# Introduction to Aerial Robotics
## Lecture 7

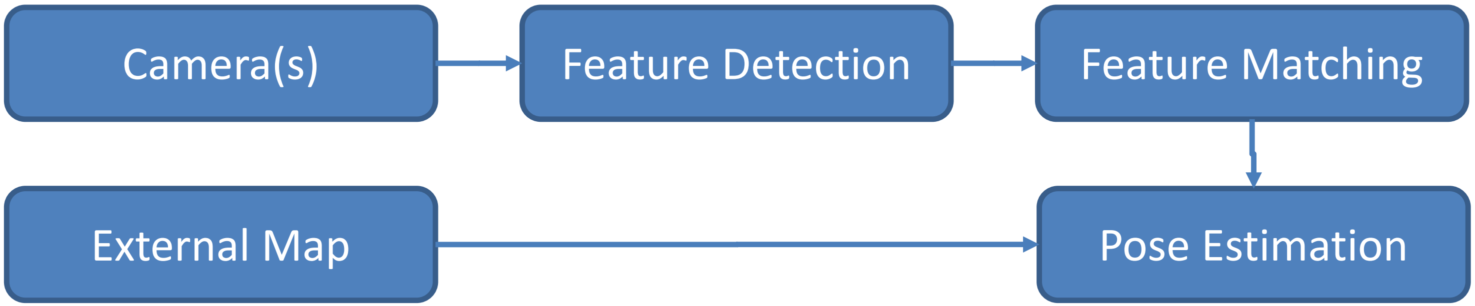
Shaojie Shen
Associate Professor
Dept. of ECE, HKUST

