Università
Ca' Foscari
Venezia

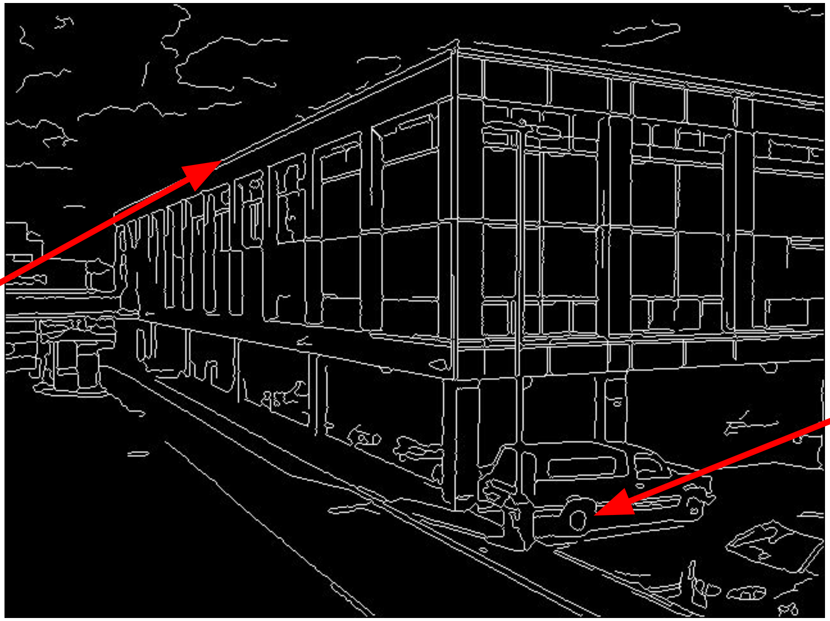
Computer Vision

Finding curves

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco
DAIS, Ca' Foscari University of Venice
Academic year 2017/2018

Finding Curves

Often, we have to work with unstructured environments in which all we have is an edge image and no knowledge about where objects of interest might be.



A line

A circle

Finding lines

For example, suppose that we want to detect street lanes to develop an autonomous vehicle



Option 1: We can limit the analysis to a specific region and do a least-squares fitting of a line

Bad idea! Why?

Finding lines

For example, suppose that we want to detect street lanes to develop an autonomous vehicle

Option 2: We can search for lines at every possible position/orientation

Computationally very expensive

Università
Ca' Foscari
Venezia

Finding lines

For example, suppose that we want to detect street lanes to develop an autonomous vehicle

Option 3: We can use a consensus-based approach: RANSAC

Università Ca' Foscari Venezia

Finding lines

For example, suppose that we want to detect street lanes to develop an autonomous vehicle

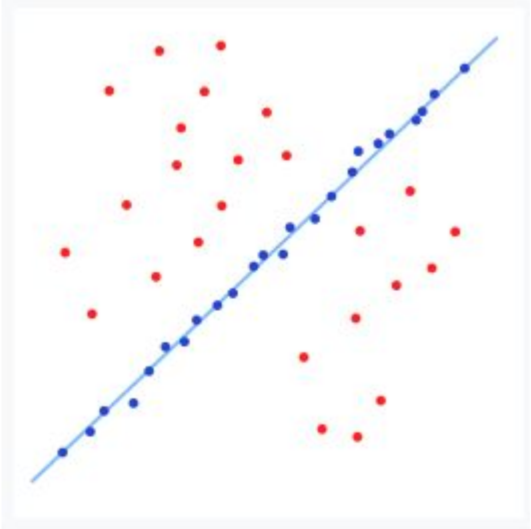
Option 4: We can use a voting scheme:

Hough Transform

RANSAC

The RANSAC algorithm is a learning technique to estimate parameters of a model by random sampling of observed data.

Given a dataset whose data elements contain both inliers and outliers, RANSAC uses a consensus scheme to find the optimal fitting result.



RANSAC

Assumptions:

1. Data consists of **inliers** (i.e., data whose distribution can be explained by some set of model parameters, though may be subject to noise) and **outliers** which are data that do not fit the model

2. Given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data

   > For example given a set of 2 points we can compute a line model that optimally explains the set

Università Ca' Foscari Venezia

RANSAC

Algorithm:

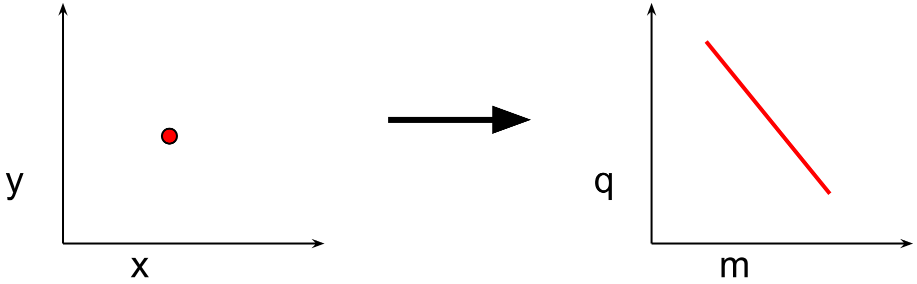
1. Select a random subset of the original data. Call this subset the hypothetical inliers.
2. Fit the model to the hypothetical inliers
3. Test all other data against the model and mark points either as inliers or outliers according to some loss function. The inliers are called “consensus set”
4. Return to step 1 until a predefined number of iterations is reached
5. The model that produced the largest consensus set is returned

Università
Ca' Foscari
Venezia

Hough transform

The basic idea is to map points from the image space to **the parameter space of the model** (for example the m-q space for lines parameterized as y=mx + q)

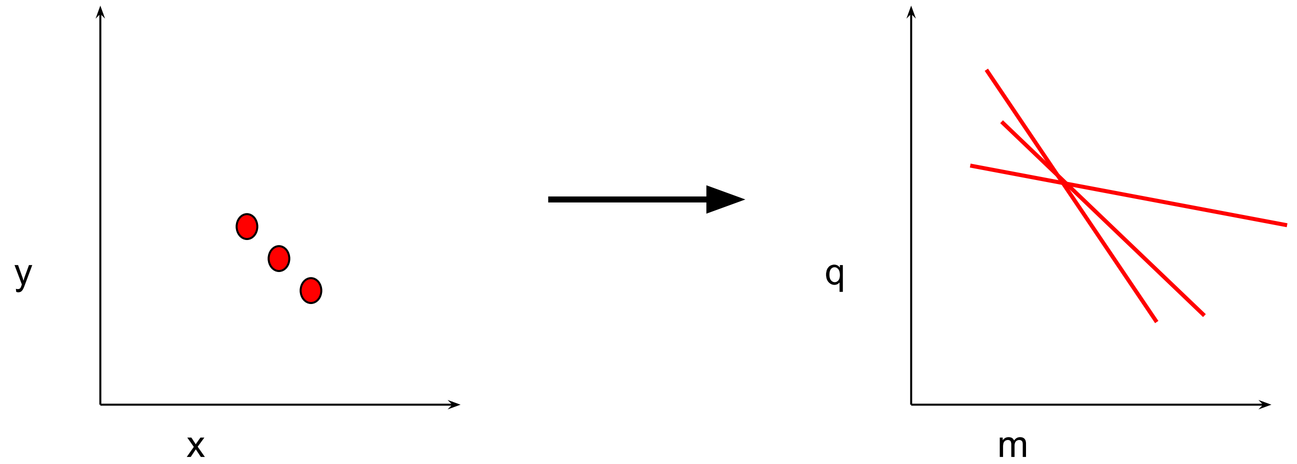
For an image point (x,y) can pass infinite lines all satisfying the equation y=mx+q. The equation can be rewritten as q=-mx+y which corresponds to a line in the m-q space



Hough transform

The principal lines in the image plane could be found by identifying points in parameter space where large numbers of parameter-space lines intersect

The parameter space is used as an accumulator of votes



Hough transform

The classical slope-intercept parameterization of the line is not convenient since the slope approach infinity when the line approaches vertical direction

Normal representation of the line:

$$r = x\cos\theta + y\sin\theta$$

The parameter space is now the r-θ-plane in which the range of values are limited

Hough transform

Università Ca' Foscari Venezia

Algorithm:

1. Initialize H[r,θ]=0
2. For each edge point p=(x,y) in the image
   a. For θ=0 to pi
      i. r = x cos θ + y sin θ
      ii. H[r,θ] += 1
3. Find (r,θ) for which H[r,θ] is maximum
4. The detected line is given by r = x cosθ + y sinθ

Hough Transform

Possible extensions/improvements:

1. Use the image gradient (no need to iterate through angles)
2. Give more votes to strongest edges
3. Change the sampling of (r,θ) to trade-off resolution with computing time
   a. High resolution -> Dispersion of votes
   b. Low resolution -> Cannot distinguish similar lines

Hough Transform

Hough transform is applicable to any function of the form g(**v**, **c**) = 0, where:

- **v** is a vector of coordinates
- **c** is a vector of coefficients

For example the Hough Transform can be used to extract all circles in the scene:

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2$$

A 3D accumulator for (c_1,c_2,c_3) is needed