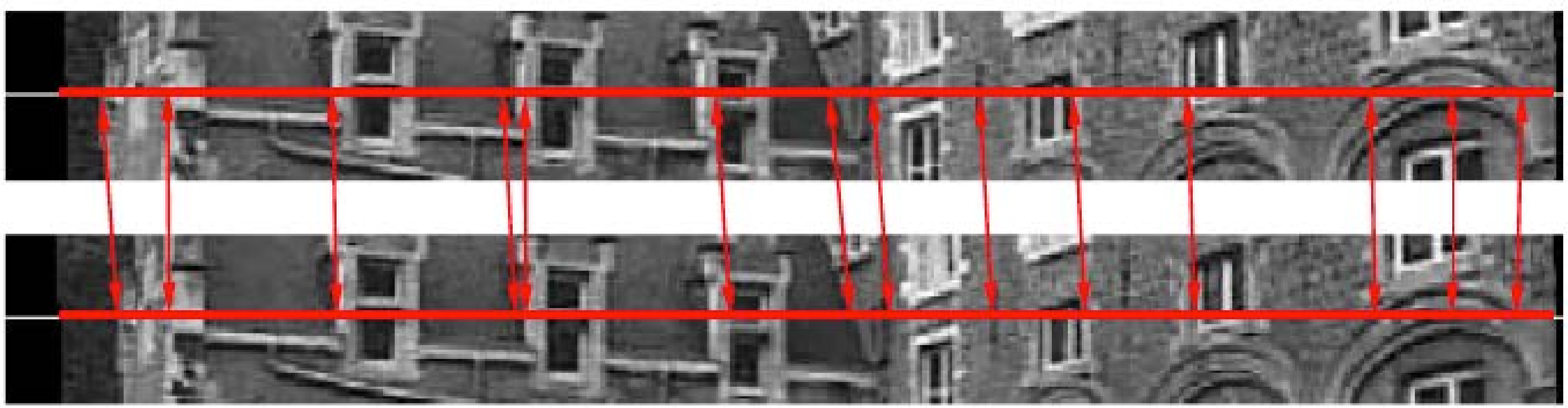

Correspondence
occlusion
Dis-occlusion
Adapted from slides from Kostas Daniilidis



Notice the disparity difference

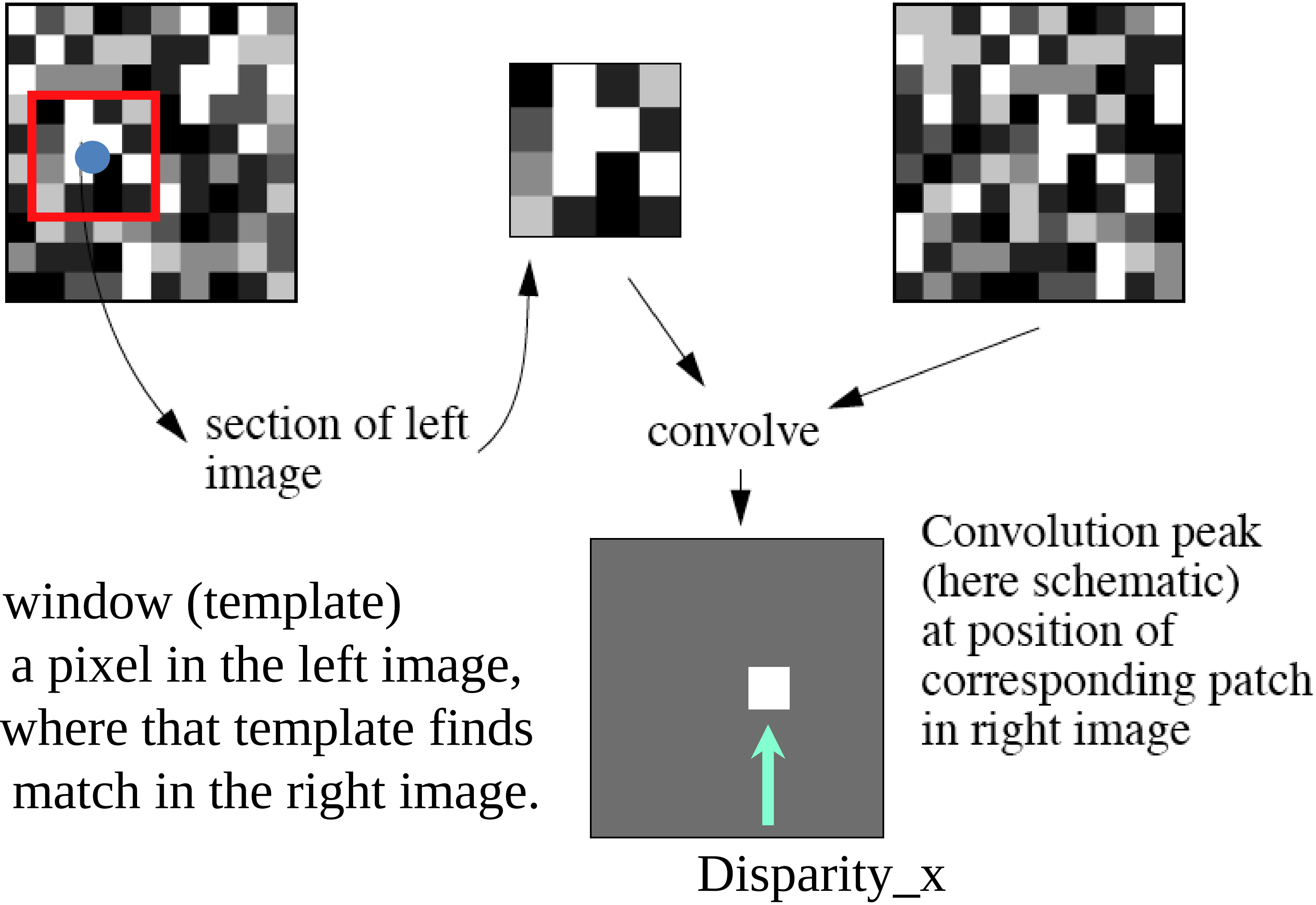
Adapted from slides from Kostas Daniilidis

