Feature Extraction

- Features: local meaningful detectable
  - Points
  - Edges
    - Step edges
    - Line edges
  - Contours – Deformable models
    - Closed contours are boundaries
    - Regions

1

Model Fitting

Fitting: find the parameters of a model that best fit the data

Alignment: find the parameters of the transformation that best align matched points

2

Fitting and Alignment

- Design challenges
  - Design a suitable **goodness of fit** measure
    - Similarity should reflect application goals
    - Encode robustness to outliers and noise
  - Design an **optimization** method
    - Avoid local optima
    - Find best parameters quickly

3

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

4

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

5

Template Matching

- The most straightforward approach to finding a particular pattern in the image is to look at every patch of pixels and to directly compare the intensity values in that patch to the intensity values in the search pattern.
- Consider two vectors of intensity values, x and y, One way to compare them is by simply computing the Sum of Squared Differences (SSD)

$$SSD(x, y) = \sum_i (x_i - y_i)^2$$

6

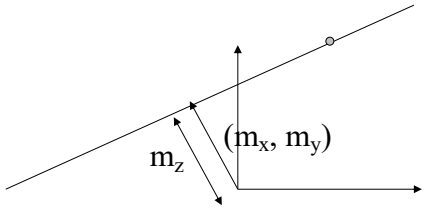
Line Fitting

- Follows edge extraction and linking
- Used to find straight line features in an image
- Simplest way split and merge techniques
  - Problem with corners

7

Parameterizing lines

- For our purposes lines in the image will be parameterized by a vector of 3 numbers (m_x, m_y, m_z) where: $m_x^2 + m_y^2 = 1$
- Points on the line are defined by the equation: $m_x * x + m_y * y + m_z = 0$



8

Least Squares Fitting

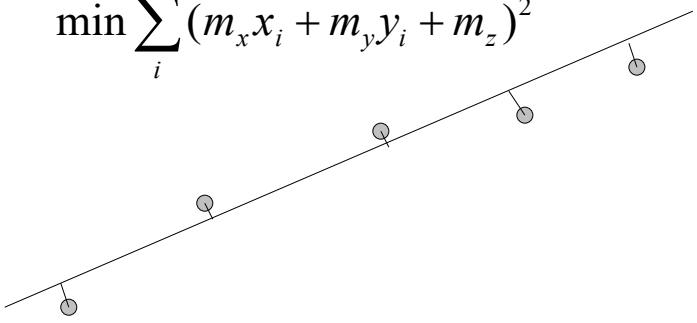
- Given a set of points (xi, yi) the goal of the fitting procedure is to find the parameters (m_x, m_y, m_z) which represent the *best* line

9

Least squares criterion

- The best fit is defined in terms of the parameters which minimize the sum of the squared residuals

$$\min \sum_i (m_x x_i + m_y y_i + m_z)^2$$



10

Steps

- Compute the centroid of the point set
- Change coordinates such that the new centroid is (0,0)

$$\bar{x} = \frac{1}{n}\sum x_i, \bar{y} = \frac{1}{n}\sum y_i$$

$$x_i' = x_i - \bar{x}, y_i' = y_i - \bar{y}$$

11

Steps (II)

- Solve for (m_x, m_y) by minimizing the following quadratic form

$$e(m_x, m_y) = \sum_i (m_x x_i' + m_y y_i')^2$$

- with $m_x = \cos\theta$, $m_y = \sin\theta$

$$\tan 2\theta = \frac{a}{b}$$

$$a = 2\sum_{i=1}^{n} x_i' y_i'$$

$$b = \sum_{i=1}^{n} x_i'^2 - \sum_{i=1}^{n} y_i'^2$$

12

Steps (III)

- Solve for m_z

$$m_z = -(m_x \bar{x} + m_y \bar{y})$$

13

Issues

- The main problem with least squares fitting techniques is that they are heavily influenced by outliers
- Solutions to this problem include
  - Robust weighting measures
  - Iteratively reweighted least squares
  - Least median squares

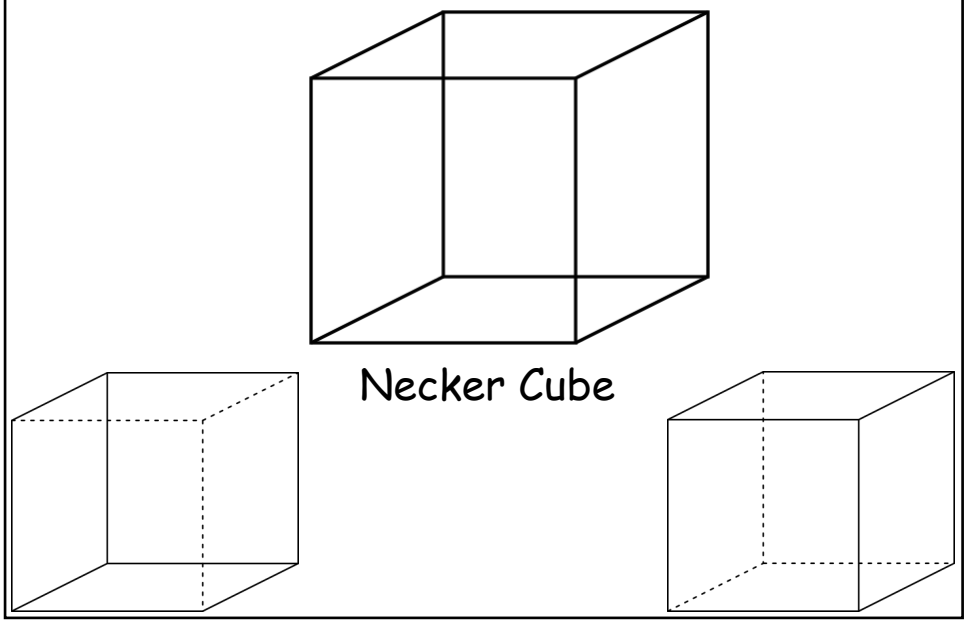
14

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

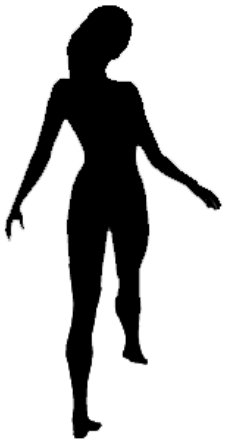
15

Multi-stable Perception



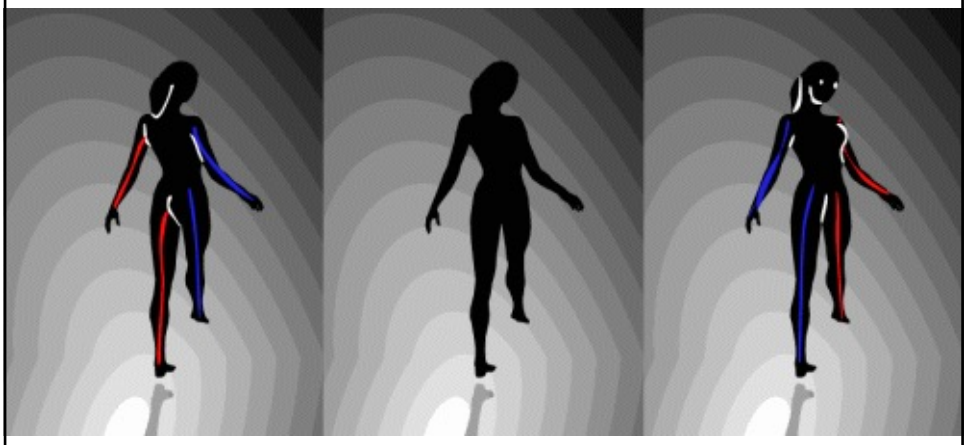
Necker Cube

16



Spinning dancer illusion,
Nobuyuki Kayahara

17



18

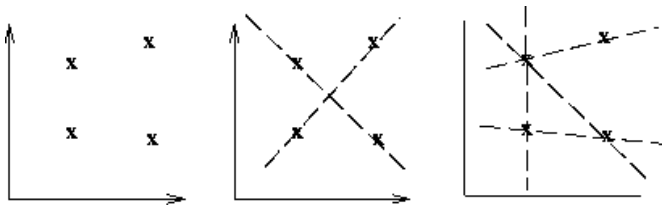
The Hough Transform

- Another approach that can be used to detect features such as lines in an image is the Hough transform
- This approach is a voting scheme based on accumulating evidence in a parameter space

19

The Hough Transform

- Each input measurement indicates its contribution to a globally consistent solution
- Here this problem is under-constrained.

