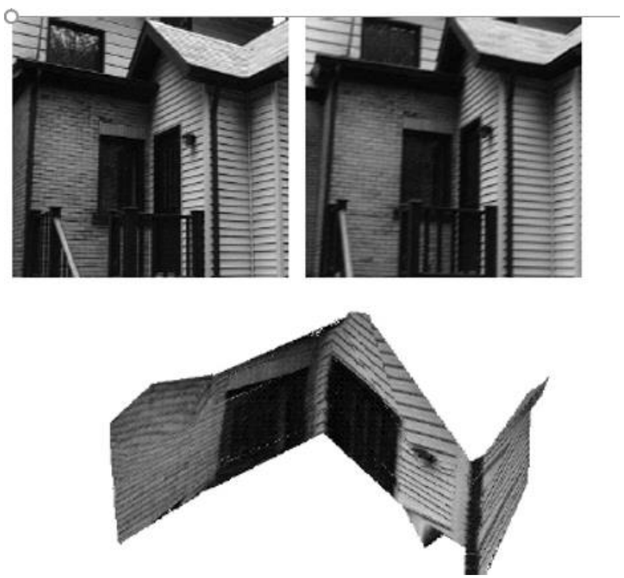
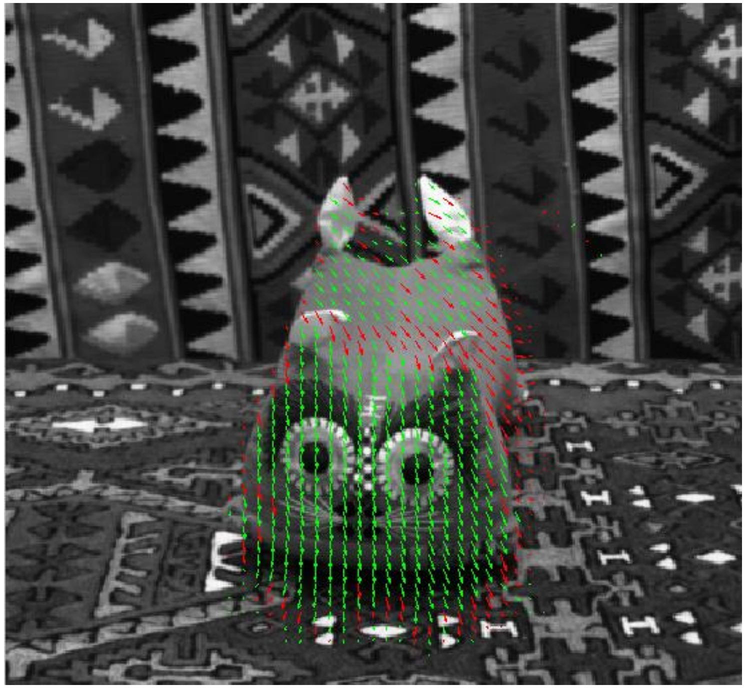
Motion
Class 7

SfM



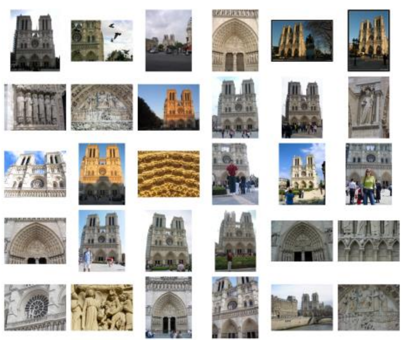
Motion



Structure from Motion (SFM)

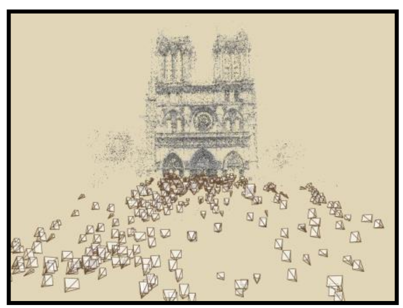
- Assumptions:
  - A set of images
  - Uncalibrated
- Applications:
  - Recover the camera locations
  - Photo Tourism
  - Improve robustness

Application: Photo Tourism Overview



Input photographs

Scene reconstruction



Relative camera positions and orientations

Point cloud

Sparse correspondence

Photo Explorer

Photo Tourism: Exploring Photo Collections in 3D,
Snavely, Seitz, Szeliski

Slide by Seitz

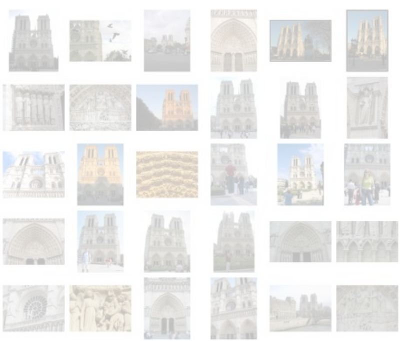
SFM

- Given a set of m images with n corresponding points $\{\tilde{p}_{ij}\}$:
  - $\{M_i\}$ the set of unknown camera projections
  - $\{P_i\}$ the set of unknown 3D points
  - $\tilde{p}_{ij} = M_i \tilde{P}_j$
- Goal: find $\{M_i\}$ and $\{P_j\}$

SFM - Observations

- $\tilde{p}_{ij} = M_i \tilde{P}_j$
- Given $\{P_j\}$ we can compute $\{M_i\}$
- Given $\{M_i\}$ we can compute $\{P_j\}$
- Ambiguity: if $\{P_j\}$ and $\{M_i\}$ is a solution, then for any full rank 4×4 array A, $\{A^{-1}P_j\}$ and $\{M_i A\}$ are also a solution
  - $\tilde{p}_{ij} = M_i \tilde{P}_j = M_i A A^{-1} \tilde{P}_j$

Photo Tourism Overview



Input photographs

Scene reconstruction

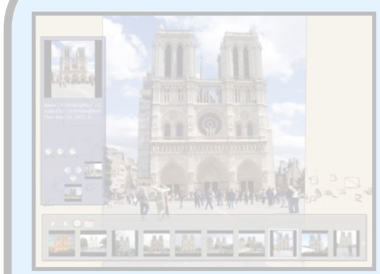
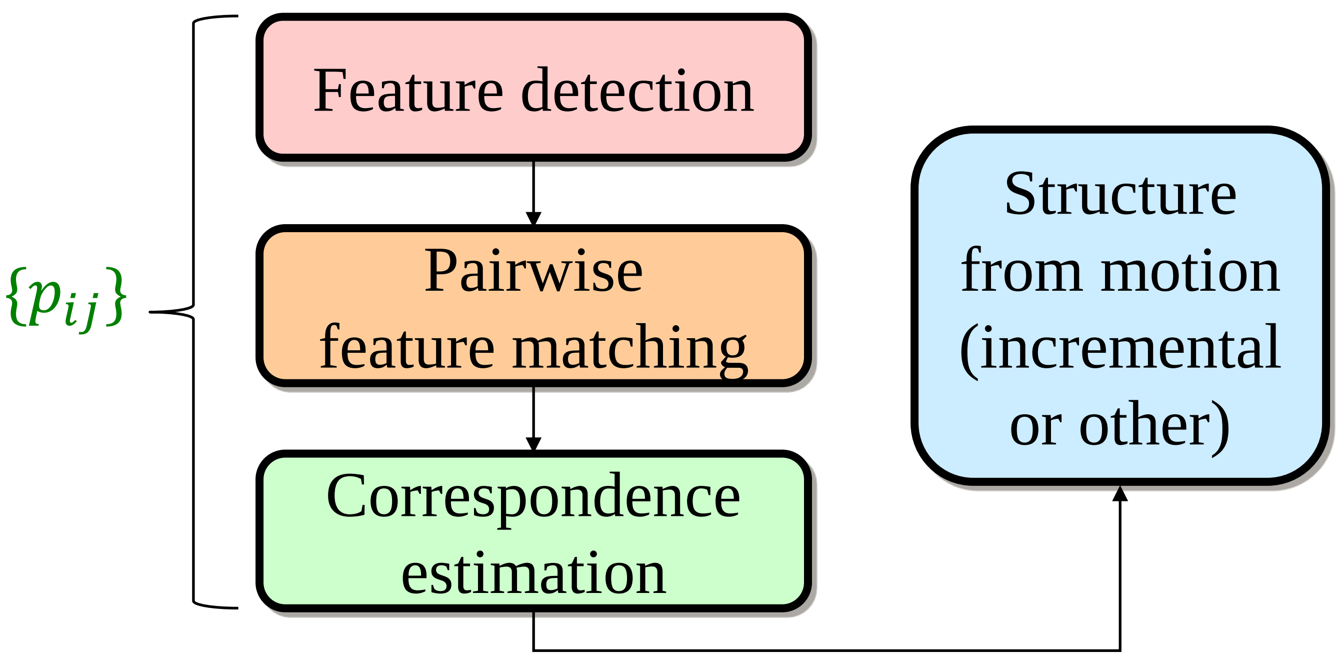


Photo Explorer

Slide by Seitz

Scene Reconstruction

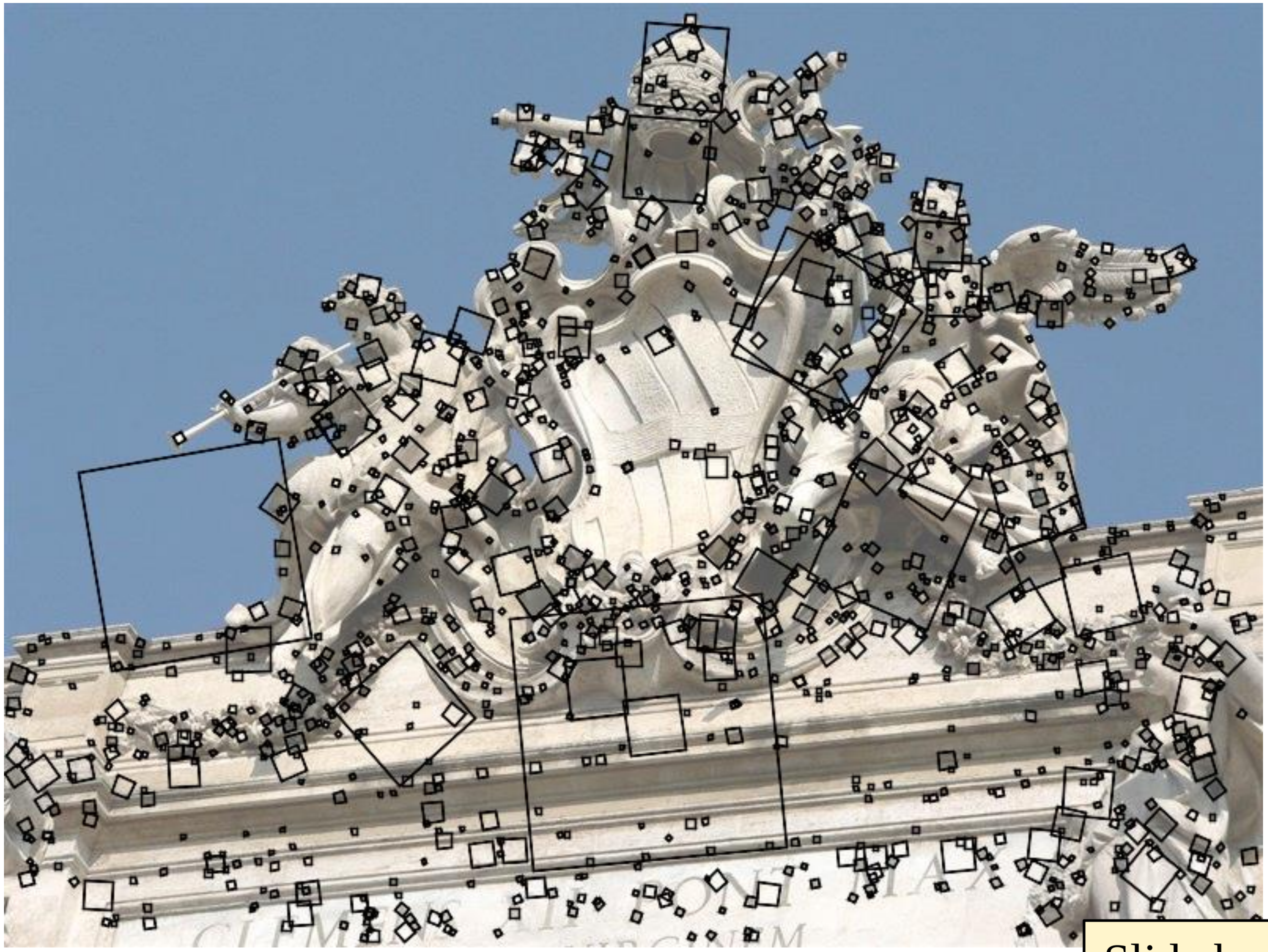
- Estimate $\{M_i\}$ and $\{P_i\}$



Slide by Seitz

Feature Detection

Detect features using SIFT [Lowe, IJCV 2004]