Correspondence



Adapted from slides from Kostas Daniilidis

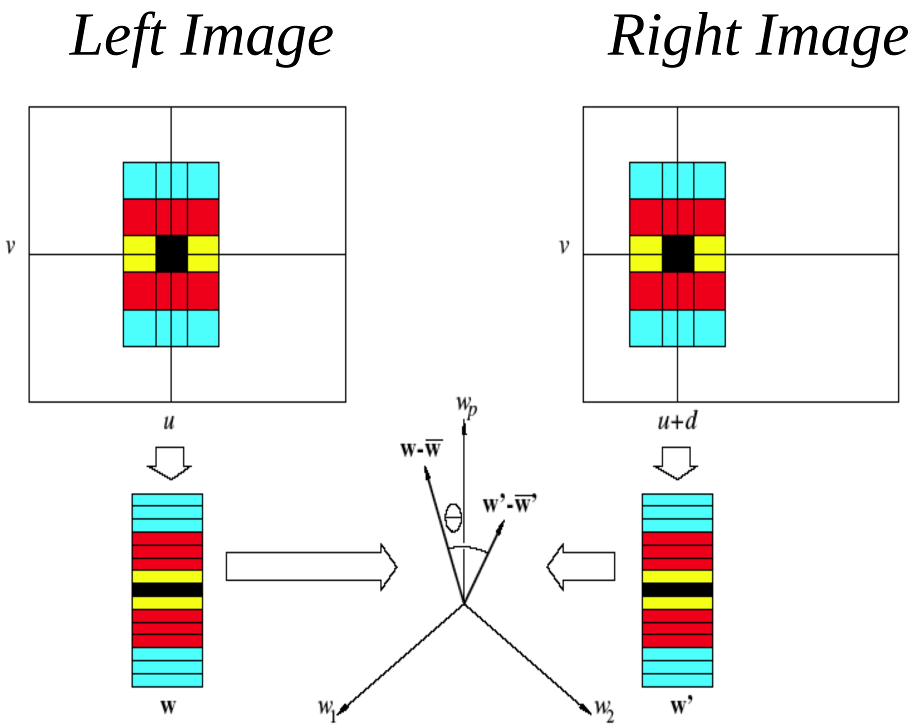
Computing Correspondence



Take a window (template) around a pixel in the left image, search where that template finds its best match in the right image.