23 March 2021

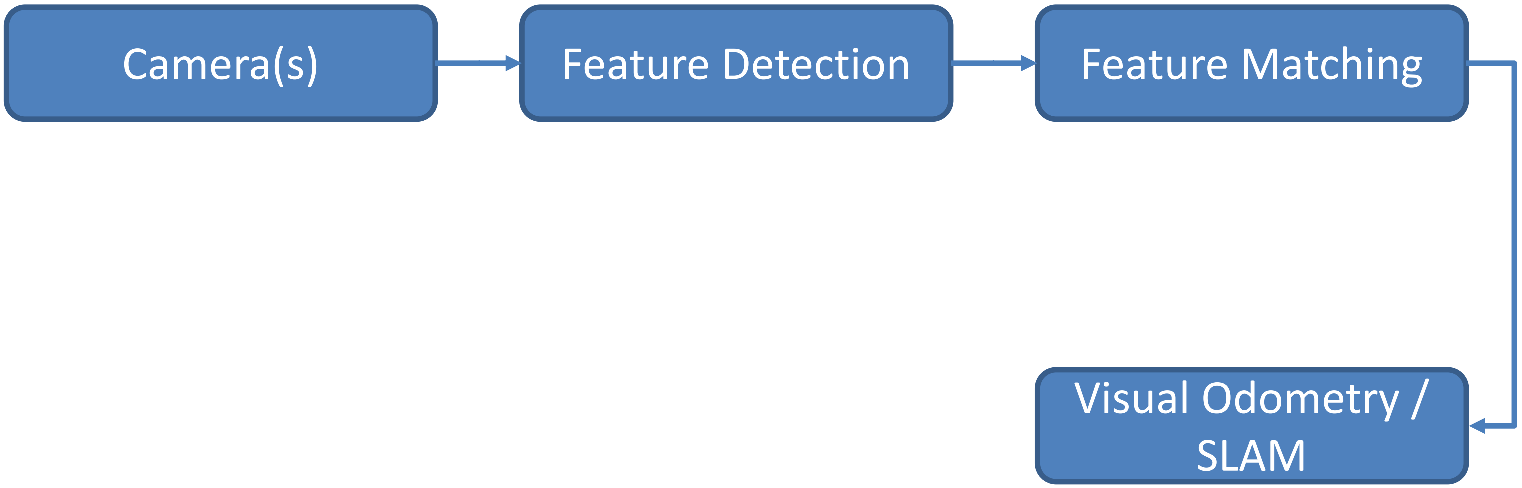
# Outline

- Optical Flow
- Stereo Vision
- Visual Odometry

# Vision-based Pose Estimation Pipeline (aka. Map-based Localization)

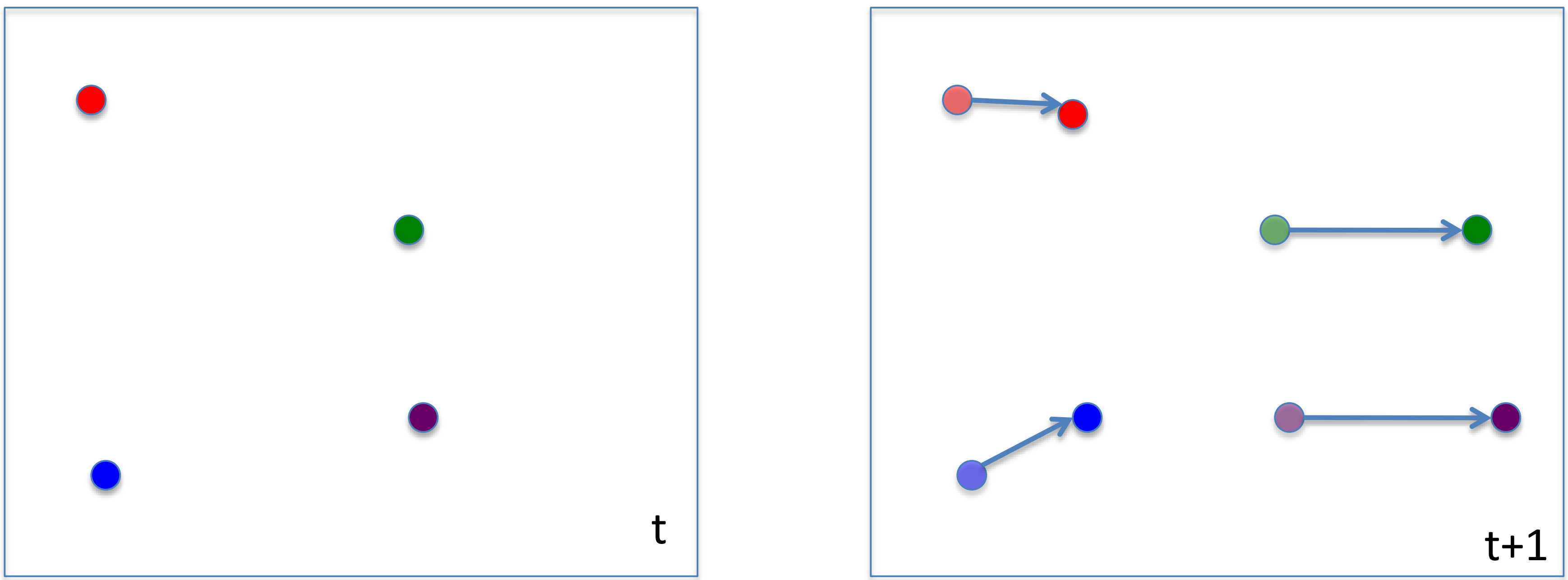


# Vision-based Incremental Pose Estimation Pipeline (aka. Visual Odometry)



# Frame-to-Frame Feature Matching 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 $t + 1$

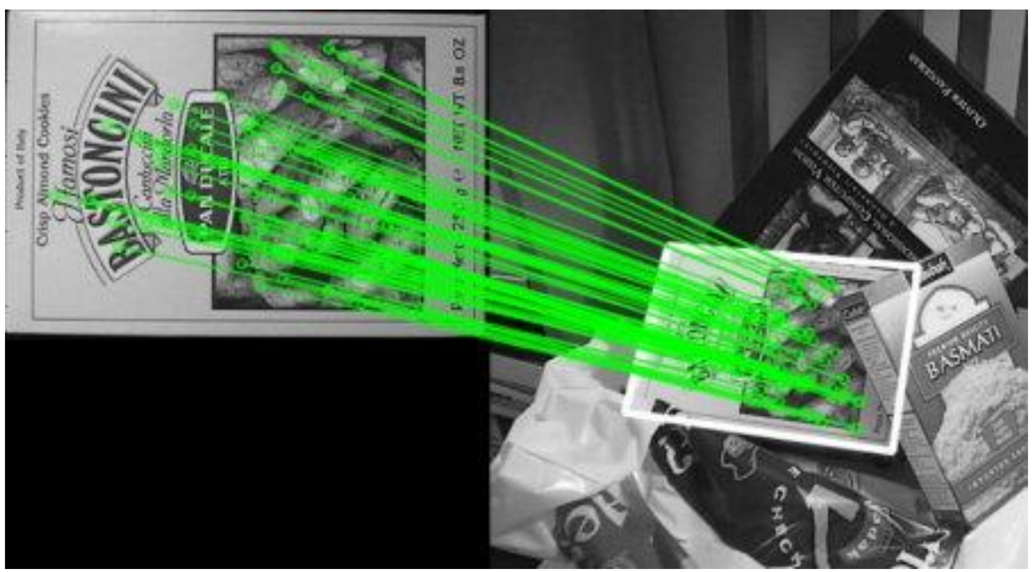





Slide adapted from Kostas Daniilidis

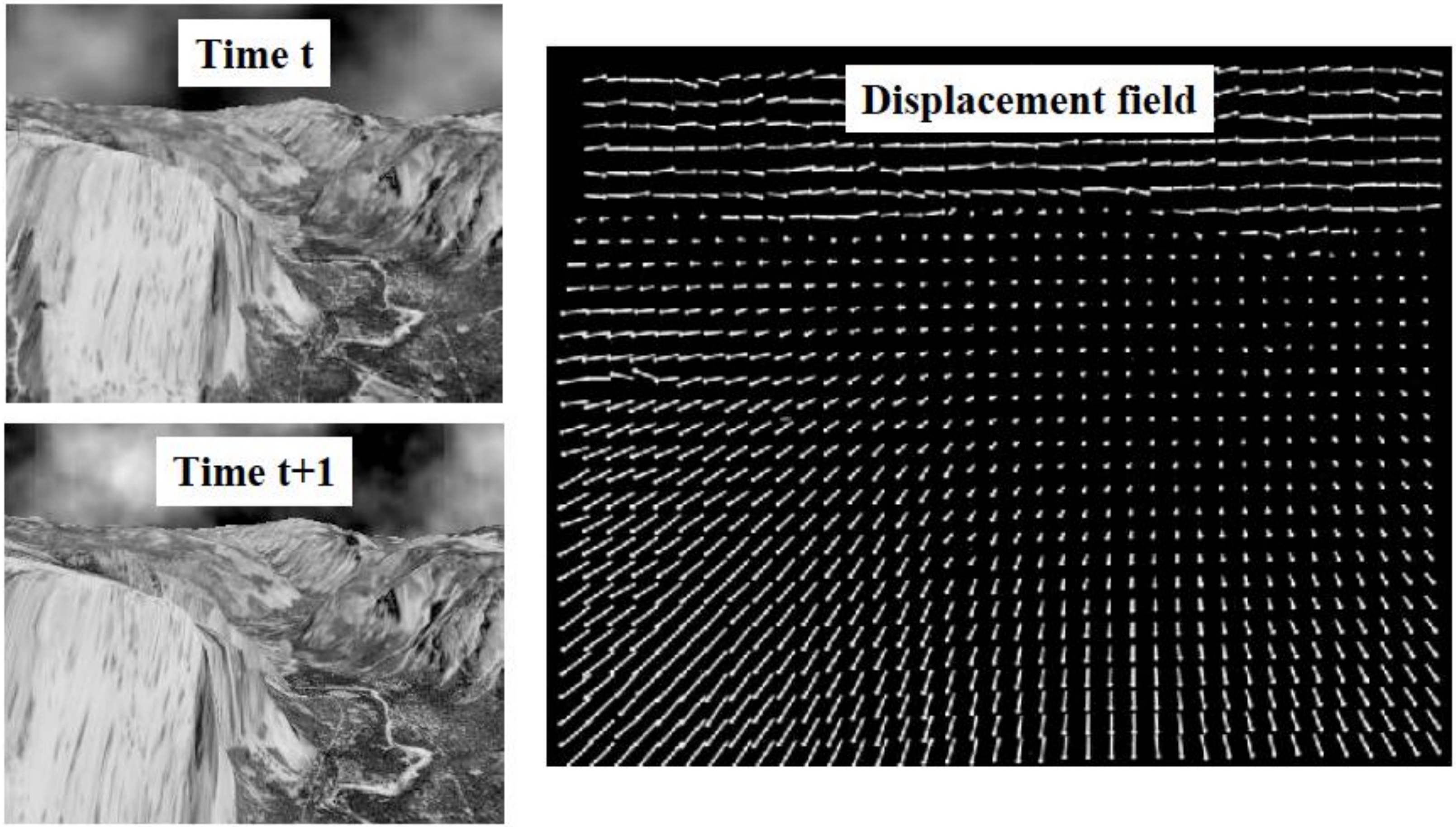
# Discrete Feature Matching Approach

- 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



# Optical Flow

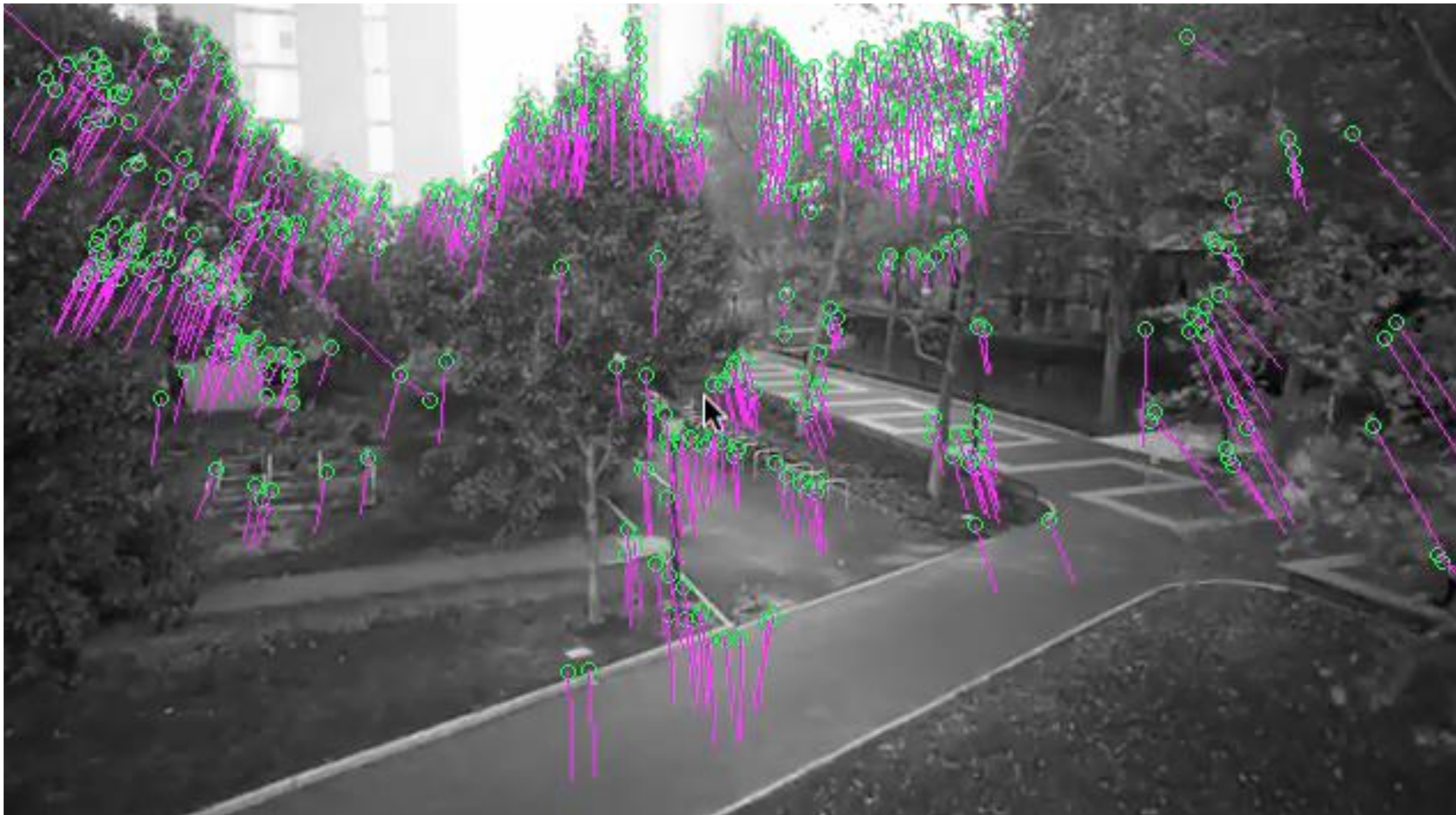
# Differential Approach: Optical Flow






Slide adapted from Kostas Daniilidis

# Differential Approach: Optical Flow





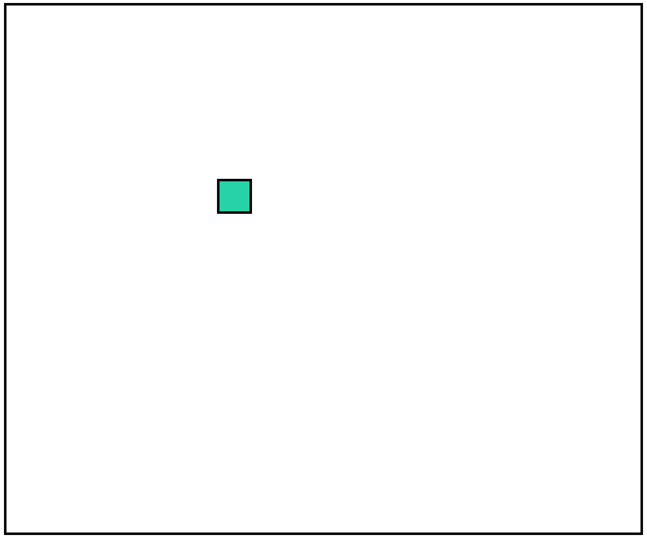
Slide adapted from Kostas Daniilidis

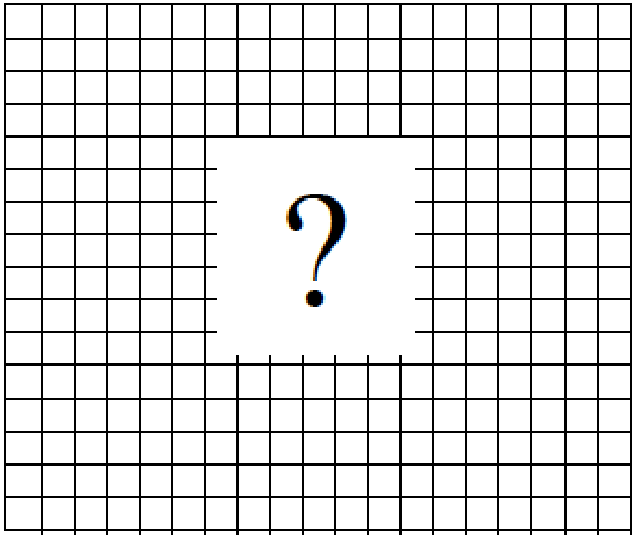
# Differential Approach: Optical Flow

- 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



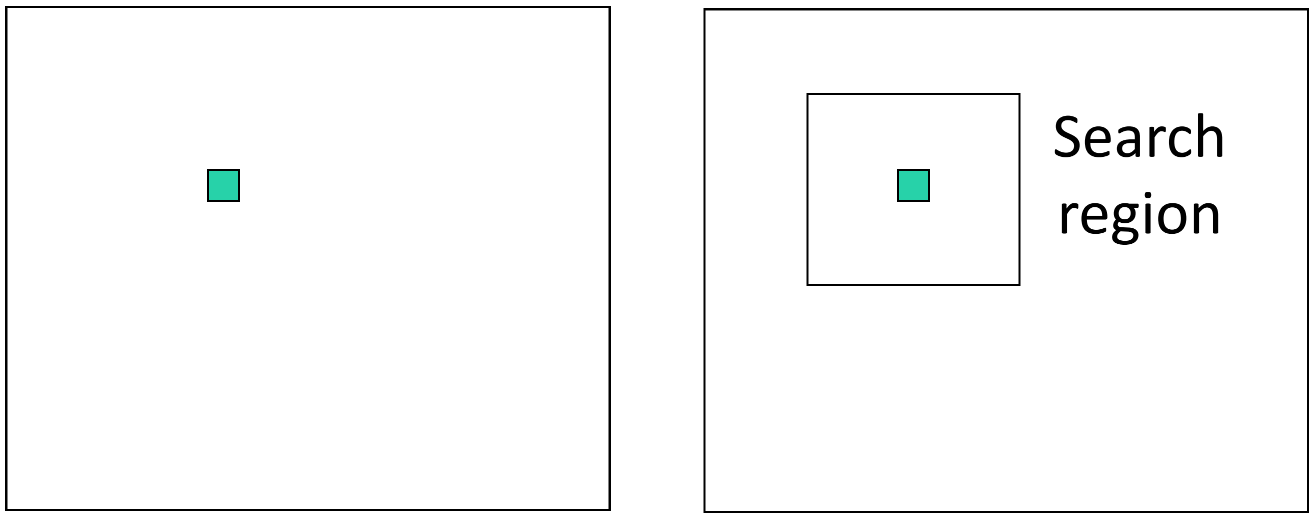





Slide adapted from Kostas Daniilidis

Given image patch in one image

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



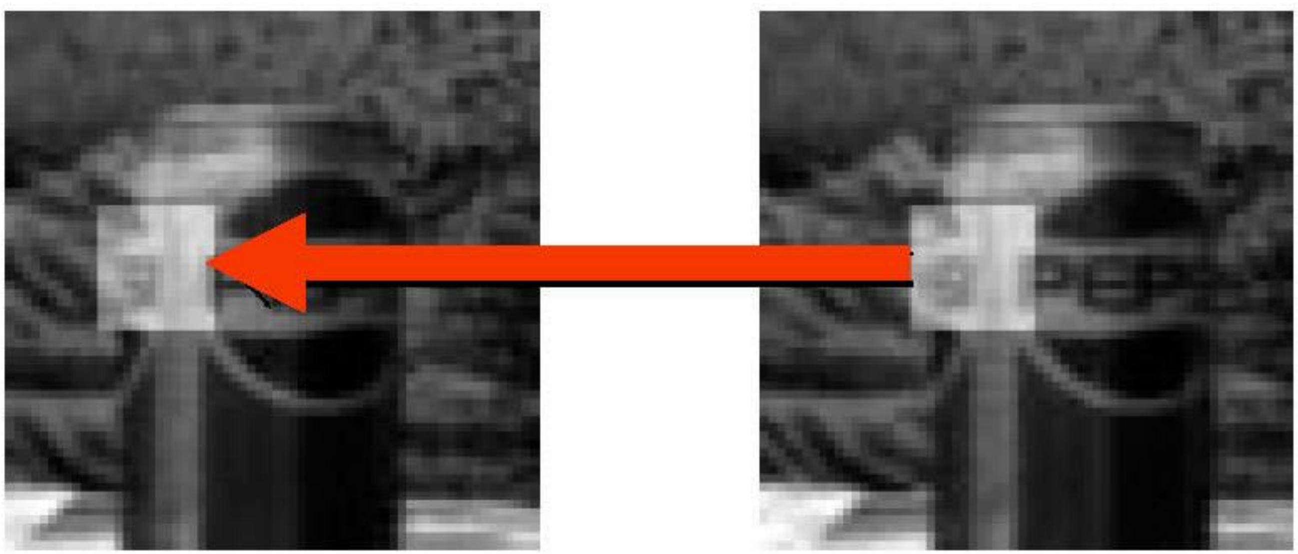

- The motion is known to be "small", we can bound the search region.



Slide adapted from Kostas Daniilidis

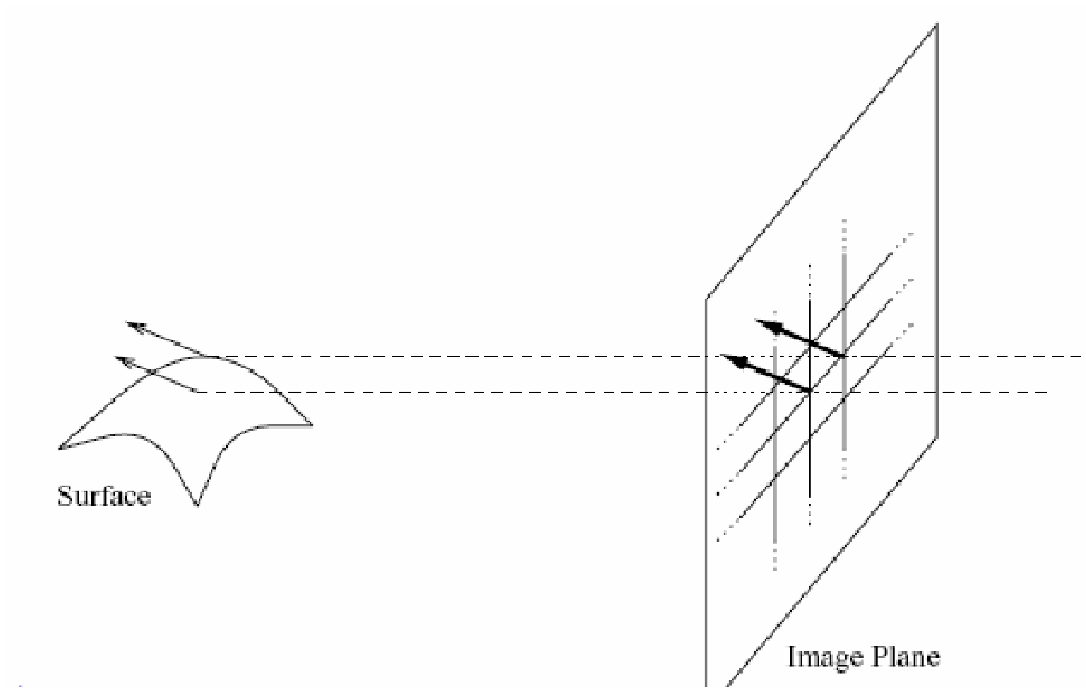
# Optical Flow Assumption: Brightness Constancy

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



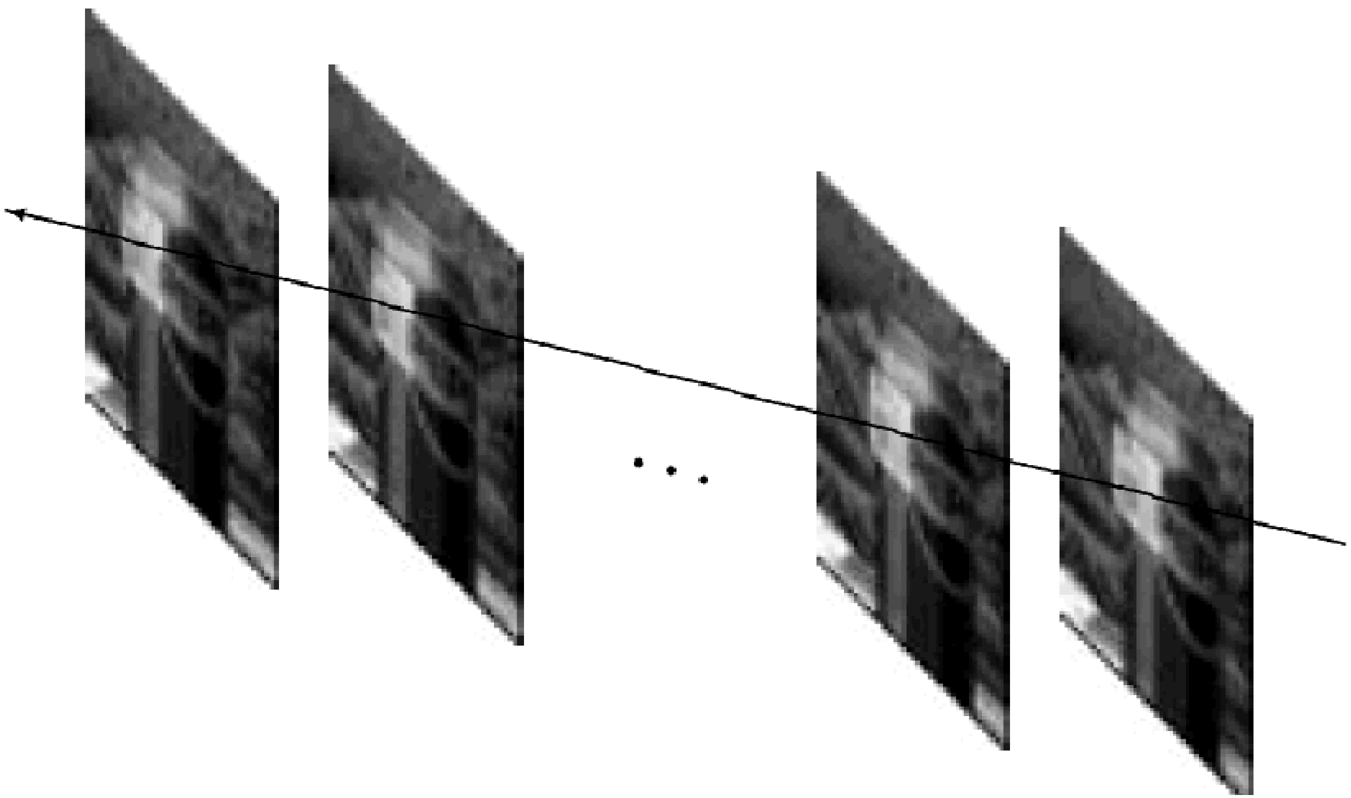
# Optical Flow Assumption: Spatial Coherence

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



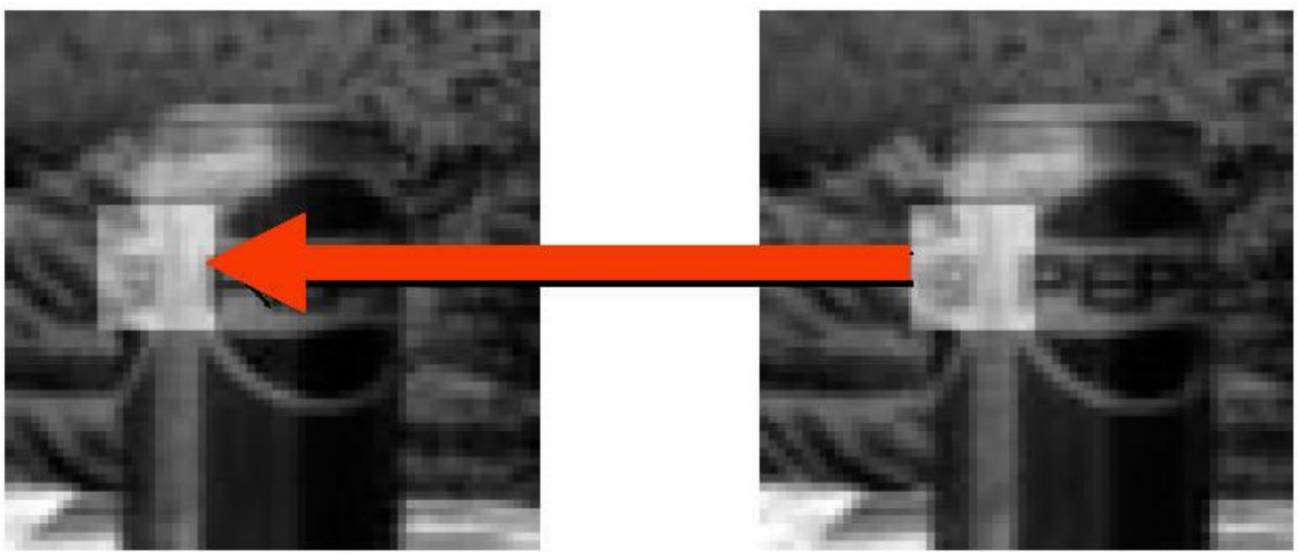
# Optical Flow Assumption: Temporal Persistence

- 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 $\epsilon$ 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:

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

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



Slide adapted from Kostas Daniilidis

# Step 2: Linearization

$$[J(x+d) - I(x)]$$

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

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

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



Slide adapted from Kostas Daniilidis

# Step 2: Linearization

- Combining previous equations:

$$\frac{1}{2}\frac{\partial_{\in}}{\partial_{d}} = \int_{w} [J(x+d) - I(x)]\, g\, w\, dx = 0$$

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

$$\downarrow$$

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



Slide adapted from Kostas Daniilidis

# Step 3: Solve Linear System

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

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

A: second moment matrix



Slide adapted from Kostas Daniilidis

# Step 3: Solve Linear System

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

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

Error vector b



Slide adapted from Kostas Daniilidis

# Step 3: Solve Linear System

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

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

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

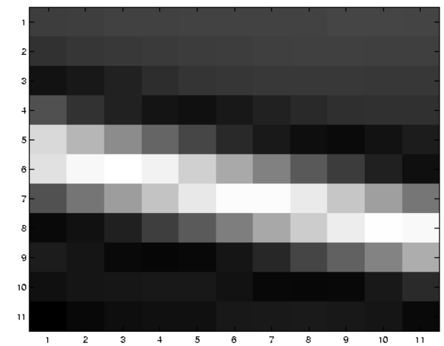
$$A\, d = b$$
$$d = A^{-1} b$$

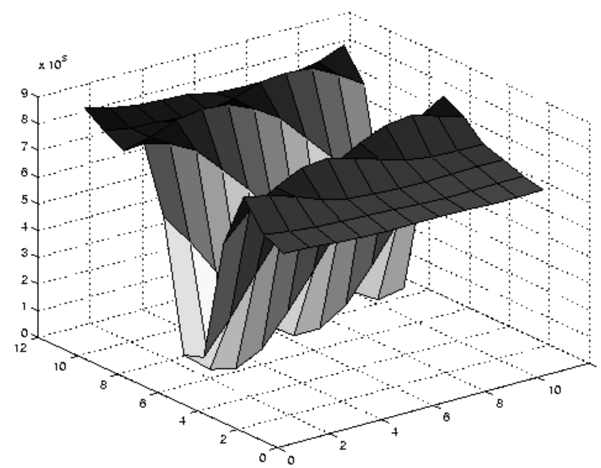
- What if $A$ is not full rank? Recall the structure of $A$:
  - Same as the one for corner detection
  - Eigenvalues and eigenvectors of $A$ tells whether we are tracking a corner



Slide adapted from Kostas Daniilidis

# Edge



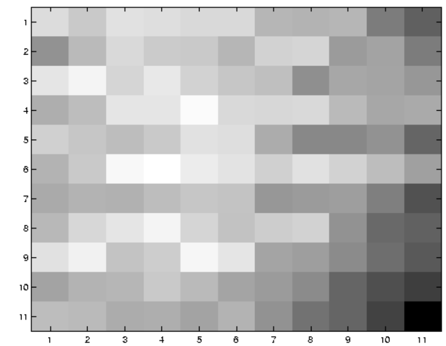
– large gradients, all the same direction

