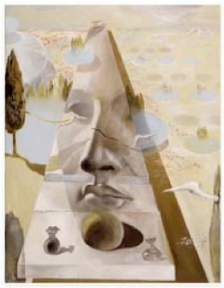
Lecture 9
Fitting and Matching



- Problem formulation
- Least square methods
- RANSAC
- Hough transforms
- Multi-model fitting
- Fitting helps matching!

Reading:
[HZ] Chapter: 4 "Estimation – 2D projective transformation"
Chapter: 11 "Computation of the fundamental matrix F"
[FP] Chapter:10 "Grouping and model fitting"

Some slides of this lectures are courtesy of profs. S. Lazebnik & K. Grauman

In this lecture, we're going to talk about a number of problems related to fitting and matching. We will formulate these problems formally and our discussion will involve Least Squares methods, RANSAC and Hough voting. We will conclude the lecture with a few remarks on how fitting can be often used to solve a matching problem.

Fitting

Goals:

- Choose a parametric model to fit a certain quantity from data
- Estimate model parameters

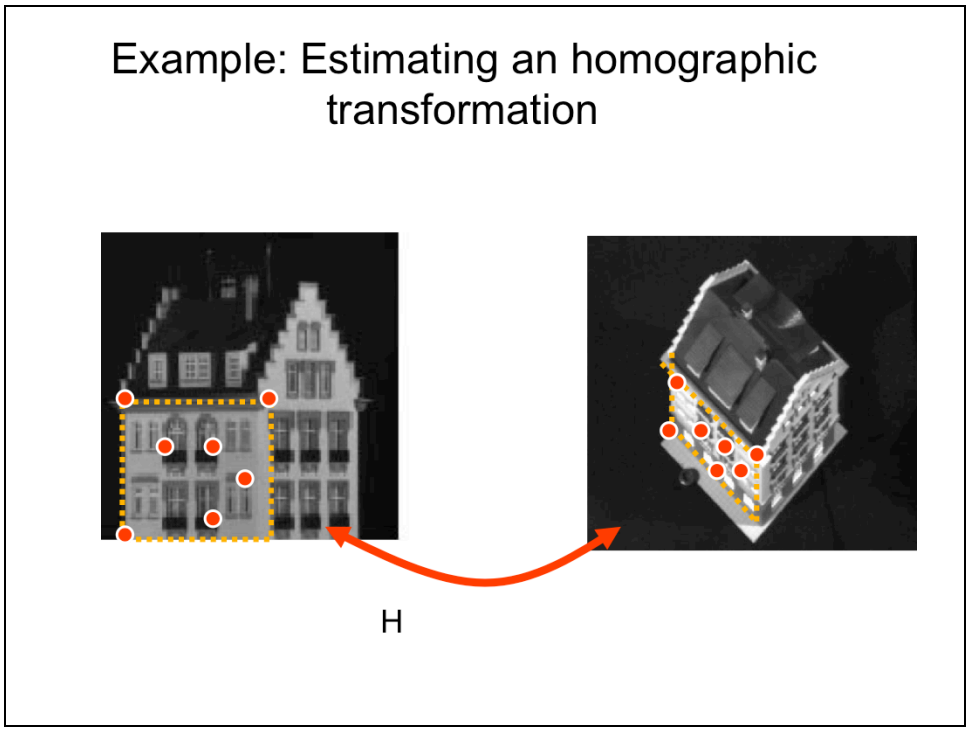
- Lines
- Curves
- Homographic transformations
- Fundamental matrices
- Shape models

Fitting is about trying to fit observed data into a parametric model that we are assuming holds. The process of fitting such model to the data involves estimating the parameters that describe the model such that the fitting error is minimized.

A classic example is line fitting: given a set of points in 2D, the goal is to find the parameters that describe the line so as to best fit the set of 2D points. Similar problems can be defined for other geometrical quantities such as curves, homographic transformations, fundamental matrices or even object shapes.

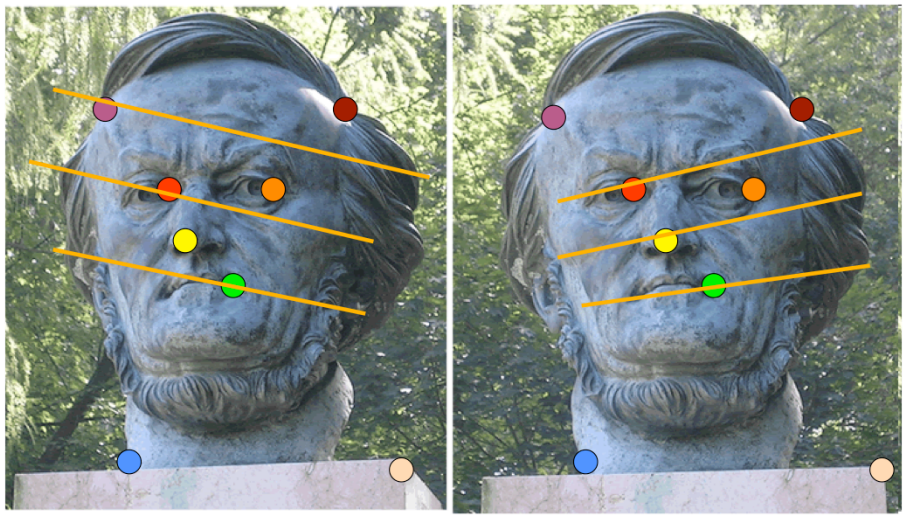
Example: fitting lines
(for computing vanishing points)

The next couple of slides show some examples of models we could be trying to fit our data to. In this example, the goal is fit a line given a number of point observations (red dots). Fitting lines in an image can be useful, for instance, to estimate the scene vanishing points.



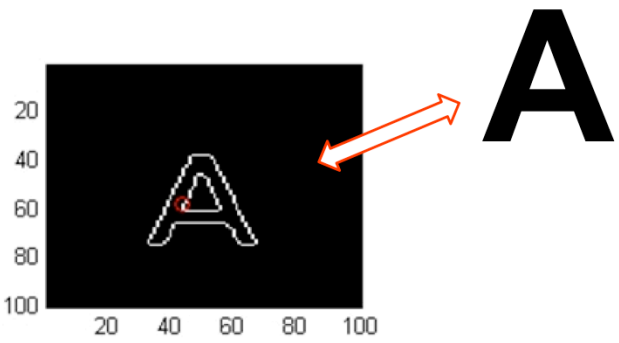
Here is another example, where we have points (red dots) that lie on the same planar region (the façade of the building) and that are observed from two different views. Because they lie on the same planar regions, we expect them to be related by a homographic transformation. So the goal here would be to fit (and estimate) such a homographic transformation, given the set of corresponding points. This homography defines a 2D to 2D mapping. In homogenous coordinates it is described by a 3x3 matrix – thus it is defined by 9 parameters up to scale, thus 8 unknown parameters. Estimating these 8 unknown parameters requires at least 4 point correspondences.

Example: Estimating F



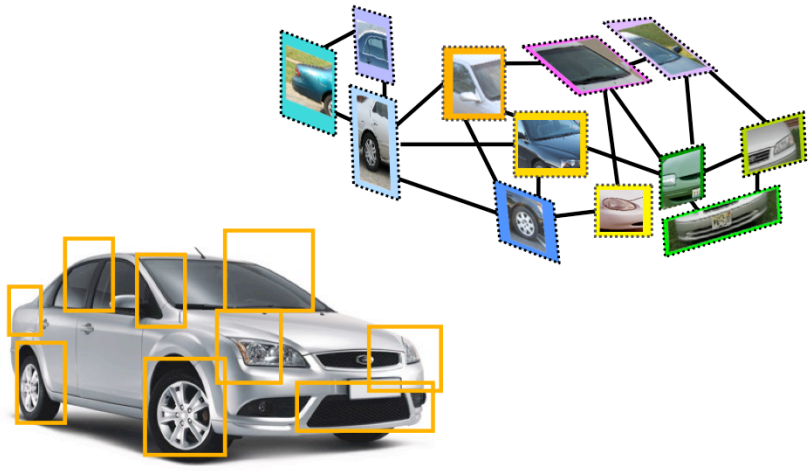
In this example, the goal is fit a fundamental matrix transformation F that relates (at least) 7 point correspondences.

Example: fitting a 2D shape template

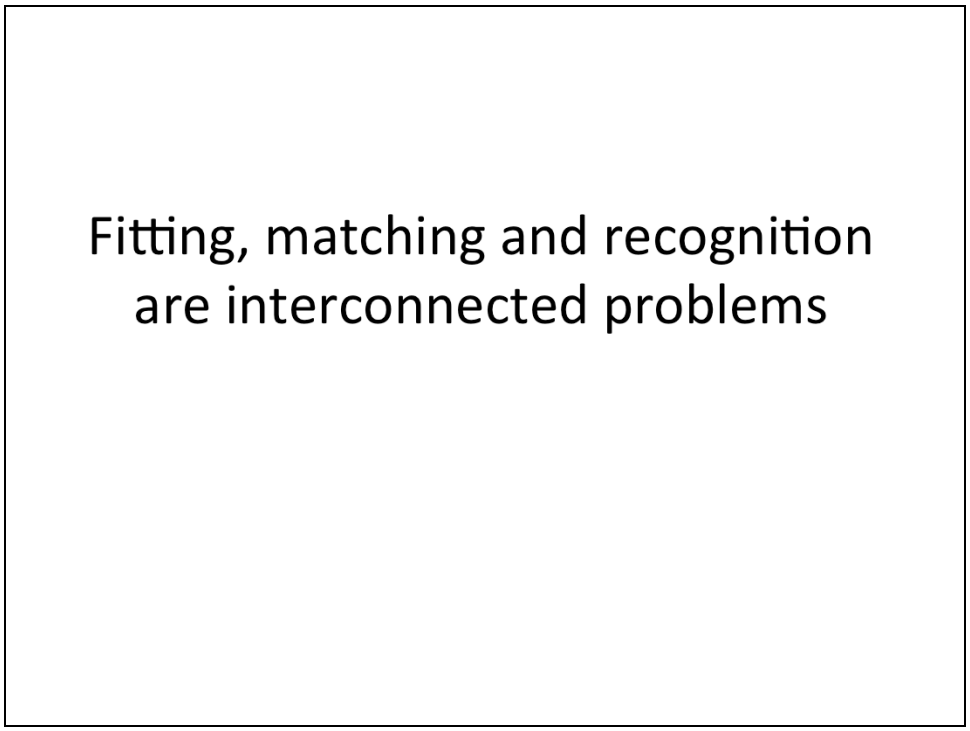


One more example is fitting a shape template (the letter A on the top-right) with an observation (image) of the letter A.

Example: fitting a 3D object model



A final example is fitting a 3D shape template of a car with an observation (image) of the car. Notice that in this case the fitting problem can also be interpreted as a recognition or matching problem.



In fact, fitting, matching and recognition are interconnected problems.