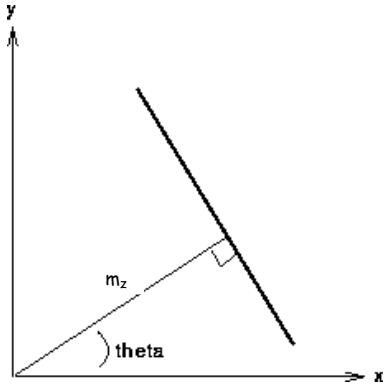


20

Approach

- For every point x, y determine the set of all θ and m_z such that:

$$\cos\theta\, x_i + \sin\theta\, y_i + m_z = 0$$



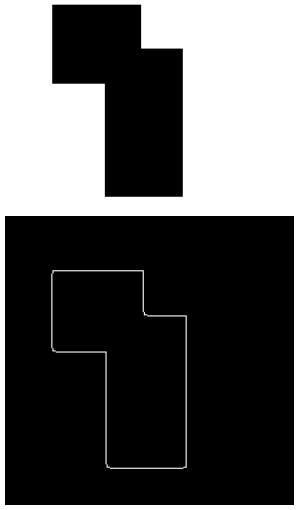
21

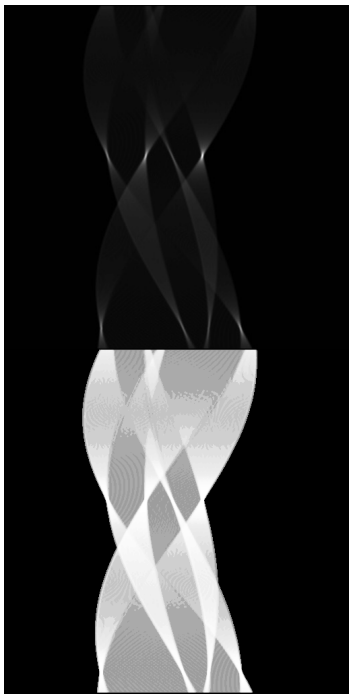
Approach

- When viewed in Hough parameter space, points which are collinear in the cartesian image space become readily apparent as they yield curves which intersect at a common point (m_z ,θ).
- Mark all of these parameter pairs in a 2D accumulator array
- Find local maxima in the accumulator array

22

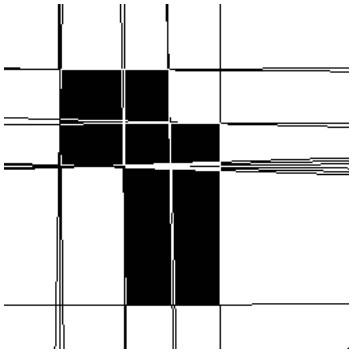
Example





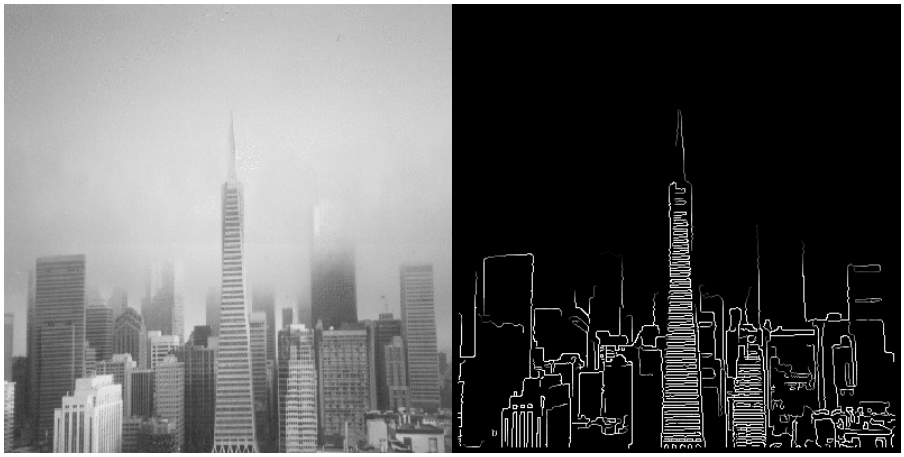
23

Example(II)



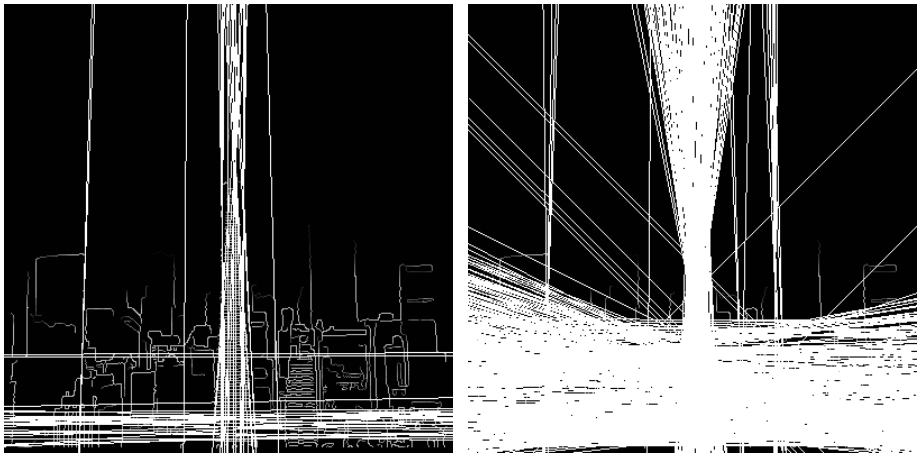
24

Real Example



25

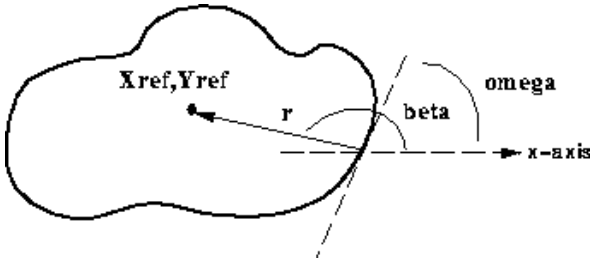
Real Example



26

Generalized Hough Transform

- Use a look-up table to define the relationship between the boundary positions and orientations and the Hough parameters



27

Issues

- The storage space required increases exponentially with the dimension of the parameter space
- Choosing the size of the buckets in the accumulator space can be an issue

28

Ellipses

- An ellipse is a type of conic section defined by the equations

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

$$b^2 - 4ac < 0$$

29

Fitting ellipses

- One approach to fitting ellipses is to find a choice of parameters which minimizes the sum of squares of the residuals

$$e = \sum_i (ax_i^2 + bx_i y_i + cy_i^2 + dx_i + ey_i + f)^2$$

30

Issues

- The preceding expression represents the algebraic distance between the feature points and the ellipse not the Euclidean distance. The resulting solution is, therefore, biased.

31

Who came from which line?

- Assume we know how many lines there are - but which lines are they?
  - easy, if we know who came from which line
- Three strategies
  - Incremental line fitting
  - K-means
  - Probabilistic (later!)

32

Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
   Transfer first few points on the curve to the line point list
   Fit line to line point list
   While fitted line is good enough
      Transfer the next point on the curve
         to the line point list and refit the line
   end
   Transfer last point(s) back to curve
   Refit line
   Attach line to line list
end
```

33

Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

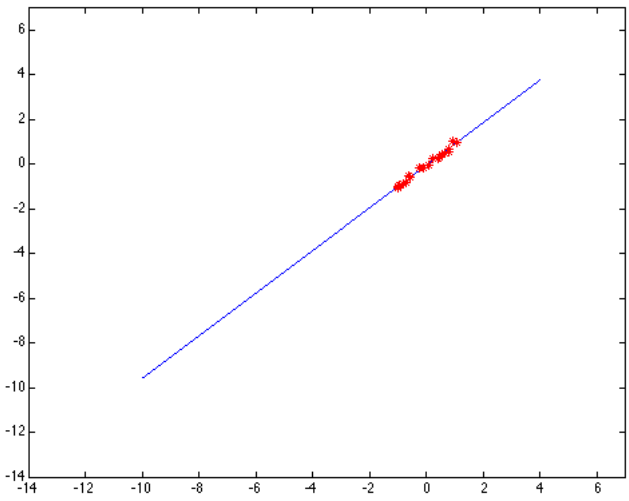
```
Hypothesize k lines (perhaps uniformly at random)
or
Hypothesize an assignment of lines to points
   and then fit lines using this assignment