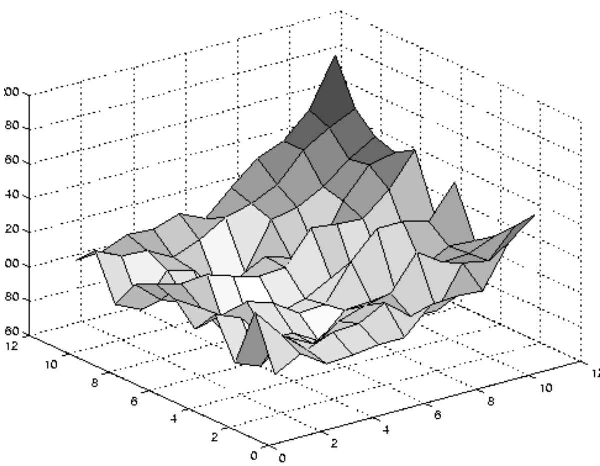
– large $l_1$, small $l_2$



Slide adapted from Kostas Daniilidis

# Low Texture Region



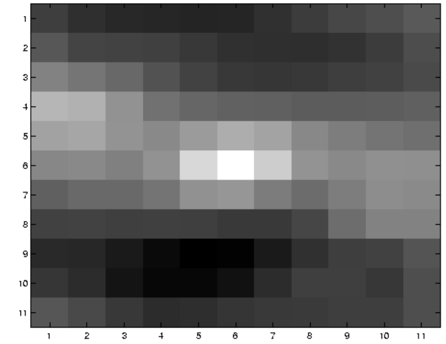
– gradients have small magnitude

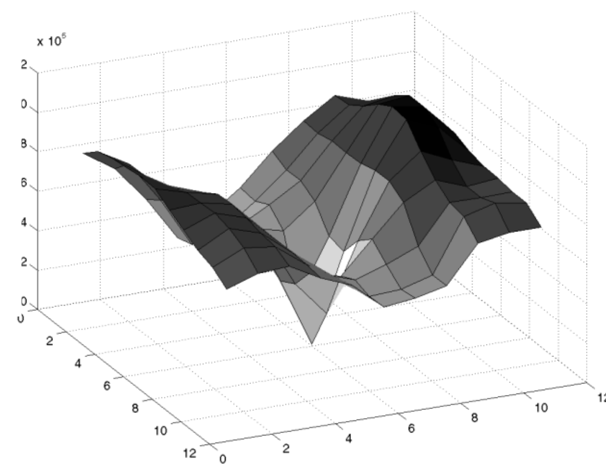
– small $l_1$, small $l_2$



Slide adapted from Kostas Daniilidis

# High Texture Region





– gradients are different, large magnitudes

– large $l_1$, large $l_2$



Slide adapted 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

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

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

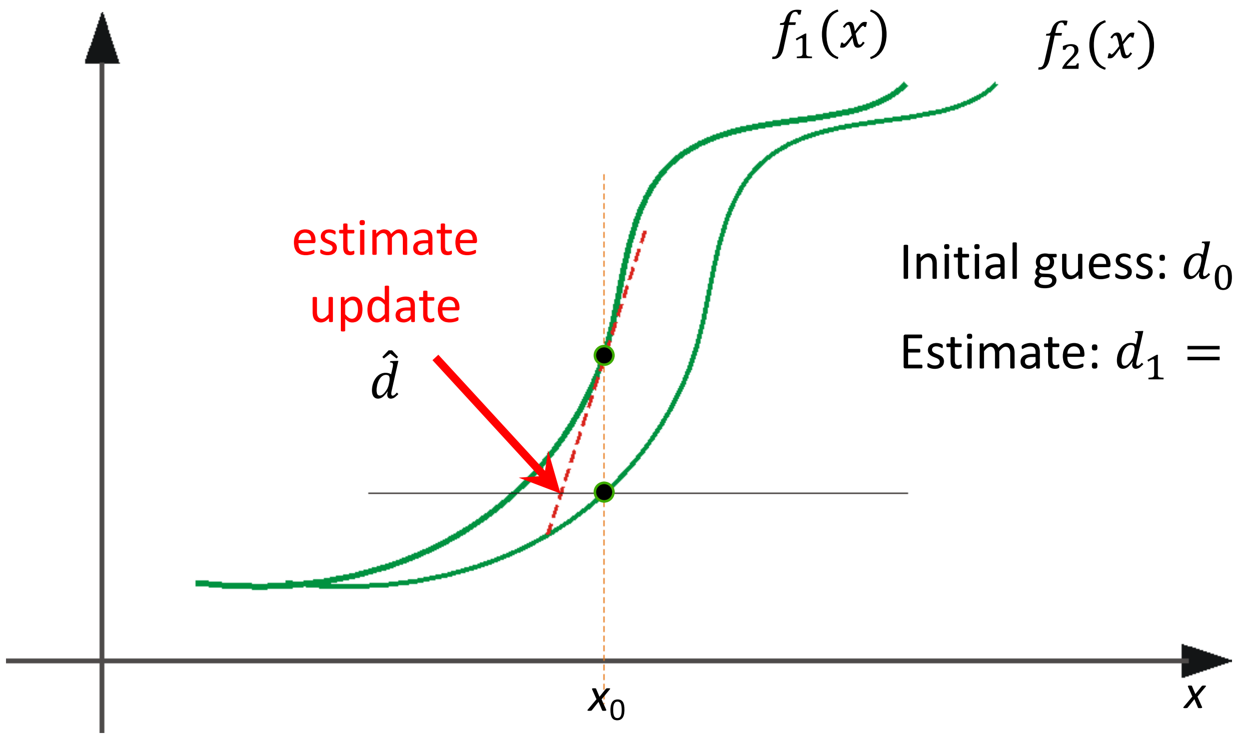
$$b = \sum_{i,j} \begin{bmatrix} g_x(i,j)[I(i,j) - J(i,j)] \\ g_y(i,j)[I(i,j) - J(i,j)] \end{bmatrix}$$

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

- 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

Compute $d$ to minimize $\| f_1(x+d) - f_2(x) \|^2$




Initial guess: $d_0 = 0$

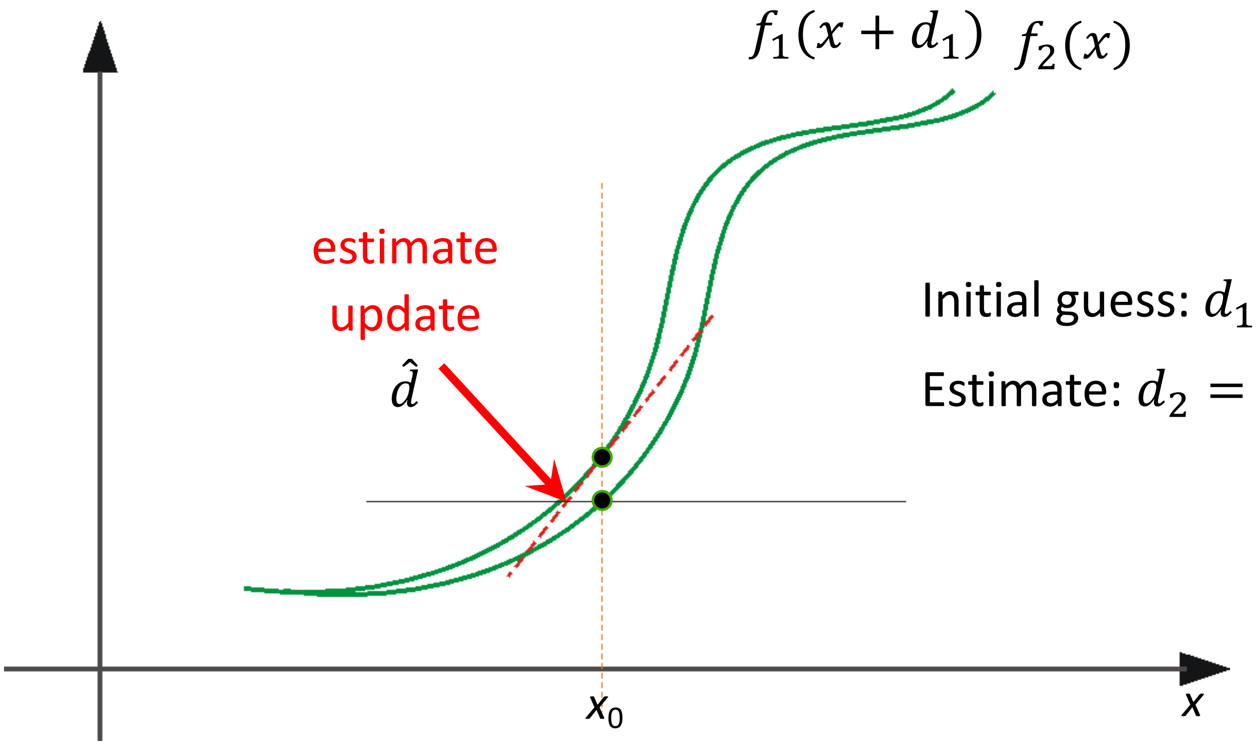
Estimate: $d_1 = d_0 + \hat{d}$



Slide adapted from Richard Szeliski

# Step 4: Iterative Refinement

Compute $d$ to minimize $\|f_1(x+d) - f_2(x)\|^2$




Initial guess: $d_1$

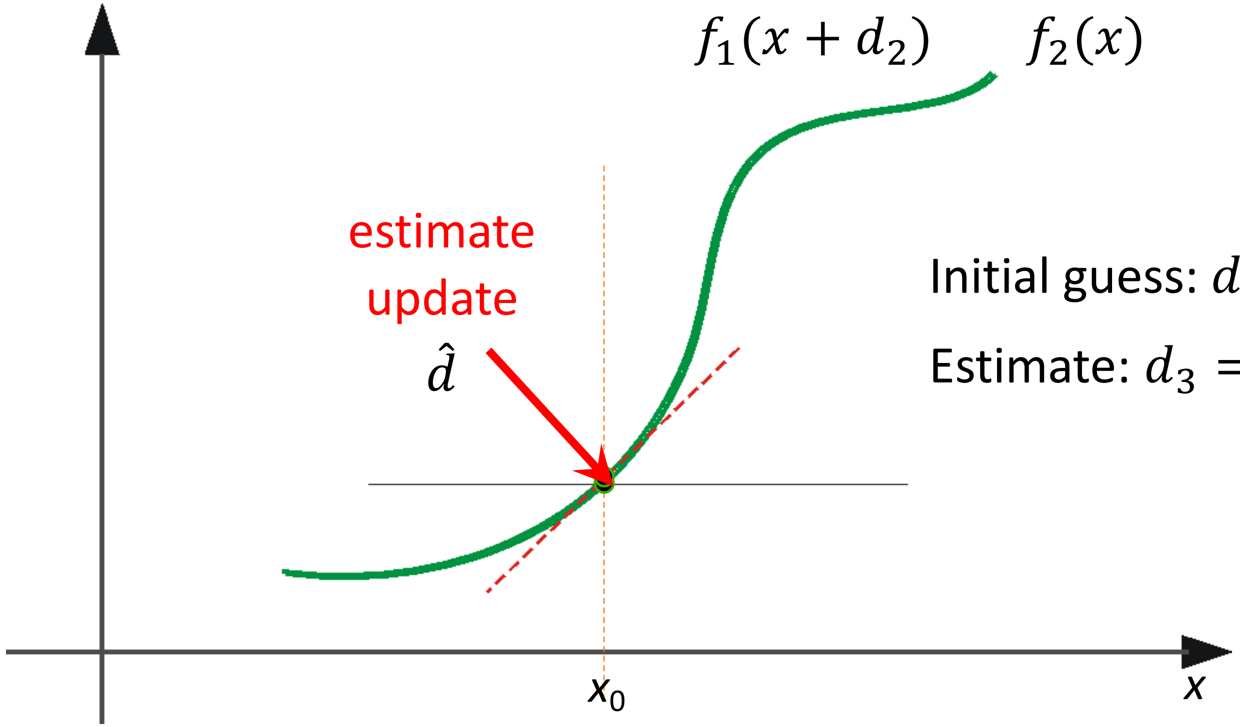
Estimate: $d_2 = d_1 + \hat{d}$



Slide adapted from Richard Szeliski

# Step 4: Iterative Refinement

Compute $d$ to minimize $\|f_1(x+d) - f_2(x)\|^2$




Initial guess: $d_2$

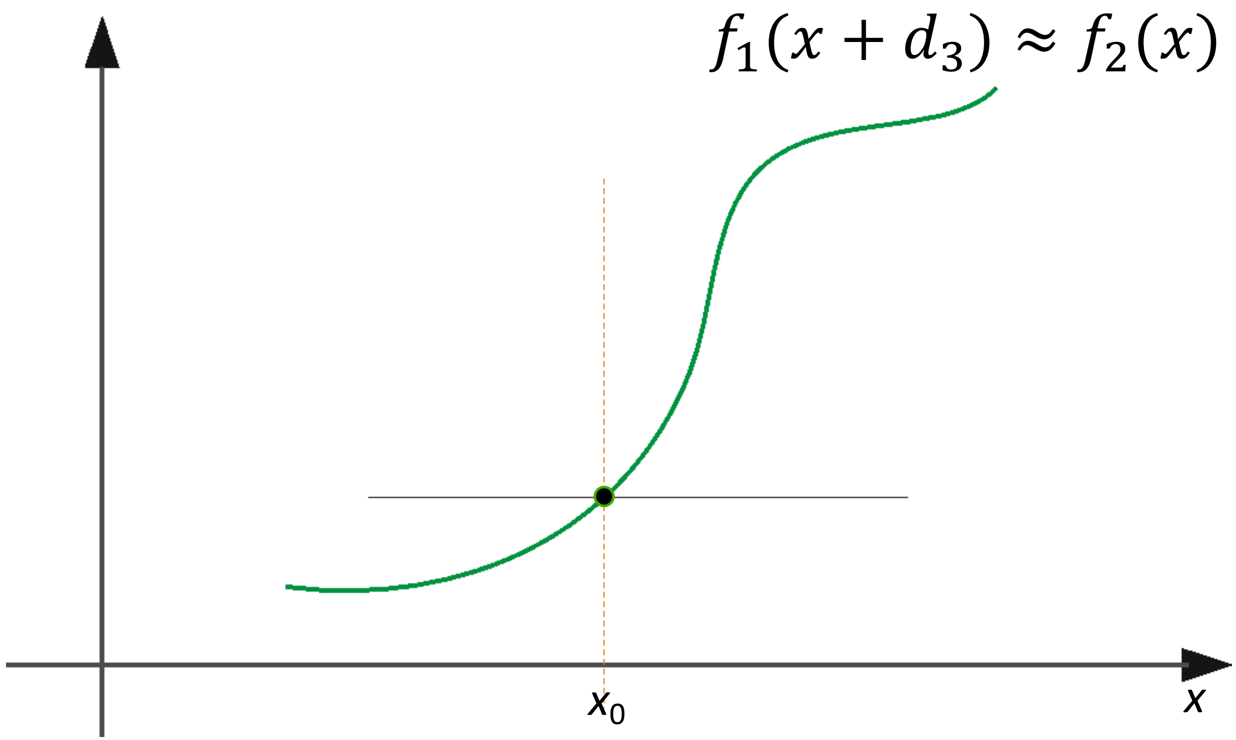
Estimate: $d_3 = d_2 + \hat{d}$



Slide adapted from Richard Szeliski

# Step 4: Iterative Refinement

Compute $d$ to minimize $\| f_1(x+d) - f_2(x) \|^2$






Slide adapted from Richard Szeliski

# Motion Models

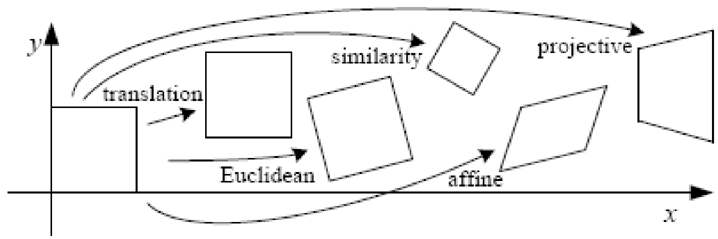
- 2D Models:
  - Affine
  - Quadratic
  - Planar projective transform (Homography)

- 3D Models:
  - Instantaneous camera motion models
  - Homography+epipole
  - Plane+Parallax

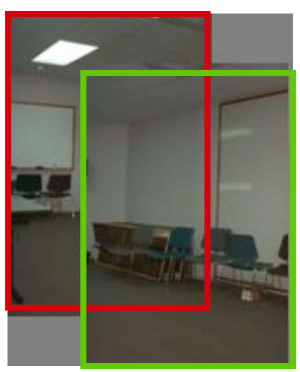


Slide adapted from Richard Szeliski

# Motion Models

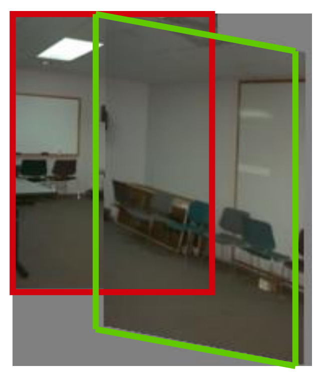



Translation



2 unknowns

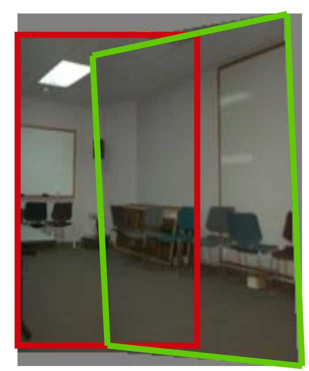
Affine



6 unknowns:

$$x' = Ax + d$$

3D rotation



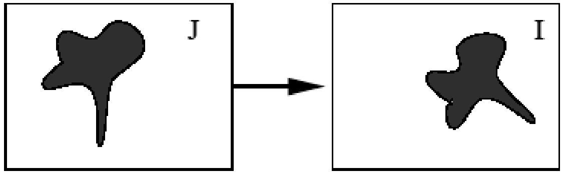
3 unknowns



Slide adapted from Richard Szeliski

# Compute Affine Motion

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




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

$$\in= \int_W [J(Ax + d) - I(x)]^2 w(x) dx \quad (1)$$

Let $A = I + D$, min. $\in$ with respect to $D \in R^{2\times2}$, and $d \in R^{2\times1}$

- Three steps for solving this problem:
  - Set $\frac{\partial_\in}{\partial_D}, \frac{\partial_\in}{\partial_d}$ to 0;
  - Taylor expression on J(Ax+d) respect to x;
  - Solve for A(D) and d



Slide adapted from Kostas Daniilidis

# Compute Affine Motion

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

- Differentiate $\in$ with respect to D and d,

$$\frac{1}{2}\frac{\partial_\in}{\partial_D} = \int_w [J(Ax+d) - I(x)]\, g\, x^T\, w\, dx = 0 \qquad (2)$$

$$\frac{1}{2}\frac{\partial_\in}{\partial_d} = \int_w [J(Ax+d) - I(x)]\, g\, w\, dx = 0 \qquad (3)$$

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

- Assume small motion, $Ax + d = x + (Dx + d) = x + u$,
  – Taylor expression of $J(Ax+d)$ is: $J(Ax+d) = J(x) + g^T u$



Slide adapted from Kostas Daniilidis

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

- 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:

$$\int_W g\, x^T\, (g^T u) w\ dx = \int_W [I(x) - J(x)] g\, x^T\, dx \qquad (5)$$
$$\int_W g\, (g^T u) w\ dx = \int_W [I(x) - J(x)] g\, w\, dx \qquad (6)$$

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


Slide adapted from Kostas Daniilidis

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

- $Tz = a$:

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

and

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

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

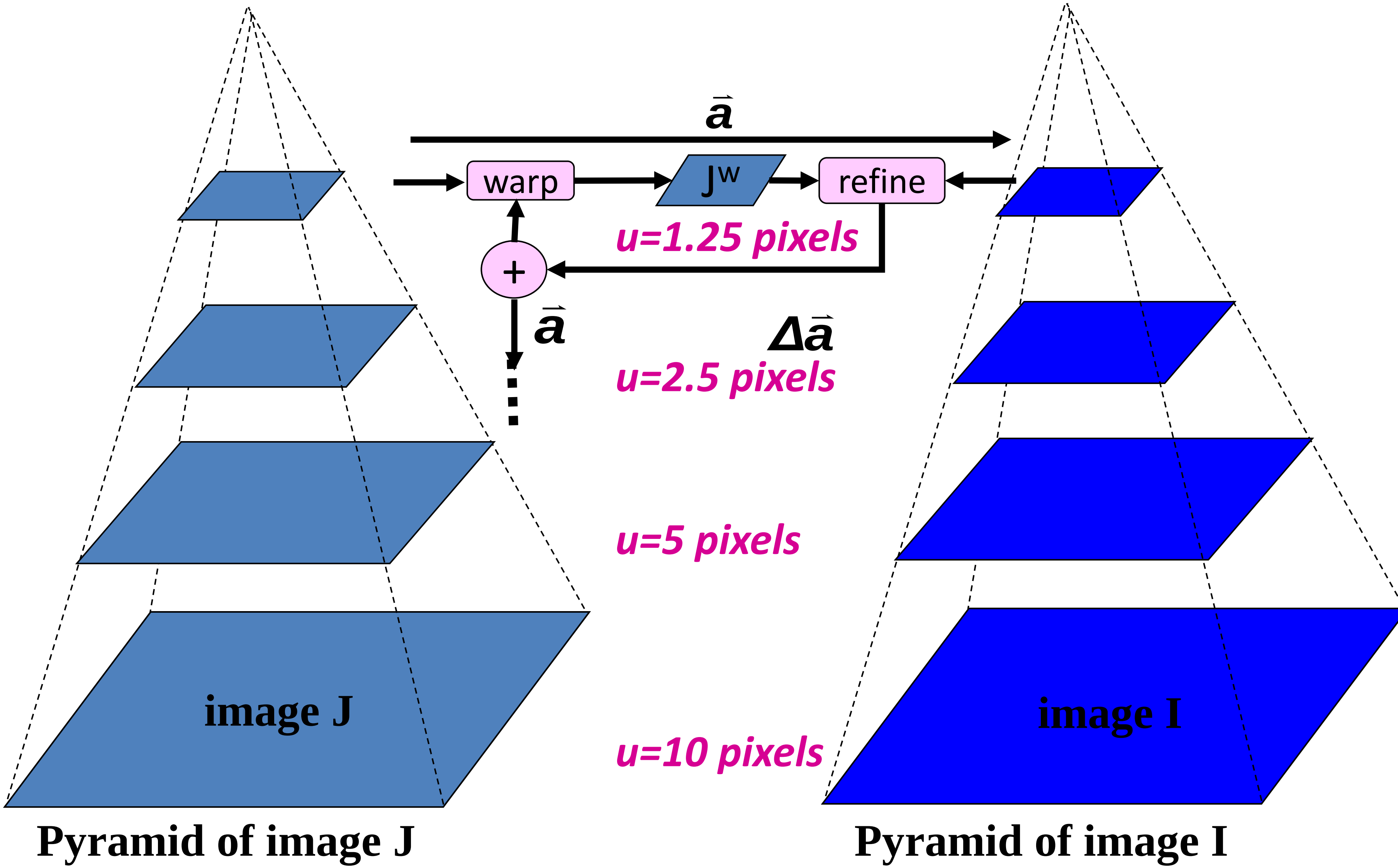


Slide adapted from Kostas Daniilidis

# 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

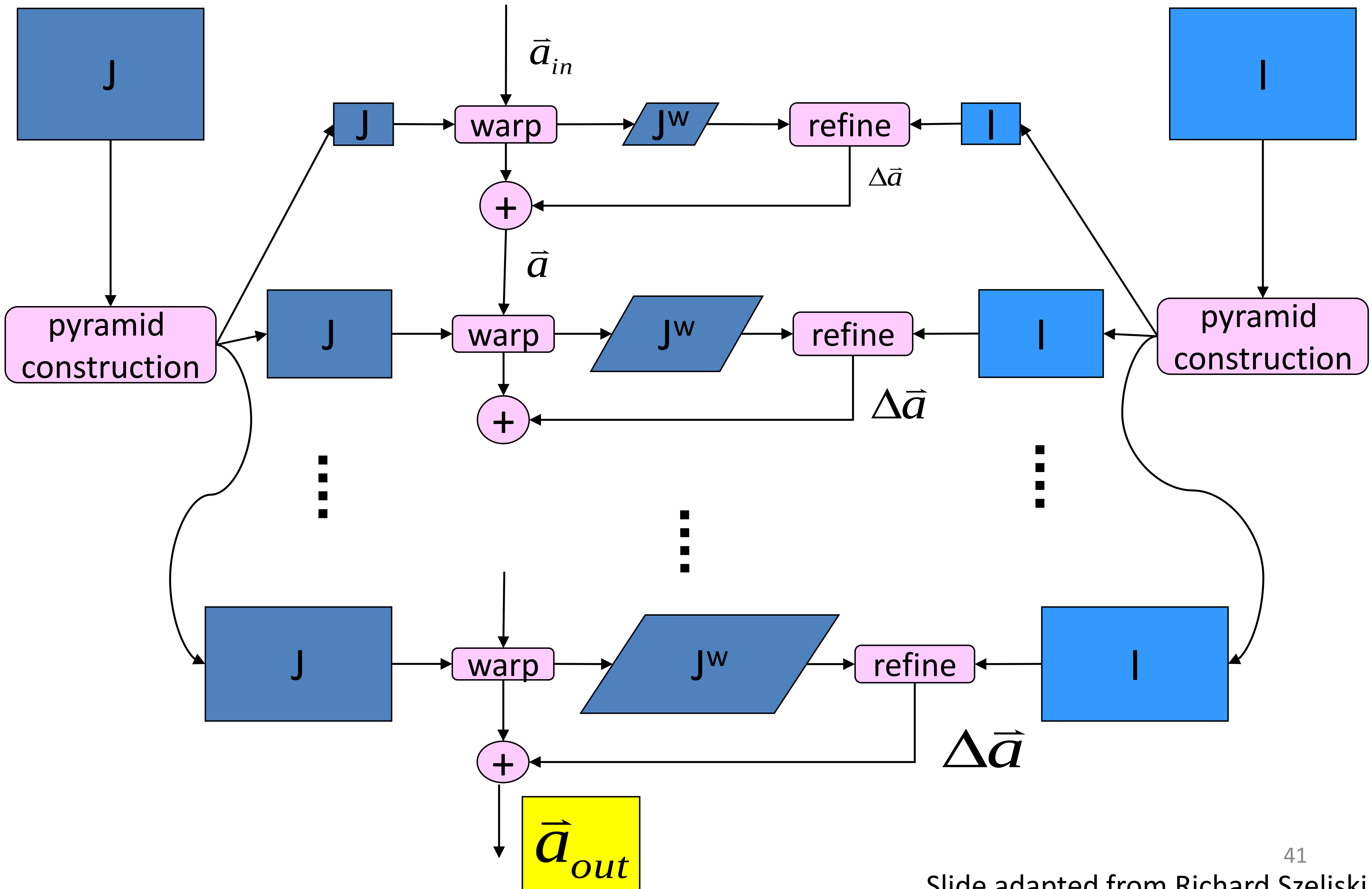
# Coarse-to-Fine Estimation






Slide adapted from Richard Szeliski

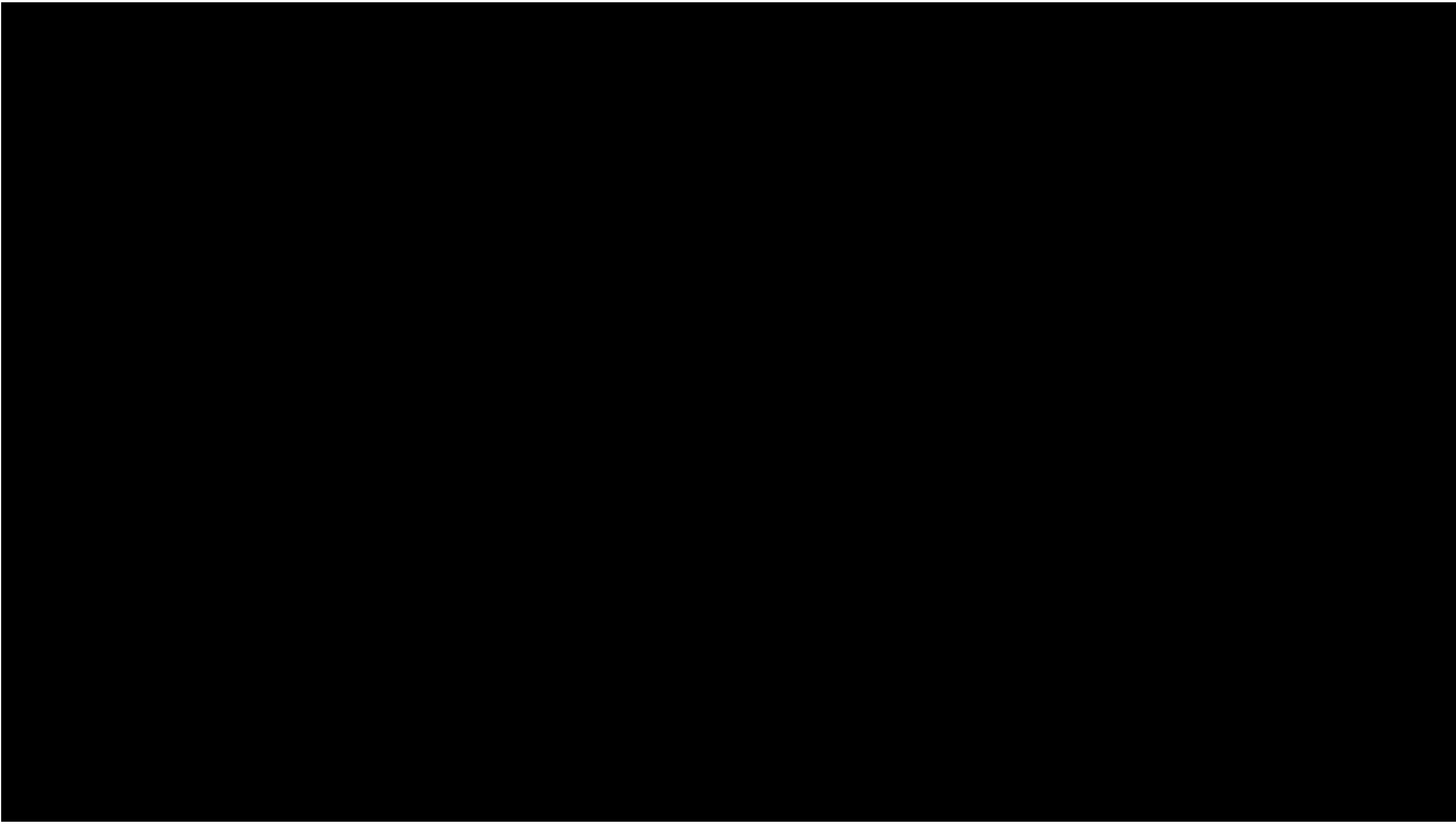
# Coarse-to-Fine Estimation






Slide adapted from Richard Szeliski

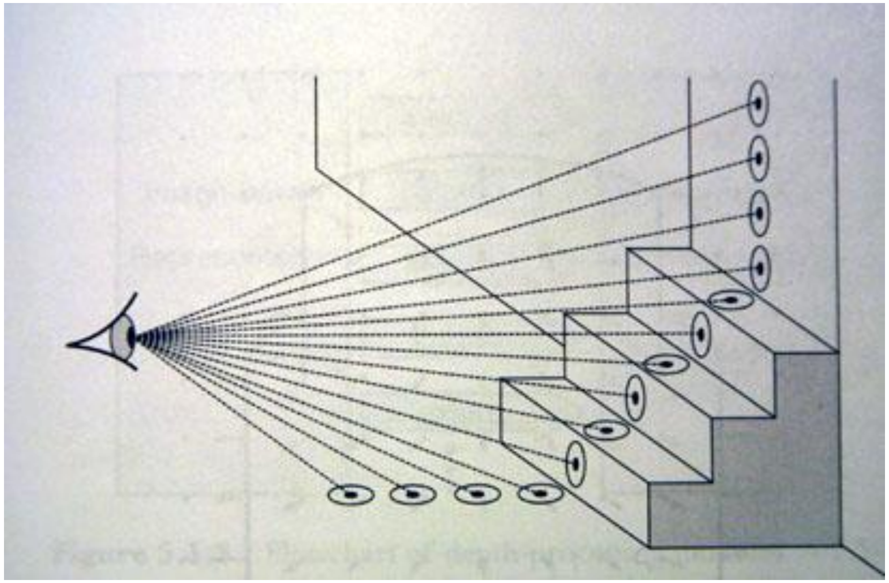
# Optical Flow-based Velocity Estimator

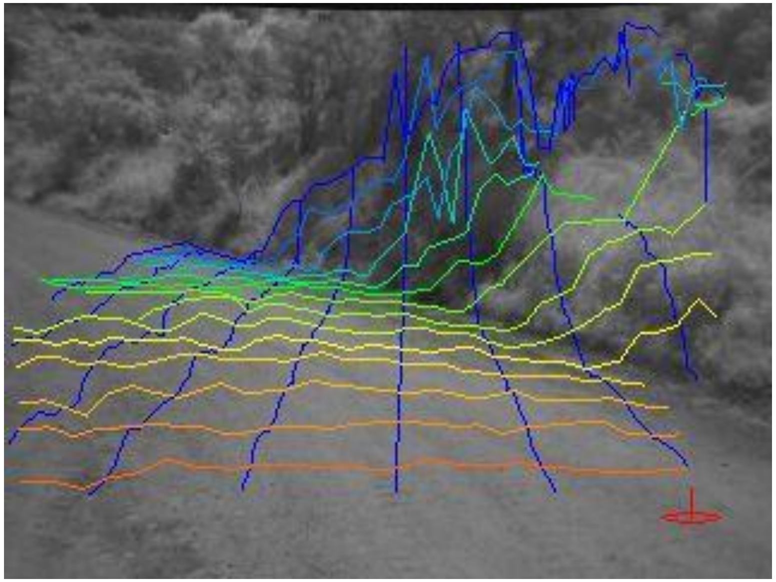


# Stereo Vision

# 3D Shape perception

- Depth: the distance of the surface from the observer
- Surface orientation: the slant and tilt of the surface with respect to observers' sight



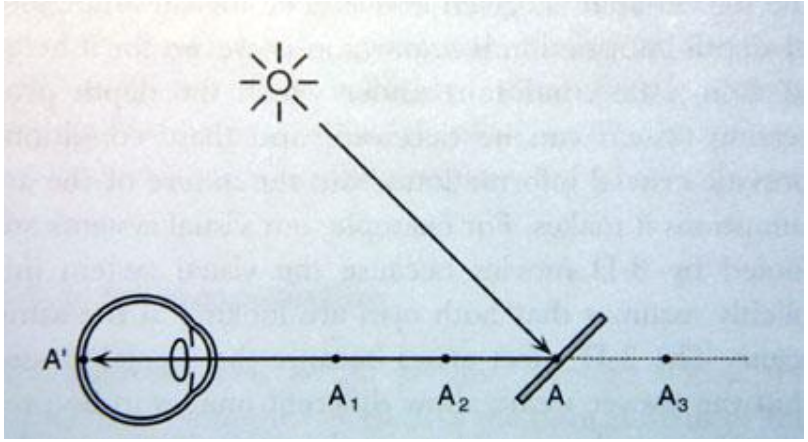


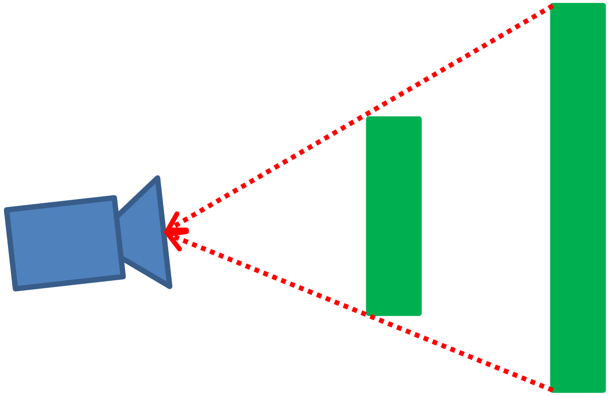


Slide adapted from Kostas Daniilidis

# Depth ambiguity

*Inverse problem: multiple solution exists*



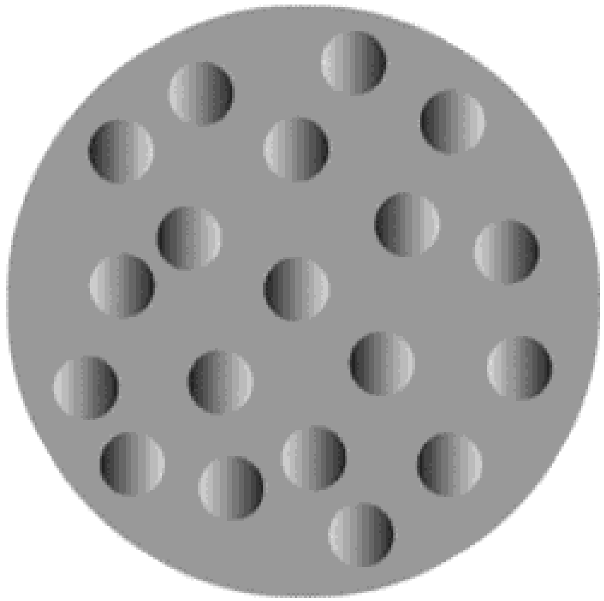


Slide adapted from Kostas Daniilidis

# Pictorial cues for 3D shape

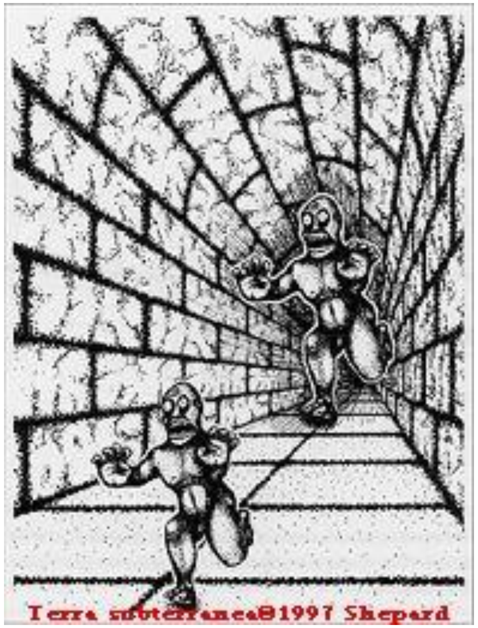
- Perspective projection gives us the relative position to horizon, therefore we can deduce its physical size

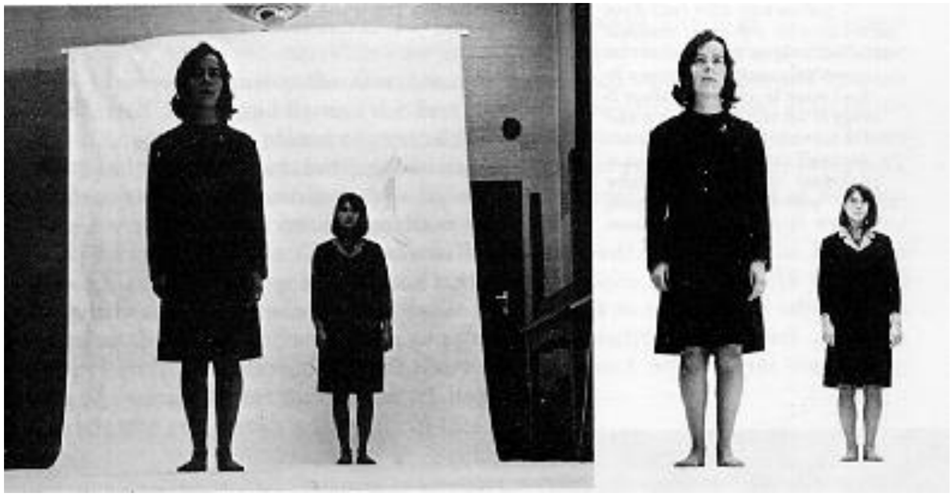
- Shading also reveal shape using illumination model