Fitting

Critical issues:
- noisy data
- outliers
- missing data

Fitting becomes problematic in presence of:

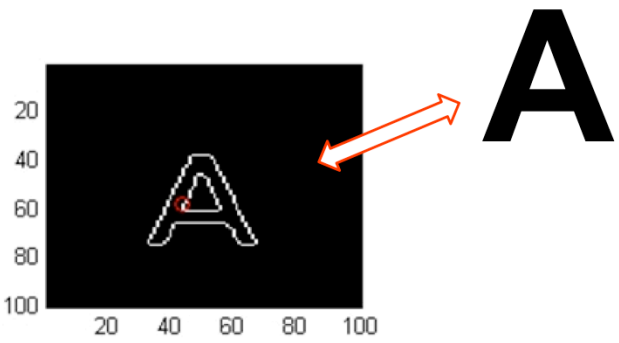
- noisy data
- outliers
- missing data

We discuss this in details in the next few slides.

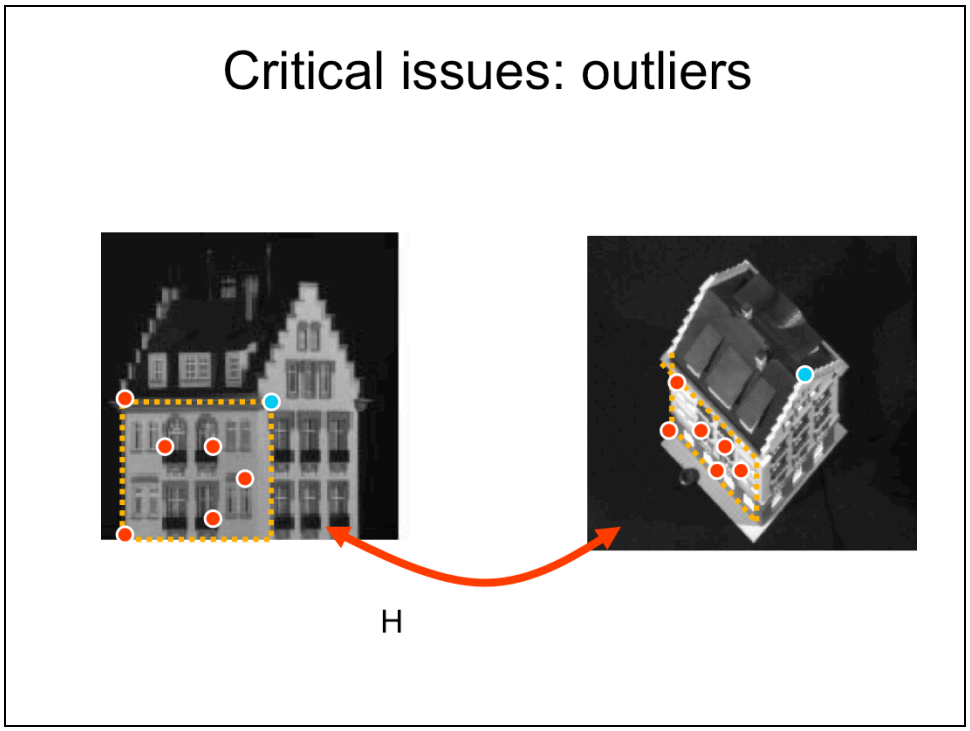
Critical issues: noisy data

Data may not always fit the model exactly because of measurement errors or detection noise.

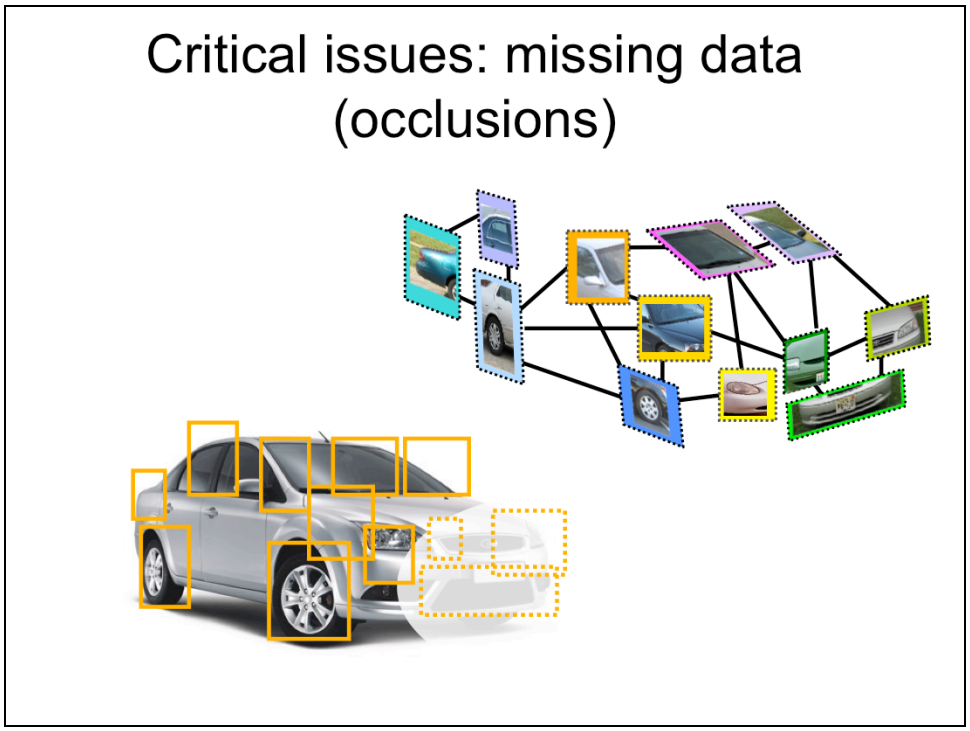
Critical issues: noisy data (intra-class variability)



Also, the model may not fit exactly the data because of intra-class variability – the shape of the template may be different from the observation. This is still true even if there is no measurement noise.



Another class of problems stem from the presence of outliers. An outlier is a data point that is not explained by the model but it still appears among the data point we wish to fit. In this example, the outlier is the pair of corresponding points shown in blue. This pair of corresponding points is clearly incorrect but nevertheless it may be produced by mistake by the algorithm that generates point correspondences. The fitting procedure should somehow be able to account for it.



A third class of problems is due to occlusions. This takes place when some of the points we wish to fit are not visible. Occlusions become a critical issue in shape fitting (or matching) problems whereby the observation of the object we want to match against is not fully visible.

Fitting

Goal: Choose a parametric model to fit a certain quantity from data

Techniques:

- Least square methods
- RANSAC
- Hough transform
- EM (Expectation Maximization) [not covered]

There are several techniques that we can use to fit a parametric model to the data:

- Least square methods
- RANSAC
- Hough transform
- EM (Expectation Maximization)

In this lecture we will explain the details of the first 3 methods; we don't cover the 4th one in this course.

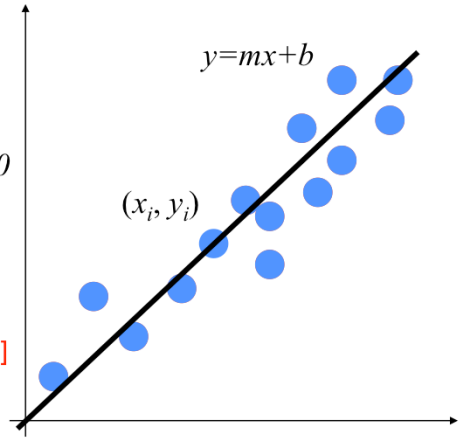
Let's start with the least square methods.

Least squares methods

- fitting a line -

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i - m x_i - b = 0$ [Eq. 1]
- Find (m, b) to minimize

$$E = \sum_{i=1}^{n} (y_i - m x_i - b)^2$$ [Eq. 2]



We illustrate the least square technique using the simple example of line fitting. In this problem, our goal is to fit a parametric model that describes the equation of the line to a series of points labeled as (x_1,y_1), (x_2,y_2),…, (x_n,y_n). Let's model the line as y = mx + b, where m and b are the slope and intercept of the line. Our goal is to estimate the parameters m, b so as to minimize the value of E given in (Eq. 2). E is the sum of residuals or how far off our estimate is for each point. We can interpret this as choosing a line to minimize the distance between the line and the observed data points.

Least squares methods

- fitting a line -

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2 \quad \text{[Eq. 2]}$$

$$E = \sum_{i=1}^{n} \left(y_i - \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right)^2 = \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right\|^2 = \|Y - Xh\|^2 \quad \text{[Eq. 3]}$$

$$= (Y - Xh)^T (Y - Xh) = Y^T Y - 2(Xh)^T Y + (Xh)^T (Xh) \quad \text{[Eq. 4]}$$

Find $h = [m, b]^T$ that minimizes E $\quad \frac{dE}{dh} = -2X^T Y + 2X^T Xh = 0$ [Eq. 5]

$$X^T Xh = X^T Y \quad \text{[Eq. 7]}$$

Normal equation

$$h = \left(X^T X \right)^{-1} X^T Y \quad \text{[Eq. 6]}$$

We can formulate this easily as a least squares problem. Eq. 2 describes the objective function we are trying to minimize. We can re-write this equation in the matrix form expressed in Eq. 3, where h = [m b]^T and X and Y collect the coordinates of the data points as indicated in the equation. By expanding the square of the norm of

|| Y – X h ||, we obtain the expression in Eq. 4.

Our goal is to find the parameters m and b that minimizes E. These can be found by taking the derivative of E with respect to h and equal it to zero (Eq. 5). Solving this equation allows to find h as expressed in Eq. 6.

This solution can be interpreted as the optimal (in the least squares sense) parameters m, b that minimize the residuals, or, equivalently, choosing a line that comes closest to containing all of the points. Eq. 7 is typically called the **normal equation**.

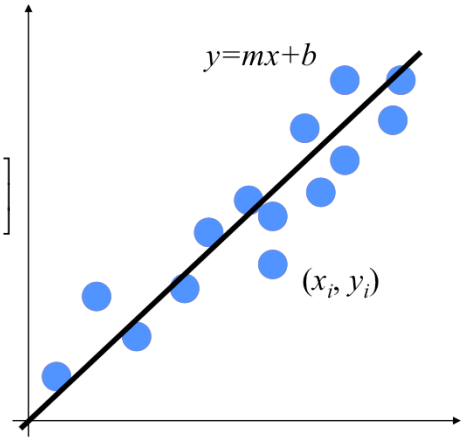
Least squares methods

- fitting a line -

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$$h = \left(X^T X\right)^{-1} X^T Y \qquad h = \begin{bmatrix} m \\ b \end{bmatrix}$$

[Eq. 6]



Limitations

- Fails completely for vertical lines

There is a major limitation to this simple method – it fails completely for vertical lines. In this case m would be infinity (or some very large number) which may introduce numerical error in our solutions.

Least squares methods

- fitting a line -

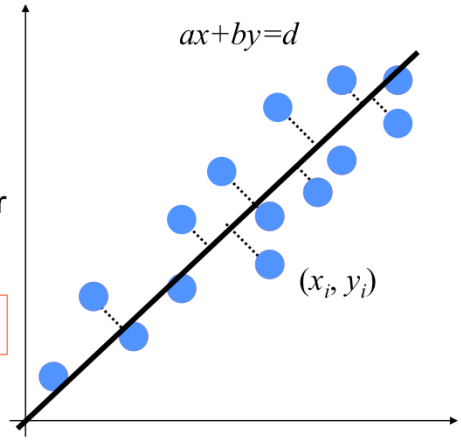
- Distance between point (x_n, y_n) and line $ax+by=d$
- Find (a, b, d) to minimize the sum of squared perpendicular distances

[Eq. 8]

$$E = \sum_{i=1}^{n} (ax_i + by_i - d)^2$$

$$Ah = 0 \quad \text{[Eq. 9]}$$

data model parameters



To remedy this issue, we can use a very similar set up with a small tweak. Instead of the slope-intercept form of a line y = mx + b, we can parameterize our line equation using ax + by = d. This new parameterization leads us to a new expression for E which is expressed in Eq. 8. This is very similar to Eq. 1 earlier, but parameterized to avoid having infinite slope. It can be shown that minimizing E with respect to a, b and d is equivalent to solving the homogenous system expressed in Eq. 9, where h =[a b d]^T and the matrix A collects all the coordinates of the data points. We leave the details of this derivation as an exercise. We can solve Eq. 9 using SVD.