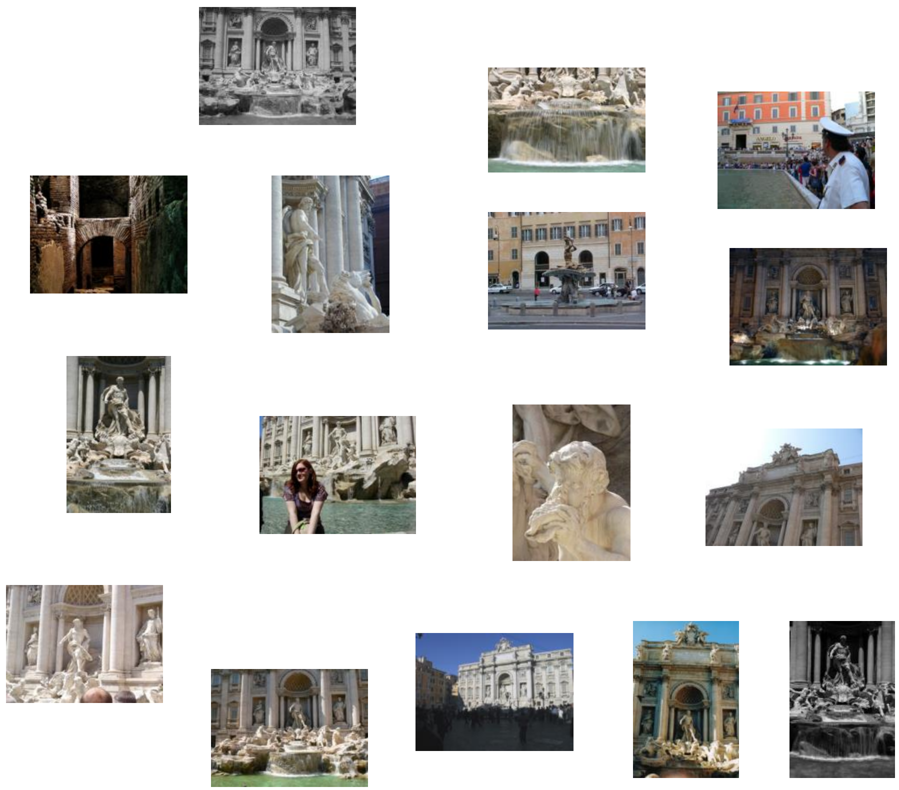


Slide by Seitz

הבינתחומי הרצליה
IDC HERZLIYA

Feature Detection

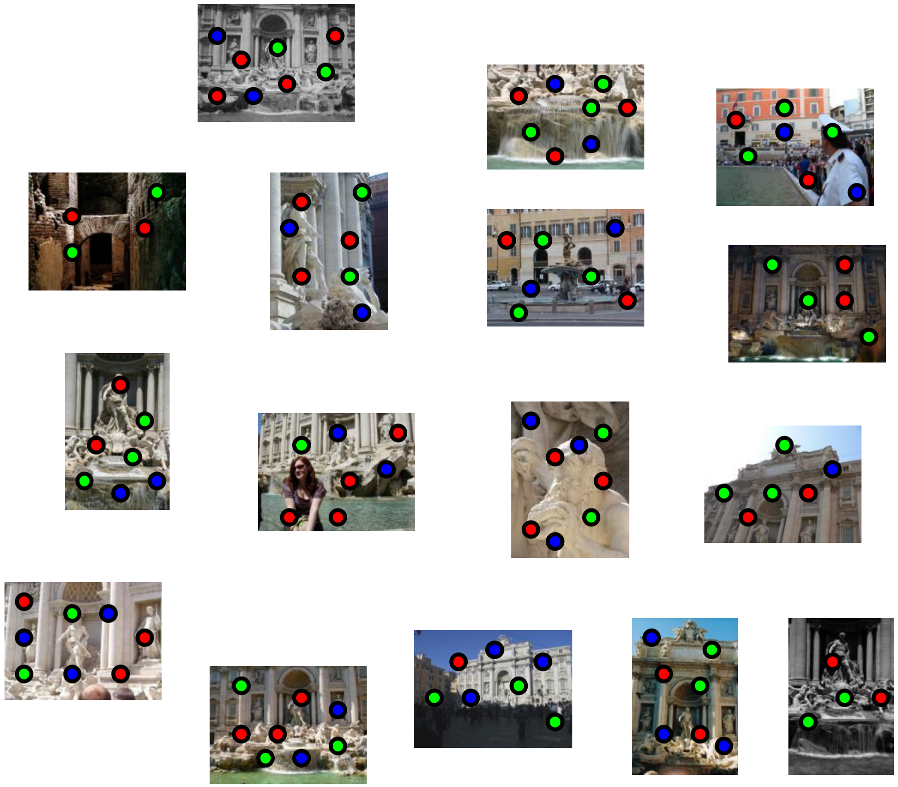
Detect features using SIFT [Lowe, IJCV 2004]



Slide by Seitz

Feature Detection

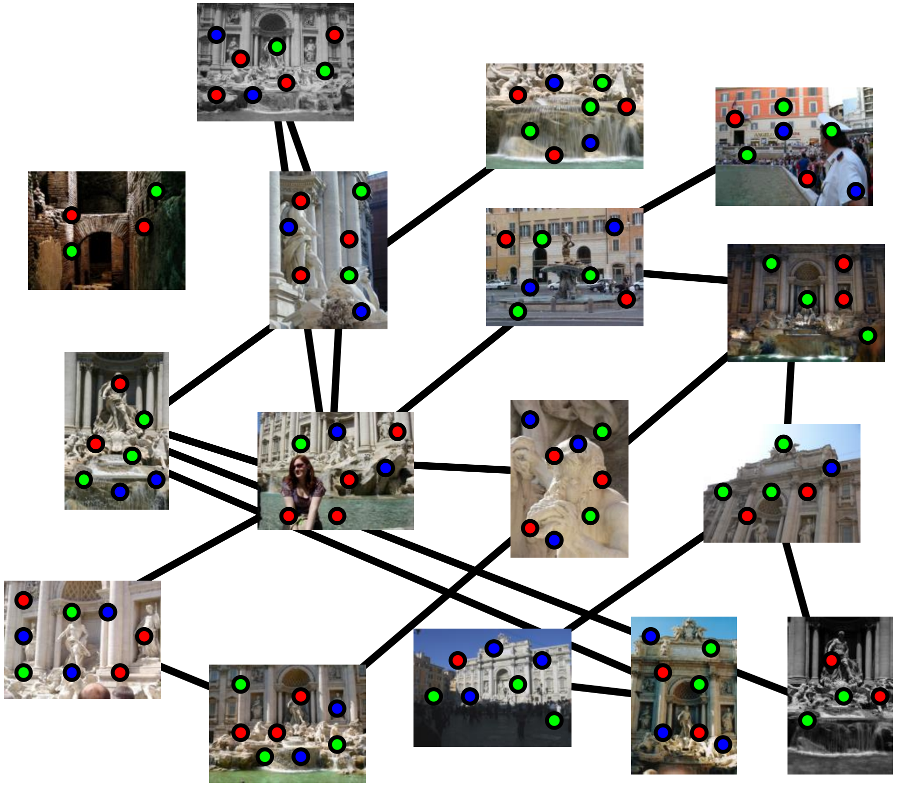
Detect features using SIFT [Lowe, IJCV 2004]



Slide by Seitz

Feature Matching

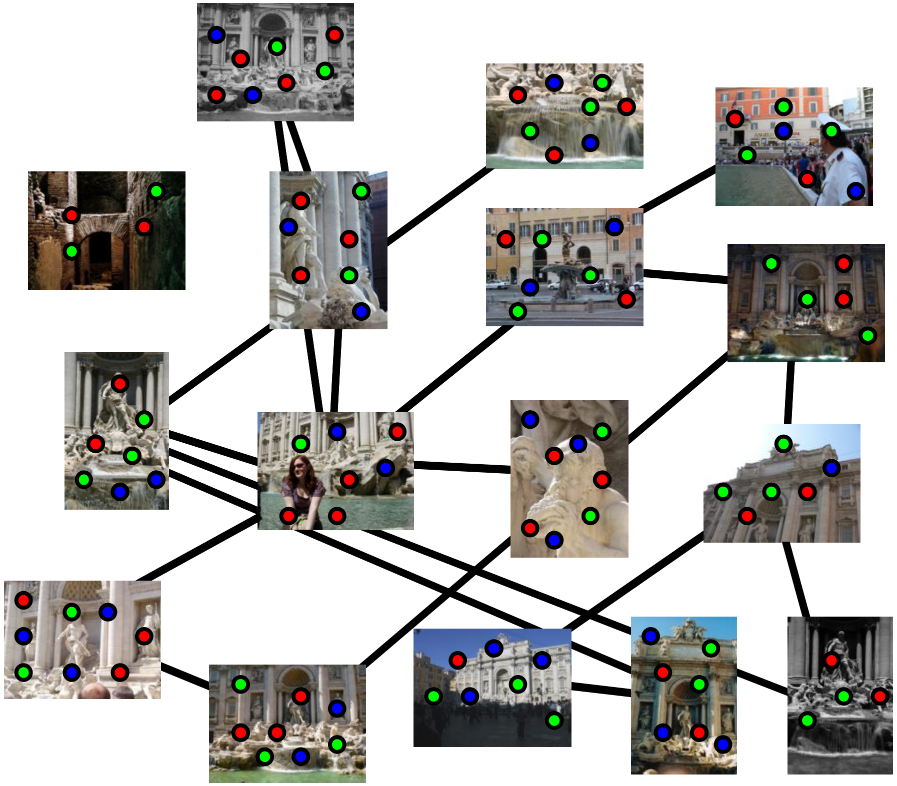
Match features between each pair of images



Slide by Seitz

Feature matching

Refine matching using RANSAC [Fischler & Bolles 1987] to estimate fundamental matrices between pairs