Slide adapted from Kostas Daniilidis

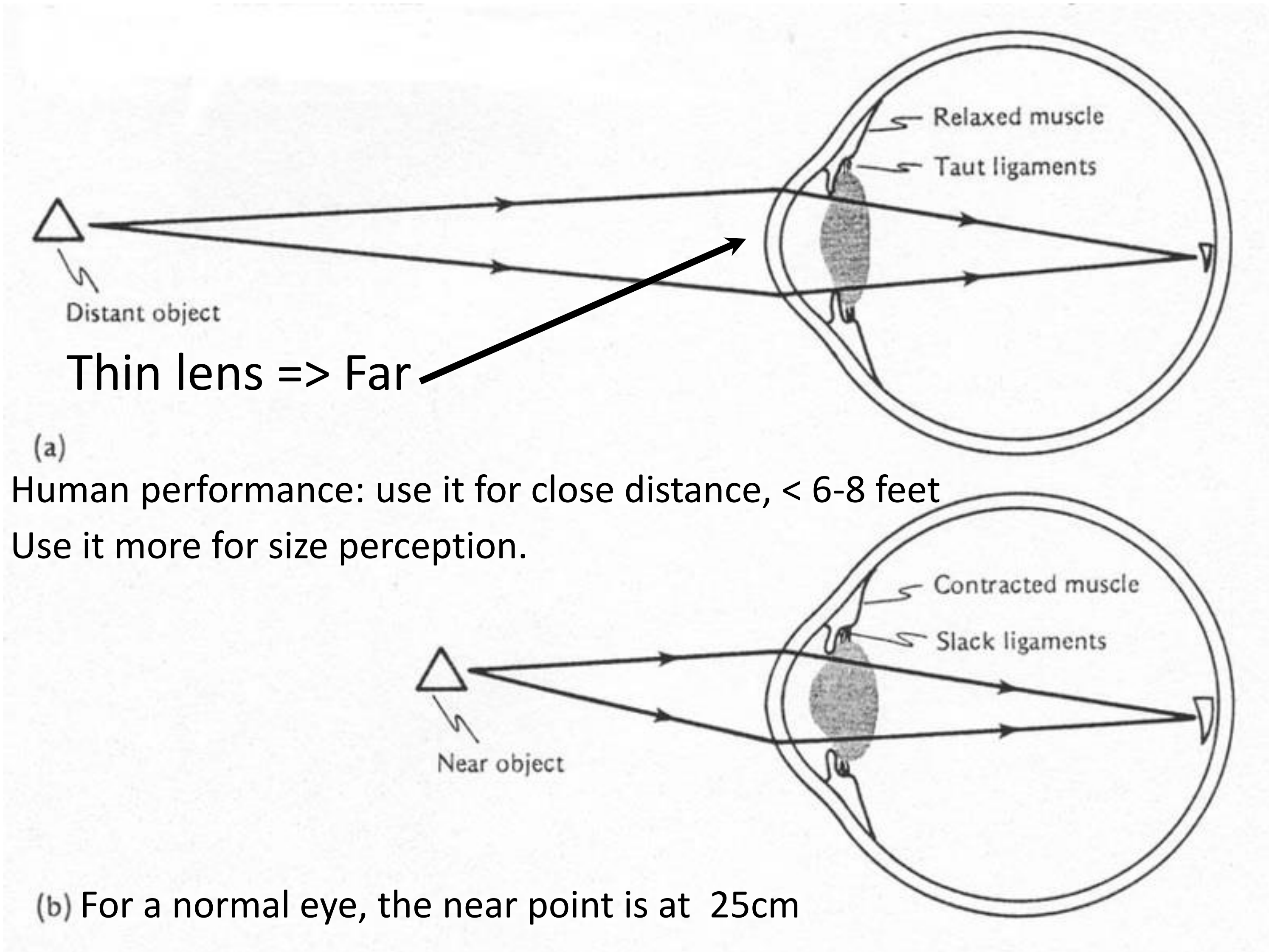




Slide adapted from Kostas Daniilidis

# Shape from Focus, Accommodation




Thin lens => Far

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



Slide adapted from Kostas Daniilidis

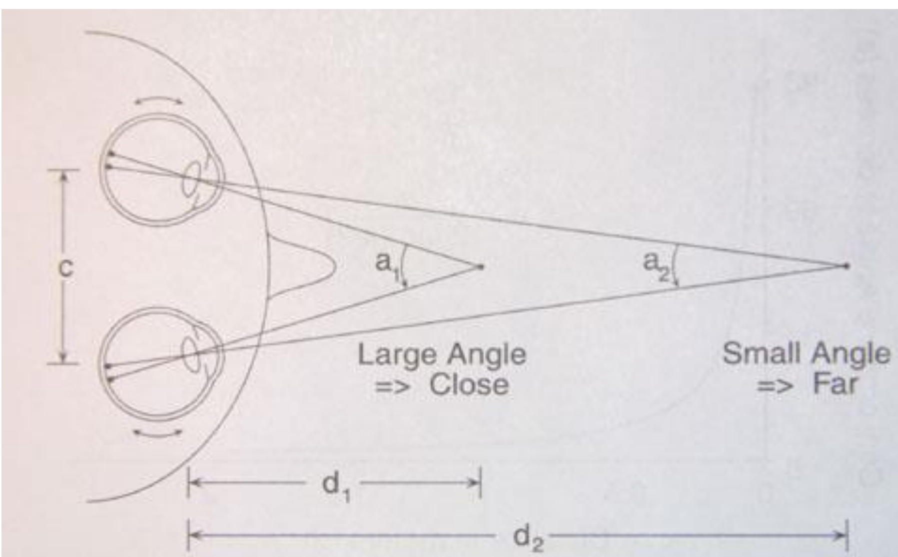
- Because of its restricted range (6-8 feet), accommodation is rarely a crucial source of depth in humans.
- In the chameleon, it is of paramount importance, for it controls this organism's ability to feed itself.  A chameleon catches its prey by slicking its sticky tongue out just the right distance to catch an insect.
- 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.



Slide adapted 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.



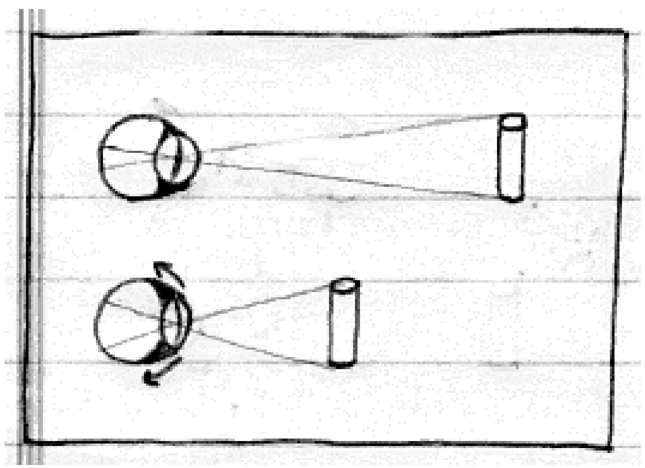

$$d = \frac{c}{2tan(\frac{a}{2})}$$

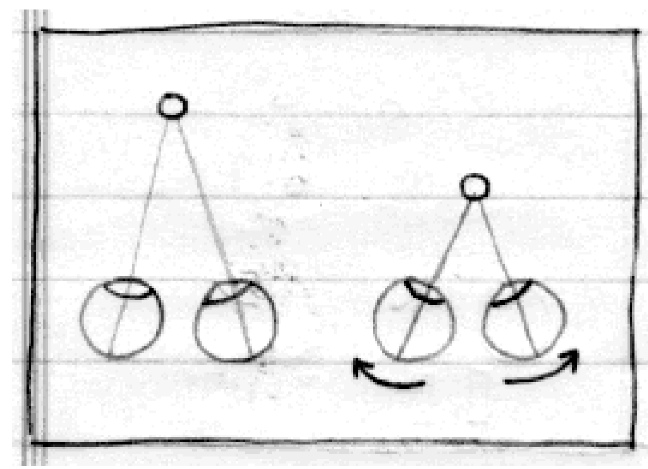


Slide adapted from Kostas Daniilidis

# Accommodation and Convergence

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



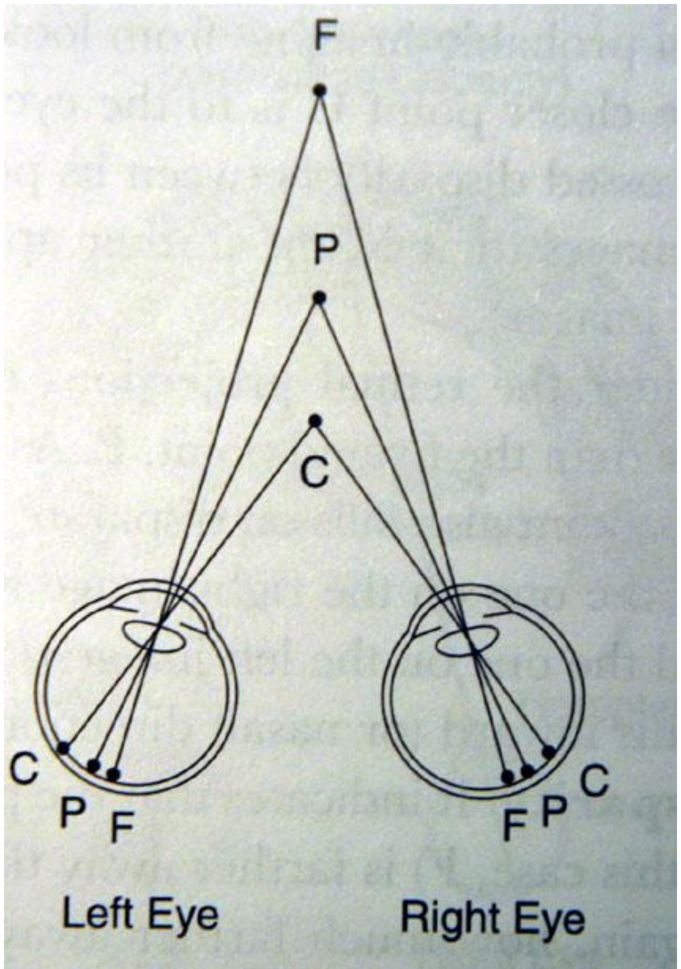


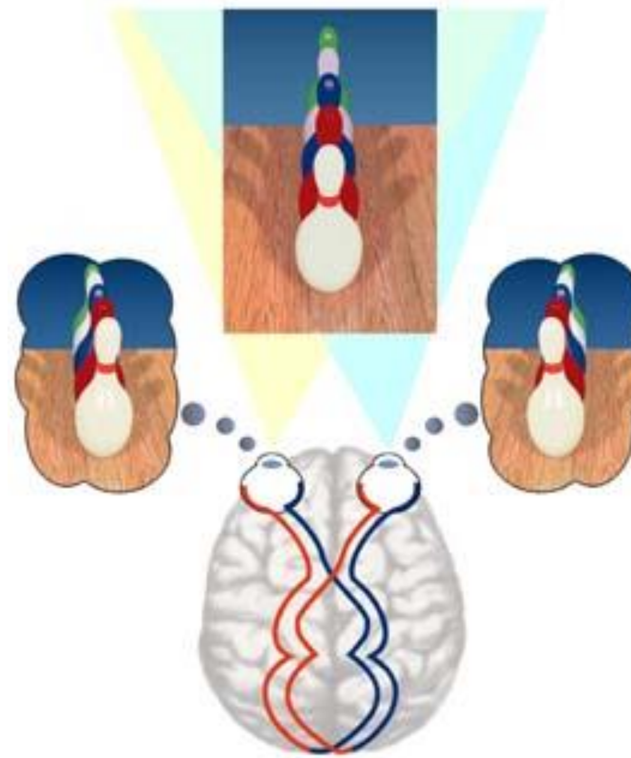
Human performance: up to a few meters



Slide adapted from Kostas Daniilidis

# Stereo Vision



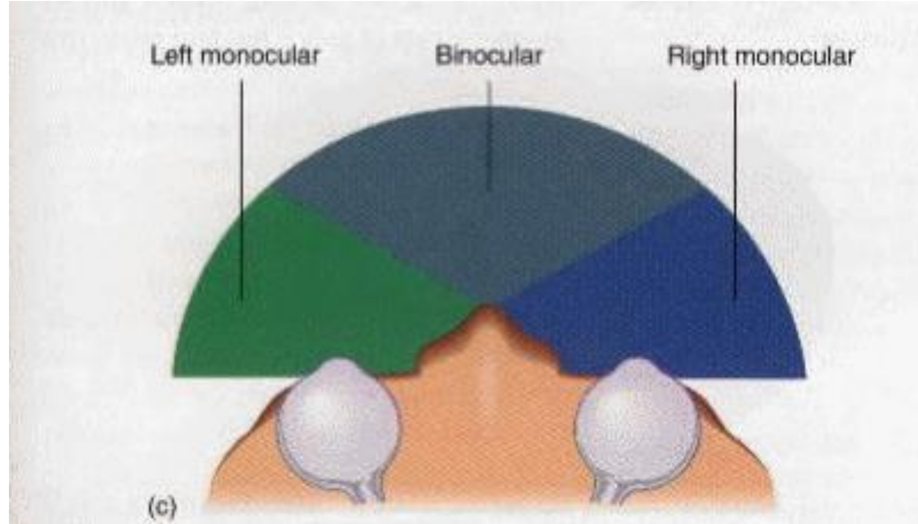



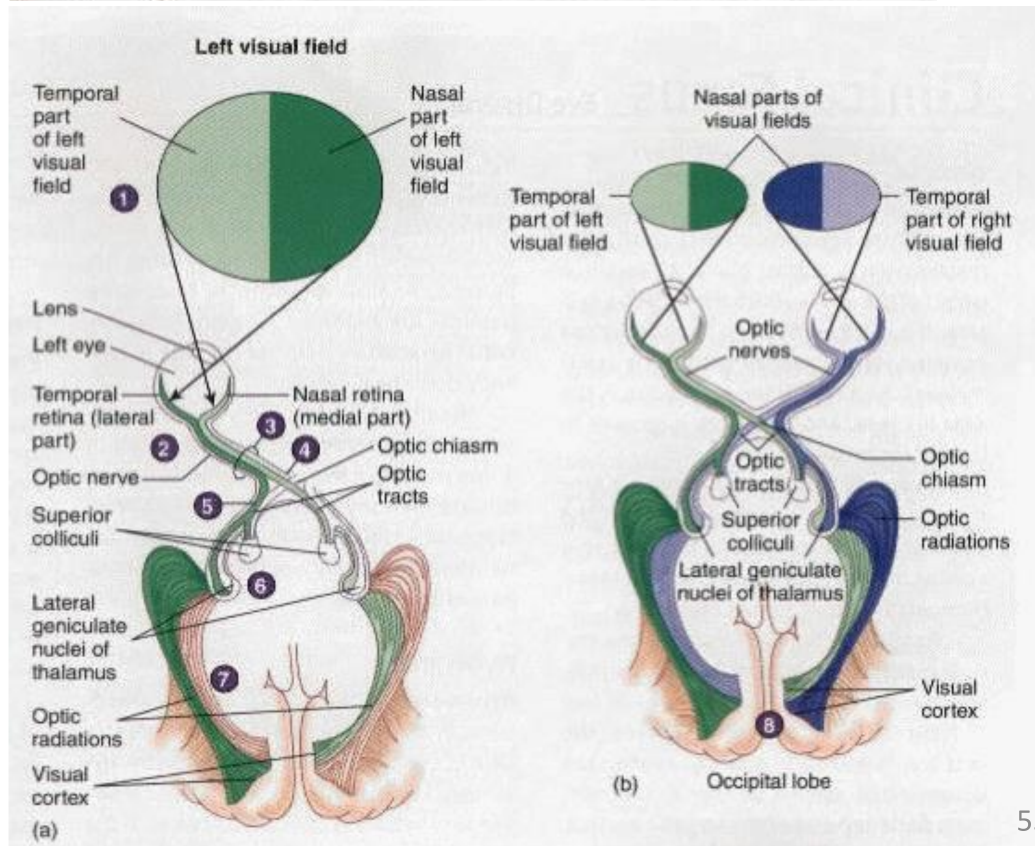


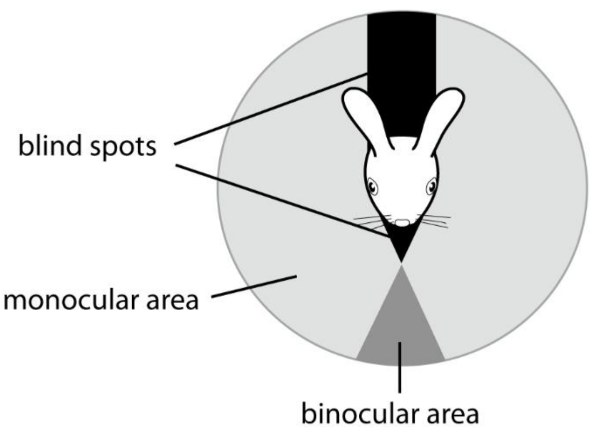
Slide adapted from Kostas Daniilidis

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

What happened to rabbit's vision?