Least squares methods

- fitting a line -

$A\,h = 0$ A is rank deficient

$\text{Minimize} \| A\,h \| \quad \text{subject to} \quad \| h \| = 1$

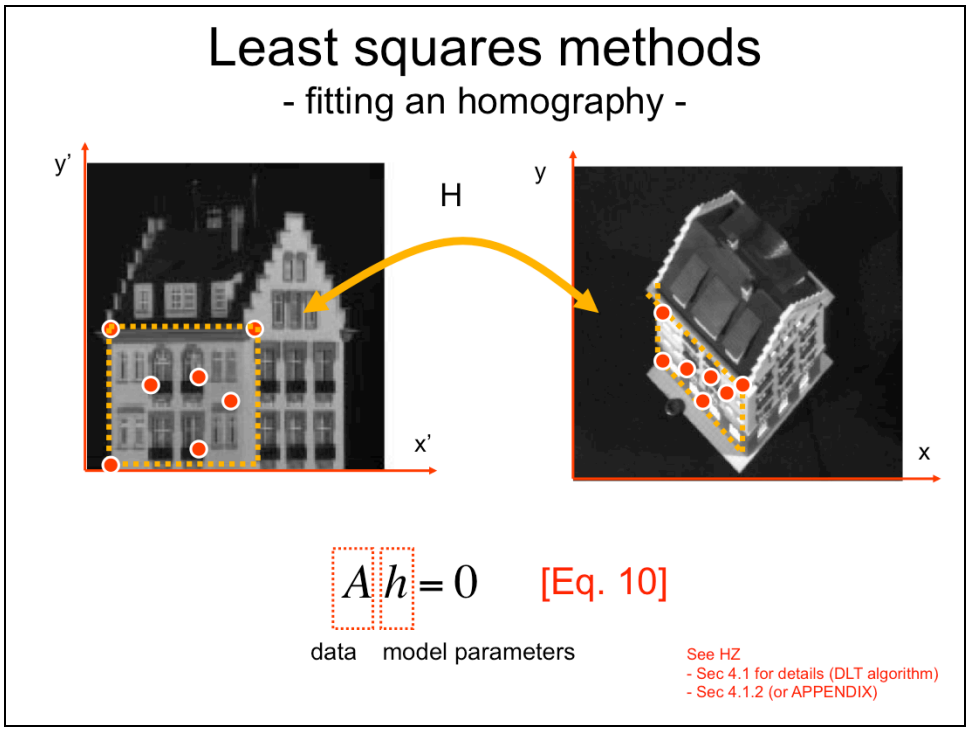
$A = UDV^T$

h = last column of V

As we have seen in previous lectures, the solution h of the system in Eq. 9 can be computed as the last column of V, where U*D*V' = svd(A). For details, see [HZ], Sec. A5.3 - page 593

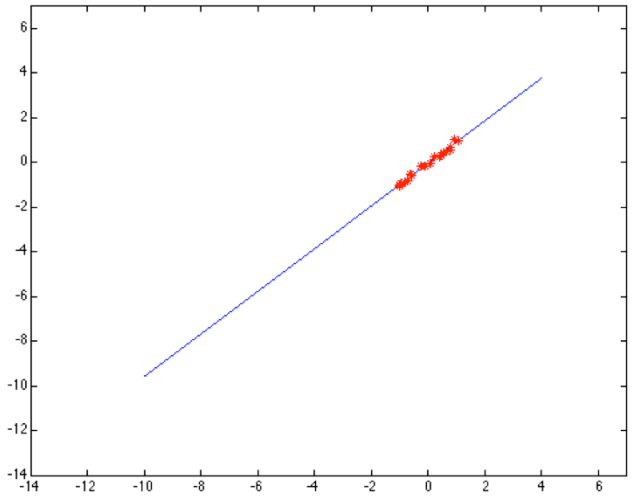
A few more details kindly offered by Phil:

When we take the SVD of a matrix, we are separating the matrix into three pieces, U (directions of output), D (a diagonal matrix with scaling amounts for different directions), and V (directions of input). In this case, input to the matrix means being multiplied by a vector on the right. We can interpret the SVD as when we have an input to A that is in the direction of the nth column of V, it will get scaled by the nth value of D, and be output as the nth column of U. Thus, when the matrix of scaling factors D is sorted (as is custom/ what matlab does), the last column of V dictates the input direction of A that will have the least output. Therefore, that is the direction of input to A that will get the output closest to zero and is the vector that will minimize Ah.



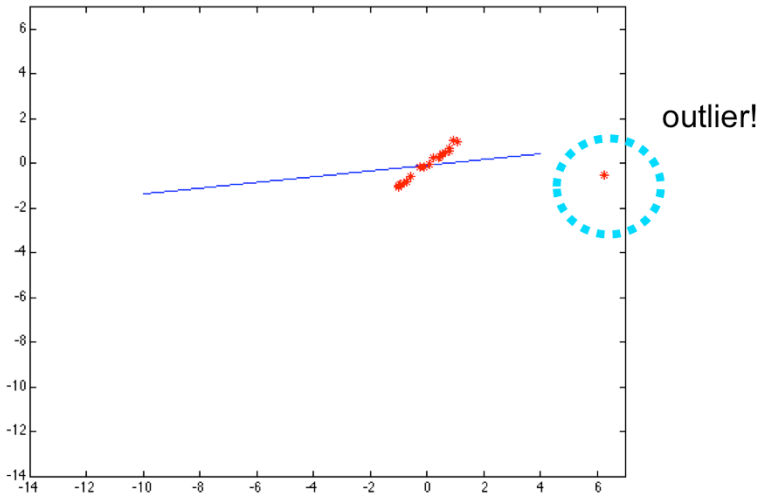
Another popular problem is the one of fitting the homographic transformation H to pairs of corresponding points (see figure). It can be shown that finding the H that best fits all the data is equivalent to solving the homogenous system in Eq. 10, where A collects all the data points (i.e., coordinates of corresponding points in the two images) and h collects the 9 unknown parameters that describe the homographic transformation H (i.e. the coefficient of H). Remember that H has 8 degrees of freedom in that it is known up to scale. As we did before, h can be found in the least square sense, through SVD.

Least squares: Robustness to noise



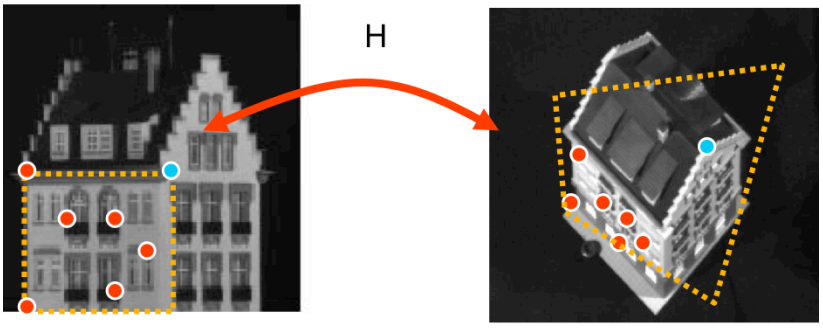
Least squares methods can work well when the data are noisy. The figure shows the result of a simulation where we used least square to fit all the red points with a line.

Least squares: Robustness to noise



However, least square is in general not very robust to the presence of outliers as the result shown here demonstrates. The outlier marked in blue significantly alters the quality of the fitting process.

Critical issues: outliers



CONCLUSION: Least square is not robust w.r.t. outliers

Another example is shown here for the case of fitting homographic transformations.

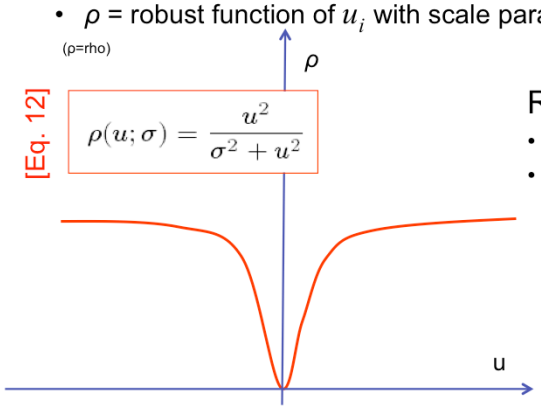
Least squares: Robust estimators

Instead of minimizing $E = \sum_{i=1}^{n} (ax_i + by_i - d)^2$ [Eq. 8]

We minimize $E = \sum_i \rho(u_i ; \sigma)$ [Eq. 11] $\quad u_i = ax_i + by_i - d$

- u_i = error (residual) of i[th] point w.r.t. model parameters h = (a,b,d)
- ρ = robust function of u_i with scale parameter σ

(ρ=rho)



[Eq. 12] $\rho(u;\sigma) = \frac{u^2}{\sigma^2 + u^2}$

Robust function ρ:
- When u is large, ρ saturates to 1
- When u is small, ρ is a function of u²

In conclusion:
- Favors a configuration with small residuals
- Penalizes large residuals

To make our estimator less sensitive to the presence of outliers, we can use a slightly modified cost function – which is called **robust estimator**. Instead of minimizing the sum of the squares of the residuals, we look to minimize Eq. 11, where the u_i's are the original residuals that we have been using and $\rho(u_i; \sigma)$ is a robust function of the residuals and of the scale parameter σ (Eq. 12). In the lower part of the slide, as an example, we see a plot of $\rho(u_i; \sigma)$ as function of u and σ. This function is more robust in that it penalizes sets of large residuals and prefer very small residuals. Indeed:

- When u is large, ρ saturates to 1
- When u is small, ρ is a function of u^2

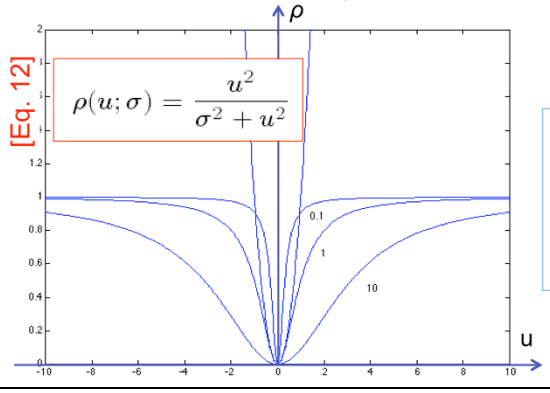
Least squares: Robust estimators

Instead of minimizing $E = \sum_{i=1}^{n} (ax_i + by_i - d)^2$ [Eq. 8]

We minimize $E = \sum_i \rho(u_i ; \sigma)$ [Eq. 11] $\quad u_i = ax_i + by_i - d$

- u_i = error (residual) of i[th] point w.r.t. model parameters h = (a,b,d)
- ρ = robust function of u_i with scale parameter σ