Until convergence
   Allocate each point to the closest line
   Refit lines
end
```

34

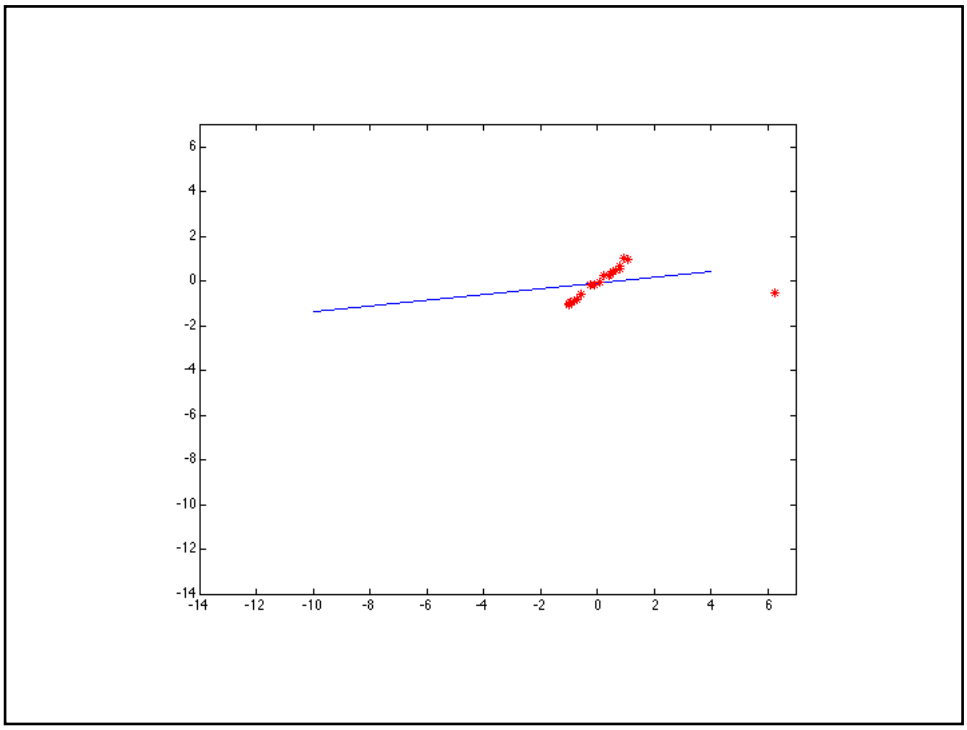
Robustness

- As we have seen, squared error can be a source of bias in the presence of noise points
  - One fix is EM - we'll do this later
  - Another is an M-estimator
    - Square nearby, threshold far away
  - A third is RANSAC
    - Search for good points

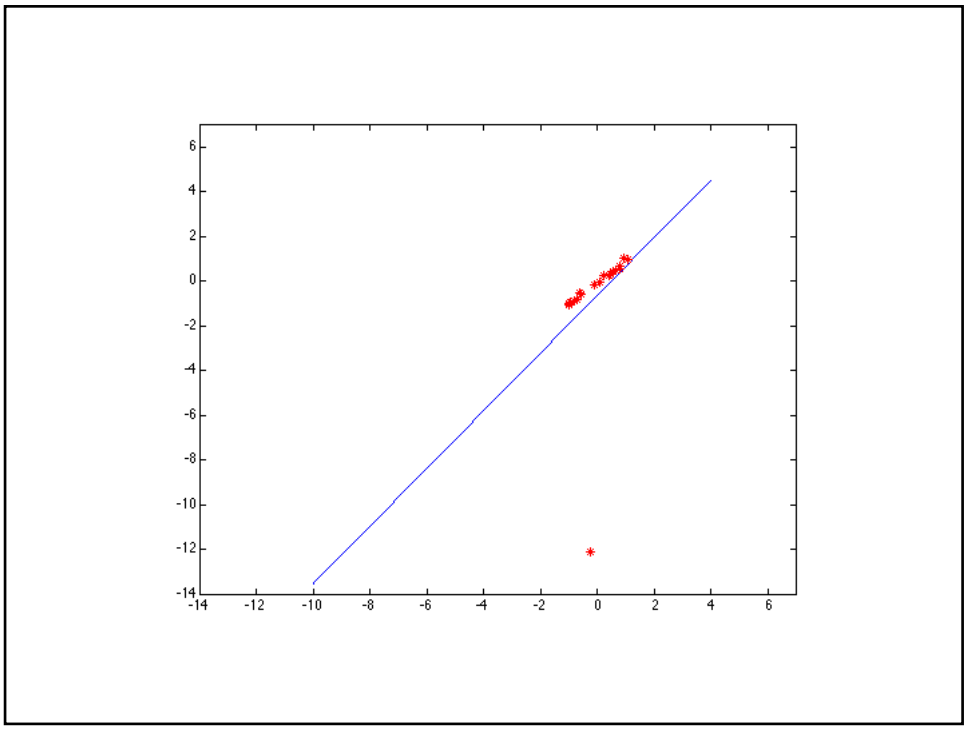
35



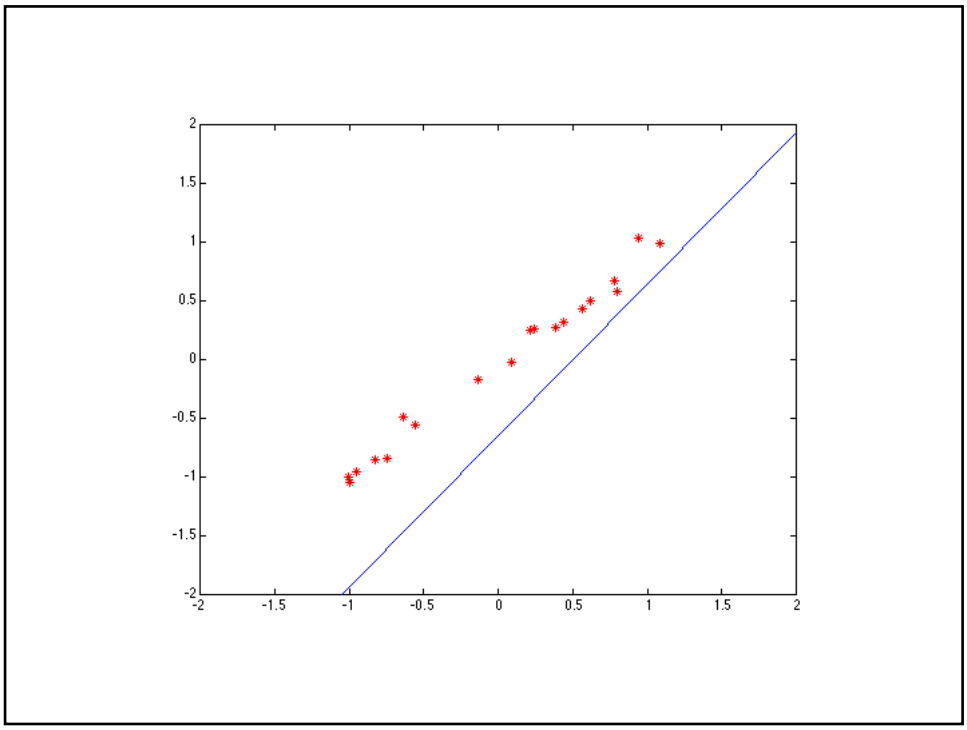
36


6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-14
-12
-10
-8
-6
-4
-2
0
2
4
6

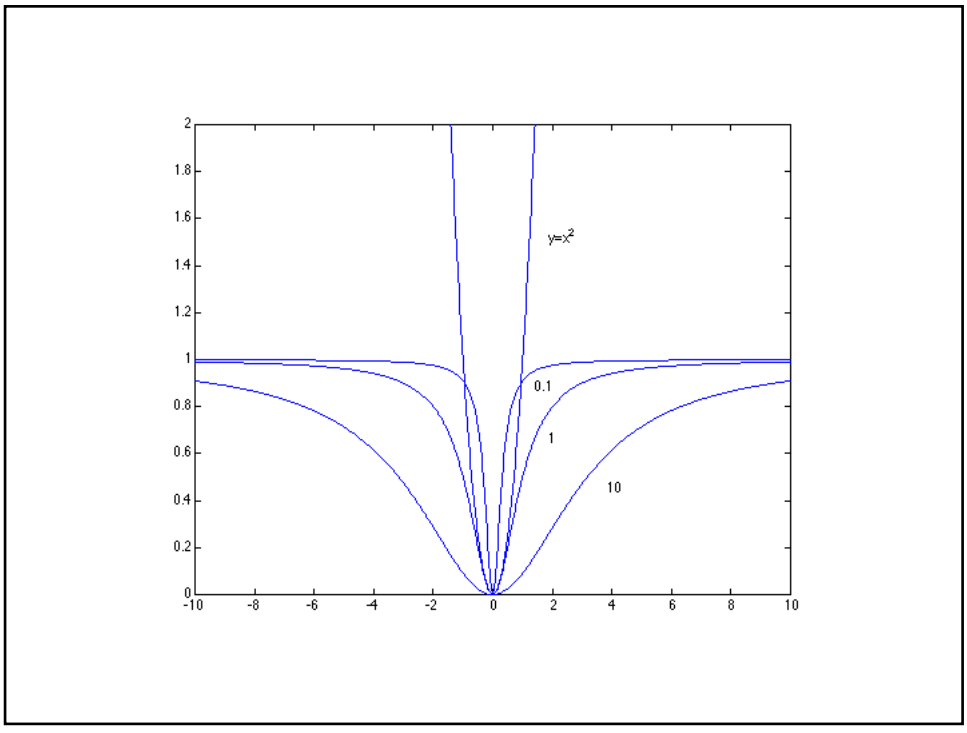
37


6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-14
-12
-10
-8
-6
-4
-2
0
2
4
6

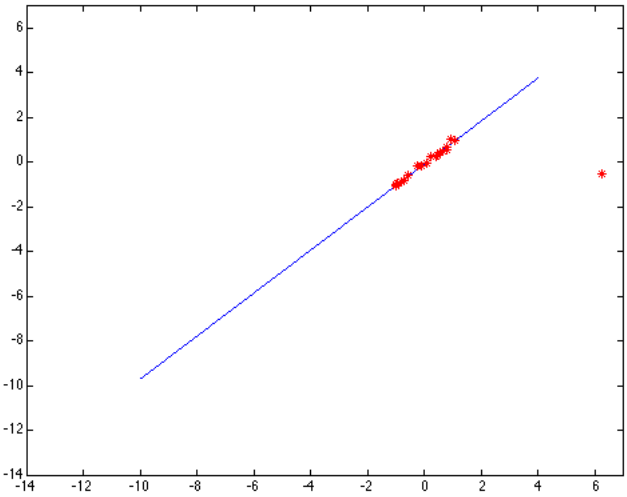
38


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

39

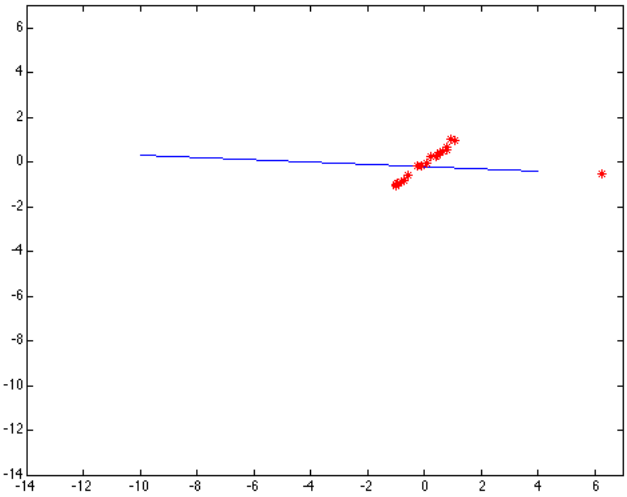

2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-10
-8
-6
-4
-2
0
2
4
6
8
10
y=x2
0.1
1
10

40



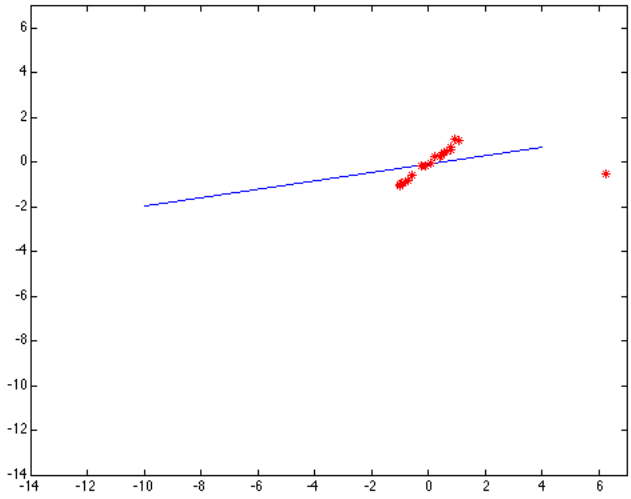
41

Too small



42

Too large



43

RANSAC

- Choose a small subset uniformly at random
- Fit to that
- Anything that is close to result is signal; all others are noise
- Refit
- Do this many times and choose the best

- Issues
  - How many times?
    - Often enough that we are likely to have a good line
  - How big a subset?
    - Smallest possible
  - What does close mean?
    - Depends on the problem
  - What is a good line?
    - One where the number of nearby points is so big it is unlikely to be all outliers

44

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
    n — the smallest number of points required
    k — the number of iterations required
    t — the threshold used to identify a point that fits well
    d — the number of nearby points required
      to assert a model fits well
Until k iterations have occurred
    Draw a sample of n points from the data
      uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
      Test the distance from the point to the line
        against t; if the distance from the point to the line
        is less than t, the point is close
    end
    If there are d or more points close to the line
      then there is a good fit. Refit the line using all
      these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion

45

Fitting curves other than lines

- In principle, an easy generalisation
  - The probability of obtaining a point, given a curve, is given by a negative exponential of distance squared
- In practice, rather hard
  - It is generally difficult to compute the distance between a point and a curve

46

Robust estimation: Details

- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Least squares solution can be used for initialization
- Scale of robust function should be chosen adaptively based on median residual

47

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

48

Other ways to search for parameters (for when no closed form solution exists)

- Line search
  1. For each parameter, step through values and choose value that gives best fit
  2. Repeat (1) until no parameter changes
- Grid search
  1. Propose several sets of parameters, evenly sampled in the joint set
  2. Choose best (or top few) and sample joint parameters around the current best; repeat
- Gradient descent
  1. Provide initial position (e.g., random)
  2. Locally search for better parameters by following gradient

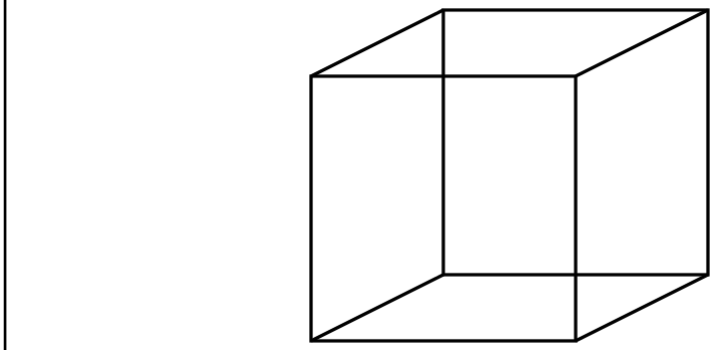