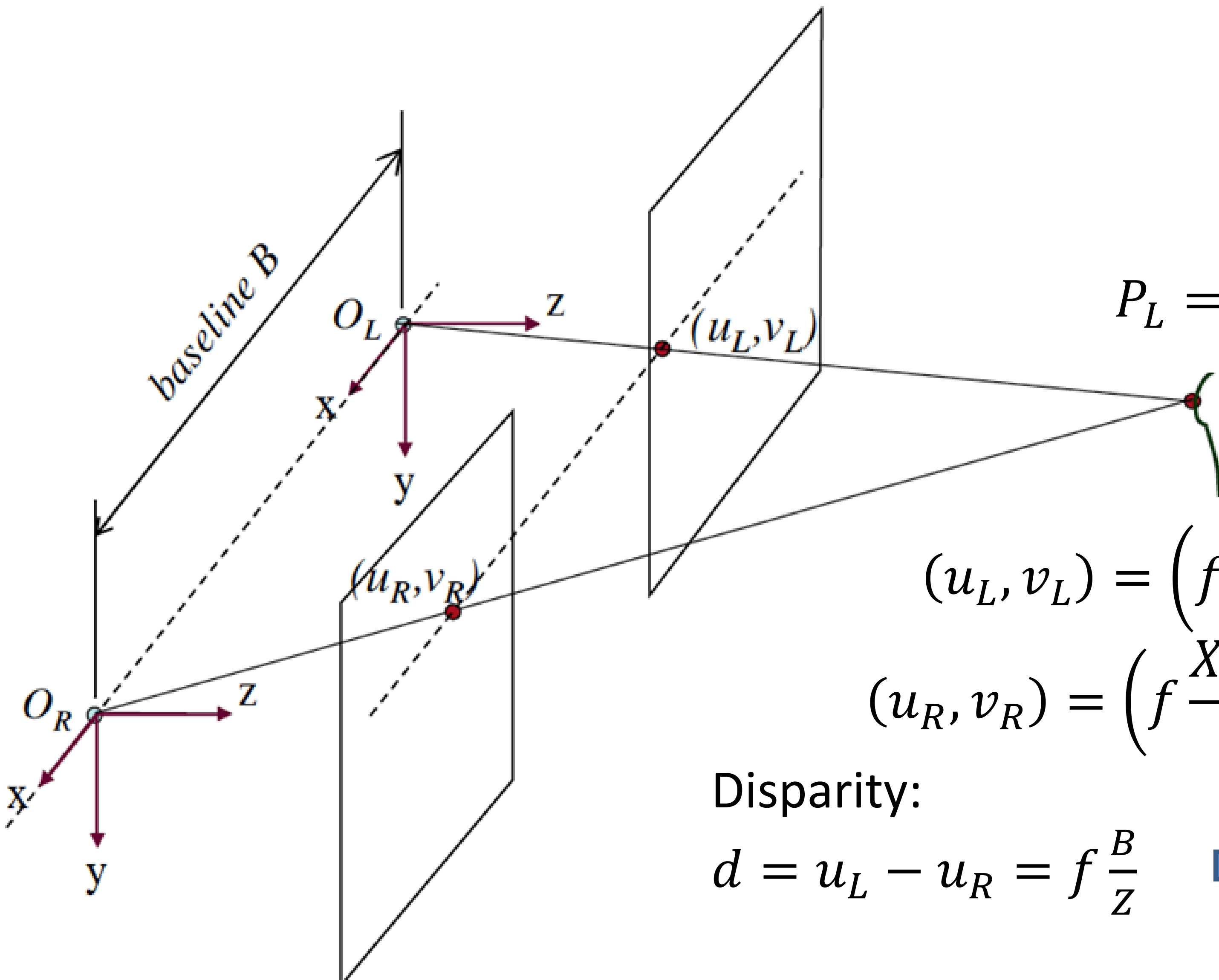






Slide adapted from Kostas Daniilidis

# Basic Stereo Derivations




$$P_L = (X, Y, Z)$$

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

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

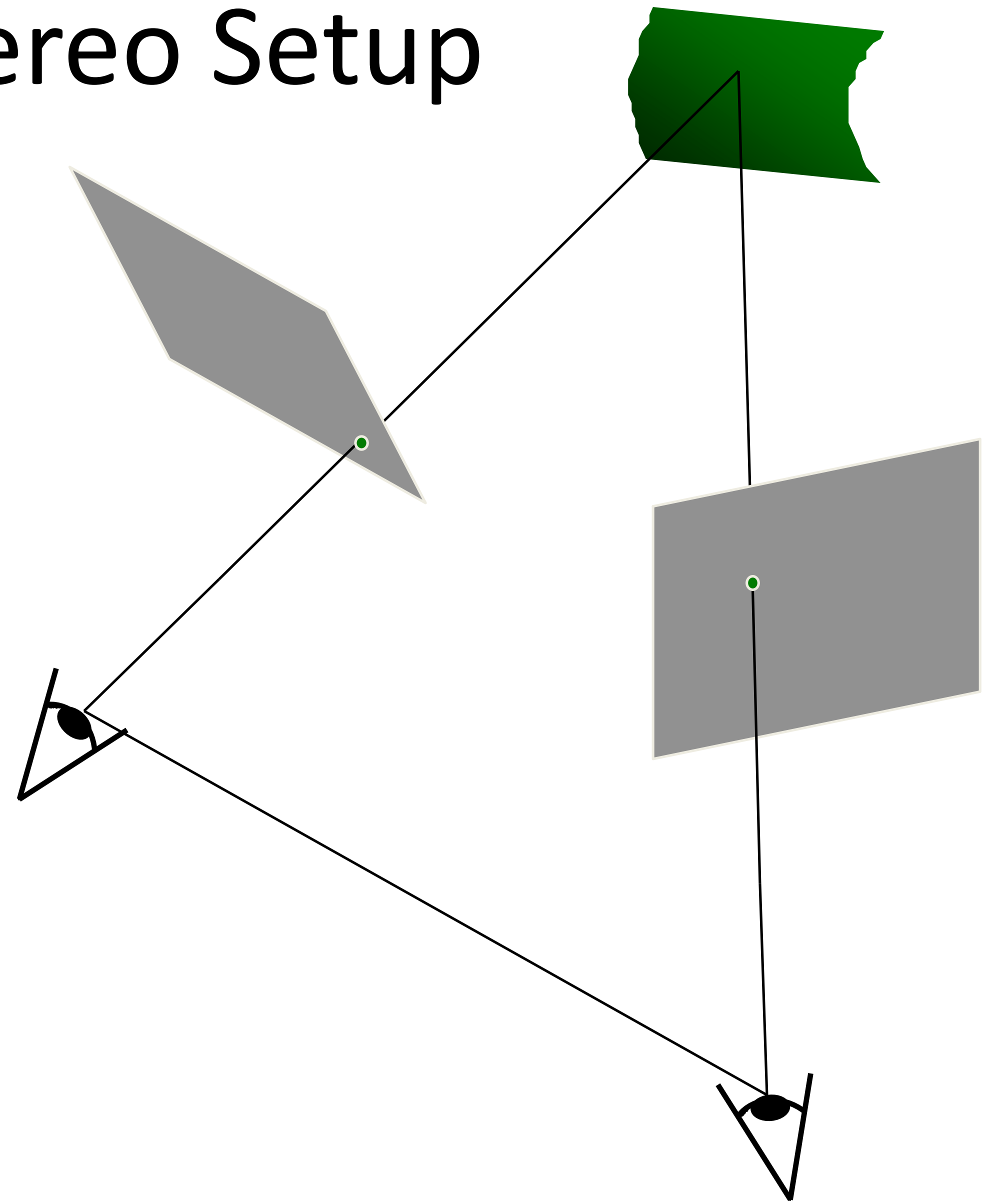
Disparity:

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



Slide adapted from Kostas Daniilidis

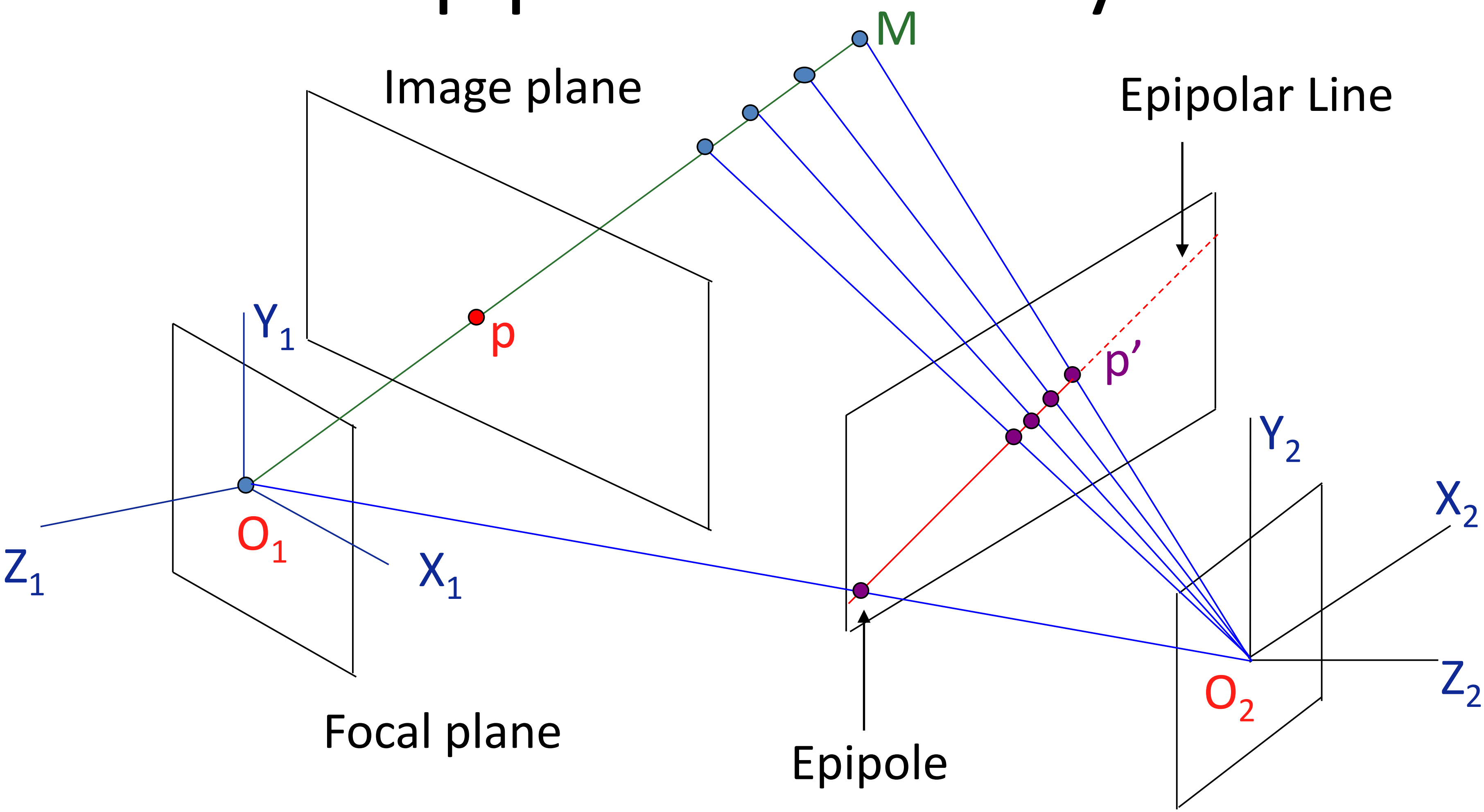
# General Stereo Setup





Slide adapted from Kostas Daniilidis

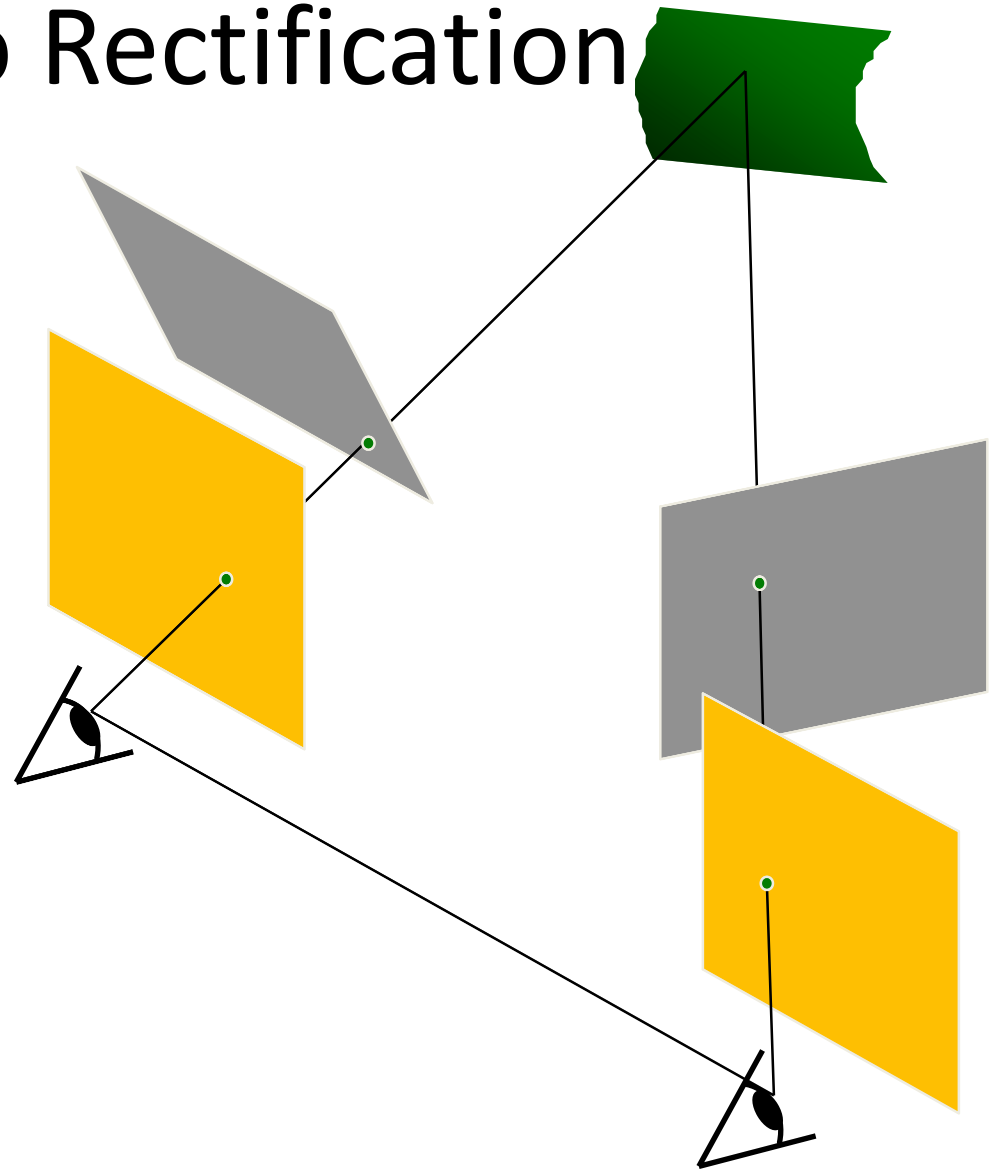
# Epipolar Geometry






Slide adapted from Kostas Daniilidis

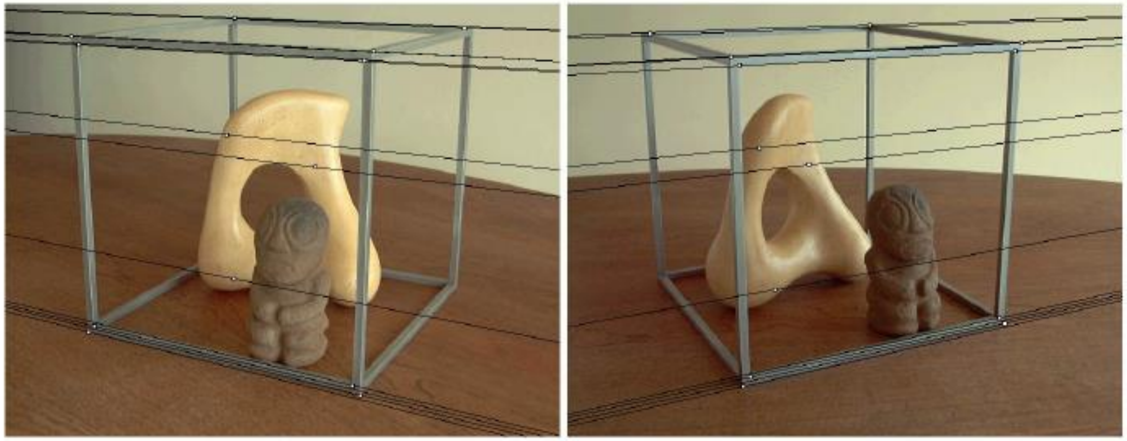
# Stereo Rectification





Slide adapted from Kostas Daniilidis

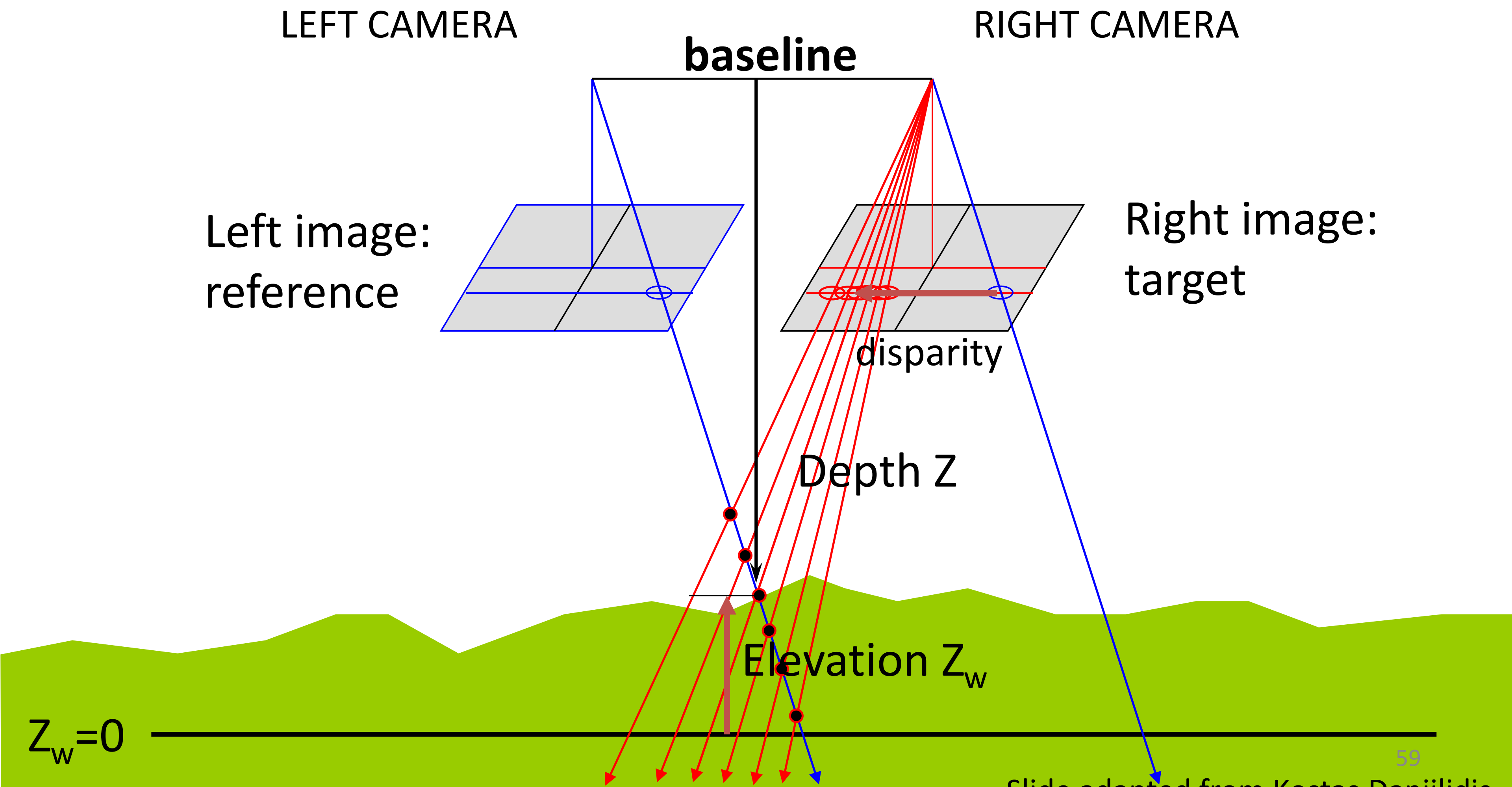
# Stereo Rectification





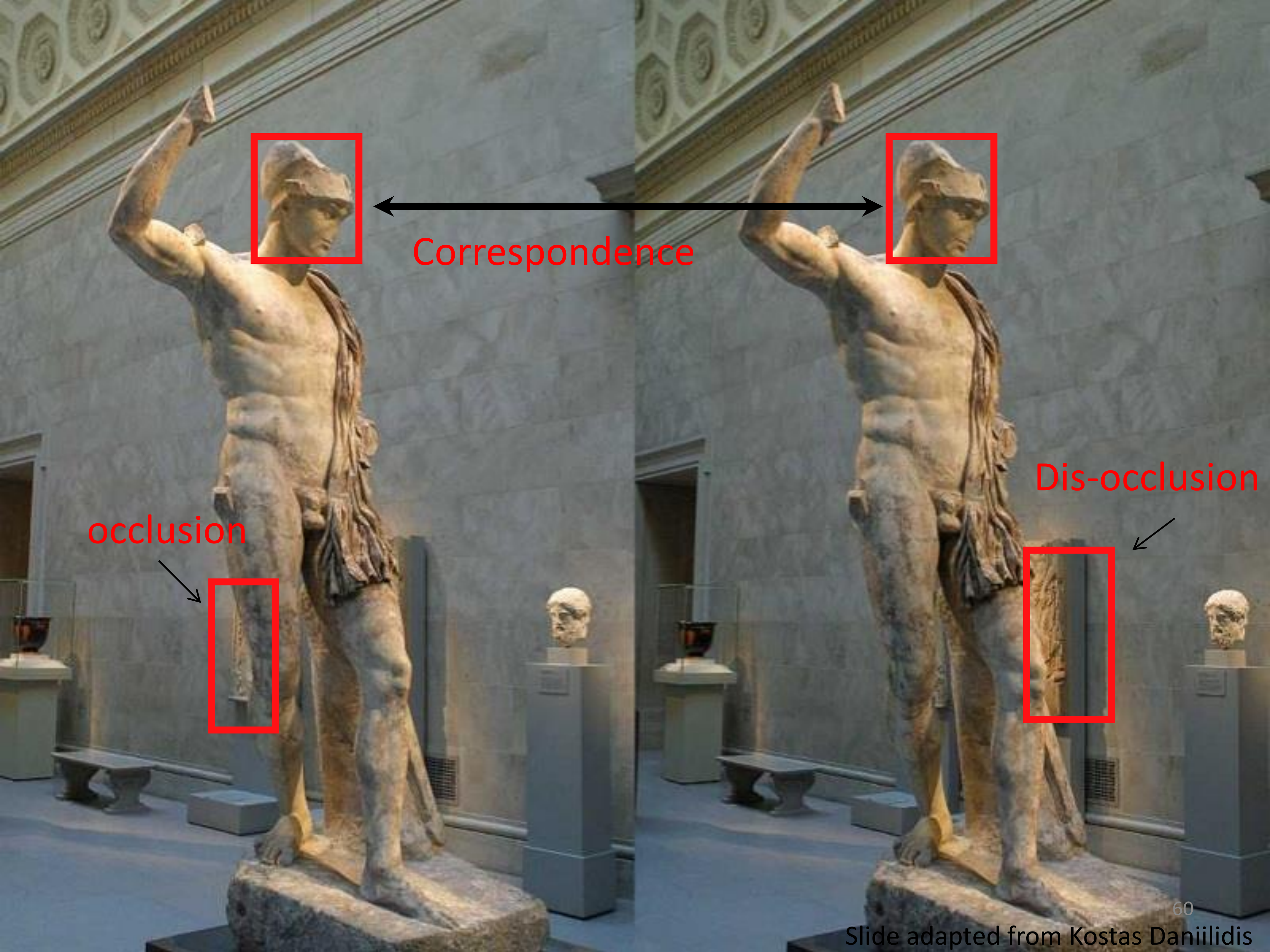
Slide adapted from Kostas Daniilidis

# A Simple Stereo System

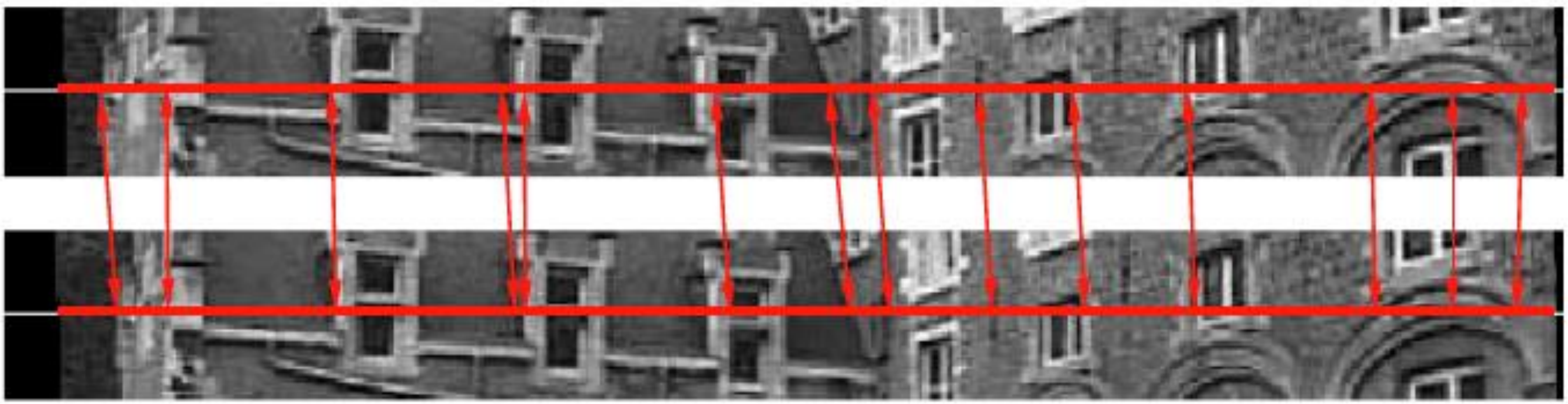





Slide adapted from Kostas Daniilidis


Correspondence
occlusion
Dis-occlusion
Slide adapted from Kostas Daniilidis

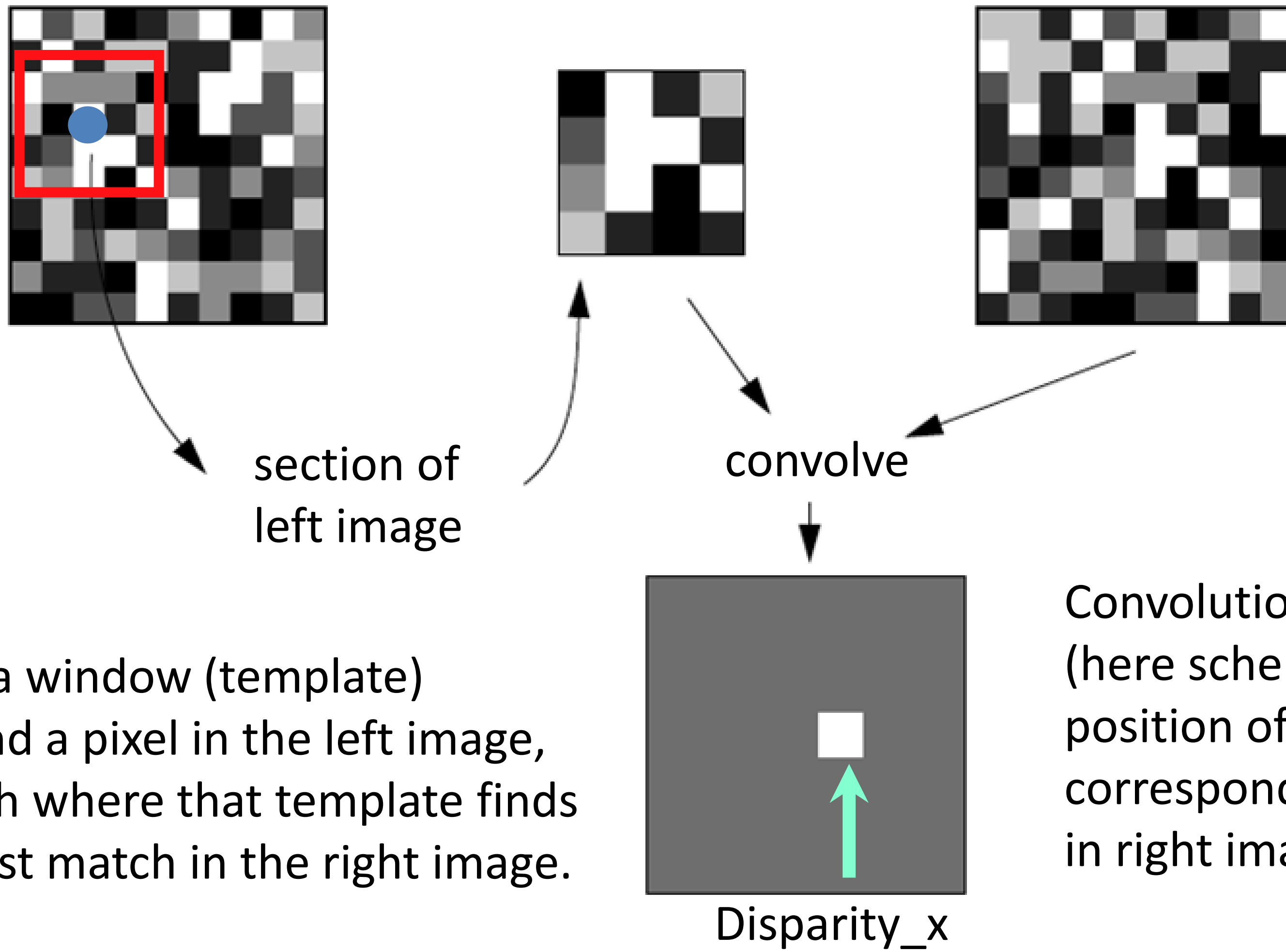
# Correspondence





Slide adapted 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.

Convolution peak (here schematic) at position of corresponding patch in right image



Slide adapted from Kostas Daniilidis

# Choice of similarity function for image patches

*Sum of squared differences*

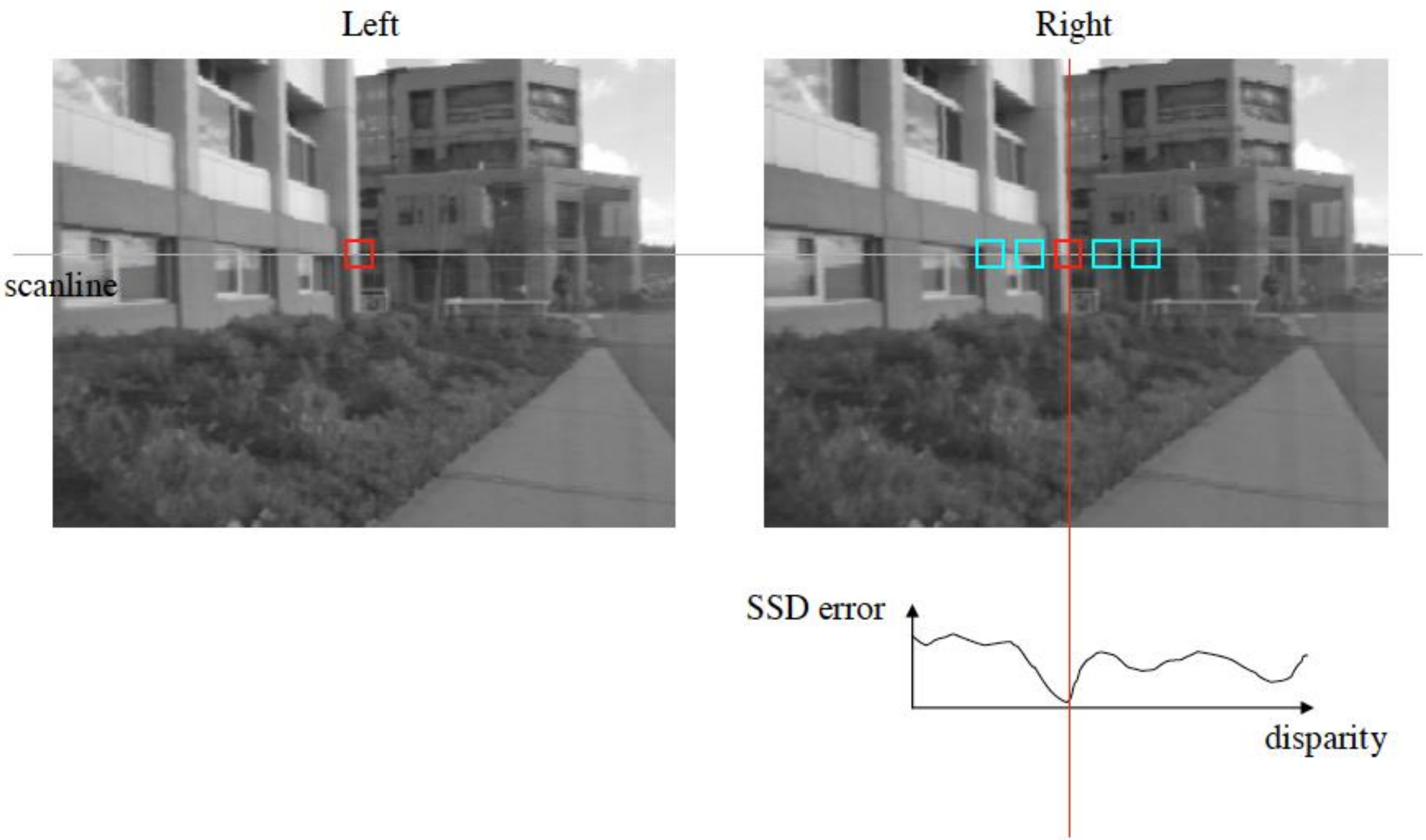
$$SSD(f,g) = \sum_{i,j} (f(i,j) - g(i,j))^2$$

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



Slide adapted from Kostas Daniilidis

# Correspondence Using Correlation



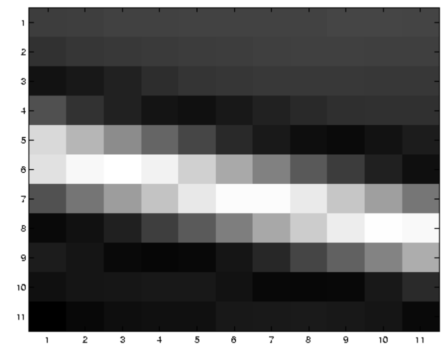


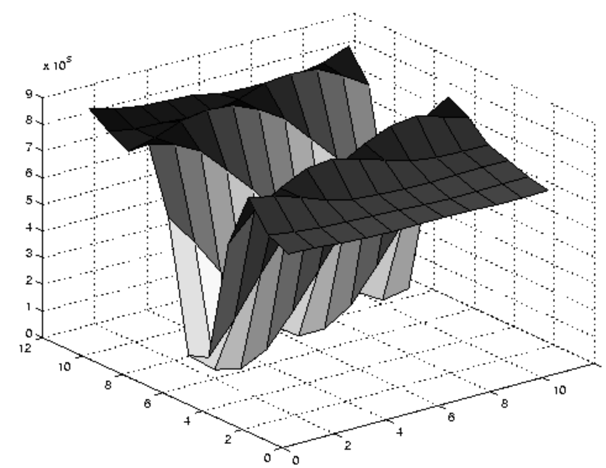

Slide adapted from Kostas Daniilidis

# Edge

*Sum of squared differences*


Err(x,y)



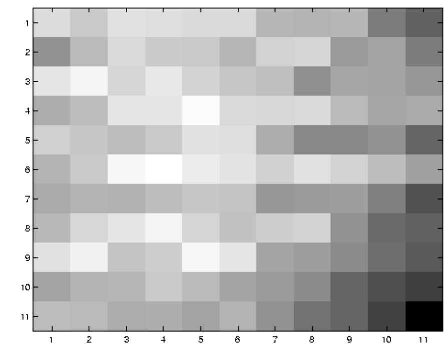


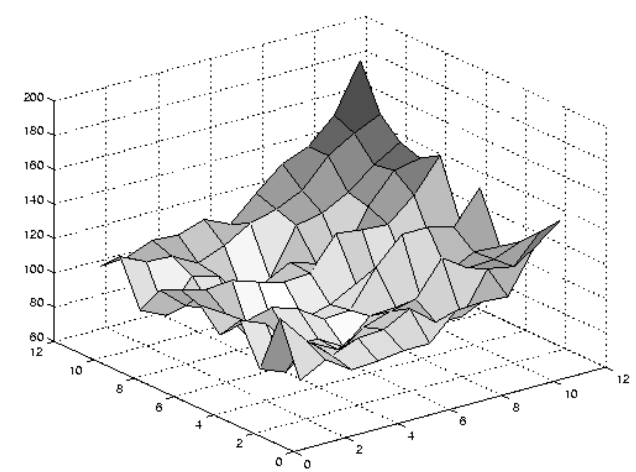

Slide adapted from Kostas Daniilidis

# Low texture region

*Sum of squared differences*

Err(x,y)

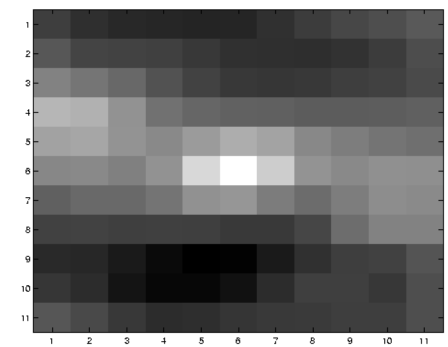


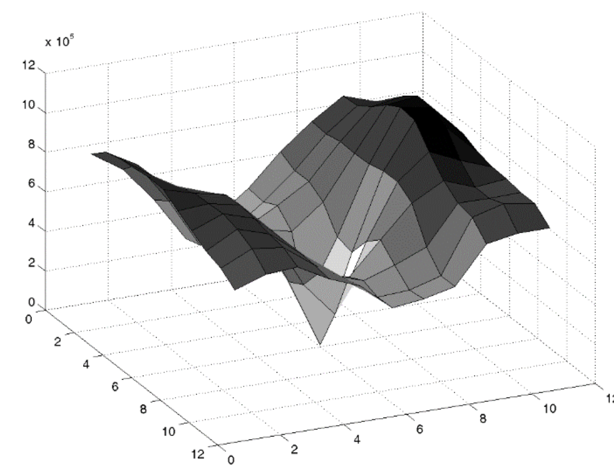

Slide adapted from Kostas Daniilidis

# High textured region

*Sum of squared differences*

Err(x,y)