[Eq. 12] $\rho(u;\sigma) = \frac{u^2}{\sigma^2 + u^2}$

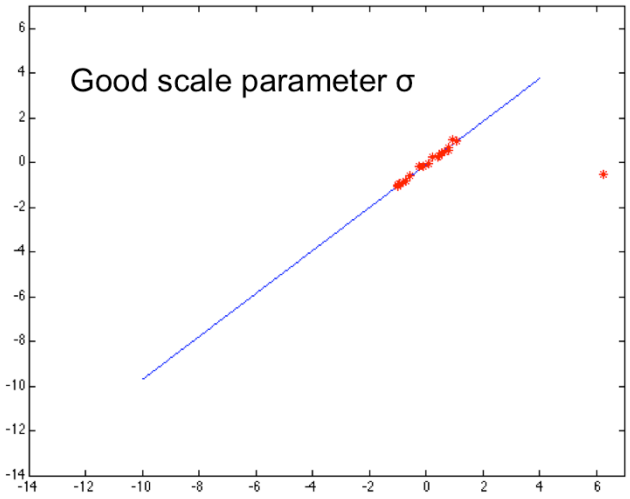


•Small sigma → highly penalize large residuals

•Large sigma → mildly penalize large residual (like LSQR)

The scale parameter σ regulates how much weight we give to potential outliers. Indeed a small value of sigma highly penalizes large residuals (which means we will end up giving less weight to the outliers). A large value of sigma mildly penalizes large residuals and act similarly to Eq. 8 (i.e., like in a standard least square solution, outliers will affect our result more). The plot in the figure shows different profiles of $\rho(u_i; \sigma)$ for different values of σ (e.g., σ=0.1, σ=1 and σ=10).

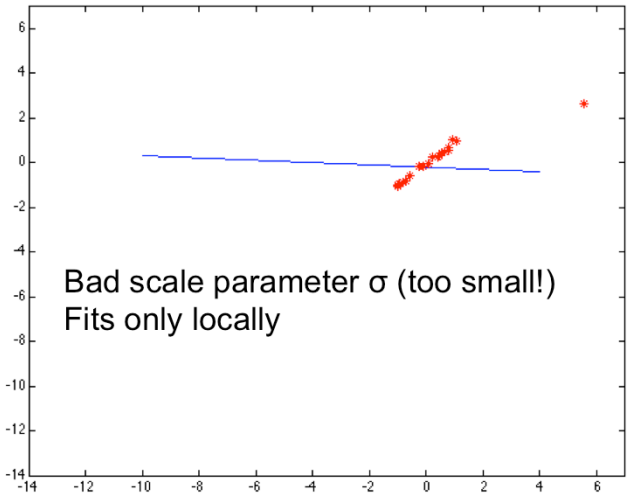
Least squares: Robust estimators



The effect of the outlier is eliminated

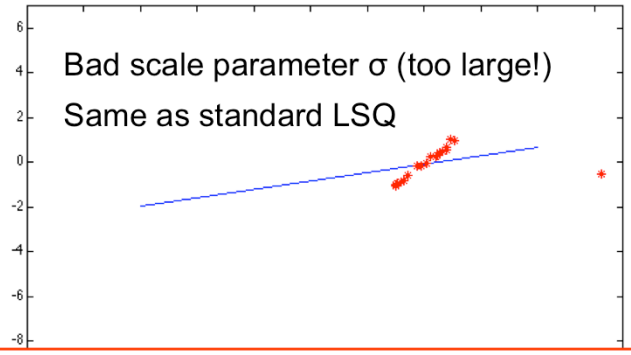
A downside to using the robust estimator is that now we have one more parameter to choose – what value of sigma to use. Here, we have an example of a good choice of sigma for the earlier data.

Least squares: Robust estimators



Here is an example of a bad choice of sigma. σ is too small and we end up favoring a solution for the line that only fits points locally (i.e., most of the points are considered as outliers). This in general produces very unstable results.

Least squares: Robust estimators



•CONCLUSION: Robust estimator useful if prior info about the distribution of points is known

•Robust fitting is a nonlinear optimization problem (iterative solution)

•Least squares solution provides good initial condition

Here is another example of a bad choice of sigma. σ is too large and we end up with a similar result to the original least squares function. In conclusion, the robust estimator is useful when we have prior knowledge about what σ should be or some way it to find out. Generally, in practice, people use least squares as a good starting point and use an iterative approach to find the optimal robust fitting method (which is a non linear problem).

Fitting

Goal: Choose a parametric model to fit a certain quantity from data

Techniques:

- Least square methods
- RANSAC
- Hough transform

Next, we will talk about RANSAC which stands for Random Sample Consensus. RANSAC was introduced by Fischler & Bolles in 1981.

Basic philosophy
(voting scheme)

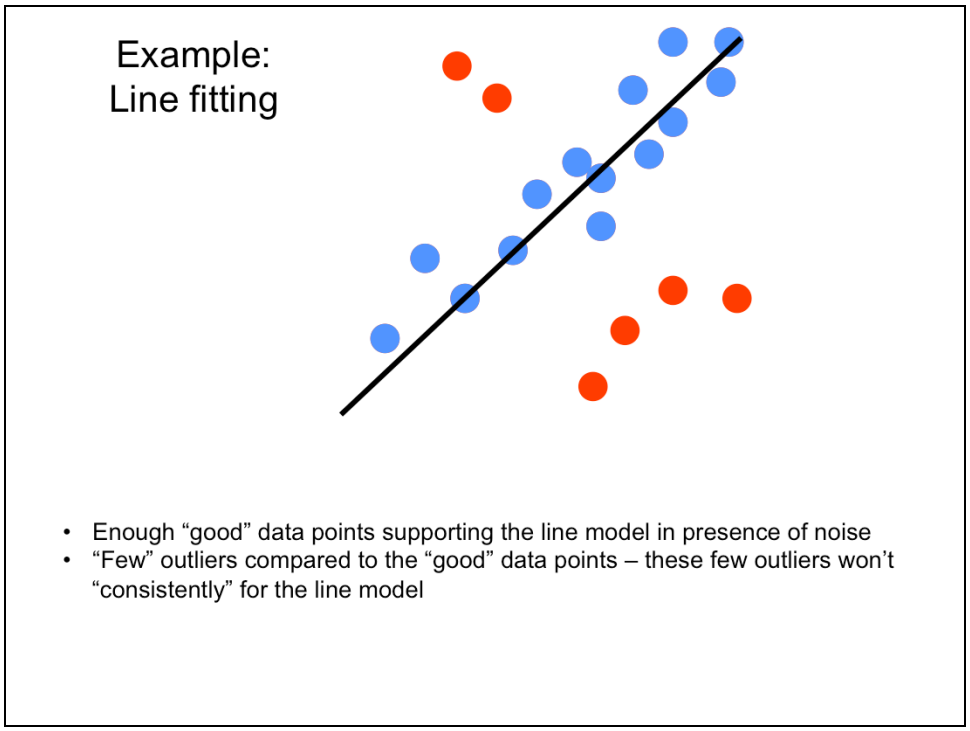
- Data elements are used to vote for one (or multiple) models
- Robust to outliers and missing data
- Assumption 1: Noisy data points will not vote consistently for any single model ("few" outliers)
- Assumption 2: There are enough data points to agree on a good model ("few" missing data)

The basic idea is that we are going to set up a voting procedure where each data points counts for a "vote" for one or more models. Ideally, points that are noisy will not vote for the same model and points that are relevant will vote for consistent models. This would make our system robust to outliers and missing data (two of the challenges we mentioned earlier). This basic idea is shared by RANSAC and Hough Voting (as we shall see next).

RANSAC is designed to be robust to outliers and missing data and follows two major assumptions:

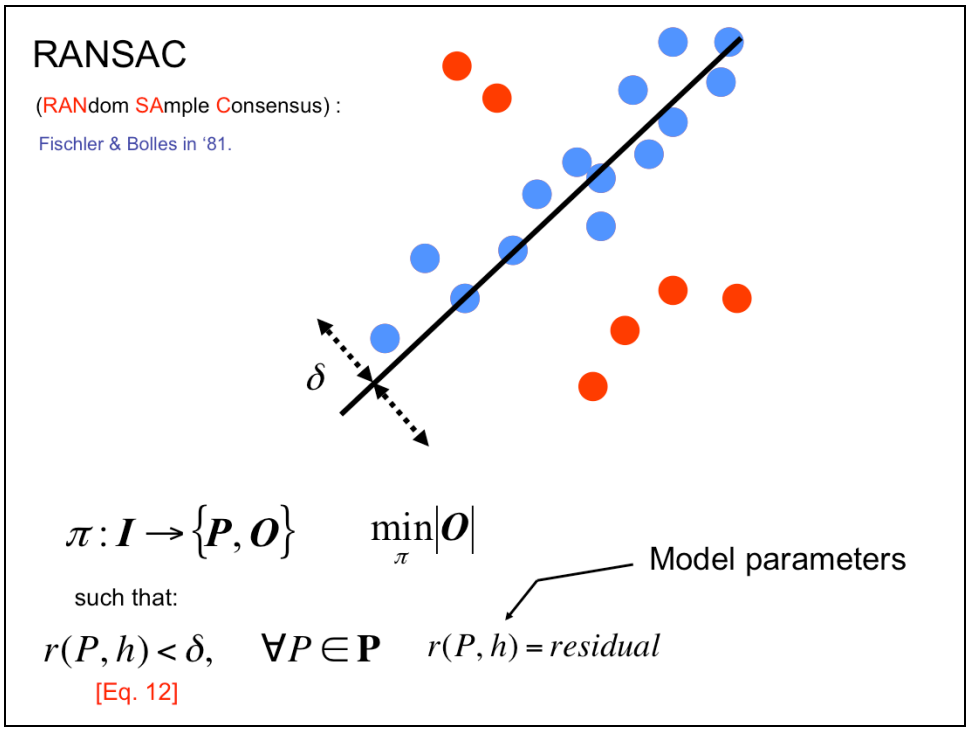
1: Noisy data points will not vote consistently for any single model ("few" outliers)

2: There are enough data points to agree on a good model ("few" missing data)



Here is an example for the case of line fitting. The assumptions are that there are:

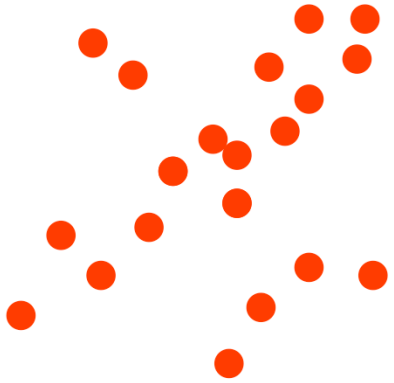
- Enough “good” data points supporting the line model in presence of noise (blue points!)
- “Few” outliers compared to the “good” data points – these few outliers won’t “consistently” for any single model (red points!)



We start we a very high level overview of RANSAC. Suppose we have the standard line fitting problem in presence of outliers.

We can formulate this problem as follows: we want to find the best partition of points in the inlier set **P** and outlier set **O** such that we keep as many inlier points as necessary, but that they also result in small residuals (that is, a residual that is smaller than δ for every inlier point). In Eq.12, r(P,h) is the residual associated to the inlier point P and the model parameter h.