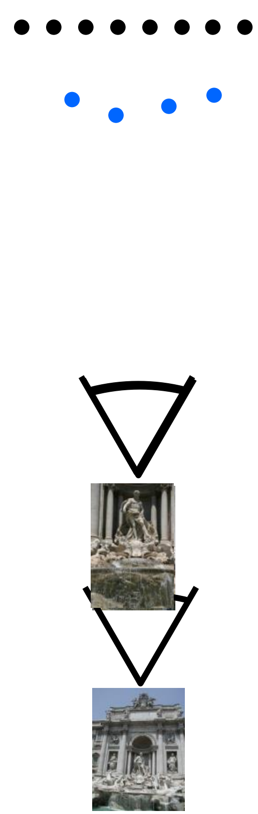
Slide by Seitz

Incremental SFM

Trevi Fountain,
Rome

Given $\{P_i\}$ compute $\{M_i\}$
Given $\{M_i\}$ compute $\{P_i\}$

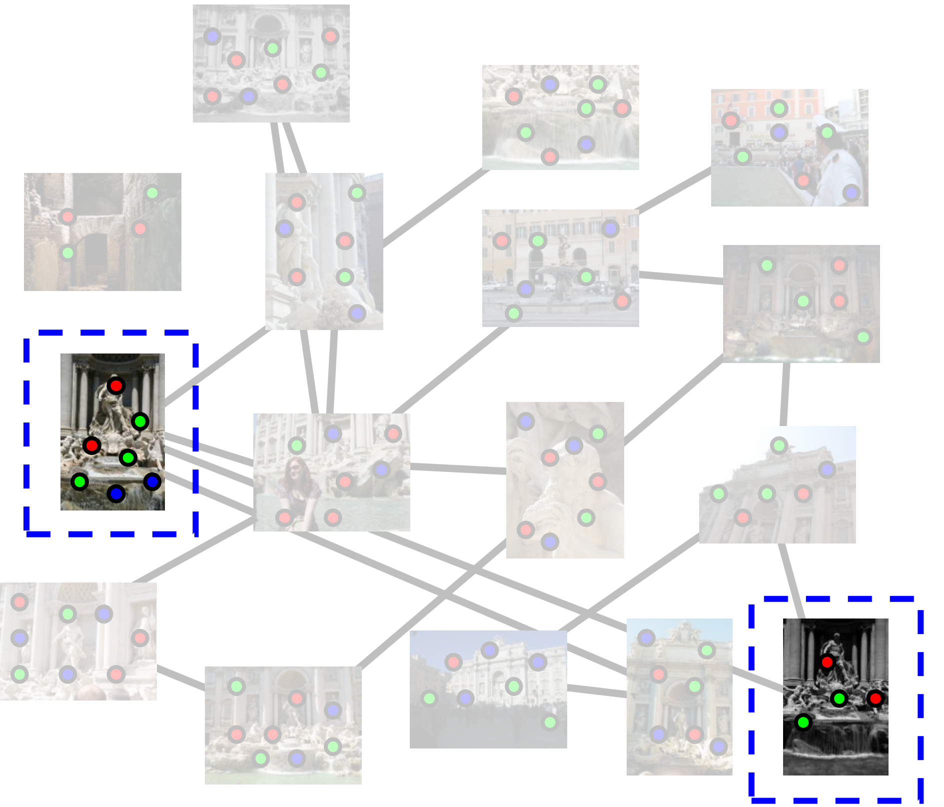
Incremental SFM



Given $\{P_i\}$ compute $\{M_i\}$

Given $\{M_i\}$ compute $\{P_i\}$

Perspective Projection: Incremental SFM



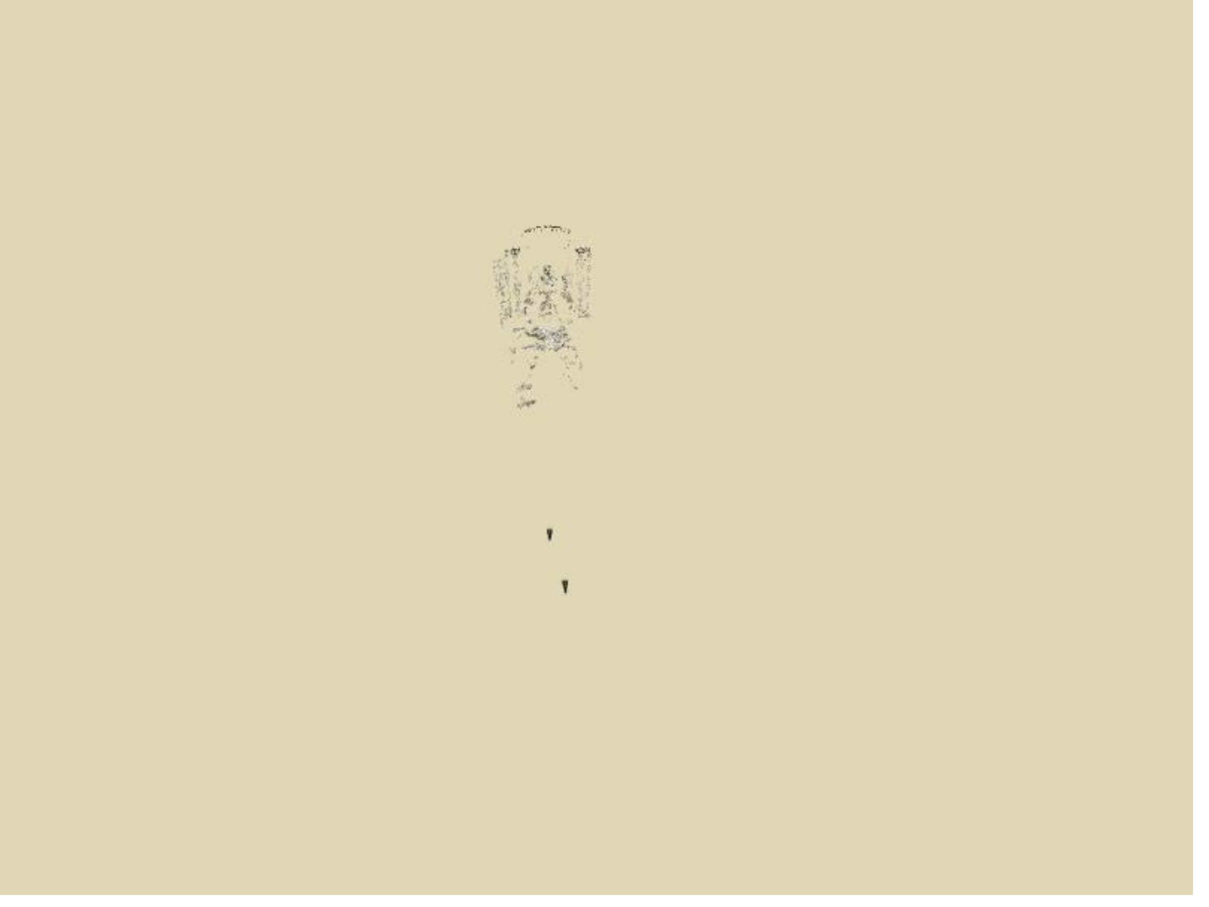
Given $\{P_i\}$ compute $\{M_i\}$

Given $\{M_i\}$ compute $\{P_i\}$

Incremental SFM

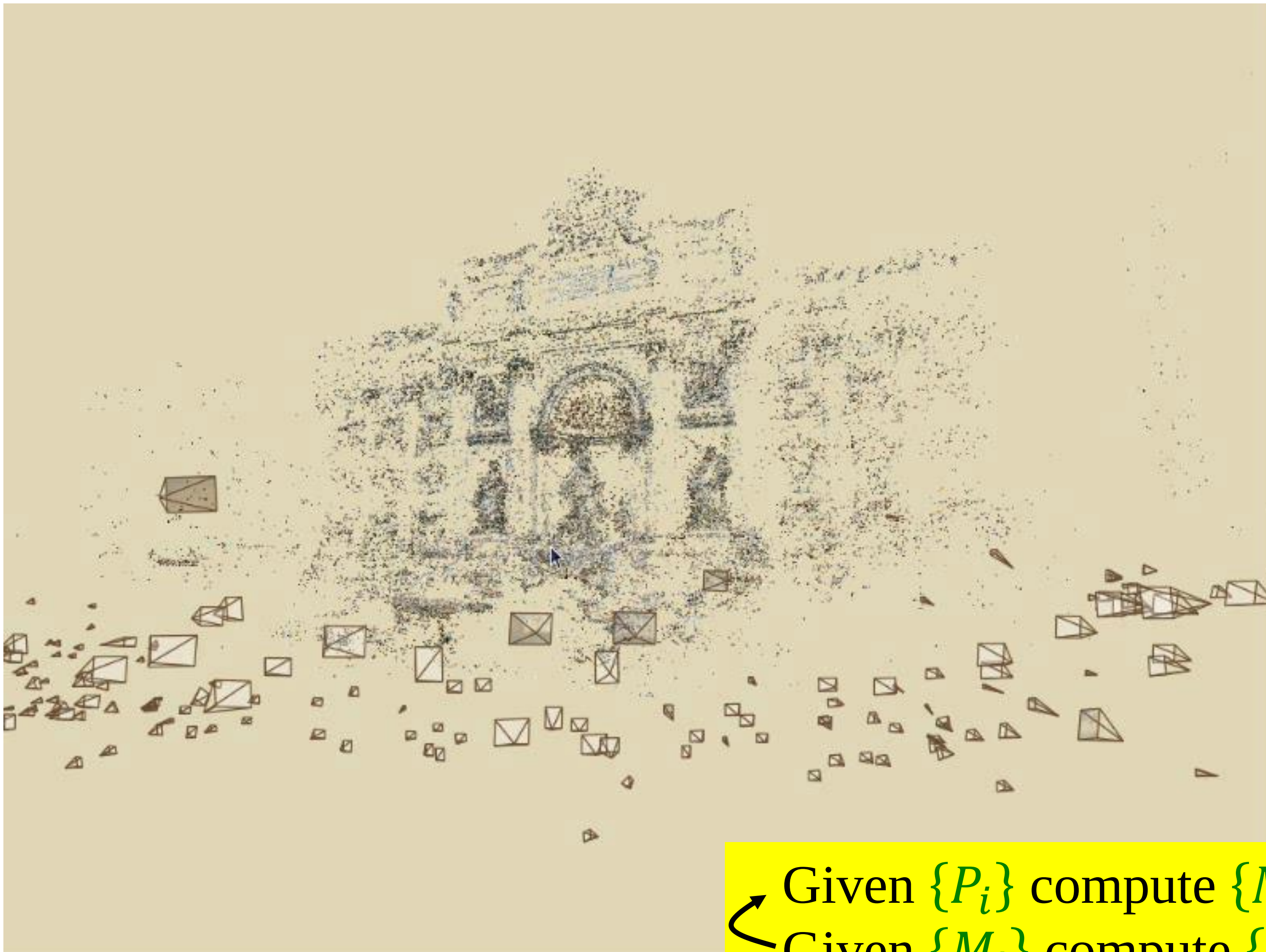
Given $\{P_i\}$ compute $\{M_i\}$
Given $\{M_i\}$ compute $\{P_i\}$

Incremental SFM



Given $\{P_i\}$ compute $\{M_i\}$
Given $\{M_i\}$ compute $\{P_i\}$

Photo Explorer



Given $\{P_i\}$ compute $\{M_i\}$

Given $\{M_i\}$ compute $\{P_i\}$

Challenges & Limitations

- A heuristic algorithm- not necessarily an optimal solution
- Which order to use the images?
- How it affects the results?
- Efficiency

Affine Structure from Motion

The Problem: Reconstruct scene geometry and camera parameters from two or more images

Assumptions:

- Known correspondence
- Orthographic projection

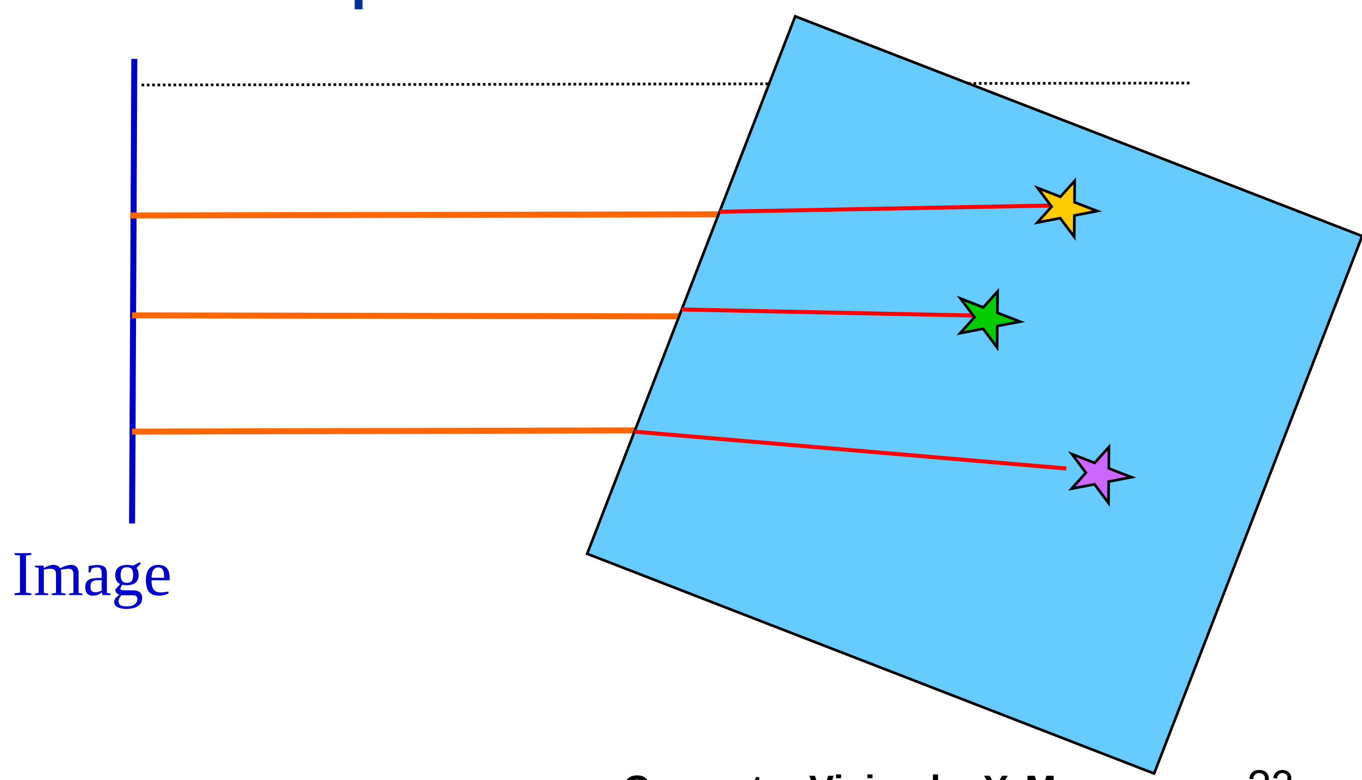
Only approximation of perspective

Advantage: Closed form solution

Affine SfM

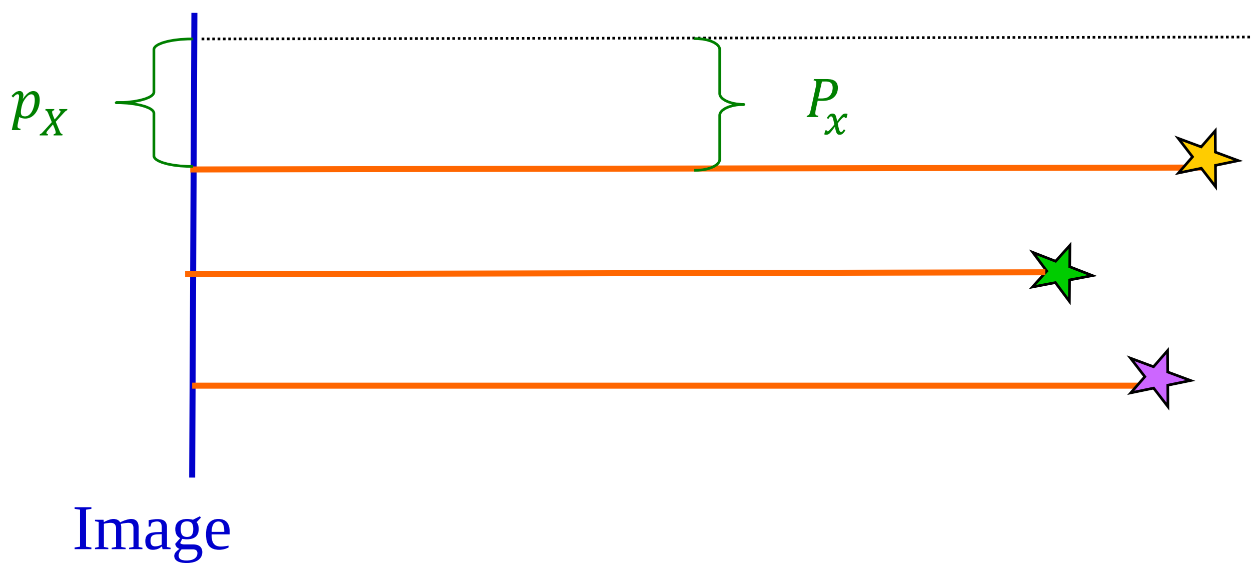
Orthographic Projection

When the object is far relative to its local depth



Orthographic Projection

When the object is far
relative to its local depth



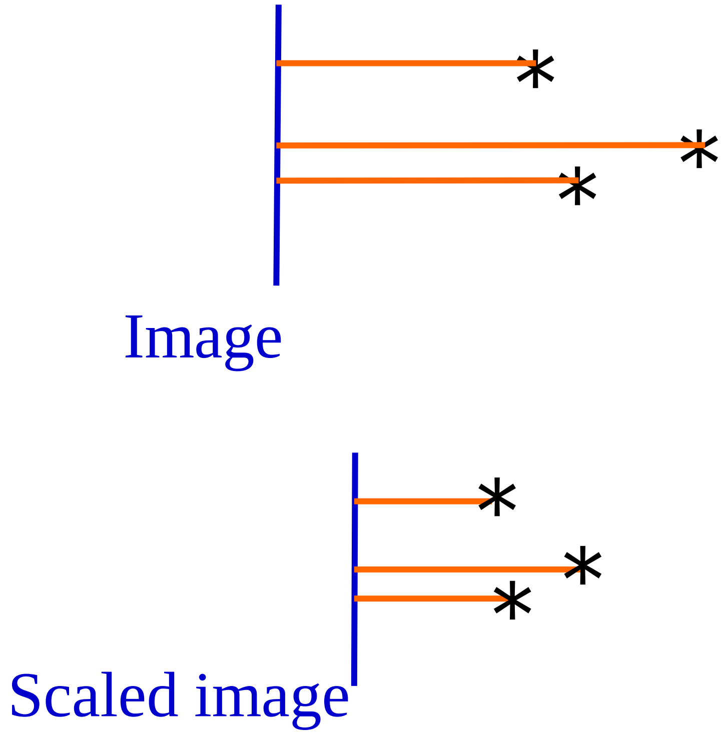
$$p_x = P_x$$
$$p_y = P_y$$

Paraperspective Projection

Orthographic + scaling s

$$p_x = sP_x$$
$$p_y = sP_y$$

s is fixed for the whole image



Orthographic+ Scale Projection

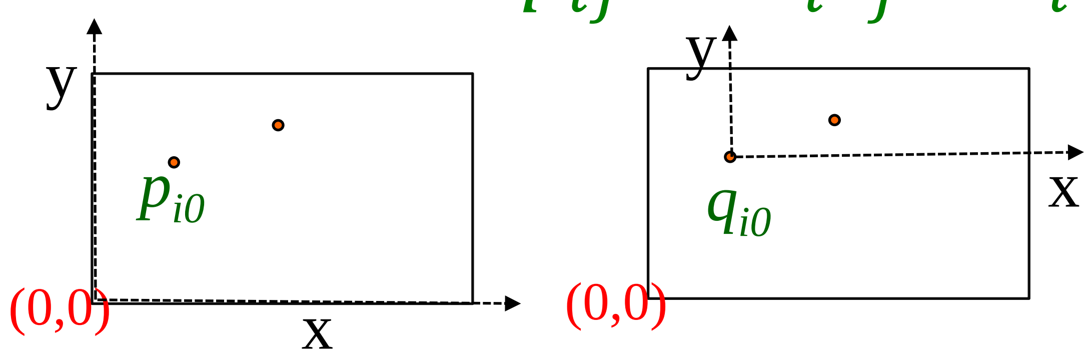
- Let P be an object point and p be an image point then $p = proj\big(sR(P-T)\big)$, where
  - R is a 3x3 rotation matrix
  - s is a scale factor
  - T is a translation vector
  - $M = proj(sR)$
- In this case $p = MP - b$, where *b=proj(RT)*
  - *M* is a *2x3* matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \Rightarrow M = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \end{pmatrix}$$

Get Rid from Translation

- Let p_{i0} be the projection of the object point P_0 in image i
- Translate all image points of image i :
  $q_{ij} \rightarrow p_{ij} - p_{i0}$
- We now have $q_{ij} = M_j P_j$

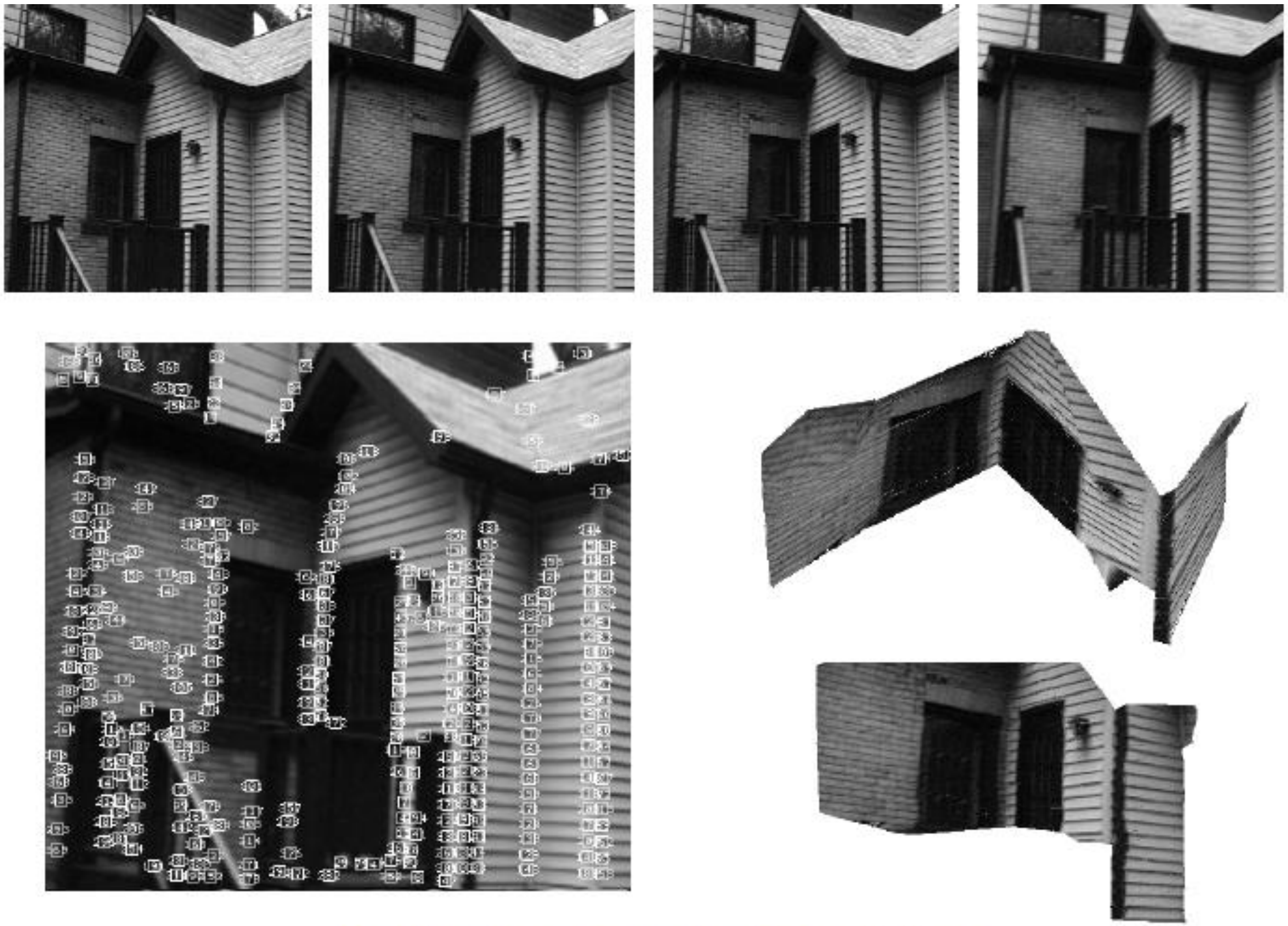
  instead of $p_{ij} = M_i P_j + b_i$



Tomasi - Kanade

- A factorization algorithm to solve SFM
- Based on SVD

Reference:
- Forsyth and Ponce book: Section 14.3
- Carlo Tomasi and Takeo Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, 9(2):137-154, November 1992.



Reprinted from Tomasi and Kanade 1992

Set Up

- *n* unknown object points, $\{P_j\}$
  - *3n* unknowns
- *m* unknown cameras, $\{M_i\}$
  - *6m* unknowns
- *mn* known images of the object points

$$p_{ij} = M_i P_j$$

  - *2mn* equations – non-linear!

Ambiguity

- Let M_i and P_j be a solution to $q_{ij} = M_i P_j$
- Let A be a non singular *3x3* matrix (affine transformation)
- $M_i A$ and $A^{-1} P_j$ is also a solution

$$q_{ij} = M_i P_j = (M_i A)(A^{-1} P_j)$$

הבינתחומי הרצליה
IDC HERZLIYA

Using Matrix Notation

- Consider the set of projection of a point P_j into *m* images: $q_{ij} = M_i P_j$

$$Q_j = \hat{M} P_j$$

$$\underbrace{\overset{Q_j}{\begin{pmatrix} \boxed{\phantom{q_{2j}}} \\ \boxed{q_{2j}} \\ \\ \boxed{q_{mj}} \end{pmatrix}}}_{2m \times 1} = \underbrace{\overset{\hat{M}}{\begin{pmatrix} \boxed{M_1} \\ \boxed{M_2} \\ \\ \boxed{M_m} \end{pmatrix}}}_{2m \times 3} P_j$$

Consider n points P_i

$$Q_j = \hat{M} P_j \quad \longrightarrow \quad \hat{Q} = \hat{M}\hat{P}$$

where

$$\underbrace{\begin{pmatrix} \boxed{Q_1} & \boxed{Q_2} & \cdots & \boxed{Q_n} \end{pmatrix}}_{\substack{\hat{Q} \\ 2m \times n}} = \underbrace{\hat{M}}_{2m \times 3} \underbrace{\begin{pmatrix} \boxed{P_1} & \boxed{P_2} & \cdots & \boxed{P_n} \end{pmatrix}}_{\substack{\hat{P} \\ 3 \times n}}$$

Ambiguity $\quad \hat{Q} = (\hat{M} A)(A^{-1} \hat{P})$

SVD: Singular Value Decomposition

Let A be an $k \times l$ matrix with $A=UDV^T$ where:

- U is an $k \times k$ column orthonormal matrix

  $U^T U=I$

- D is a diagonal matrix, $k \times l$
- V is an $l \times l$ orthonormal matrix

  $V^T V=I$

SVD: properties

- The columns of U, are the eigenvectors of AA^T
- The columns of V , are the eigenvectors of A^TA
- The elements of D, are the square roots of A^TA eigenvalues (singular values)
  - Which are the same as the square roots of and AA^T
- The singular values appear in decreasing order

Now What?

$$\hat{Q} = M\hat{P}$$

- What is the max rank of Q?
- What is the max rank of U, D, V ?

$$\underset{2m \times n}{\begin{pmatrix} \boxed{Q_1} & \boxed{Q_2} & \cdots & \boxed{Q_n} \end{pmatrix}} = \underset{2m \times 3}{\hat{M}} \underset{3 \times n}{\begin{pmatrix} \boxed{P_1} & \boxed{P_2} & \cdots & \boxed{P_n} \end{pmatrix}} = UDV^T$$

$$= \underset{\mathrm{U}}{\begin{pmatrix} \boxed{U_3} & \quad \end{pmatrix}} \underset{\mathrm{D}}{\begin{pmatrix} \lambda_1 & 0 & 0 & 0.. \\ 0 & \lambda_2 & 0 & 0 \ldots \\ 0 & 0 & \lambda_3 & 0.. \\ 0 & 0 & 0 & 0 \ldots \end{pmatrix}} \underset{V^T}{\begin{pmatrix} \boxed{V_3^T} \\ \quad \end{pmatrix}}$$

- U col. eig. vec. AA^T
- V col. eig. vec. A^TA
- D: sqrt eig. Val A^TA

The Algorithm

$$\hat{Q} = M\hat{P}$$

- Compute SVD: $Q=UDV^T$
- Compute U_3, V_3 and D_3
- $M_0=U_3$ and $P_0= D_3 V_3^T$
- M_0 is the camera motion estimation
  (6m parameters)
- P_0 is the 3D point estimation
  (3n parameters)

$$\underbrace{\begin{pmatrix} \boxed{Q_1} & \boxed{Q_2} & \cdots & \boxed{Q_n} \end{pmatrix}}_{\hat{Q}} = \hat{M} \underbrace{\begin{pmatrix} \boxed{P_1} & \boxed{P_2} & \cdots & \boxed{P_n} \end{pmatrix}}_{\hat{P}}$$

Questions

- Can we choose a rigid solution?
- What are the assumptions we made about the objects in the scene?
- How can we solve the ambiguity?
- How correspondence outliers expect to affect the algorithm?

Summary - Geometry

- Projection models:
  - Perspective
  - Weak perspective
- Tasks:
  - 3D reconstructions
  - Homography
  - SFM

- Algebra:
  - Projective geometry
  - Homography
  - Epipolar geometry
- Outlier removal:
  - RANSAC

Motion Anlysis

A Sequence of Images

- Tracking
- Ego motion
- Segmentation
- 3D shape
- Object motion:
  - Gesture recognition
  - Action recognition
  - Gait recognition

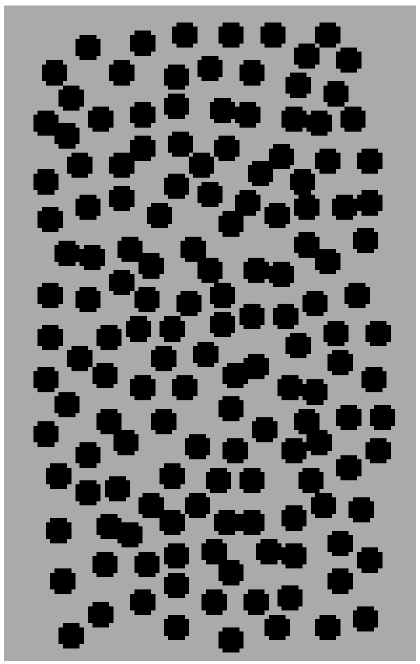
Possible Setups

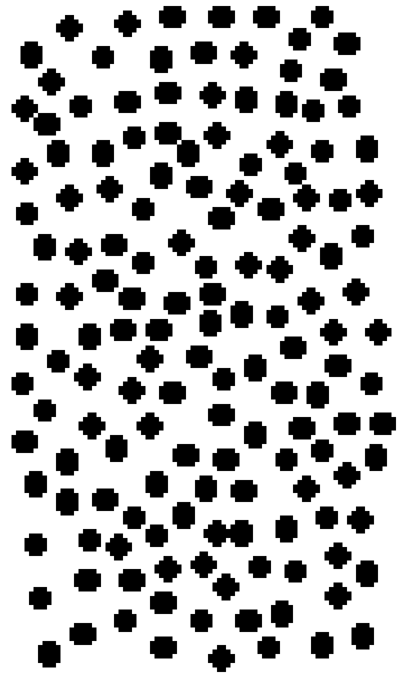
- Still Camera
  - Single moving object
  - Several moving object
- Moving Camera:
  - Still scene
  - One or more moving objects
- Many cameras…

Typical Quetions

- Segmentation:
  - The region of each moving object
  - How many moving objects are there?
- Object motion:
  - Direction in 2D/3D
  - Speed
- Ego motion recovery
- Rigidity?