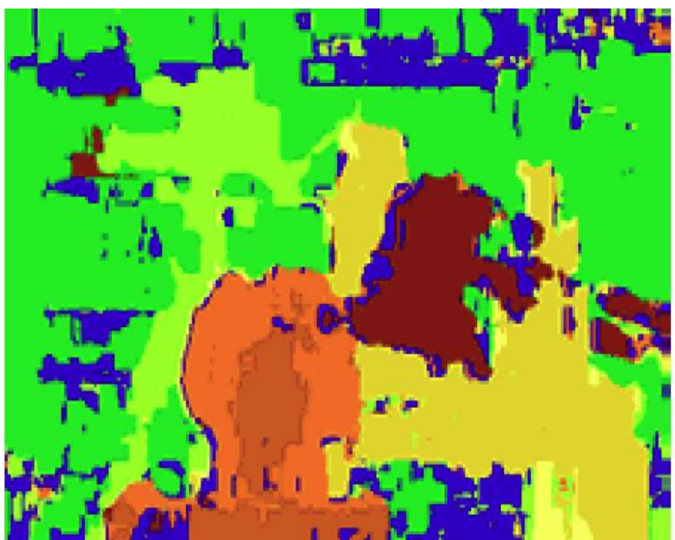



Slide adapted from Kostas Daniilidis

# Disparity computation using SSD

Scene

Ground truth





Slide adapted 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



Slide adapted from Kostas Daniilidis

# Census Measure

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

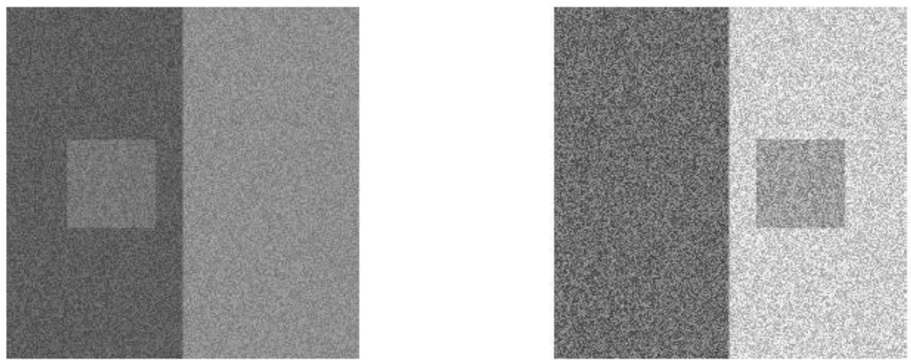


Fig. 2. Right and left random dot stereograms

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



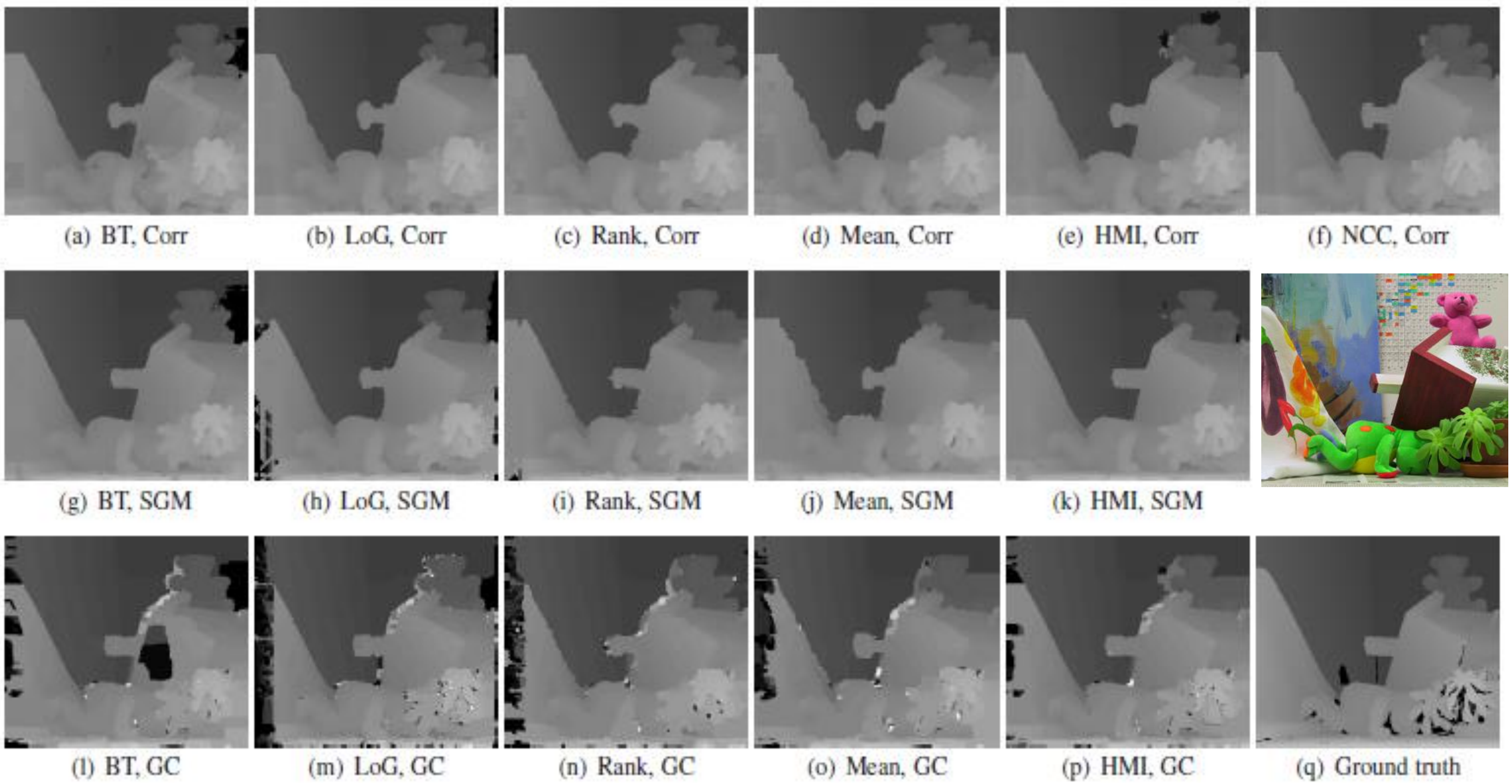
Slide adapted from Kostas Daniilidis

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



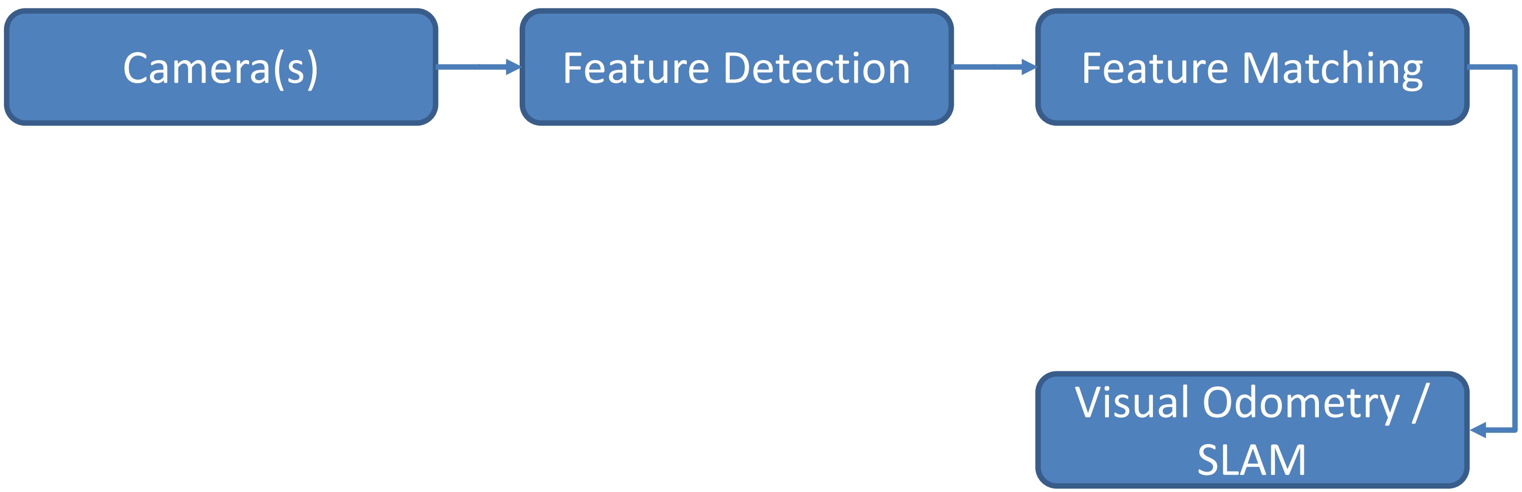
Slide adapted from Kostas Daniilidis

# Comparison of different similarity measures



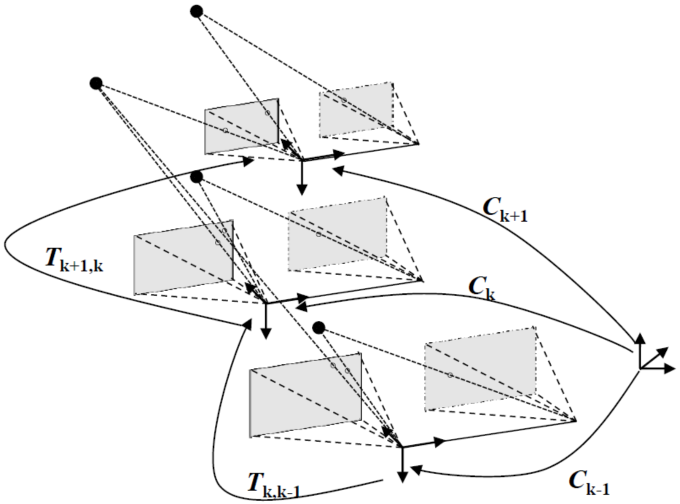
# Visual Odometry

# Vision-based Incremental Pose Estimation Pipeline (aka. Visual Odometry)



# Visual Odometry

- Visual odometry is the process of real-time estimation of incremental motion of the camera (sensor suite) using only sequential images as input
- Analogy to odometer on cars

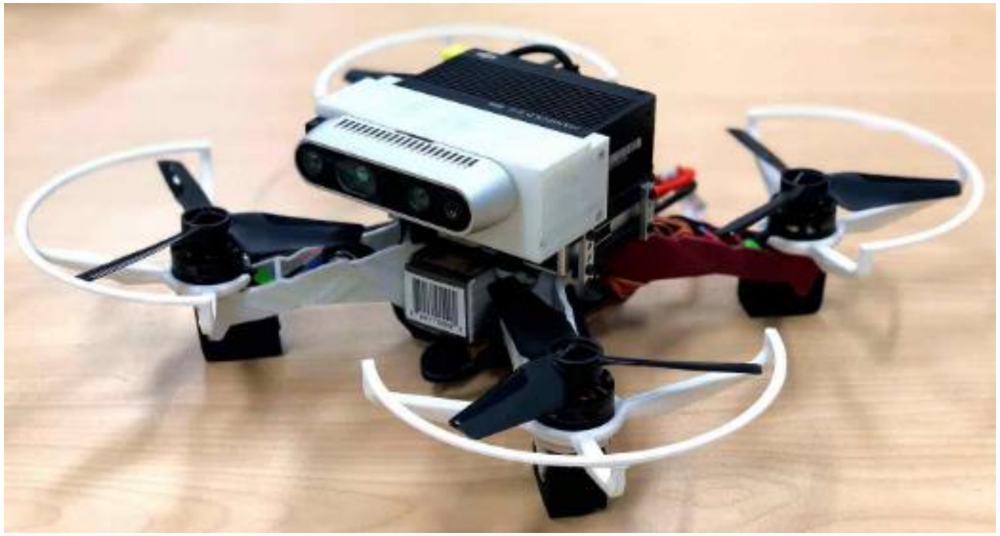



# Visual Odometry v.s. Map-based Localization

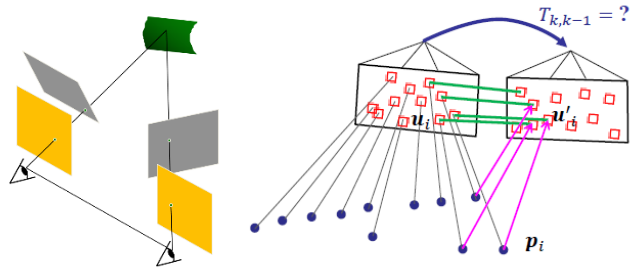
- VO Setup
  - Applicable to different camera configurations (monocular, stereo, etc.)
  - Sufficient illumination and texture
  - Dominance of static scene
  - Unknown environment
  - Sufficient overlapping between consecutive frames
  - Focus on local consistency
- Localization Setup
  - Applicable to different camera configurations (monocular, stereo, etc.)
  - Sufficient illumination and texture
  - Dominance of static scene
  - Known map
  - Sufficient observation of map features
  - Focus on global consistency

# Stereo Visual Odometry

- Setup
  - Known stereo intrinsic and extrinsic calibration
  - Rectified stereo image pairs
  - Set starting point of the dataset as the origin
  - Estimate camera movement with respect to the origin



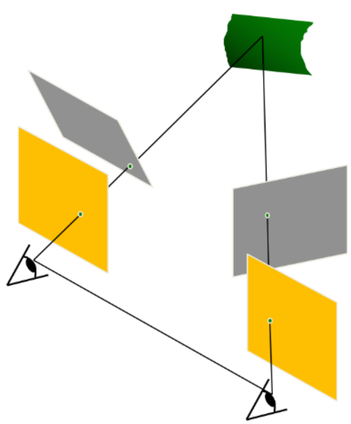
# Stereo Visual Odometry

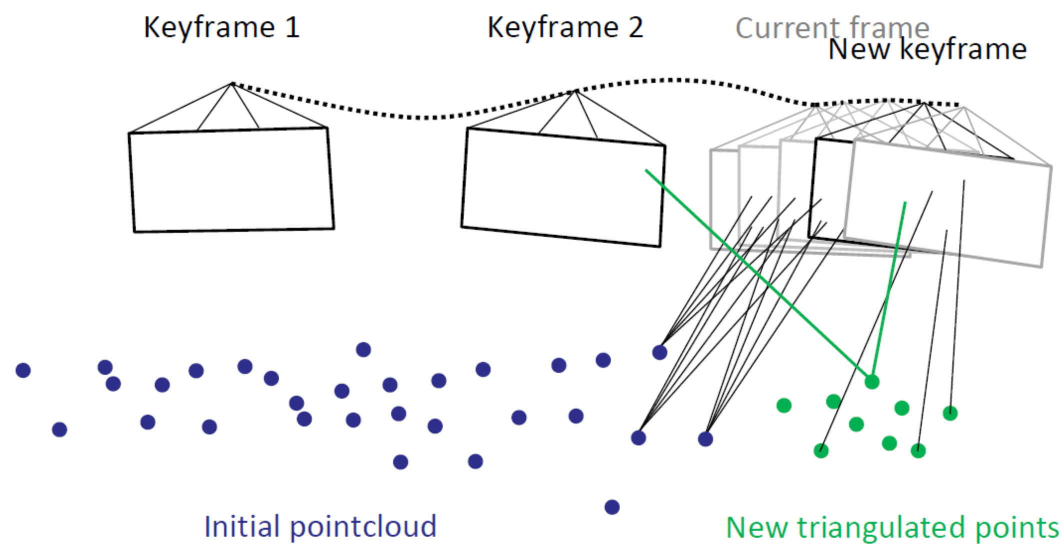



- Frame-to-frame stereo visual odometry
  1. Input Frame1 (two images)
  2. Detect 2D features in Frame1
  3. Compute depth of each 2D feature in Frame1 using feature matching or optical flow between left and right images
  4. Input Frame2 (two images)
  5. Detect 2D features in Frame2
  6. Compute the incremental pose displacement between Frame1 and Frame2 using 2D-3D pose estimation
  7. Accumulate incremental pose displacement
  8. Set Frame2 as Frame1, goto Step 3
- Question: How to set initial values?
- Question: What are the disadvantages of frame-to-frame setup?

# Stereo Visual Odometry

- Keyframe-based stereo visual odometry
  - No pose drift when there is no keyframe change
  - Only initiate new keyframe when:
    - Displacement between the current frame and the latest keyframe is large
    - Number of features between the current frame and the latest keyframe is insufficient
  - Question: Can we do even better?

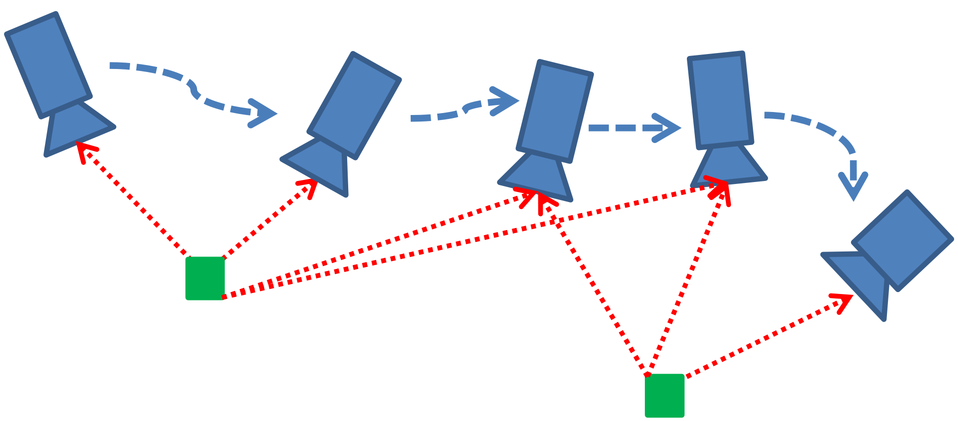




香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
電子及計算機工程學系
DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING

# More on Visual Odometry

- Sliding window visual odometry
- Sliding window visual-inertial odometry
- Full visual SLAM
- Full visual-inertial SLAM
- …
- To be covered in Lecture 10



# Logistics

- Project 2, phase 2 is released (03/23)
  - You have a lot of time to finish it: 04/13