RANSAC



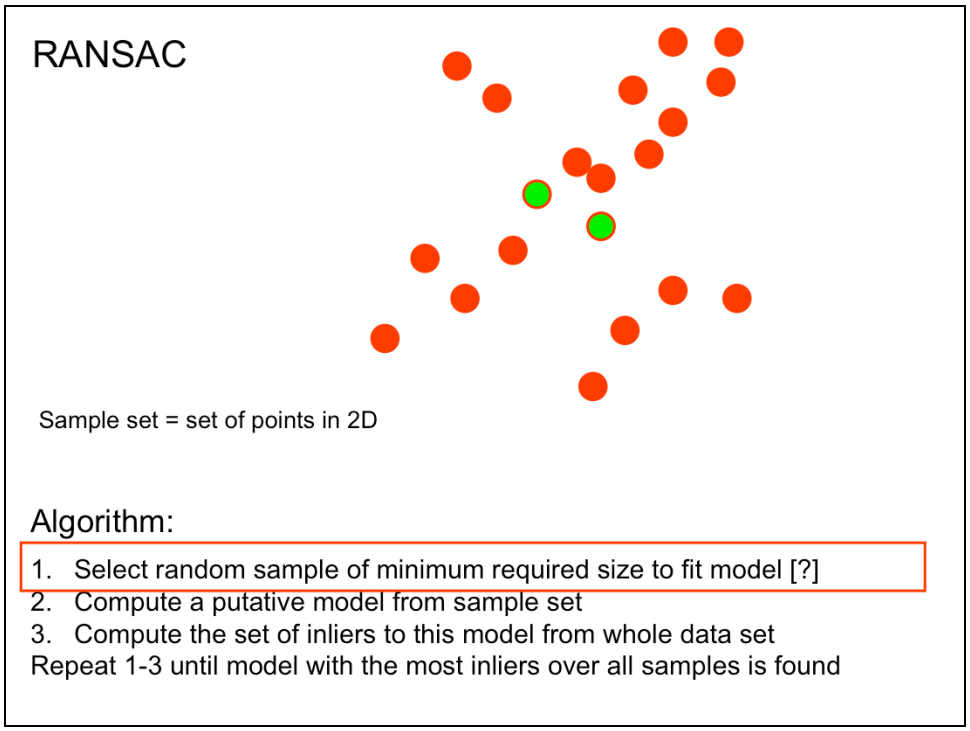
Sample set = set of points in 2D

Algorithm:

1. Select random sample of minimum required size to fit model
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

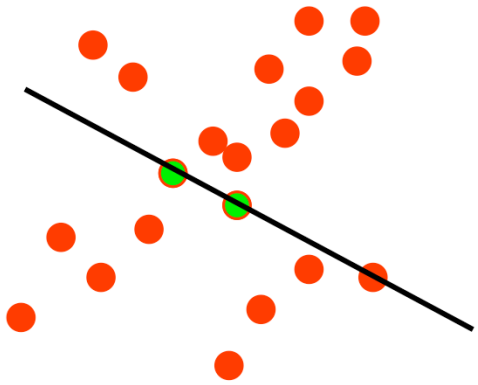
Repeat 1-3 until model with the most inliers over all samples is found

In practice, RANSAC can be formulated as follows: As before, we have a set of 2D points that we want to fit a line to. There are three major steps to RANSAC.



First, we select a random sample of the minimum required size to fit the model. In this case, a line is determined by at least two points. Thus, to fit a model of a line, we select two points at random which constitutes the random sample. In this example, the random sample is made up by the two green points.

RANSAC



Sample set = set of points in 2D

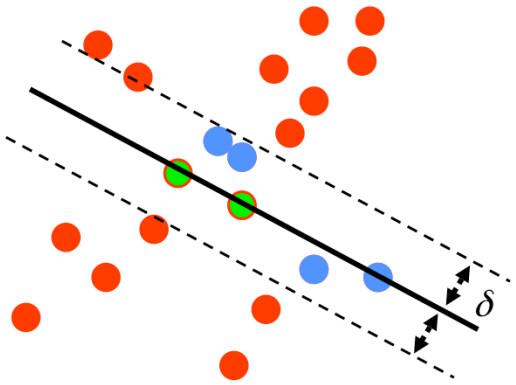
Algorithm:

1. Select random sample of minimum required size to fit model [?]
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

Next, we calculate a model from this sample set. Using the two green points that we have already randomly sampled, we calculate the line that these two represent.

RANSAC



Sample set = set of points in 2D

$|\mathbf{O}| = ? = 14$

$|\mathbf{P}| = ? = 6$

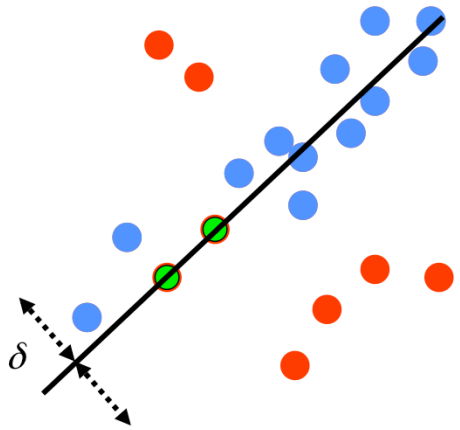
Algorithm:

1. Select random sample of minimum required size to fit model [?]
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

From here, we see how much of our entire set of 2D points agrees with this model up to a tolerance δ. In this case, we see that all of the points that are now blue agree with the model hypothesized by the two green points. Thus, we have an inlier set P of 6 (2 green + 4 blue) and an outlier set (denoted by O in the slide) of 14.

RANSAC



$|\mathbf{O}| = 6$

$|\mathbf{P}| = 14$

Algorithm:

1. Select random sample of minimum required size to fit model [?]
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

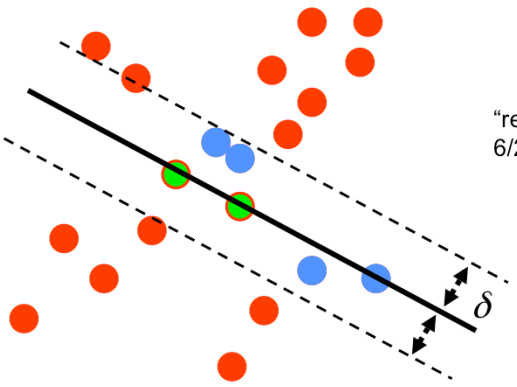
We repeat these steps until the number of points in the inlier set is minimized or we meet some other criteria (to be discussed in the following slides). In this example, we've chosen the two points in green. The model that these two points imply would include all of the blue points. This gives us 14 inliers and 6 outliers, which looks to be a reasonable model. Indeed, this is the model that maximizes the inlier set.

How many samples?

- Number N of samples required to ensure, with a probability p, that at least one random sample produces an inlier set that is free from "real" outliers.
- Usually, p=0.99

It is often computationally unnecessary (and infeasible) to explore the entire sample space. A typical assumption is that only N samples are drawn, where N is the number of samples that is required to ensure, with a probability p, that at least one random sample produces an inlier set that is free from "real" outliers. This N is function of the outier ratio e and the minimum number of data points needed to fit the model s. Usually, p=0.99.

Example

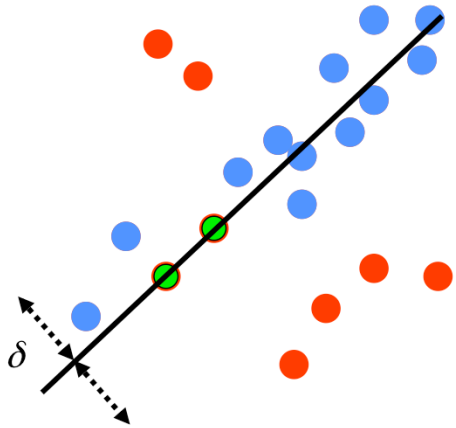


"real" outlier ratio is 6/20 = 30%

- Here a random sample is given by two green points
- Estimated inlier set is given by the green+blue points
- How many "real" outliers we have here? 2

Here we see an example of a random sample that produces an inlier set that contains 2 "real" outliers.

Example



"real" outlier ratio is 6/20 = 30%

- Random sample is given by two green points
- Estimated inlier set is given by the green+blue points
- How many "real" outliers we have here? 0

Here we see an example of a random sample that produces an inlier set that does not contain any "real" outlier (it's free from real outliers).

Thus, N is the number of times we need to sample our data (and thus repeat the steps 1-3 in the previous slides) before we find the configuration above with probability *p*. Again this is function of *e* and *s* as well.

How many samples?

- Number N of samples required to ensure, with a probability p, that at least one random sample produces an inlier set that is free from "real" outliers for a given s and e.
- E.g., p=0.99

$$N = \log(1-p) / \log\left(1-(1-e)^s\right) \quad \text{[Eq. 13]}$$

	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

e = outlier ratio
s = minimum number needed to fit the model

Note: this table assumes "negligible" measurement noise

N can be estimated using equation 13, where *e* denotes the outlier ratio (6/20 = 30% in the previous example), s is the minimum number of points needed to fit the model (2 in the previous example of a line), and p is the probability that at least one random sample produces an inlier set that is free from "real" outliers for a given s and e.

The chart shows numbers of samples needed N for a 99% chance that at least one random sample produces an inlier set that is free from "real" outliers . To read this chart, we have s points needed to fit the model in the first column and the proportion of outliers e in the top row. The circled example shows that for a model that requires 2 points and data that has 30% outliers, we need 7 samples (of two points each) to have a 99% chance that at least one random sample produces an inlier set that is free from "real" outliers.

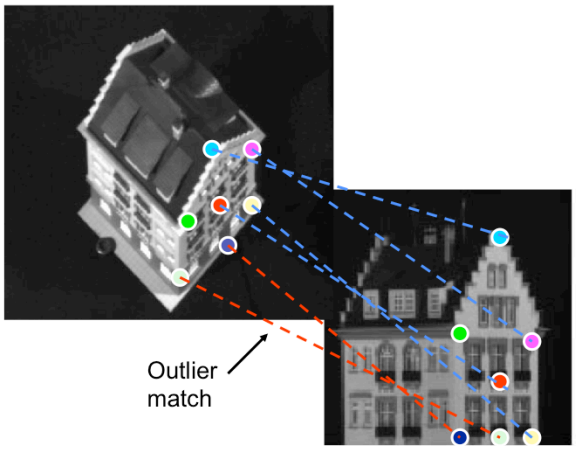
A justification of Eq 13 is reported below (kindly provided by Phillip Lee):

We can say that p = (1-e)^s. This is the chance that if we picked one set of samples (s points), all of them will be inliers (w/ replacement).

Then, we know that 1-p = 1-(1-e)^s. This is the chance that if we picked one set of samples (s points), at least one of them will be an outlier.