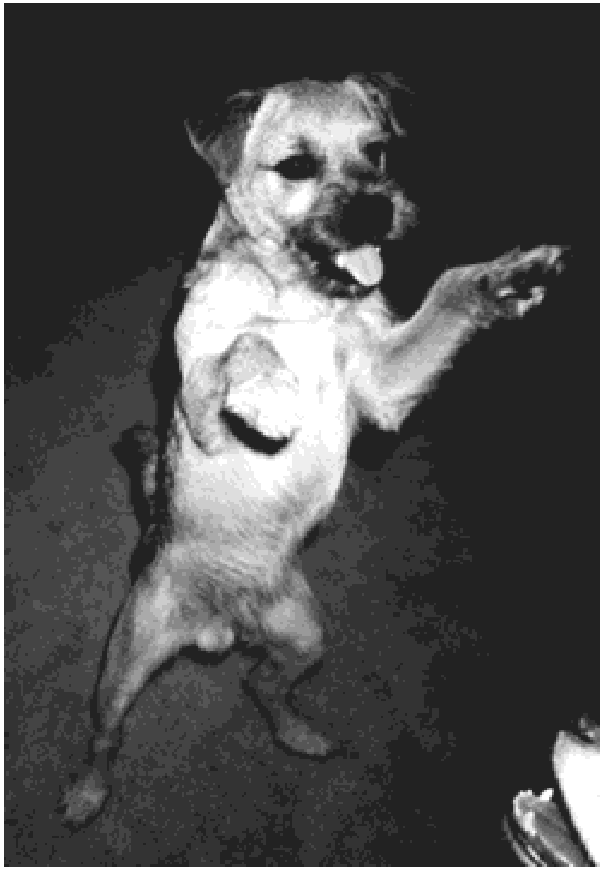
Segmentation





Taken from http://www.cquest.utoronto.ca/psych/psy280f/ch8/rdc.html

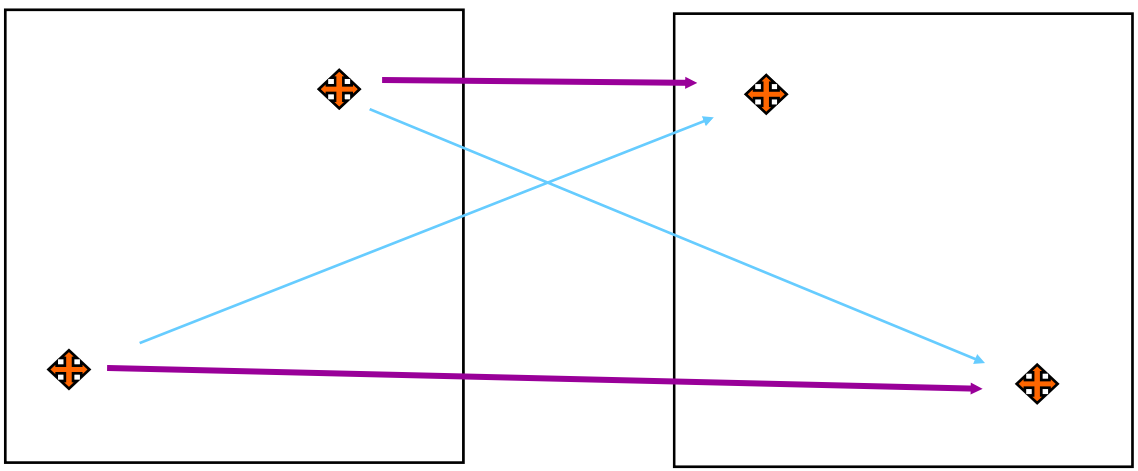
http://www.cquest.utoronto.ca/psych/psy280f/ch8/archieMove.html

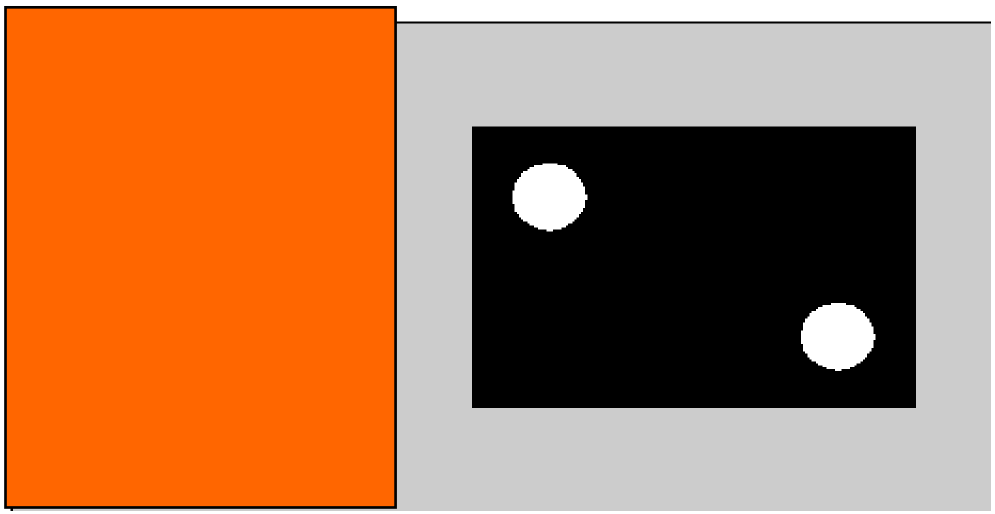


Feature Correspondence and Perception

Correspondence

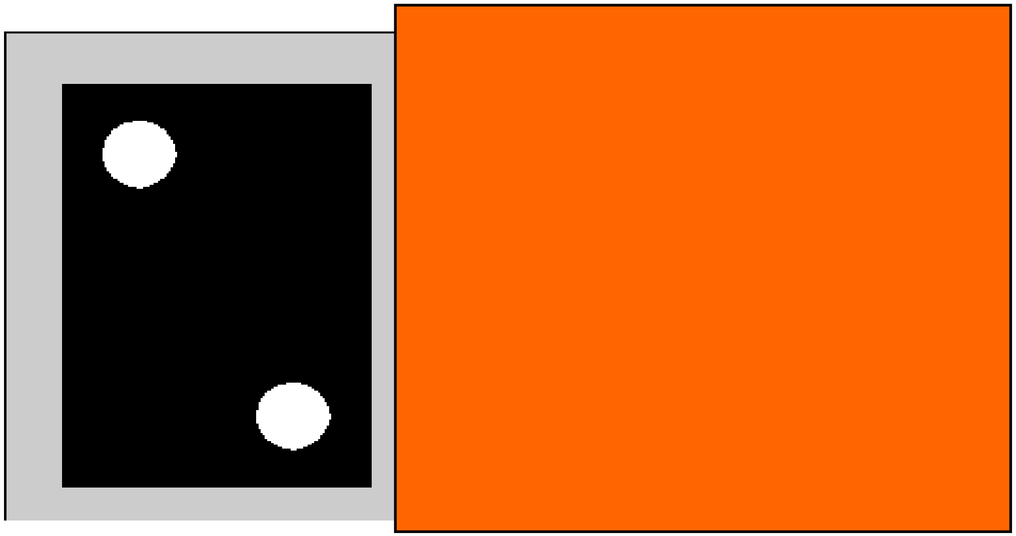
- Perceived motion
- Which point in image I_t corresponds to which point in image I_{t+1}?



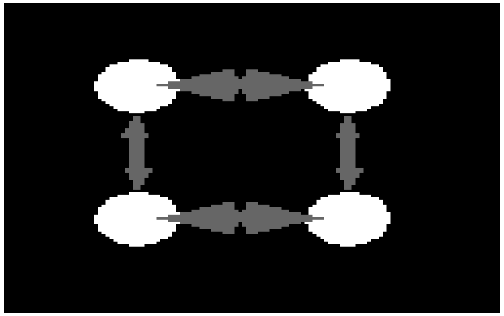


Taken from http://www-psy.ucsd.edu/~sanstis/motion.html

Distance

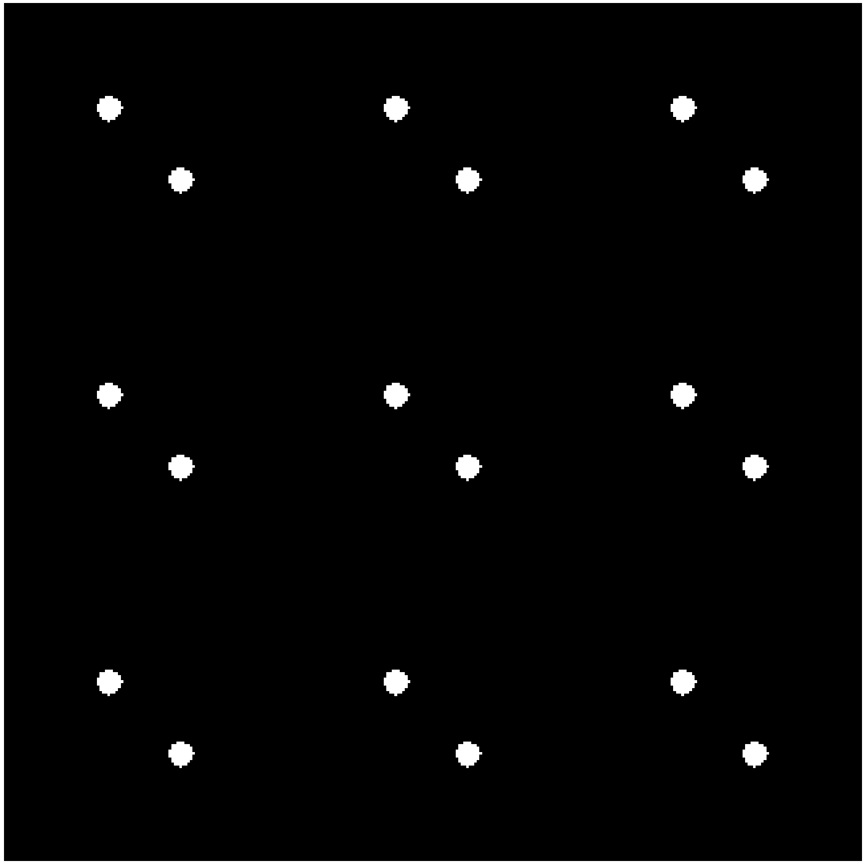


Taken from http://www-psy.ucsd.edu/~sanstis/motion.html



Taken from http://www-psy.ucsd.edu/~sanstis/motion.html

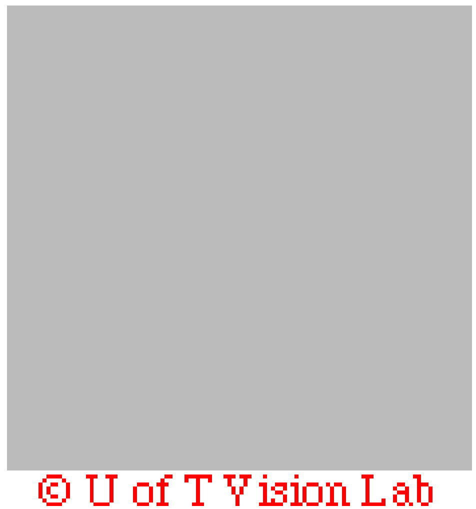
Global movement



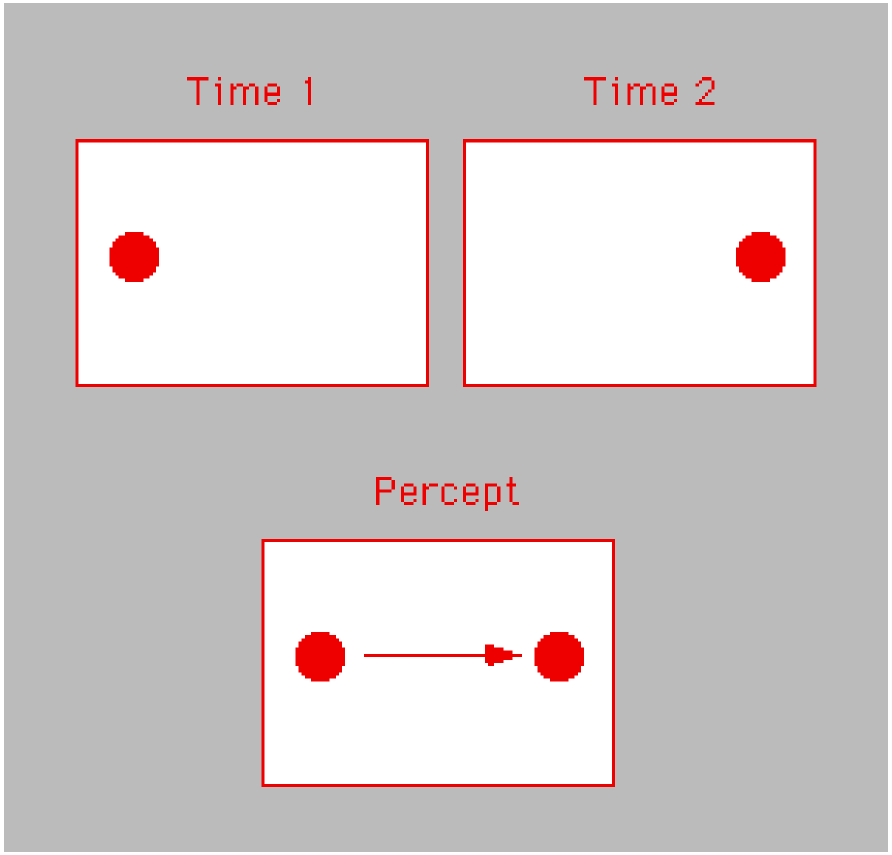
Taken from http://www-psy.ucsd.edu/~sanstis/motion.html