Adapted from slides from Kostas Daniilidis

Correspondence using convolution as image patch similarity function



Convolution distance:

$$d(f,g) = \sum_{i,j} f(i,j)g(i,j)$$

Adapted from slides from Kostas Daniilidis

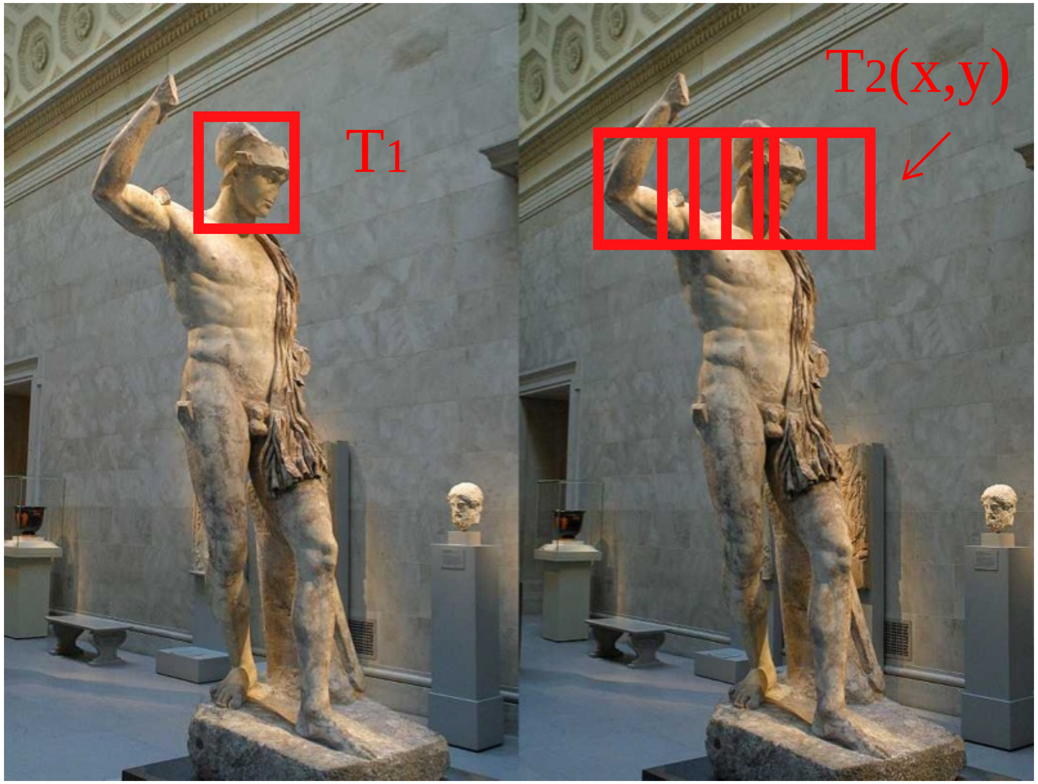
Choice of similarity function for image patches

Sum of squared differences

$$SSD(f,g) = \sum_{i,j} (f(i,j) - g(i,j))^2$$

We want similarity function to be resistant to image noise, illumination changes.

Adapted from slides from Kostas Daniilidis



To find the corresponding image patch of T_1, in the second image, we need to do a search:

For each location (x,y) in the second image J,

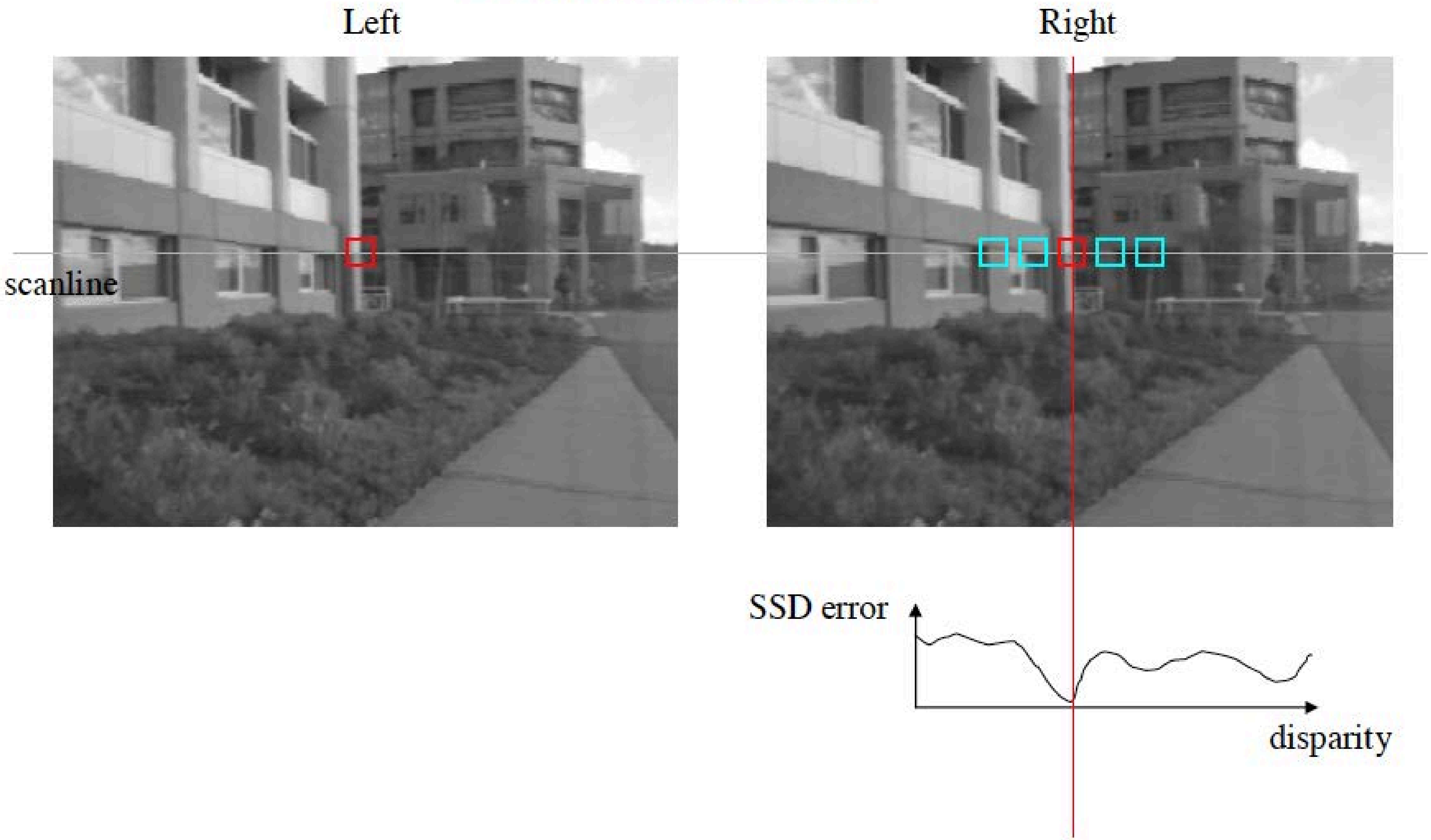
Err(x,y) = SSD(T_1, T_2(x,y));

End

Disparity of T_1 = arg.min. Err(x,y)

Adapted from slides from Kostas Daniilidis

Correspondence Using Correlation

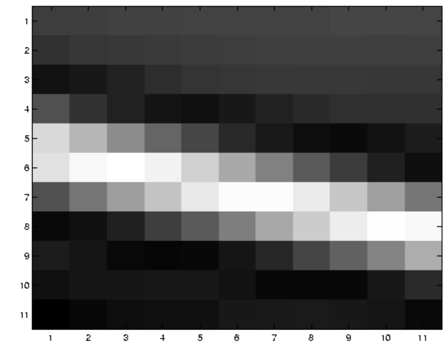


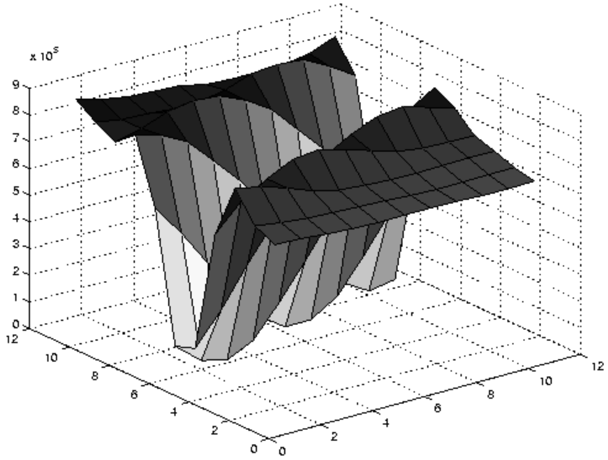
Adapted from slides from Kostas Daniilidis