49

Fitting and Alignment: Methods

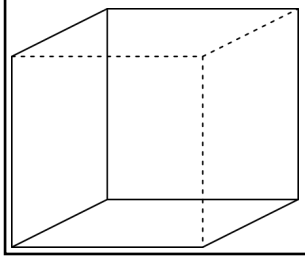
- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

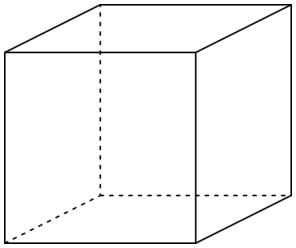
50

Multi-stable Perception

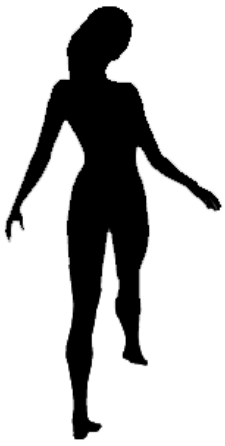


Necker Cube



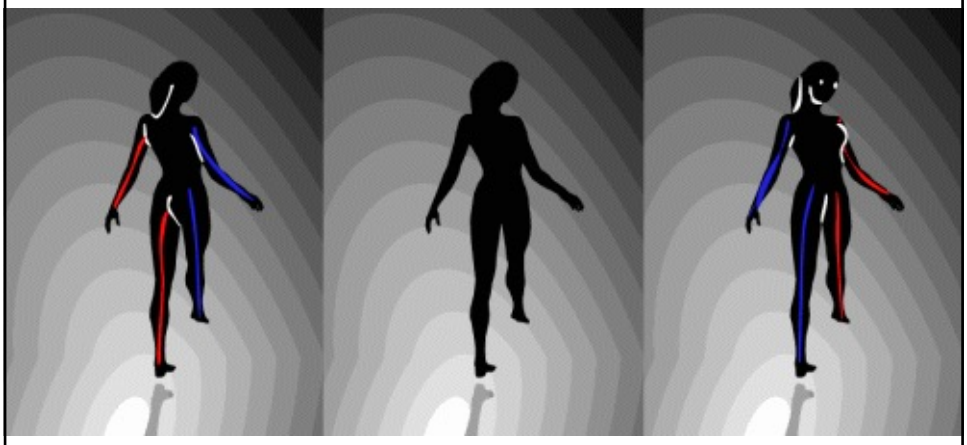


51



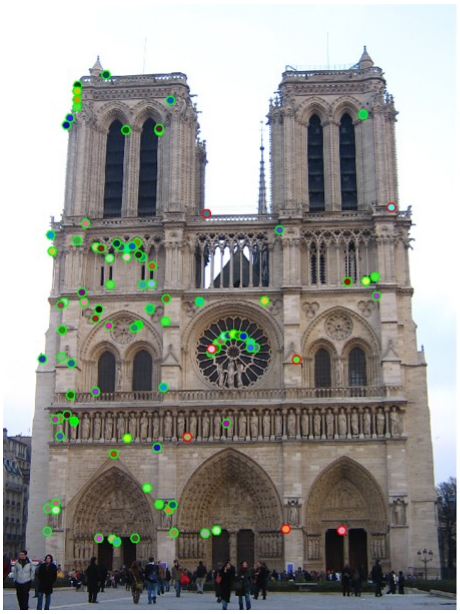
Spinning dancer illusion,
Nobuyuki Kayahara

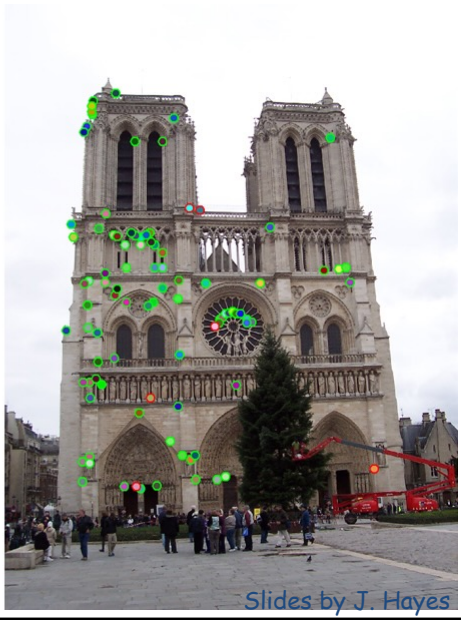
52



53

How do we fit the best alignment?





Slides by J. Hayes

54

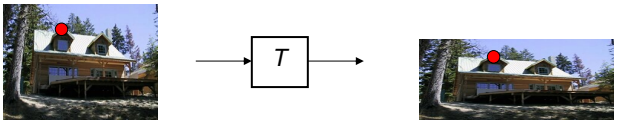
Alignment

- Alignment: find parameters of model that maps one set of points to another
- Typically want to solve for a global transformation that accounts for *most* true correspondences
- Difficulties
  - Noise (typically 1-3 pixels)
  - Outliers (often 50%)
  - Many-to-one matches or multiple objects

Slides by J. Hayes

55

Parametric (global) warping



p = (x,y) **p'** = (x',y')

Transformation T is a coordinate-changing machine:

$p' = T(p)$

What does it mean that *T* is global?
- Is the same for any point p
- can be described by just a few numbers (parameters)

For linear transformations, we can represent T as a matrix

$p' = \mathbf{T}p$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slides by J. Hayes

56

Common transformations

original

Transformed

translation

rotation

aspect

affine

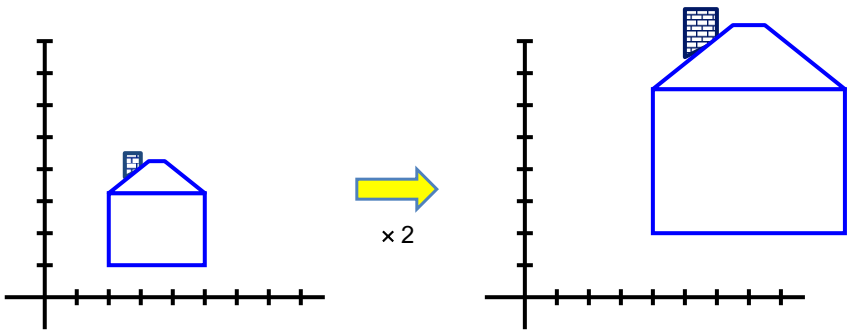
perspective

Slide credit (next few slides): A. Efros and/or S. Seitz

57

Scaling

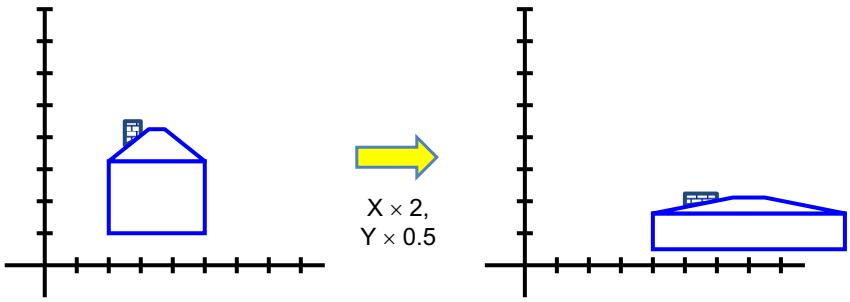
- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



58

Scaling

- *Non-uniform scaling*: different scalars per component:



59

Scaling

- Scaling operation: $x' = ax$

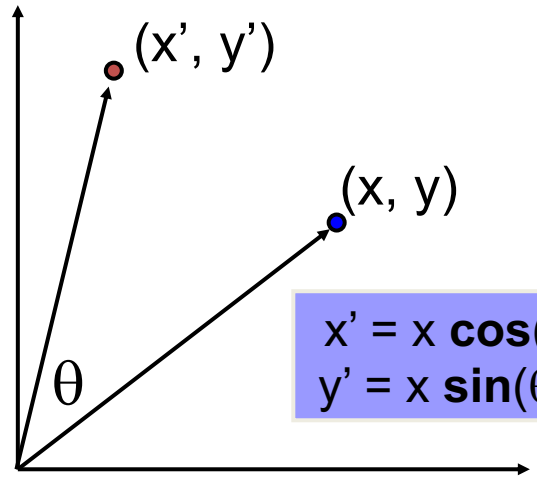
  $y' = by$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

60

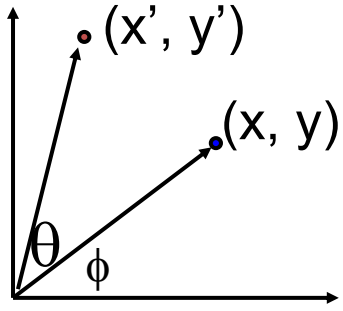
2-D Rotation



$$x' = x\cos(\theta) - y\sin(\theta)$$
$$y' = x\sin(\theta) + y\cos(\theta)$$

61

2-D Rotation



Polar coordinates…

$x = r\cos(f)$
$y = r\sin(f)$
$x' = r\cos(f + \theta)$
$y' = r\sin(f + \theta)$

Trig Identity…

$x' = r\cos(f)\cos(\theta) - r\sin(f)\sin(\theta)$
$y' = r\sin(f)\cos(\theta) + r\cos(f)\sin(\theta)$

Substitute…

$x' = x\cos(\theta) - y\sin(\theta)$
$y' = x\sin(\theta) + y\cos(\theta)$

62

2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though sin(q) and cos(q) are nonlinear functions of q,

- ***x' is a linear combination of x and y***
- ***y' is a linear combination of x and y***

What is the inverse transformation?

- Rotation by -q
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^T$

63

Basic 2D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, shear

64

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

or

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

65

Projective Transformations

Projective transformations are combos of

- Affine transformations, and
- Projective warps

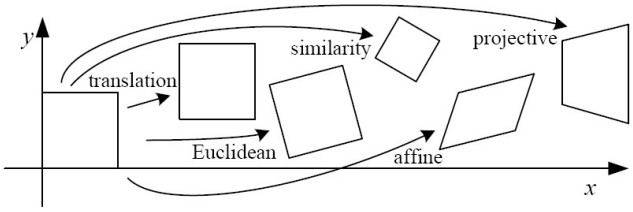
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis
- Projective matrix is defined up to a scale (8 DOF)

66

2D image transformations (reference)

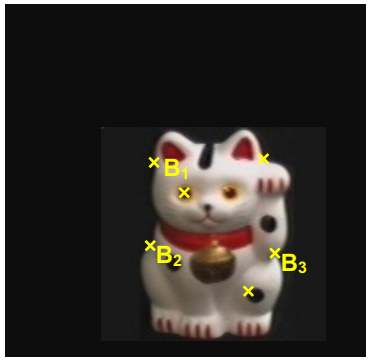


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation + ···	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths + ···	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles + ···	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism + ···	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