Then, (1-p) = (1-(1-e)^s)^n. This is the chance that if we picked n sets of

Estimating H by RANSAC



•H → 8 DOF
•Need 4 correspondences

Sample set = set of matches between 2 images

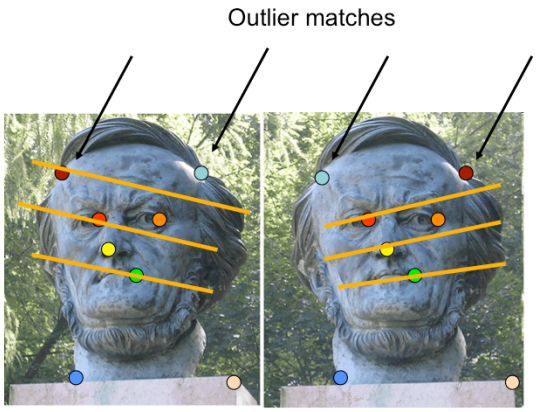
Algorithm:

1. Select a random sample of minimum required size [?]
2. Compute a putative model from these
3. Compute the set of inliers to this model from whole sample space

Repeat 1-3 until model with the most inliers over all samples is found

Here are some other examples of when RANSAC can be useful. In this example the goal is to fit the homographic transformation H given a set of pairs of corresponding points (correspondences) in presence of outliers. In this example, the blue dashed lines indicate correct correspondences whereas the red ones are outliers. The steps of the algorithm are the same as before. The only difference is the size of the random sample which is 4 in this case. Why? H has 8 degrees of freedom, so we need 4 correspondences (because each point has two coordinates) to compute a minimum model. Thus, in this case s=4.

Estimating F by RANSAC



•F → 7 DOF
•Need 7 (8) correspondences

Sample set = set of matches between 2 images

Algorithm:

1. Select a random sample of minimum required size [?]
2. Compute a putative model from these
3. Compute the set of inliers to this model from whole sample space

Repeat 1-3 until model with the most inliers over all samples is found

In this example the goal is to fit the fundamental matrix F given a set of pairs of corresponding points (correspondences) in presence of outliers. The outliers in this case are related to the dark red and cyan points, whereas all the other points are in correct correspondence. In this case, the random sample is 8. Why? F has 7 degrees of freedom, so we need 7 correspondences (in fact, 8 if we use the 8 points algorithm) to compute a minimum model. Thus, in this case s=7(8).

RANSAC - conclusions

Good:

- Simple and easily implementable
- Successful in different contexts

Bad:

- Many parameters to tune
- Trade-off accuracy-vs-time
- Cannot be used if ratio inliers/outliers is too small

In conclusion, RANSAC is an easily implementable method to estimate a model that often works well in practice. However, there are many parameters to tune; it may take a long time to get to the accuracy that you need, and requires the inlier to outlier ratio to be reasonable. Empirical studies show that RANSAC doesn't work well if the inlier/outlier ration is greater than 50%.

Fitting

Goal: Choose a parametric model to fit a certain quantity from data

Techniques:

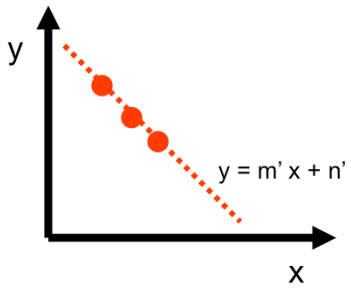
- Least square methods
- RANSAC
- Hough transform

The Hough transform is the last method that we'll discuss today.

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Given a set of points, find the curve or line that explains the data points best



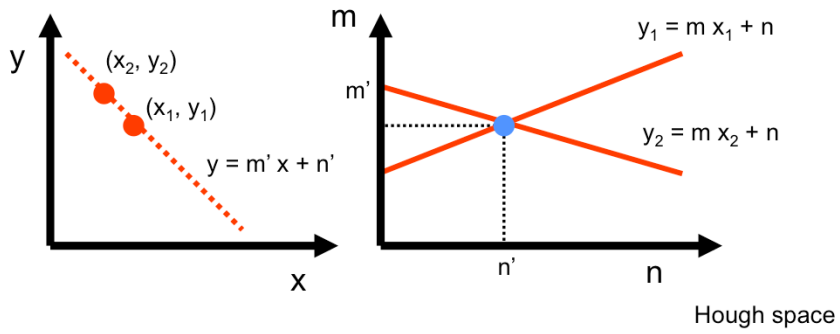
Hough voting is a voting procedure as well, where each data points counts for a “vote” for one or more models that we want to fit to our data. Let’s consider the example of line fitting again. Our goal is to estimate the parameter m’ and n’ of the line (in dashed red color) that fits our data points (in red).

Hough voting uses the concept of parameter (or dual) space. If the model we want to fit (i.e. the line) is represented parametrically by, for instance, m and n, then we can establish a relationship between the original space where the data points lie, and the dual parameter space which is defined by the parameters (i.e., m and n) that describe the model we want to fit the data to (each axis corresponds to the a model parameter).

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Given a set of points, find the curve or line that explains the data points best



$y = m x + n$

In this slide, on the left we have the original image x,y (Cartesian) coordinates and on the right, we have the parameter (Hough) space. We want to show that the parameters that describe the line $y = m' x + n'$ (i.e. m' and n') can be computed by intersecting lines in the Hough space. Let's pick up a point (x_1, y_1) that belongs to the line $y = m' x + n'$ in the original image. This point corresponds to the line $y_1 = m x_1 + n$ with parameters x_1 and y_1 in the Hough space. Let's pick up a second point (x_2, y_2) that belongs to $y = m' x + n'$ in the original image. This corresponds to the line $y_2 = m x_2 + n$ with parameters x_2 and y_2 in the Hough space. Thus, it is easy to realize that the point of intersection of $y_1 = m x_1 + n$ and $y_2 = m x_2 + n$ in the Hough space returns the coordinates m' and n' of the line $y = m' x + n'$ in the original space. This it is true for any arbitrary point we pick up from the original space as long as it belongs to the line $y = m' x + n'$.

Thus, this mechanism can be used for fitting lines to points: given a set of data points we want to fit a line to, we associate for each of these points a line in the Hough space. Then, we intersect these lines in the Hough space. The point of intersections returns the parameter of the line fitting the points in the original space.

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Any Issue? The parameter space [m,n] is unbounded…

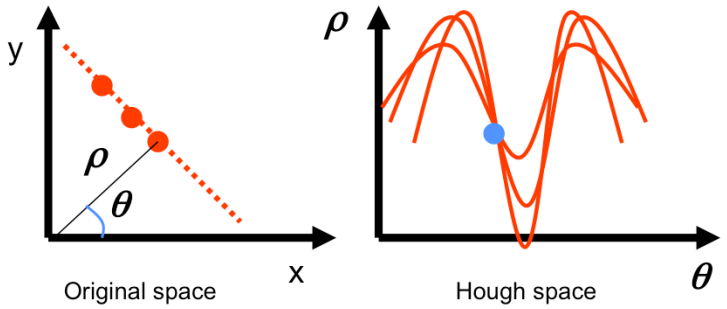
As we already discussed for the least square case, the parameter space [m,n] is unbounded; that is, m and n can range from minus infinity to infinity (think of a vertical line).

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Any Issue? The parameter space [m,n] is unbounded…

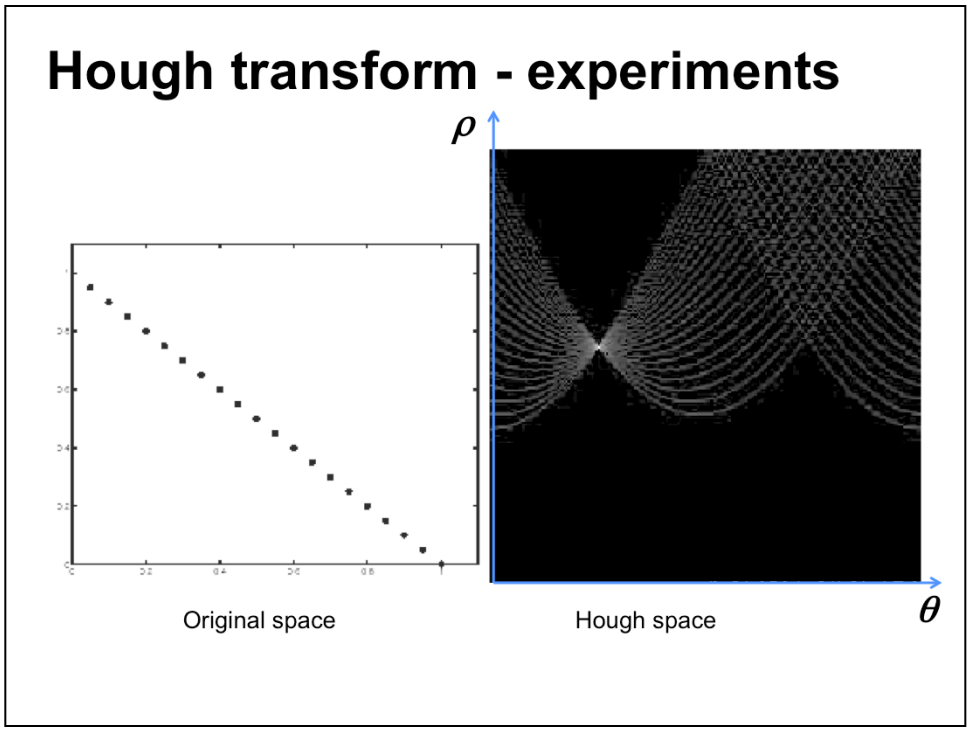
•Use a polar representation for the parameter space



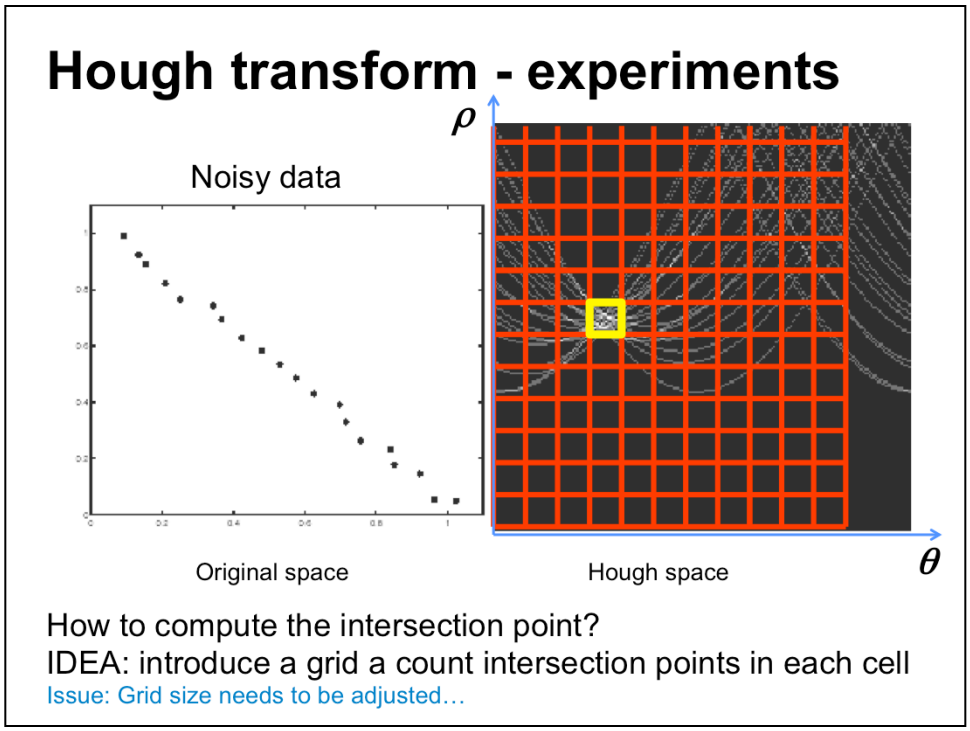
$$x\cos\theta + y\sin\theta = \rho \quad \text{[Eq. 13]}$$

Thus, it is common to consider the polar parameterization of the line as shown in Eq. 13, instead of y = mx + n.