Edge

Sum of squared differences

Err(x,y)



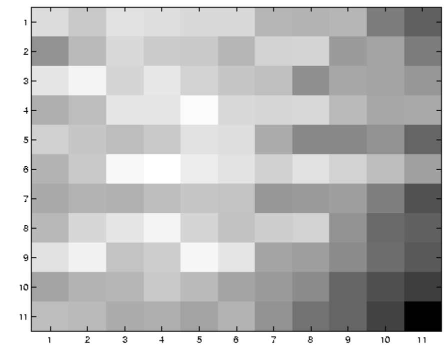


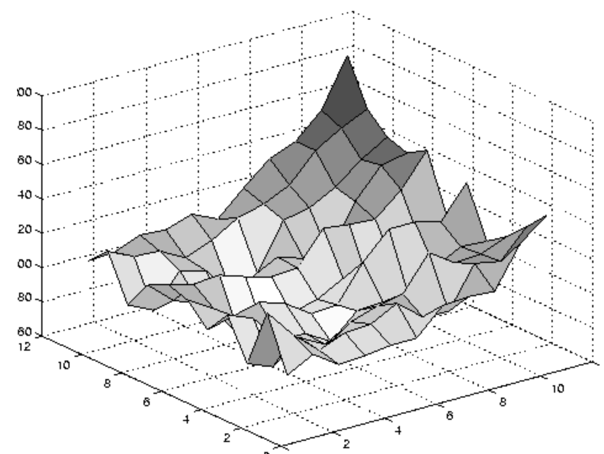
Adapted from slides from Kostas Daniilidis

Low texture region

Sum of squared differences

Err(x,y)

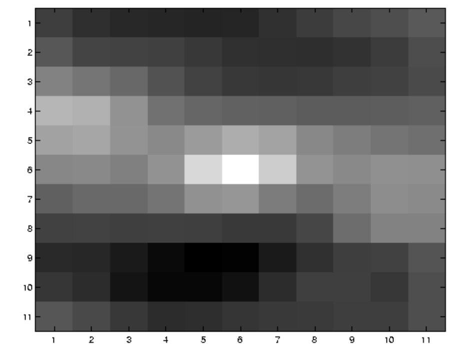


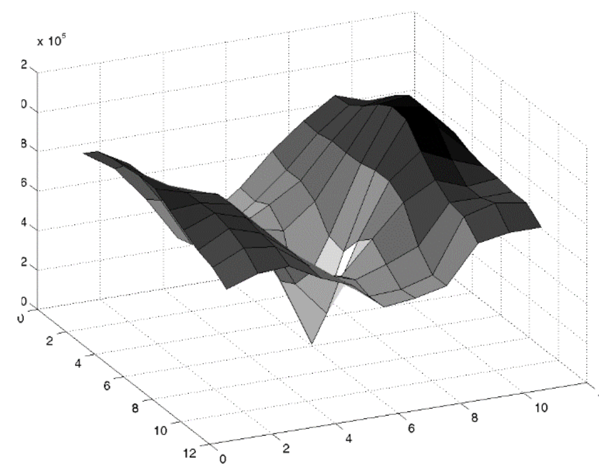


Adapted from slides from Kostas Daniilidis

High textured region *Sum of squared differences*

Err(x,y)