Szeliski 2.1

67

Example: solving for translation



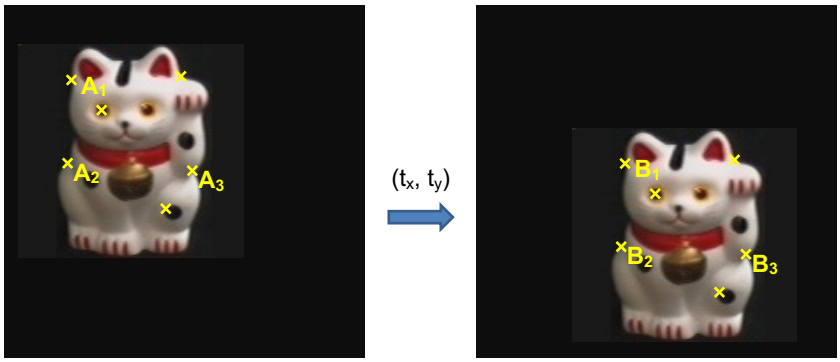
Given matched points in {A} and {B}, estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Slides by J. Hayes

68

Example: solving for translation



Least squares solution

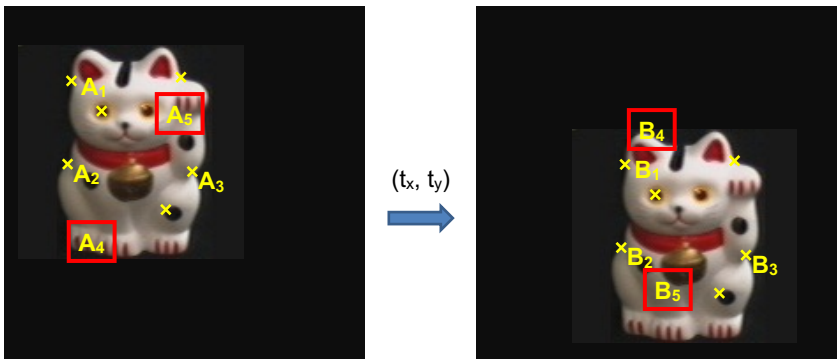
1. Write down objective function
2. Derived solution
   a) Compute derivative
   b) Compute solution
3. Computational solution
   a) Write in form Ax=b
   b) Solve using pseudo-inverse or eigenvalue decomposition

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

69

Example: solving for translation



Problem: outliers

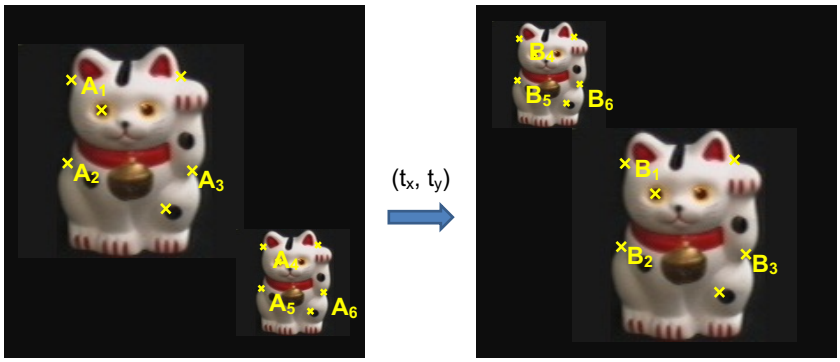
RANSAC solution

1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

70

Example: solving for translation



Problem: outliers, multiple objects, and/or many-to-one matches

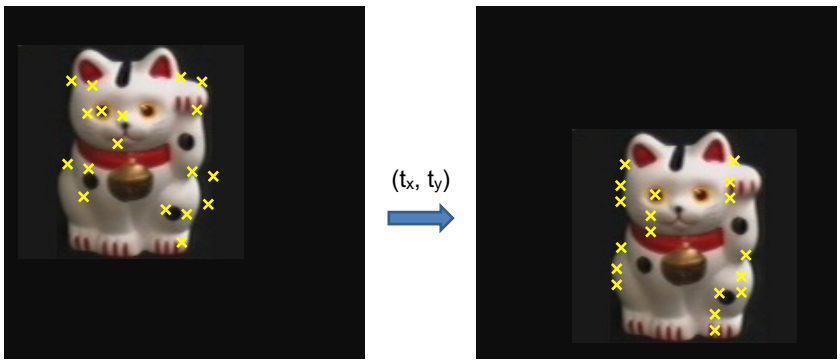
Hough transform solution

1. Initialize a grid of parameter values
2. Each matched pair casts a vote for consistent values
3. Find the parameters with the most votes
4. Solve using least squares with inliers

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

71

Example: solving for translation



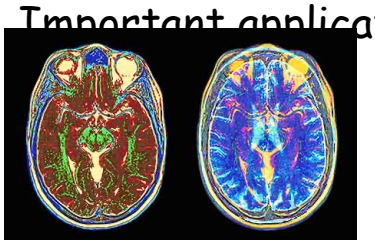
Problem: no initial guesses for correspondence

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

72

What if you want to align but have no prior matched pairs?

- Hough transform and RANSAC not applicable
- Important applications



Medical imaging: match brain scans or contours

Robotics: match point clouds

73

Iterative Closest Points (ICP) Algorithm

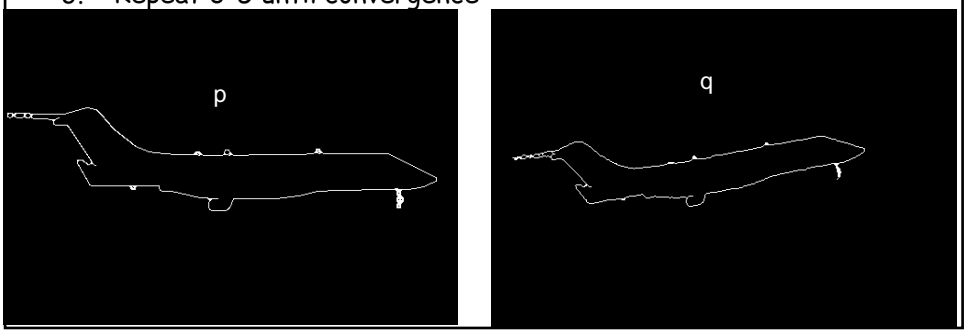
Goal: estimate transform between two dense sets of points

1. **Initialize** transformation (e.g., compute difference in means and scale)
2. **Assign** each point in {Set 1} to its nearest neighbor in {Set 2}
3. **Estimate** transformation parameters
   - e.g., least squares or robust least squares
4. **Transform** the points in {Set 1} using estimated parameters
5. **Repeat** steps 2-4 until change is very small

74

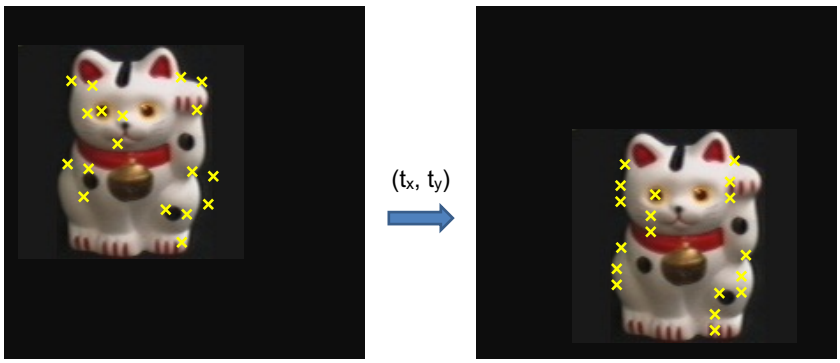
Example: aligning boundaries

1. Extract edge pixels $p_1..p_n$ and $q_1..q_m$
2. Compute initial transformation (e.g., compute translation and scaling by center of mass, variance within each image)
3. Get nearest neighbors: for each point p_i find corresponding $\text{match(i)} = \underset{j}{\text{argmin}}\, dist(pi, qj)$
4. Compute transformation ***T*** based on matches
5. Warp points ***p*** according to ***T***
6. Repeat 3-5 until convergence



75

Example: solving for translation



Problem: no initial guesses for correspondence

ICP solution

1. Find nearest neighbors for each point
2. Compute transform using matches
3. Move points using transform
4. Repeat steps 1-3 until convergence

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

76

Algorithm Summaries

- Least Squares Fit
  - closed form solution
  - robust to noise
  - not robust to outliers
- Robust Least Squares
  - improves robustness to outliers
  - requires iterative optimization
- Hough transform
  - robust to noise and outliers
  - can fit multiple models
  - only works for a few parameters (1-4 typically)
- RANSAC
  - robust to noise and outliers
  - works with a moderate number of parameters (e.g, 1-8)
- Iterative Closest Point (ICP)