הבינתחומי הרצליה
IDC HERZLIYA

Shape similarity



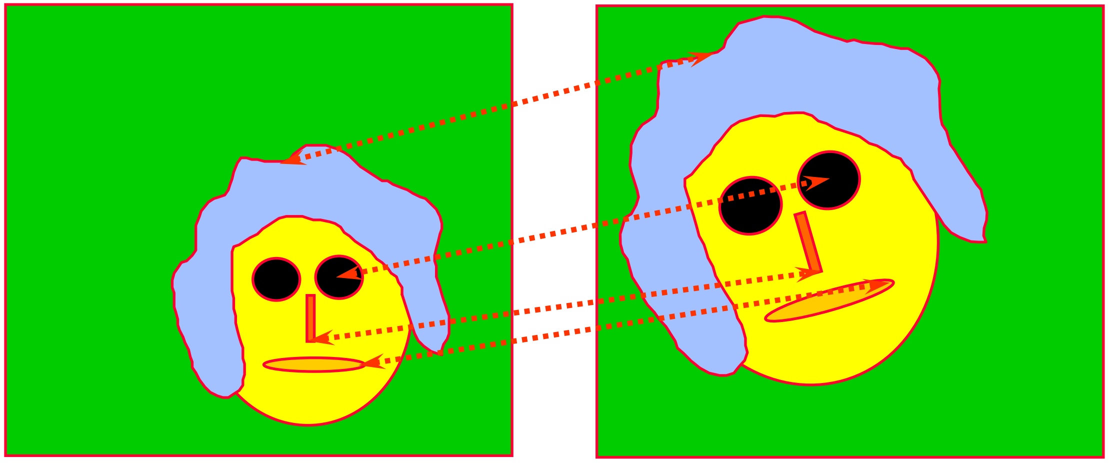
http://www.cquest.utoronto.ca/psych/psy280f/ch8/ternus.html



Taken from
http://www.cquest.utoronto.ca/psych/psy280f/ch8/amTiming.html

Motion Estimation Direction

Where are the moving regions moving to?



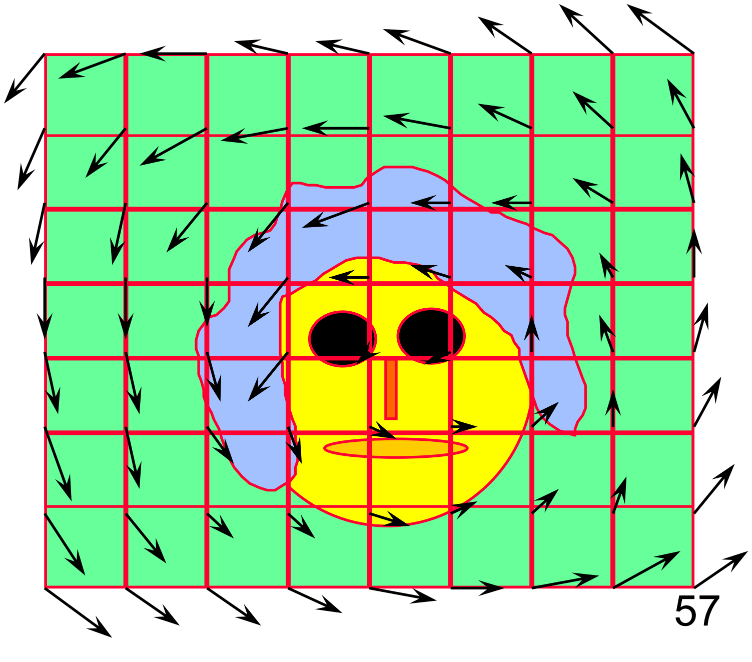
Existing Approaches for Motion Estimation

- Correlation based
- Feature based
- Gradient based

We focus here

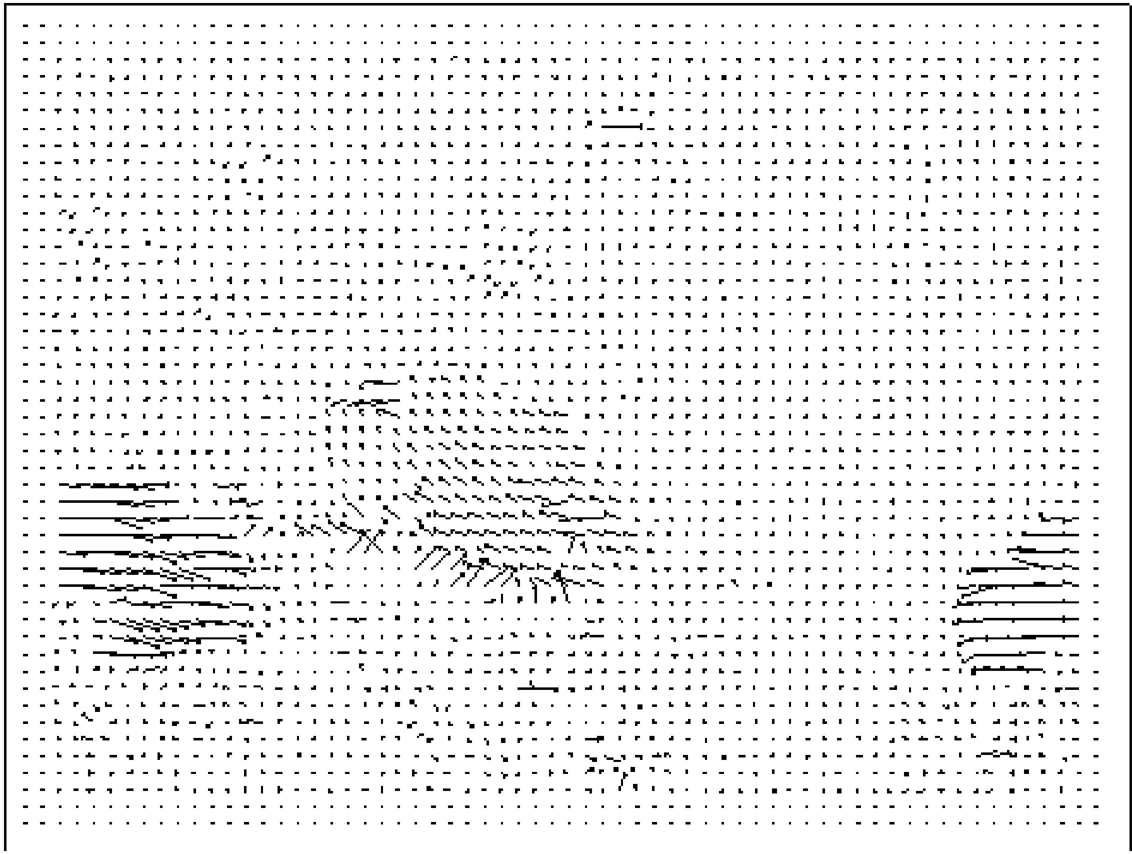
Optical Flow

- Pixel motion between consecutive frames:
  - Caused by camera or object motion
- Introduced by James J. Gibson 1940



Optical Flow

For each pixel, a velocity vector (u,v)

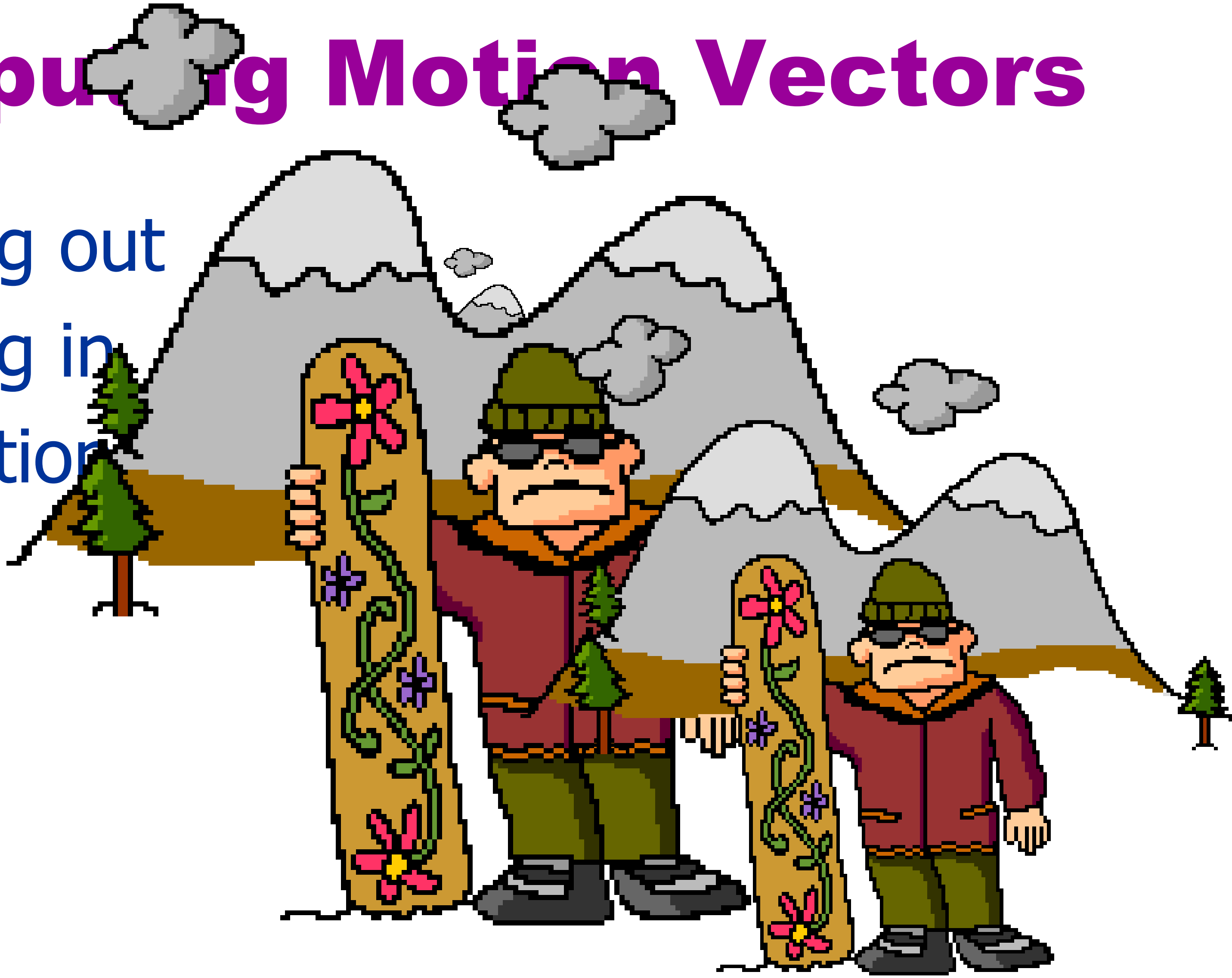


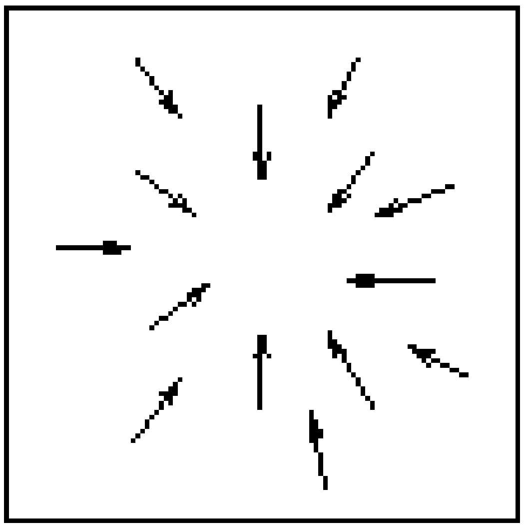
http://extra.cmis.csiro.au/IA/changs/motion/