Now all possible lines in Cartesian space have a sinusoidal profile in hough space. The Hough voting procedure is still the same in that the parameters rho and theta of the line (fitting the data points in the original space) are estimated as the point of intersection of all the sinusoidal profiles in the Hough space.



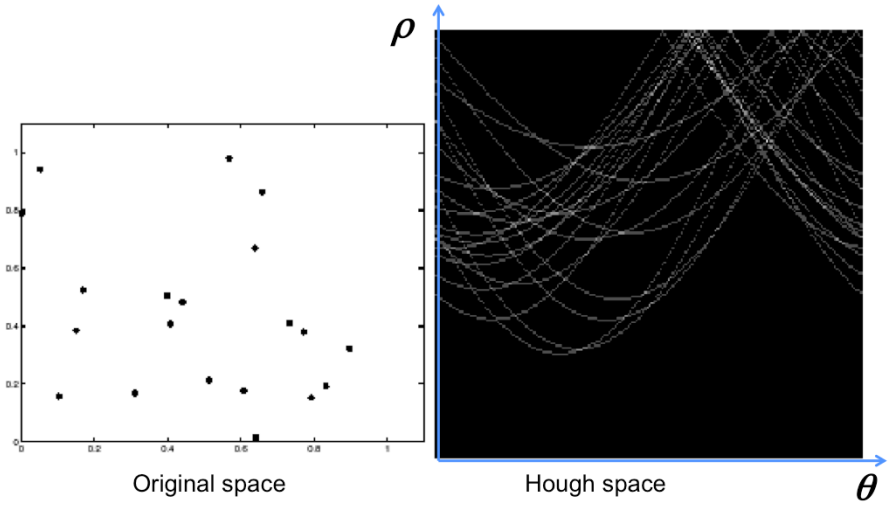
Here is an example of the Hough transform in action.



When the data points are noisy, the corresponding sinusoidal profiles don't intersect at the exactly same point. So how do we compute the parameters of the line in this case? Let's call a point of intersection of two sinusoids a "vote". The idea is to divide the Hough space into grid and count how many votes we have in each cell of the grid. The cell that contains the largest number of votes (the yellow cell in the figure) returns the parameters of the line we want to fit. For instance, such parameters can be estimated as the coordinates of the center of the cell that contains the largest number of vote. The name "Hough voting" comes from this concept of counting votes in the Hough space.

While the idea of using a grid helps dealing with the case of noisy data, the grid size is an additional parameter that we need to tune when running Hough voting. Small grid sizes makes it harder to find the cell that contains the largest number of votes. Large grid sizes decreases the accuracy in estimating the parameters of the model (all of the values of θ and ρ within the most voted cell are good candidates for describing the line).

Hough transform - experiments



Issue: spurious peaks due to uniform noise

An interesting case is the one where we have a more or less uniform distribution of points in the image (uniform noise). In presence of uniform noise, we don't find a clear consensus of an appropriate model in the Hough space. However, random aggregations of points along a line in the original space may produce consistent voting in certain cells which result in lowering the overall signal to noise ratio.

Hough transform - conclusions

Good:

- All points are processed independently, so can cope with occlusion/outliers
- Some robustness to noise: noise points unlikely to contribute consistently to any single cell

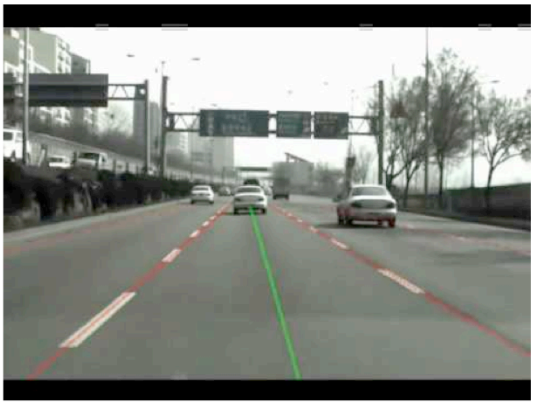
Bad:

- Spurious peaks due to uniform noise
- Trade-off noise-grid size (hard to find sweet point)

Of course, the Hough transform has pros and cons. An important pros is that it can successfully handle some degree of outliers and noise: An outlier just produces an additional sinusoid that is very unlikely to matter as we count the number of votes in each cell. This is true as long as the outliers don't get aggregated in a way that they consistently vote for another model.

The cons include the problem of having uniform noise to produce spurious peaks that may decrease the signal to noise ratio. Also, it's not easy to find the right size of the grid unless we have some prior knowledge about how data are distributed.

Applications – lane detection



Courtesy of Minchae Lee

A typical application of Hough voting (for lines) is to detect lanes in images for autonomous navigation applications.

Applications – computing vanishing points

Another typical application of Hough voting (for lines) is to estimate vanishing lines (and thus vanishing points).

Generalized Hough transform

[more on forthcoming lectures]

D. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, Pattern Recognition 13(2), 1981

- Parameterize a shape by measuring the location of its parts and shape centroid
- Given a set of measurements, cast a vote in the Hough (parameter) space
- Used in object recognition! (the implicit shape model)

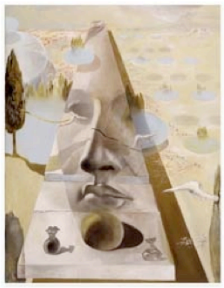
B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

We can generalize the Hough voting scheme to find shapes that are much more complicated than a straight line.

The general concept is to parameterize a shape in term of the location of its parts with respect to the shape centroid. Given a set of measurements, we can cast votes in the Hough (parameter) space; the cell with the largest number of votes returns the probability that the shape has been found.

This idea has been used in designing methods for object detection (see for instance the work by Leibe et al., 2004). We will discuss this in details in one of the upcoming lectures.

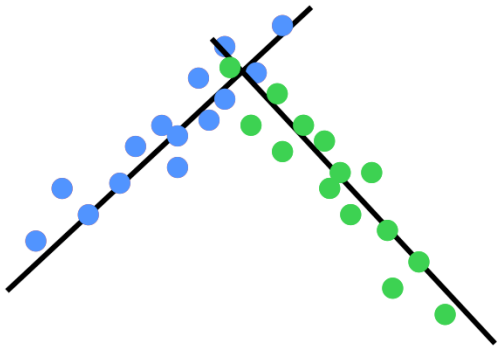
Lecture 9
Fitting and Matching



- Problem formulation
- Least square methods
- RANSAC
- Hough transforms
- Multi-model fitting
- Fitting helps matching!

We now discuss the case of fitting data when these data can be explained by more than one model.

Fitting multiple models



- Incremental fitting
- E.M. (probabilistic fitting)
- Hough transform

Here is an example: given this distribution of 2D points, we want to:

- Discover how many lines explain the points— say 2.
- Estimate the parameters of these2 lines that best fits the points.

All of this in presence of noise and outliers.

There are many approaches to this problem. Incremental fitting (sweeping through possible numbers of lines), Expectation maximization (not covered in this class, but using the fit of a model to estimate how many lines should be fit, then using the estimate to fit the model in an iterative fashion), and the Hough transform (to vote for how many lines are needed while estimating their parameters).

Incremental line fitting

Scan data point sequentially (using locality constraints)

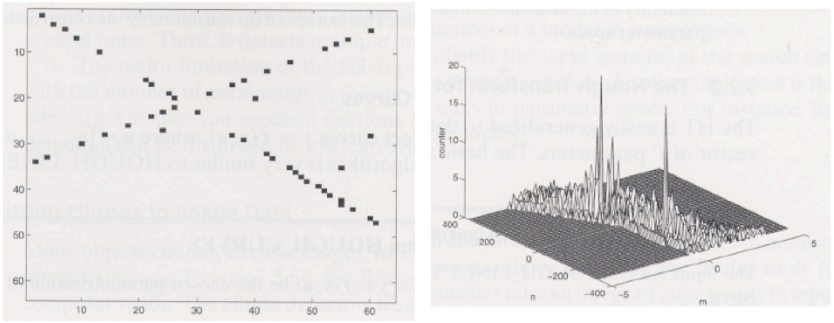
Perform following loop:

1. Select N point and fit line to N points
2. Compute residual R_N
3. Add a new point, re-fit line and re-compute R_{N+1}
4. Continue while line fitting residual is small enough,

- When residual exceeds a threshold, start fitting new model (line)

The idea of incremental line fitting is fairly straightforward. We fit a line to N point and calculate our residuals. Then we add a new point and re-fit the line and residual. If the residual ever goes over a threshold, we add another line to our model. We continue until all (or enough) points are accounted for.

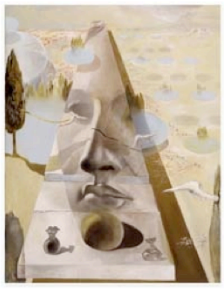
Hough transform



Same cons and pros as before…

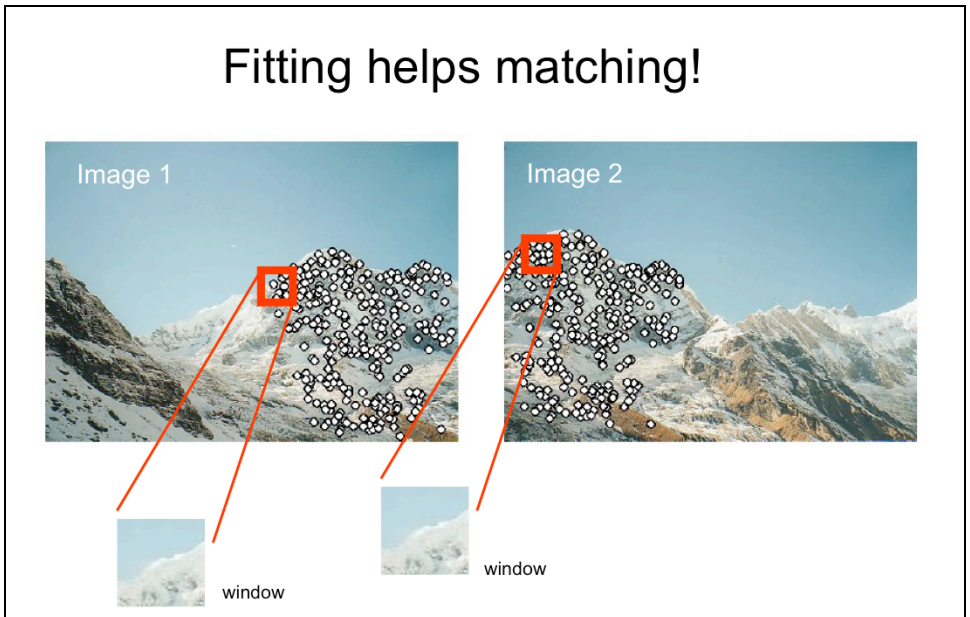
Hough voting provides an appealing alternative to incremental fitting. The problem is essentially solved using the same procedure as discussed before. If data points are explained by multiple lines (two in this case), we will observe more than one cell that contains a number of votes that is significantly higher than those due to noise. The figure shows an example where Hough voting is applied to the point distribution on the left hand side. In the right figure, the n-axis and m-axis describe the Hough voting space, and the vertical axis (counter) reports the number of votes that appears in each cell. We can clearly see two peaks (hence two lines) emerging from all the other peaks that are due to noise. The parameters associated to these 2 peaks characterize the two lines that fit the points in the original space.