  - For local alignment only: does not require initial correspondences

77

Active or deformable models

- Deformable models represent
  - class of objects of differing shape (bananas)
  - objects which change shape (such as lips)
- Deformable models may be
  - 3D surfaces, (a balloon squeezed out of shape),
  - 3D space curves, which we bend to form figures,
  - 2D contours, eg. the Snake.

78

Deformable Contours

- The idea behind deformable contours is to find a contour c(s) which best approximates the perimeter of an object
- The approach is to construct an *energy functional* which measures the appropriateness of a contour and to optimize this functional with respect to the contour parameters

79

Energy Functional

$$E = \int (\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})\,ds$$

$$E_{cont} = \left\| \frac{dc}{ds} \right\|^2$$

$$E_{curv} = \left\| \frac{d^2c}{ds^2} \right\|^2$$

$$E_{image} = -\|\nabla I\|^2$$

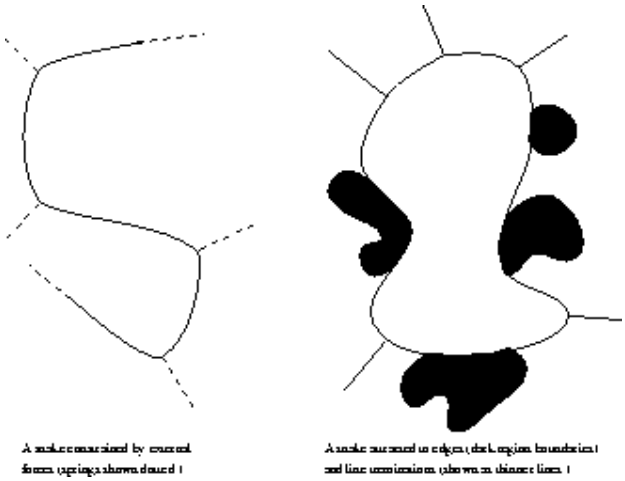
80

Energy Functional

- Internal forces
  - a(s) is a measure of the elasticity of the snake.
  - whereas β(s) is a measure of the stiffness of the snake.
- External forces
  - image features such as contours
  - higher level understanding of the general shape
  - user-applied energy.

81

Energy Functional



82

Discrete Version

- In the discrete version the contour is represented by a sequence of point locations $\mathbf{p}_i$

83

Discrete Functional

$$E = \sum_{i=1}^{N} (\alpha_i E_{cont} + \beta_i E_{curv} + \gamma_i E_{image})$$

$$E_{cont} = (d - \|p_i - p_{i-1}\|)^2$$

$$E_{curv} = \|p_{i-1} - 2p_i + p_{i+1}\|^2$$

$$E_{image} = -(I_x^2 + I_y^2)$$

84

Greedy Optimization

- On each iteration every point is moved to a location in its immediate neighbourhood which minimizes the associated energy functional
- This process is repeated until the contour points converge to a stable configuration

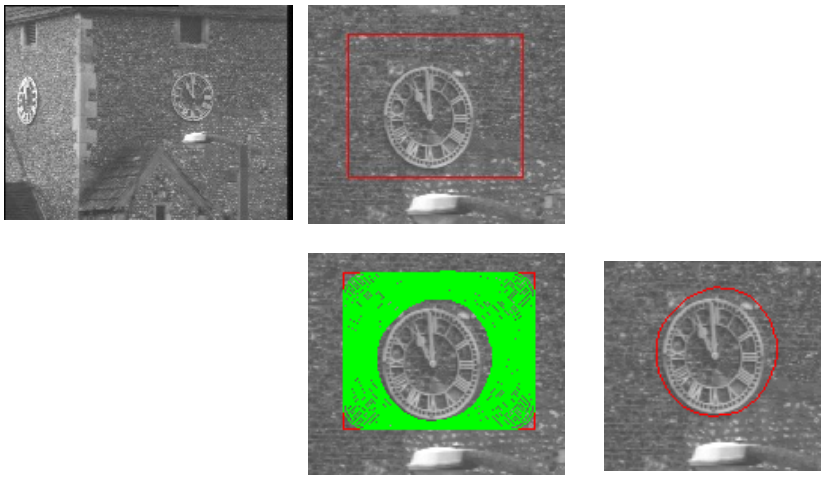
85

Other Optimization Methods

- gradient descent, rather crude, requires small movements
- Dynamic programming
- Bayesian techniques

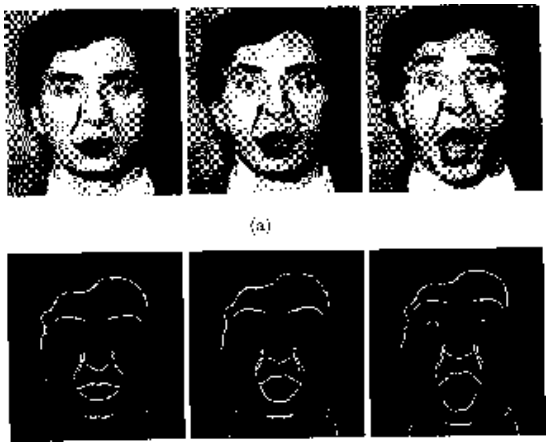
86

Example



87

Example



(a)

88

Advantages of Snakes

- Controlled interactively by using appropriately placed springs and volcanoes.
- Easy to manipulate (intuitive)
- Autonomous and self-adapting
- Sensitive to image scale by Gaussian smoothing in the image energy function.
- Insensitive to noise and other ambiguities in the images.
- They can be used to track dynamic objects in temporal as well as the spatial dimensions.

89

Disadvantages of Snakes

- Often get stuck in local minima states
  - overcome by simulated annealing techniques at the expense of longer computation times.
- Often overlook minute features in the process of minimizing the energy over the entire path of their contours.
- Their accuracy is governed by the convergence criteria used in the energy minimization technique;
  - higher accuracies require tighter convergence criteria and hence, longer computation times.

90

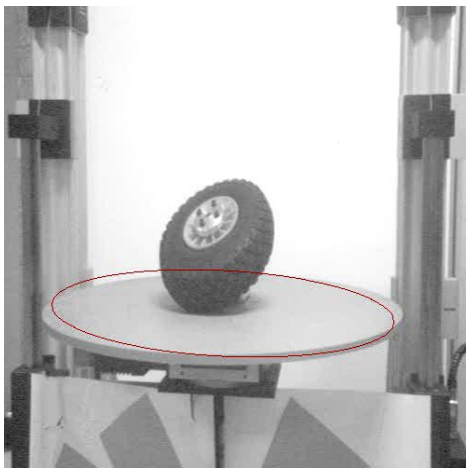
Other Types of Snakes

- balloon force (by user)
- attractor and tangent hard constraints
- Spline-based Deformable Template Models
- Analytical Form-based Parametric Deformable Models
- Prototype-based Parametric Deformable Models
- Level Sets

91

Other Types of Snakes: Parametric

- Few global parameters only, eg. An ellipse



92

Other Types of Snakes: B-splines