demo

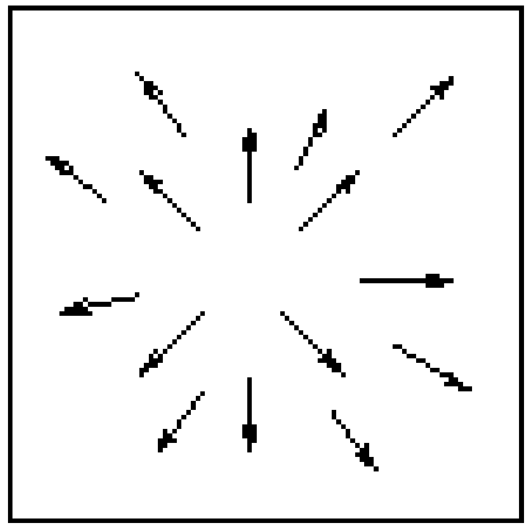
Computing Motion Vectors

- Zooming out
- Zooming in
- Translation

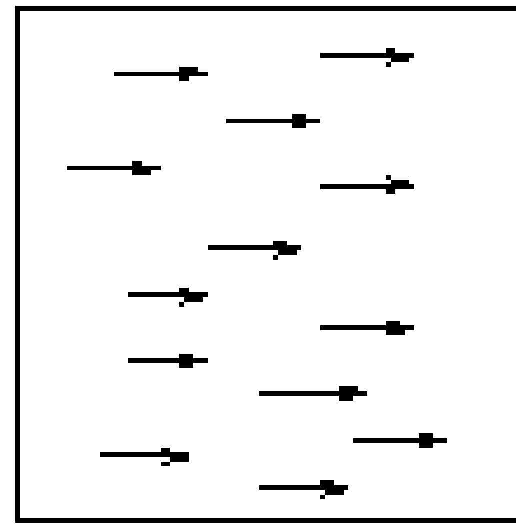




Zoom out



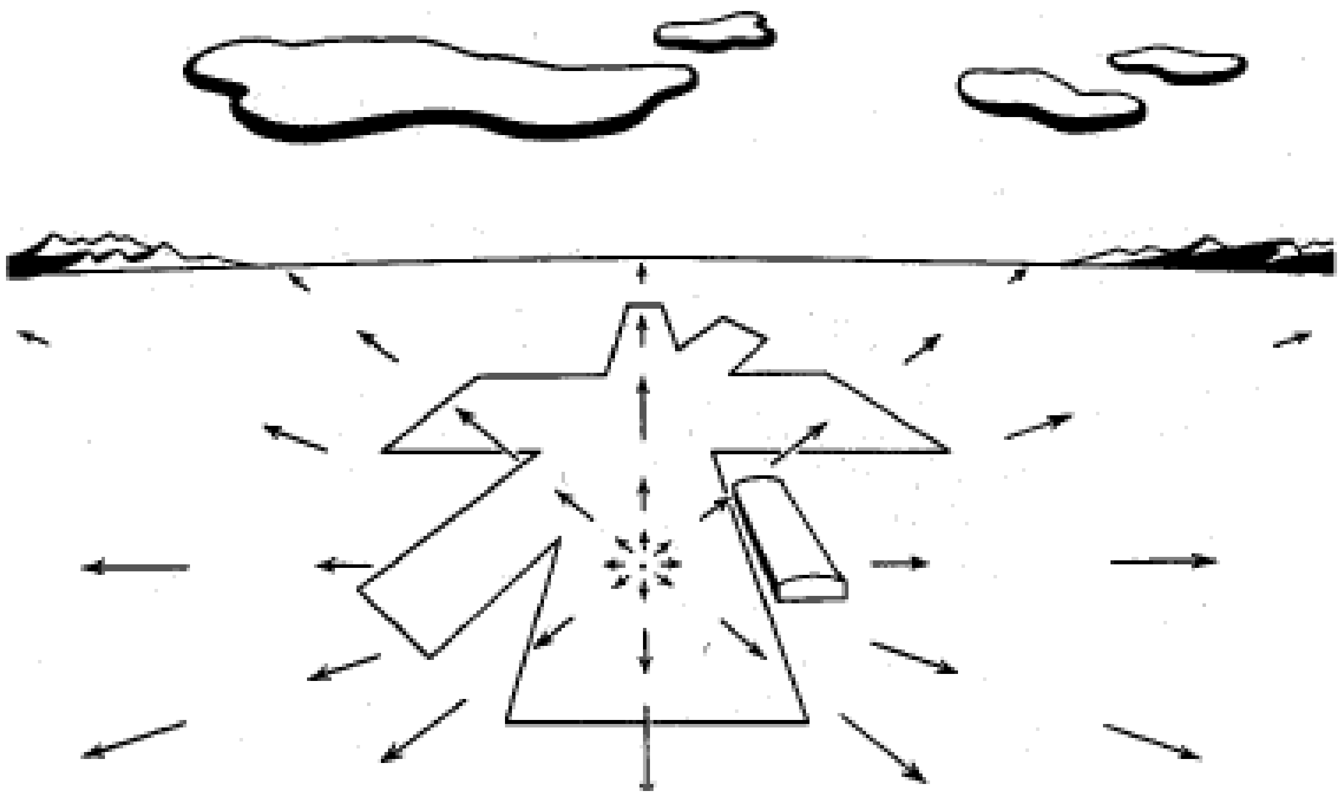
Zoom in



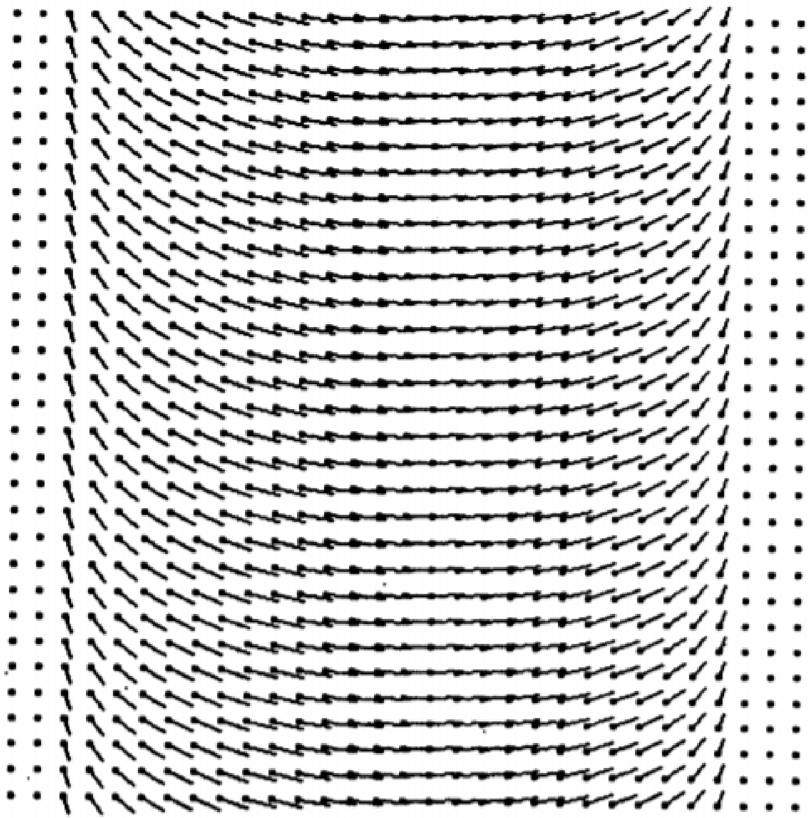
Pan Right to Left

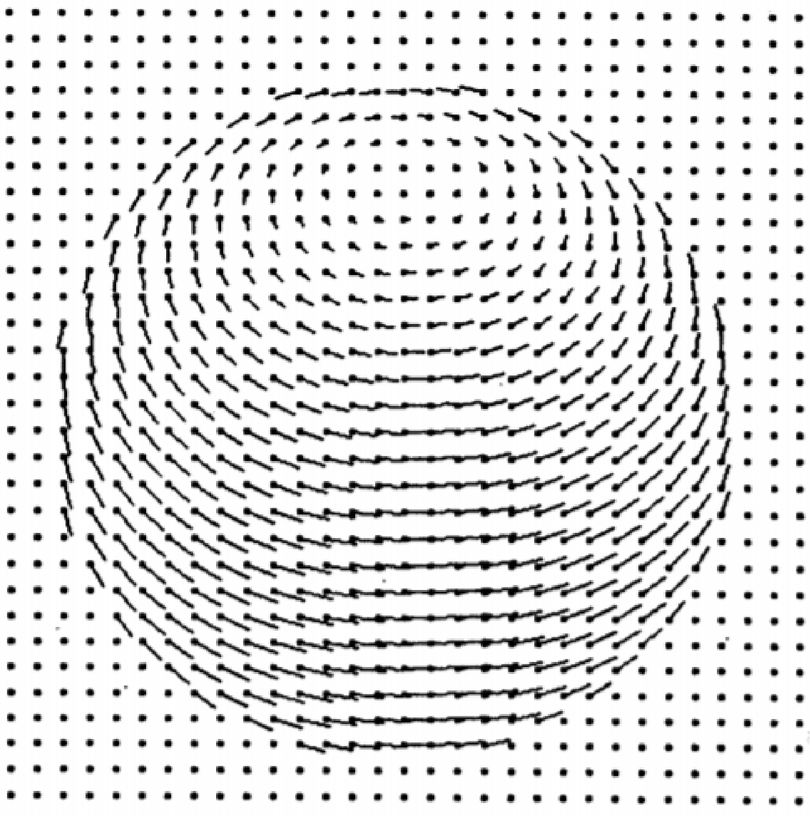
Forward Translation & Focus of Expansion

[Gibson, 1950]

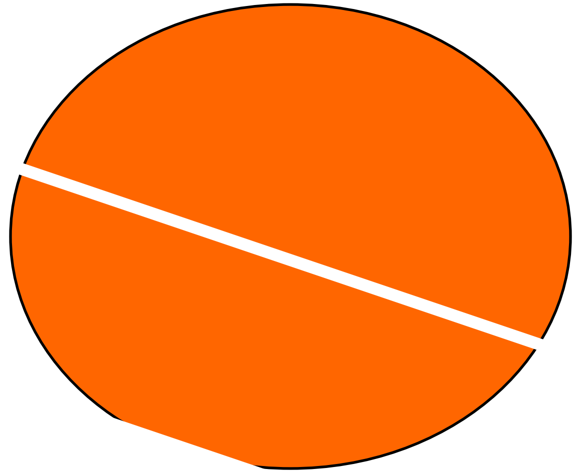


What is Moving and How?