Lecture 9
Fitting and Matching



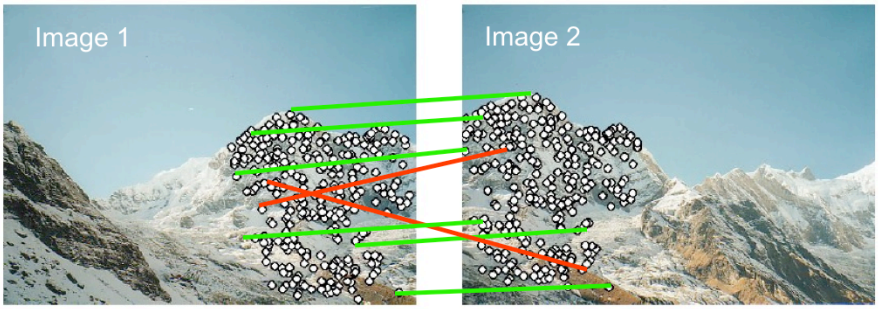
- Problem formulation
- Least square methods
- RANSAC
- Hough transforms
- Multi-model fitting
- Fitting helps matching!

We will wrap up this lecture with how fitting (estimating models) interact with matching (finding correspondences between images).



A common task in vision is to match feature points between images. Each image in the figure shows a number of detected points obtained, for instance, using a feature detector; these are are depicted as white points. Feature points can be matched across images using a similarity criterion such as the cross correlation (see lecture 6) that is computed over a small window around the feature of interest.

Fitting helps matching!



Matches based on appearance only
Green: good matches
Red: bad matches

Idea:
- Fitting an homography H (by RANSAC) mapping features from images 1 to 2
- Bad matches will be labeled as outliers (hence rejected)!

Unfortunately, for a number of reasons that we already extensively discussed during lecture 6, this matching process is faulty and can produce a large number of mismatched (outliers) which are indicated in red in the figure.

A technique such as RANSAC can help. Suppose that these images are related by an homographic transformation H. This is actually true in this case, since the scene we are observing is very far from the camera. Then, we can run RANSAC to estimate the H that relates corresponding points. Not only does RANSAC produce a robust estimation of H by rejecting the outliers, but also returns the list of outliers (i.e., the mismatches).

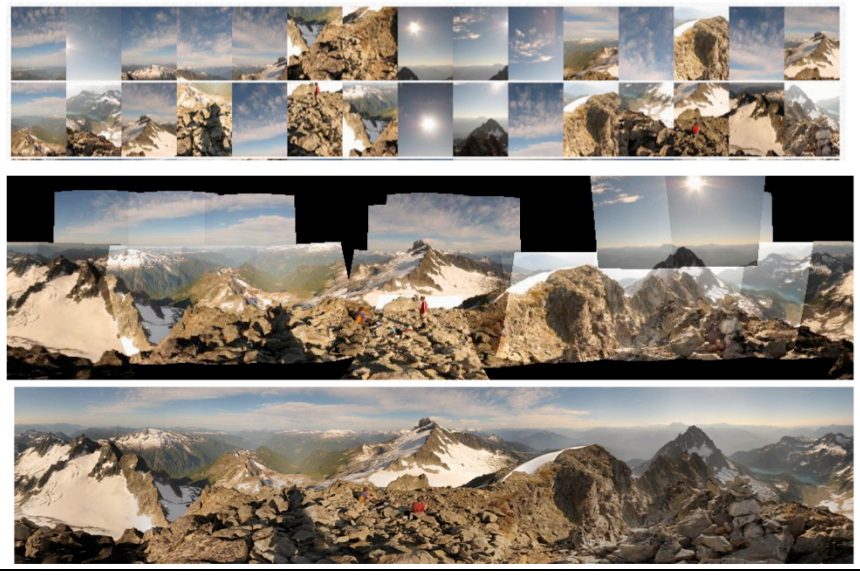
This example shows how matching and fitting are very much interrelated problems.

Fitting helps matching!

Using our homography, we can be use to blend the two images on top of each other after.

Recognising Panoramas

M. Brown and D. G. Lowe. Recognising Panoramas. In Proceedings of the *9th International Conference on Computer Vision -- ICCV2003*



This approach can be used to stitch images so as to form panoramic images and it is currently commonly used in many commercial applications for panorama processing.

Next lecture:
Feature detectors and descriptors

Back up slides for Least squares methods. Please see Linear algebra

Least squares methods

- fitting a line -

$$Ax = b$$

- More equations than unknowns
- Look for solution which minimizes $||Ax-b|| = (Ax-b)^T(Ax-b)$
- Solve $\frac{\partial (Ax-b)^T (Ax-b)}{\partial x_i} = 0$
- LS solution

$$x = (A^T A)^{-1} A^T b$$

We saw earlier that the Least Squares solution to Ax = b is x = (A^T A)^-1 A^T b. This means that given an A and b, this is the value of x that comes closest to Ax=b with respect to the cost function ||Ax-b||. X may not be an exact solution when A is tall and full rank (meaning we have more equations than unknowns).

Least squares methods

- fitting a line -

Solving $x = (A^t A)^{-1} A^t b$

$A^+ = (A^t A)^{-1} A^t$ = pseudo-inverse of A

$A = U \sum V^t$ = SVD decomposition of A

$A^{-1} = V \sum^{-1} U$

$A^+ = V \sum^+ U$

with $\sum^+$ equal to $\sum^{-1}$ for all nonzero singular values and zero otherwise

Another way to look at the least squares solution is as a singular valude decomposition (SVD) problem. The pseudo-inverse A-dagger, where x = (A-dagger) y can also be derived from SVD.

Least squares methods

- fitting an homography -

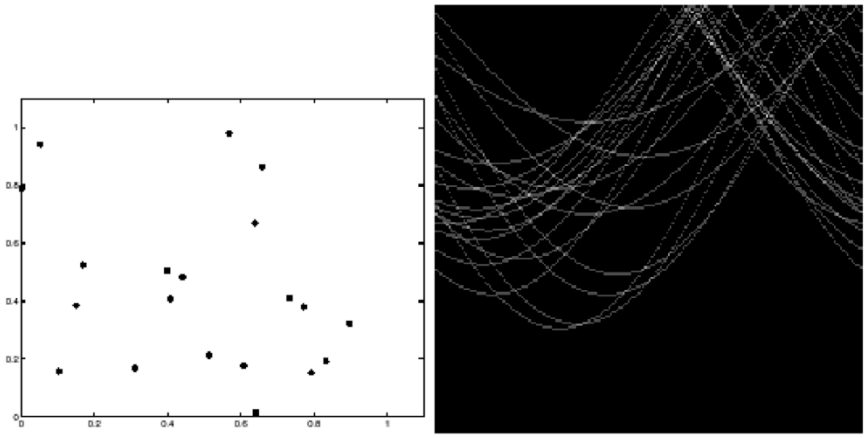
$$
\begin{aligned}
h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' - x' &= 0 \\
h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' - y' &= 0
\end{aligned}
$$

From n>=4 corresponding points: $\mathrm{A\,h} = 0$

$$
\begin{pmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' & -x_1' \\
0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' & -y_1' \\
x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' & -x_2' \\
0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' & -y_2' \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_n & y_n & 1 & 0 & 0 & 0 & -x_nx_n' & -y_nx_n' & -x_n' \\
0 & 0 & 0 & x_n & y_n & 1 & -x_ny_n' & -y_ny_n' & -y_n'
\end{pmatrix}
\begin{bmatrix}
\mathrm{h}_{1,1} \\ \mathrm{h}_{1,2} \\ \vdots \\ \mathrm{h}_{3,3}
\end{bmatrix}
= 0
$$

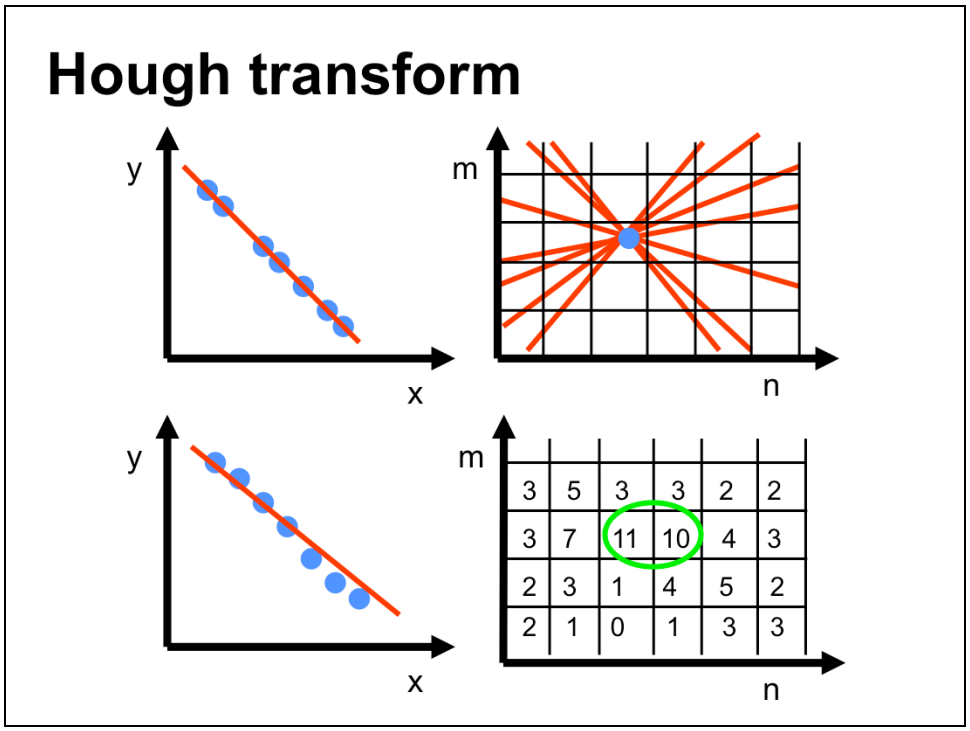
To fit a homography, with 4 points, we find the least squares solution to Ah=0, where A and h are defined above. The solution to this will be the last right singular vector of A.

Hough transform - experiments



Issue: spurious peaks due to uniform noise

Here is an example of very noisy data. It does not seem like a line should fit most of the points in Cartesian space. Accordingly, in Hough space, there does not seem to be a clear consensus of an appropriate model. However there does seem to be a few points that seem to be potential lines. These may correspond to lines that would fit some of the data points, but not all of them.



We can make this idea a little more robust with a few small modifications. The first is to remedy the issue of points having some small amounts of noise. If the points don't fit exactly on the line, the intersections in the Hough transform may not all align perfectly. To fix this, we discretize the Hough space into a grid and count how many lines occur in a specific box of the grid. Thus, we can count how many lines come close to intersecting, but we do not need them to intersect exactly (which is difficult to do numerically). Although we lost some precision in our model estimate, we add some robustness to noise to our model.

When a line passes through a grid in the Hough space, we call that a "vote" for that grid.

Fitting helps matching!

Images courtesy of Brandon Lloyd

Here's another example where we can use fitting and matching to estimate the change in perspective of this image.