$$\mathbf{V}(t) = [f_x(t)\ f_y(t)] = \sum_{i=0}^{k-m-1} \mathbf{c}_i \mathcal{B}_i^m(t), \quad t \in [t_m, t_{k-m}],$$

93

Notation for Curve Evolution

- $\Gamma(t)$ is evolving curve
  $\Gamma(0)$ is a simple smooth closed curve

Position vector: $\vec{C}(p,t) = (x(p,t), y(p,t))$

Velocity Vector:

$$\vec{C}_p = \frac{\partial \vec{C}(p,t)}{\partial p}$$

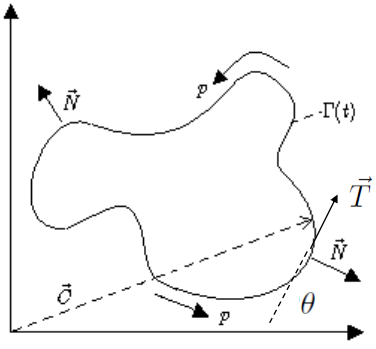
Tangent vector: $\vec{T}(p,t) = \dfrac{\vec{C}_p}{|\vec{C}_p|}$

Arc-length parameterization:

$$s(p,t) = \int_0^p |\vec{C}_p(\xi,t)| d\xi$$

$$|\vec{C}_s| = 1$$

Curvature: $\kappa(p,t) = \dfrac{\partial \theta}{\partial s}$



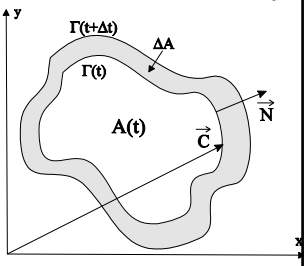
94

Curve evolution

- Each point on the curve moves based on its own velocity

$$\frac{\partial \vec{C}(p,t)}{\partial t} = V(p,t)\vec{N}(p,t)$$

$$\frac{\partial \vec{C}}{\partial t} = V\vec{N}$$



- Geometric heat equation

$$\frac{\partial \vec{C}(p,t)}{\partial t} = -\kappa(p,t)\vec{N}(p,t)$$

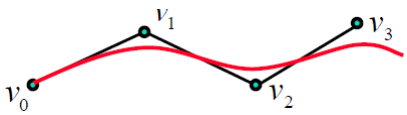
$$\kappa(p,t) = \frac{\partial \theta}{\partial s}$$

95

95

Contour Representation

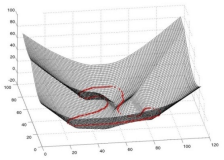
- Parametric (`Lagrangian')
  - E.g. 20-30 points + spline interpolation



  - Point motion -> curve motion

- Non-parametric ('Eulerian')
  - Curve= zero-set of embedding function

$$\Phi(x,y,t) : \{\Phi(x,y,t) = 0\} = \Gamma(t)$$

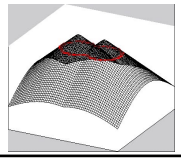


  - Differential geometry:

$$\mathcal{N} = \frac{\nabla\Phi}{|\nabla\Phi|}$$

$$\kappa = \frac{\Phi_{xx}\Phi_y^2 - 2\Phi_{xy}\Phi_x\Phi_y + \Phi_{yy}\Phi_x^2}{(\Phi_x^2 + \Phi_y^2)^{\frac{3}{2}}}$$

$$\mathcal{N} = -\frac{\nabla\Phi}{|\nabla\Phi|}$$



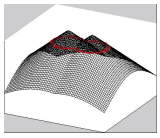
96

Level Set Methods for Curve Evolution

- **Contour representation:** $\Phi(x,y,t) : \{\Phi(x,y,t) = 0\} = \Gamma(t)$
- **Equation** $\Phi(\Gamma(t),t) = 0$ **holds for all t**

$$\nabla\Phi(\Gamma)\cdot\frac{\partial\Gamma}{\partial t} + \frac{\partial\Phi(\Gamma(t),t)}{\partial t} = 0 \qquad \frac{\partial\Gamma}{\partial t} = V\mathcal{N}$$

$$\frac{\partial\Phi(\Gamma(t),t)}{\partial t} = -\nabla\Phi(\Gamma)V\cdot\mathcal{N} \qquad \mathcal{N} = -\frac{\nabla\Phi}{|\nabla\Phi|}$$



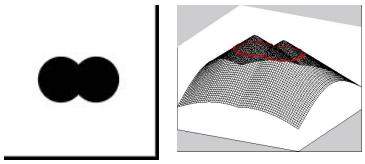
$$\frac{\partial\Phi}{\partial t} = V|\nabla\Phi|$$

$$\frac{\partial\Gamma}{\partial t} = V\mathcal{N}$$

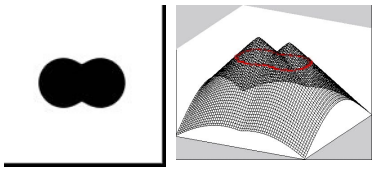
97

97

Curve Evolution using Level Set methods



$$\frac{\partial\Gamma}{\partial t} = -c\mathcal{N}$$



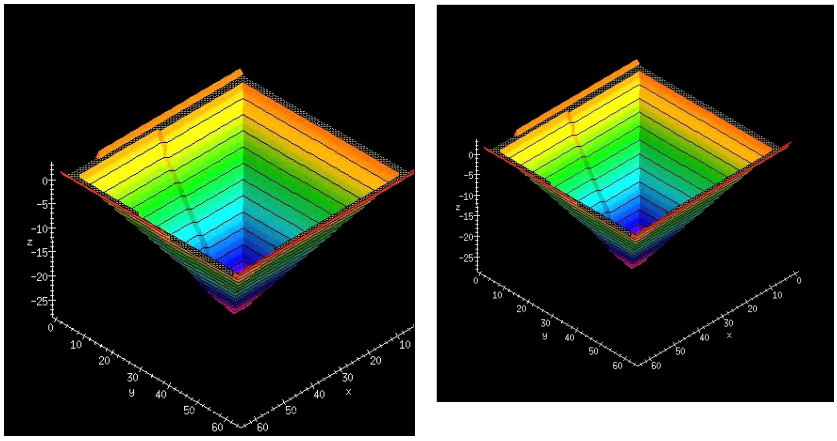
$$\frac{\partial\Gamma}{\partial t} = -\kappa\mathcal{N}$$

- **Pros**
  - **Stability**
  - **Topogical Flexibility**
  - **Treatment of corners, junctions**
  - **Accurate estimates of normal and curvature**
- **Cons**
  - **Time demanding**
  - **Topological flexibility (potentially)**

98

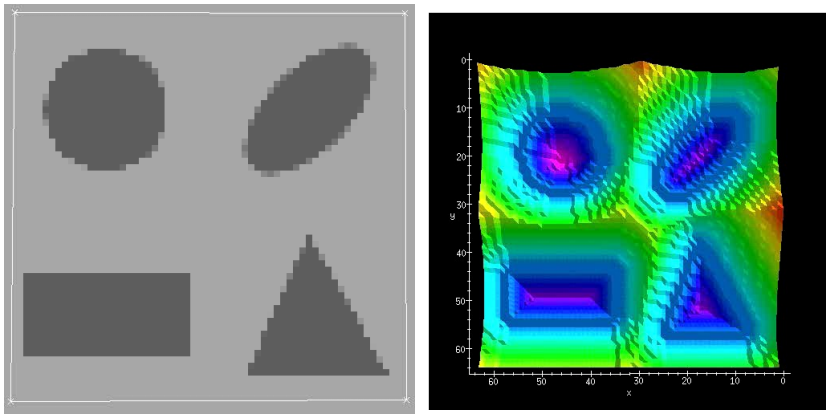
98

Level Sets



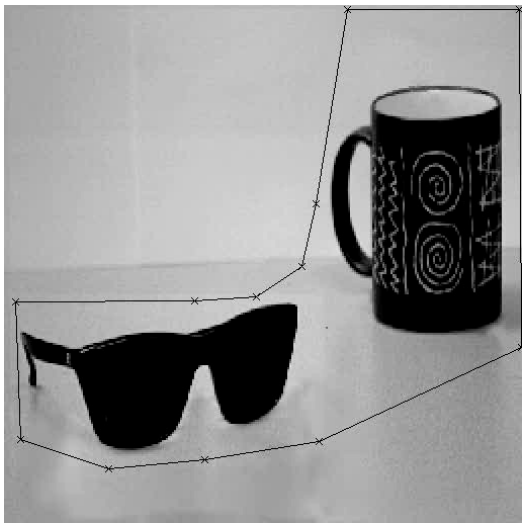
99

Level Sets II



100

Level Sets III



101

3D deformable models

- Generalization in 3D
- Global and local deformations
- Lagrange dynamics of motion

102

Calculus of variations

- **Minimize functional (generalized 'energy')**

$$E[u] = \iint_D F(x, y, u, u_x, u_y)\,dxdy$$

- **Euler PDE:**

$$\frac{\partial u}{\partial t} = \underbrace{F_u - \frac{\partial}{\partial x} F_{u_x} - \frac{\partial}{\partial y} F_{u_y}}_{\text{Euler derivative } [F]_u} = 0$$

- **Gradient Descent:**