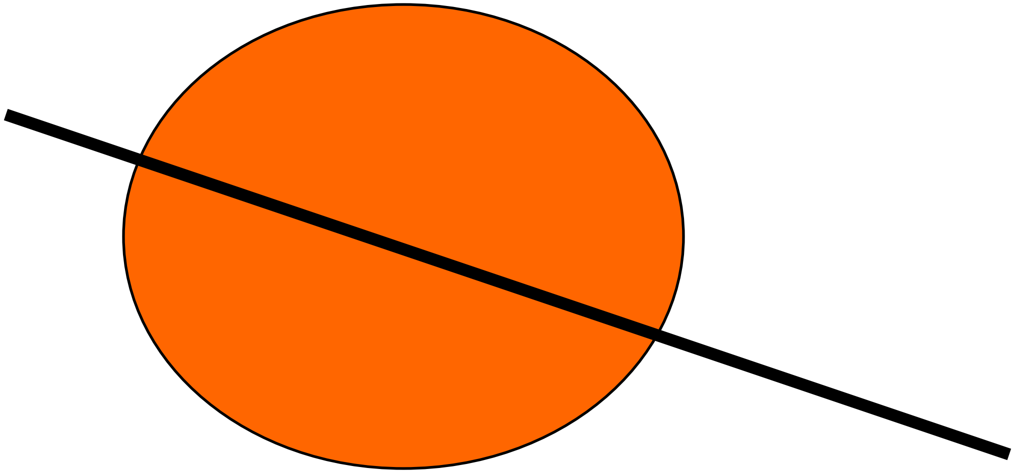




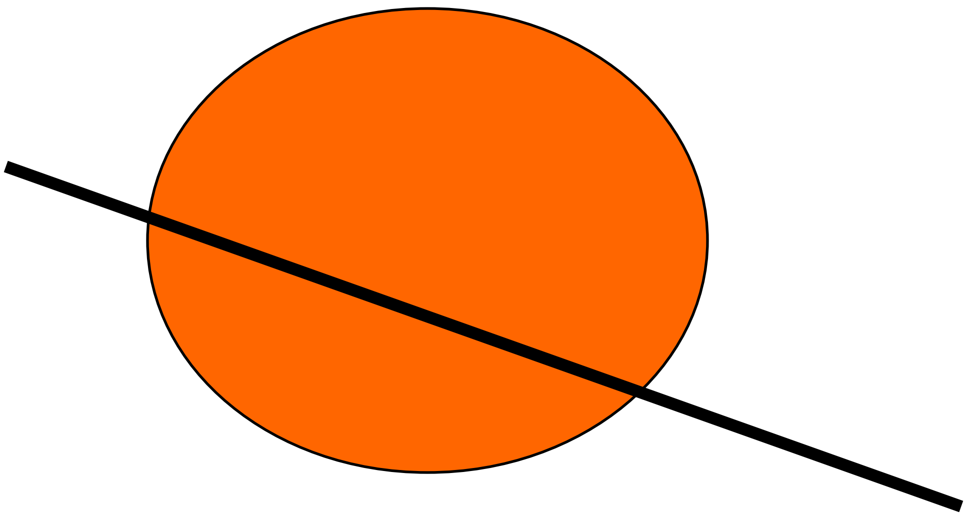
Aperture Problem



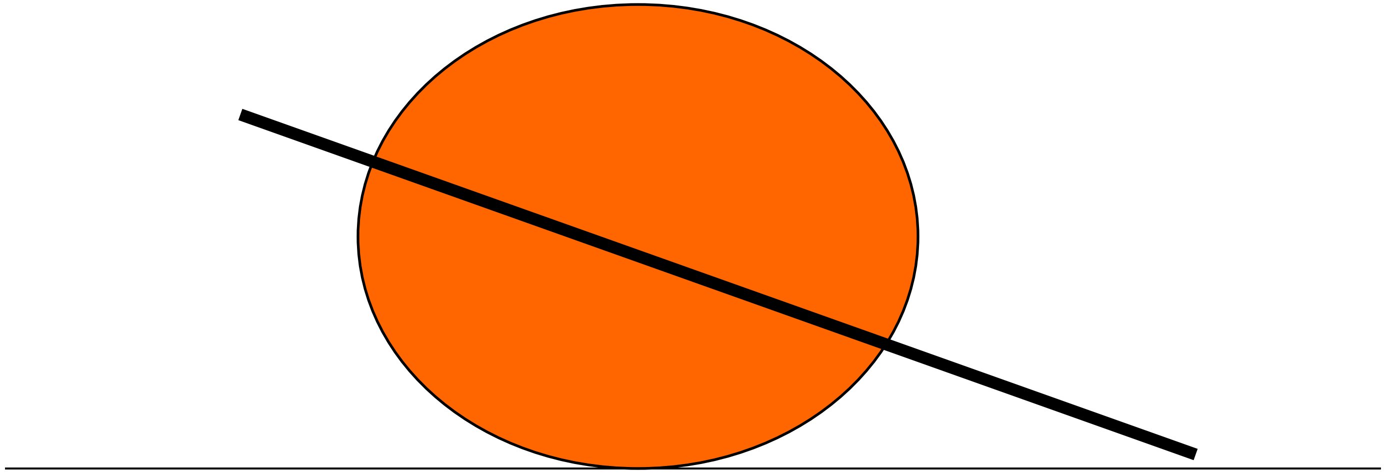
Aperture Problem



Aperture Problem



Aperture Problem



Barber Pole

- demo

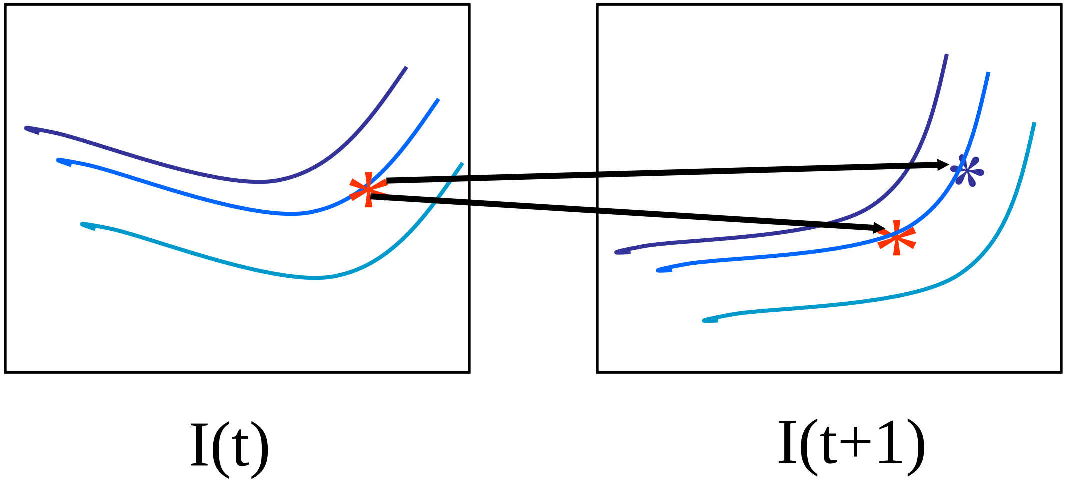
Optical Flow: Gradient Based

Assumptions:

- The movement is small
- Brightness constancy assumption (BCA):
  - The intensity of a given object point does not change between frames
  - $I(x + dx, y + dy, t + dt) = I(x, y, t)$

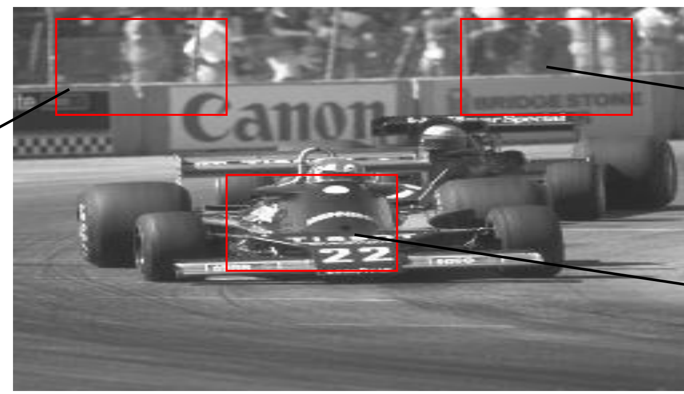
Ambiguity

- $I(x+dx, y+dy, t+dt) = I(x, y, t)$
- Brightness constancy assumption: insufficient!



Resolve Ambiguity

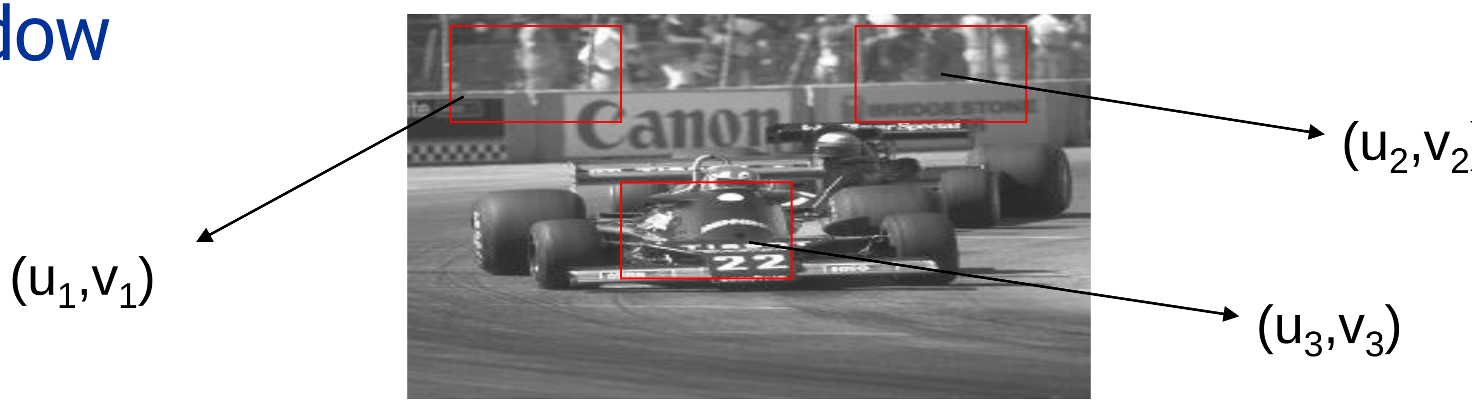
- Local constraint
  - Assume constant motion in a small local window



- Global constraints - later today

Solution of Ambiguity

- Local constraint
  - Assume constant motion in a small local window



- How to use local constraint?
  - Naïve search: expensive!

Next

- Gradient Based method

Gradient Based Method

- Relates spatial and temporal gradients
- Assumptions:
  - First order approximation of the flow: $u(p)$ and $v(p)$ are small
  - $u(p)$ and $v(p)$ are constant (or smooth) in a small neighborhood of p

An under-determined problem (aperture Problem)

Optical Flow Equation

Which assumption we used?

- Taylor Series for a pixel $p = (p_x, p_y)$

$$I(p_x + dx, p_y + dy, t + dt)$$

$$= I(p_x, p_y, t) + \frac{\partial I}{\partial x}(p)dx + \frac{\partial I}{\partial y}(p)dy + \frac{\partial I}{\partial t}(p)dt + \cancel{\ldots}$$

- Brightness constancy assumption:
  - $I(p_x + dx, p_y + dy, t + dt) = I(p_x, p_y, t)$
  - $\frac{\partial I}{\partial x}(p)dx + \frac{\partial I}{\partial y}(p)dy + \frac{\partial I}{\partial t}(p)dt = 0$
  - Notation: $\boxed{I_x(p)dx + I_y(p)dy = -I_t(p)dt}$

Lucas–Kanade Algorithm

- Optical flow computation
- Gradient based algorithm

Optical Flow Equation

- We can compute: $I_x(p)$, $I_y(p)$, $I_t(p)$
  - $I_x(p)dx + I_y(p)dy = -I_t(p)dt$
- Divide by dt
  - $I_x(p)\frac{dx}{dt} + I_y(p)\frac{dy}{dt} = -I_t(p)$
  - $\boxed{I_x(p)u(p) + I_y(p)v(p) = -I_t(p)}$
- Matrix notation: $\left(I_x(p), I_y(p)\right)\begin{pmatrix} u(p) \\ v(p) \end{pmatrix} = -I_t(p)$
- We search for $(v(p), u(p))$

Local Constant Flow

- For each p_i we have:

$$\left(I_x(p_i), I_y(p_i)\right)\begin{pmatrix} u(p_i) \\ v(p_i) \end{pmatrix} = -I_t(p_i)$$

- Let $w(p_0)$ be a small patch around p_0
- Assume constant motion $\forall p_i \in w(p_0)$:

$$\begin{pmatrix} I_x(p_1), I_y(p_1) \\ I_x(p_2), I_y(p_2) \\ I_x(p_3), I_y(p_3) \end{pmatrix}\begin{pmatrix} u(p_0) \\ v(p_0) \end{pmatrix} = -\begin{pmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_k) \end{pmatrix}$$

IDC HERZLIYA הבינתחומי הרצליה

Local Constant Flow

Let $p_1 \dots p_k \in w(p_0)$:

$$\underbrace{\begin{pmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_k) & I_y(p_k) \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} u(p_0) \\ v(p_0) \end{pmatrix} = -\underbrace{\begin{pmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_k) \end{pmatrix}}_{\mathbf{b}}$$

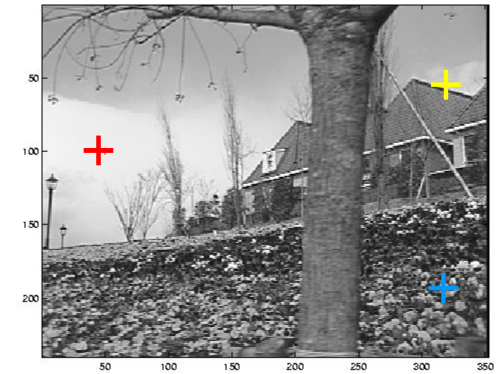
That is: $A\begin{pmatrix} u(p_0) \\ v(p_0) \end{pmatrix} = b \quad \Rightarrow \quad \begin{pmatrix} u(p_0) \\ v(p_0) \end{pmatrix} = A^+ b$

Is there a problem?

$$\boxed{A^+ = (A^T A)^{-1} A^T}$$

IDC HERZLIYA הבינתחומי הרצליה

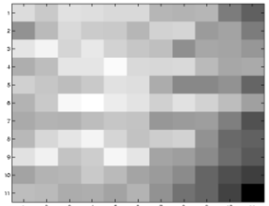
Cases

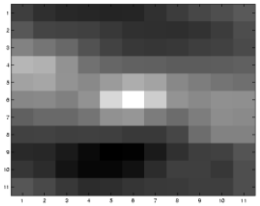


$$A\begin{pmatrix} u(p_0) \\ v(p_0) \end{pmatrix} = b \quad \longrightarrow \quad A^T A\begin{pmatrix} u(p_0) \\ v(p_0) \end{pmatrix} = A^T b$$

Let $C = A^T A = \begin{pmatrix} \sum I_x^2(p_i) & \sum I_x(p_i) I_y(p_i) \\ \sum I_x(p_i) I_y(p_i) & \sum I_y^2(p_i) \end{pmatrix}$

- $rank(C) = 0$ blank wall problem
- $rank(C) = 1$ aperture problem
- $rank(C) = 2$ enough texture





Algebra: definition of C

- $A = \begin{pmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \\ I_x(p_k) & I_y(p_k) \end{pmatrix}$

- $A^T A = \begin{pmatrix} I_x(p_1) & I_x(p_2) \dots I_x(p_k) \\ I_y(p_1) & I_y(p_2) \dots I_y(p_k) \end{pmatrix} \begin{pmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \\ I_x(p_k) & I_y(p_k) \end{pmatrix}$

$$= \begin{pmatrix} \sum I_x^2(p_i) & \sum I_x(p_i) I_y(p_i) \\ \sum I_x(p_i) I_y(p_i) & \sum I_x^2(p_i) \end{pmatrix}$$

The Algorithm

- Smooth the image in the special domain
- Smooth the image in the temporal domain (not always necessary)
- For each pixel, p_0:
  - Compute $A(p_0)$, $b(p_0)$, and $C(p_0)$
  - If $rank(C(p_0)) = 2$, compute $u(p_0)$ and $v(p_0)$ by:
    $$\begin{pmatrix} u(p_0) \\ v(p_0) \end{pmatrix} = A^{+}b$$

Modification

- Replace Σ in $c(p_0) = \begin{pmatrix} \sum I_x^2(p_i) & \sum I_x(p_i)I_y(p_i) \\ \sum I_x(p_i)I_y(p_i) & \sum I_y^2(p_i) \end{pmatrix}$

$p_i \in w(p_0)$

by Convolution with Gaussian:

  - $E.g.,\quad \sum I_x^2(p_i) \Rightarrow (G * I_x^2)(p_0)$

$(G * I_x^2)$ is a matrix

- $c(p_0) = \begin{pmatrix} (G * I_x^2)(p_0) & (G * I_xI_y)(p_0) \\ (G * I_xI_y)(p_0) & (G * I_y^2)(p_0) \end{pmatrix}$

What can go wrong?

- Brightness constancy is **not** satisfied
- The motion is **not** small
- The motion is **not** translation
- A point does **not** move like its neighbors
  - window size is too large
  - what is the ideal window size?

Next

- Dealing with large motion
  - Use a pyramid of OF
- More general motion:
  - Affine rather than just translation
- Global solutions