Adapted from slides from Kostas Daniilidis

MATCH METRIC	DEFINITION
Normalized Cross-Correlation (NCC)	$$\frac{\sum_{u,v}\left(I_1(u,v)-\bar{I}_1\right)\cdot\left(I_2(u+d,v)-\bar{I}_2\right)}{\sqrt{\sum_{u,v}\left(I_1(u,v)-\bar{I}_1\right)^2\cdot\sum_{u,v}\left(I_2(u+d,v)-\bar{I}_2\right)^2}}$$
Sum of Squared Differences (SSD)	$$\sum_{u,v}\left(I_1(u,v)-I_2(u+d,v)\right)^2$$
Normalized SSD	$$\sum_{u,v}\left(\frac{\left(I_1(u,v)-\bar{I}_1\right)}{\sqrt{\sum_{u,v}\left(I_1(u,v)-\bar{I}_1\right)^2}}-\frac{\left(I_2(u+d,v)-\bar{I}_2\right)}{\sqrt{\sum_{u,v}\left(I_2(u+d,v)-\bar{I}_2\right)^2}}\right)^2$$
Sum of Absolute Differences (SAD)	$$\sum_{u,v}\left\|I_1(u,v)-I_2(u+d,v)\right\|$$
Zero Mean SAD	$$\sum_{u,v}\left\|(I_1(u,v)-\bar{I}_1)-(I_2(u+d,v)-\bar{I}_2)\right\|$$
Rank	$$I_k'(u,v)=\sum_{m,n}I_k(m,n)<I_k(u,v)$$ $$\sum_{u,v}\left(I_1'(u,v)-I_2'(u+d,v)\right)$$
Census	$$I_k'(u,v)=BITSTRING_{m,n}\left(I_k(m,n)<I_k(u,v)\right)$$ $$\sum_{u,v}HAMMING\left(I_1'(u,v),I_2'(u+d,v)\right)$$

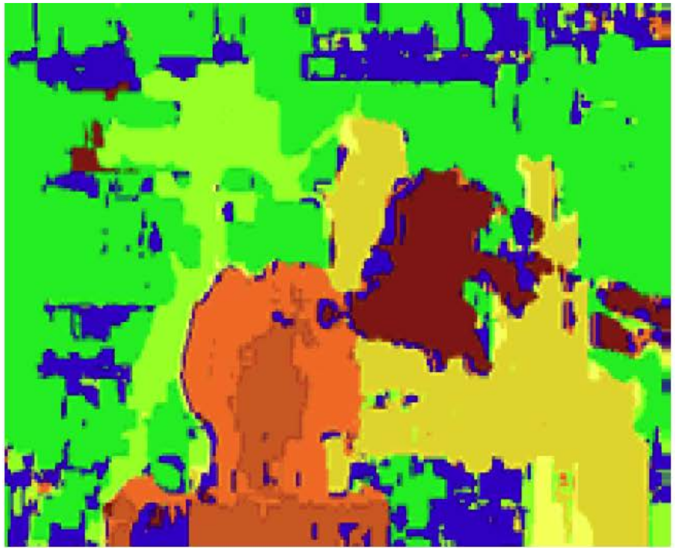
These two are actually the same

Adapted from slides from Kostas Daniilidis

Disparity computation using SSD

Scene

Ground truth



Adapted from slides from Kostas Daniilidis

Alternative Dissimilarity Measures

- Rank and Census transforms [Zabih ECCV94]
- Rank transform:
  - Define window containing R pixels around each pixel
  - Count the number of pixels with lower intensities than center pixel in the window
  - Replace intensity with rank (0..R-1)
  - Compute SAD on rank-transformed images
- Census transform:
  - Use bit string, defined by neighbors, instead of scalar rank
- Robust against illumination changes

Adapted from slides from Kostas Daniilidis

Census measure

$$\begin{matrix} 127 & 127 & 129 \\ 126 & 128 & 129 \\ 127 & 131 & A \end{matrix} \longrightarrow \begin{matrix} 1 & 1 & 0 \\ 1 & & 0 \\ 1 & 0 & a \end{matrix} \longrightarrow \{1,1,0,1,0,1,0,a\}$$