$$\frac{\partial u}{\partial t} = -[F]_u$$

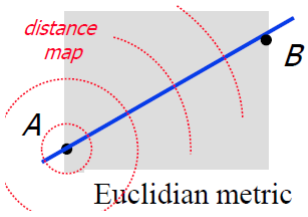
- **Solution reached at steady state:** $\partial u / \partial t = 0$

- **Computer Vision: Energy penalizes undesirable solutions**

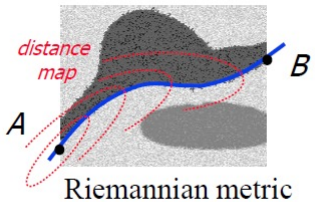
103

Image dependent metric

- **Shortest curve between two points is a geodesic**

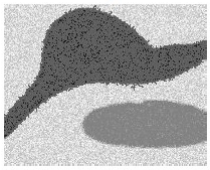


Euclidian metric

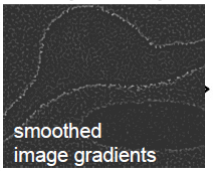


Riemannian metric

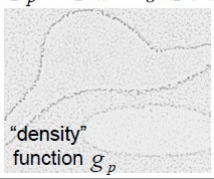
Image I

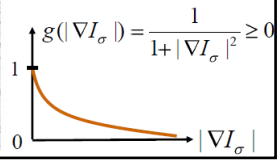


$\nabla I_\sigma = \nabla * H_\sigma * I$



$g_p = g(|\nabla I_\sigma(p)|)$





104

Geodesic Active Contours

Caselles, Kimmel, Shapiro 1997

- Euclidean length functional:

$$J(\Gamma) = \int_0^p 1 ds = \int_0^1 |C_p| dp$$

- Riemannian length functional

$$J[\Gamma] = \int_0^p g(|\nabla I(\vec{C}(s)|) ds$$

  - Image derived metric
  - `Geodesics' (Shortest paths): along boundaries

- Euler-Lagrange equations:

$$\frac{\partial \Gamma}{\partial t} = g(|\nabla I|)\kappa \mathcal{N} - (\nabla g(|\nabla I|) \cdot \mathcal{N})\mathcal{N}$$

105

Kass, Witkin & Terzopoulos: Snakes, 1987

- Snakes functional:

$$J(\Gamma) = \alpha \int_0^1 E_{int}(\vec{C}(p)) dp + \beta \int_0^1 E_{img}(\vec{C}(p)) dp + \gamma \int_0^1 E_{con}(\vec{C}(p))$$

$$E_{int} = a|\vec{C}_p|^2 + b|\vec{C}_{pp}|^2$$

$$E_{edge} = -\lambda |\nabla I(\vec{C})|^2$$

- Geodesic Active Contours functional:

$$J[\Gamma] = \int_0^p g(|\nabla I(\vec{C}(s)|) ds$$

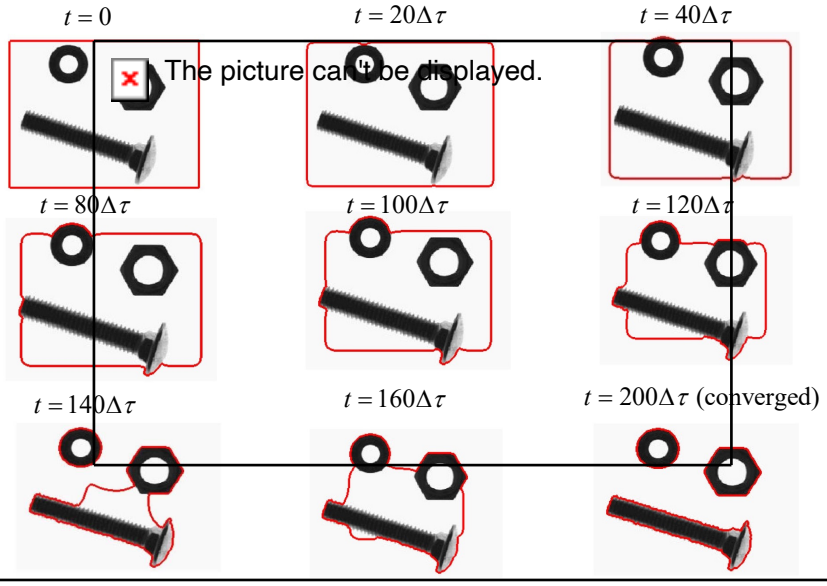
  - Intrinsic geometric formulation
  - Parametrization independent criterion

106

106

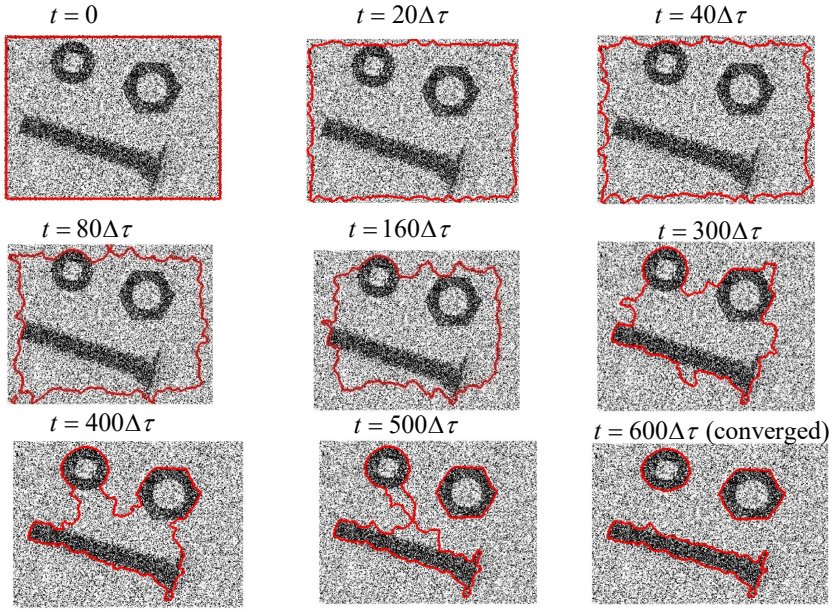
Geodesic Active Contours

$$\frac{\partial \Gamma}{\partial t} = g(|\nabla I|)\kappa\mathcal{N} - (\nabla g(|\nabla I|) \cdot \mathcal{N})\mathcal{N}$$



107

Geodesic Active Contours



108

Region-based: Mumford-Shah functional

- D. *Mumford* and *J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems*. Comm. Pure Appl. Math., 42:577--685, 1989

$$J[U,\Gamma] = \iint_{\Omega} (I-U)^2 dxdy + \alpha \iint_{\Omega\setminus\Gamma} |\nabla U|^2 dxdy + \beta|\Gamma|$$

- 1st term: fidelity of approximation, U to image, I
- 2nd term: smoothness of U
- 3rd term: short boundaries

I U Γ

109

Some insight into the MS functional

$$J[U,\Gamma] = \iint_{\Omega} (I-U)^2 dxdy + \alpha \iint_{\Omega\setminus\Gamma} |\nabla U|^2 dxdy + \beta|\Gamma|$$

- **Compare with previous functionals**

$$J[U] = \iint_{\Omega} |\nabla U|^2 dxdy$$

$$J[U] = \iint_{\Omega} |\nabla U| dxdy = \iint_{\Omega} \frac{1}{|\nabla U|} |\nabla U|^2 dxdy$$

$$J[\Gamma] = \int_0^p 1 ds = |\Gamma|$$

$$J[\Gamma] = \int_0^p g(|\nabla I(\vec{C}(s)|) ds$$

110

110

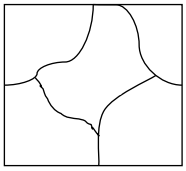
Cartoon Approximation to MS functional

$$E(U,\Gamma) = \iint_R (I-U)^2 dxdy + \alpha \iint_{R-\Gamma} \|\nabla U\|^2 dxdy + \beta|\Gamma|$$

- Consider $\alpha \rightarrow \infty$
- Piecewise Smooth -> Piecewise constant (`Cartoon')
- New functional

$$J[U,\Gamma] = \sum_{i=1}^{K} \iint_{R_i} (I-c_i)^2 dxdy + \frac{\beta}{2}|\partial R_i|$$

$$U(x,y) = \sum_{i=1}^{K} H_{R_i}(x,y)c_i$$



- What does this remind you of?

111

111

Cartoon approximation to MS

- Effect of modifying the length weight, β

$$E(U,\Gamma) = \iint_R (I-U)^2 dxdy + \alpha \iint_{R-\Gamma} \|\nabla U\|^2 dxdy + \beta|\Gamma|$$

$\alpha \rightarrow \infty$

- Decreasing: left to right?

112

112