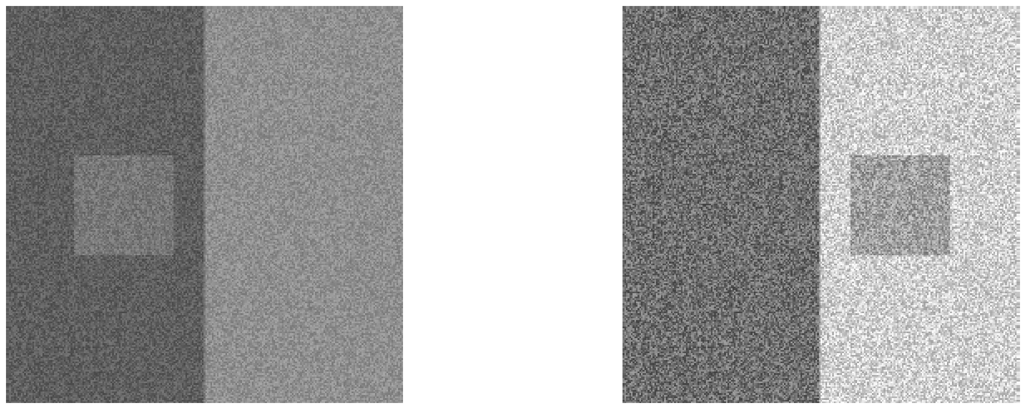
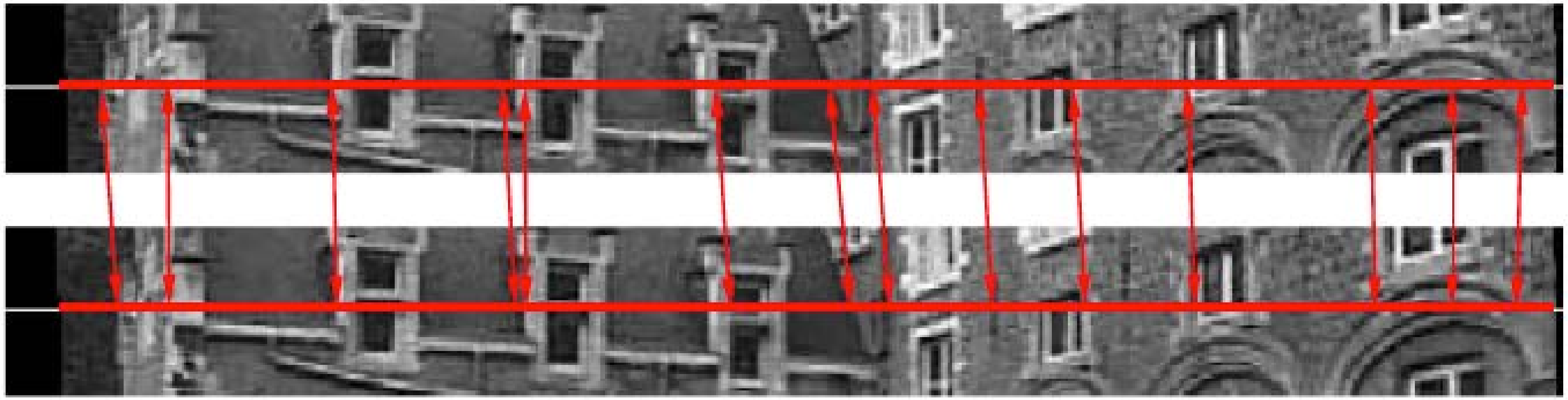


Fig. 2. Right and left random-dot stereograms

Fig. 3. Disparities from normalized correlation, rank and census transforms

Adapted from slides from Kostas Daniilidis

Correspondence



Adapted from slides from Kostas Daniilidis

Vision-Based Autonomous Navigation in Complex Environments with a Quadrotor

Shaojie Shen, Nathan Michael, and Vijay Kumar

Penn
University of Pennsylvania

The Robotics Institute

Reading

- "Computer Vision: Algorithms and Applications", Richard Szeliski, Chapter 8.4, Chapter 11
  - Download at: http://szeliski.org/Book/

Logistics

- Lab session this week - fly a robot!
  - Come to your dedicated lab session only
  - You have two weeks to complete Project 1, phase 3
- Tips
  - Maximum 2 groups fly simultaneously in the field
  - Use your hand to hold the robot while testing
  - Write controller in C++ following the code structure we provide to you
  - You do not have to write the trajectory generator in C++. You may generate the trajectory in MATLAB and move the coefficients onboard.
  - No need to worry about system setup. We have done it for you.
- Expectations:
  - This week: Get familiar with the setup; complete controller; hovering;
  - Next week: Straight line & multi-waypoint trajectory tracking

Logistics

- Proposed change of lecture time for next week
  - 24 Oct (Saturday), 10am