

Robust Fitting: RANSAC and the Hough Transform

Computer Vision I (CSCI 3240U)

Faisal Z. Qureshi

<http://vclab.science.ontariotechu.ca>



Week 7 at a glance

► **Lecture 1: RANSAC (Sec. 4.1.3, 8.1.4)**

- Robust cost functions
- Hypothesise-and-verify
- How many iterations?
- Choosing the inlier threshold

► **Lecture 2: The Hough transform (Sec. 7.4.2)**

- Voting in parameter space
- The (ρ, θ) parameterisation
- Accumulator resolution
- Using gradient direction

Reading: Szeliski Sec. 4.1.3, 8.1.4, 7.4.2. Exercise w07a. **Lab 6** is the payoff for Weeks 4–7.

Lecture 1: RANSAC

Where we left off

Least squares, ordinary or total, has **breakdown point zero**.

In vision that is fatal:

- ▶ feature matching gives 30–50% wrong correspondences,
- ▶ an edge map contains many edges that are not the structure you want,
- ▶ a LiDAR sweep of a road contains cars, kerbs and signs as well as road.

We need an estimator that **expects** a large fraction of the data to be irrelevant.

Why the squared cost is the problem

$$E = \sum_i r_i^2$$

The contribution of a point **grows without bound** with its residual. A point 1000 units away contributes a million.

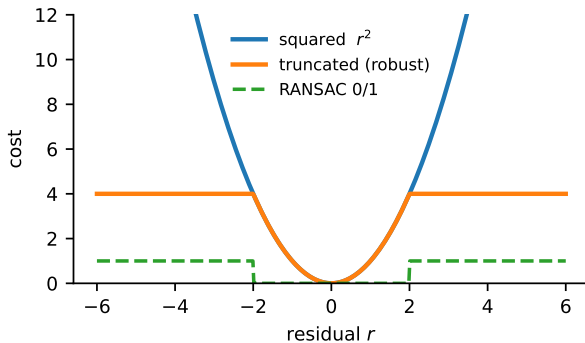
A **robust** cost saturates:

$$E = \sum_i \rho(r_i), \quad \rho \text{ bounded}$$

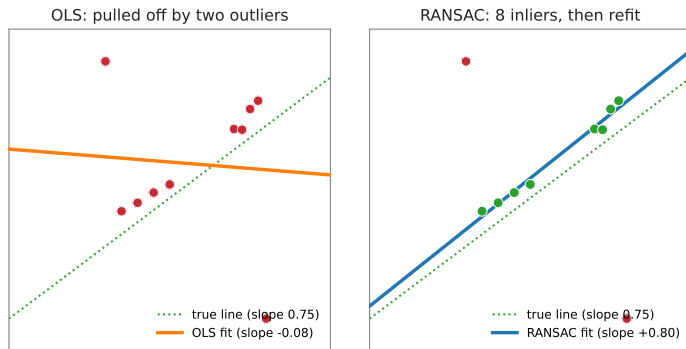
so a wildly wrong point contributes no more than a moderately wrong one.

RANSAC takes this to the limit: ρ is **0 or 1**. A point either agrees with the model or it does not, and how badly it disagrees is irrelevant.

Squared versus robust cost



OLS versus RANSAC on the same contaminated data



Eight true inliers around $y = 0.75x + 1$, plus two planted outliers. OLS minimises $\sum r_i^2$ over *all ten* points and lands on a slope of -0.08 . RANSAC finds an 8-point inlier set, refits on it, and recovers $+0.80$. Same data, two very different answers.

RANSAC: the algorithm

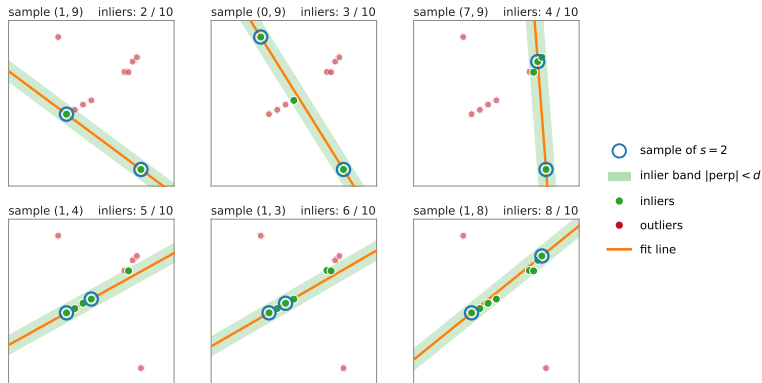
RAndom **SA**mples **C**onsensus.

1. **Sample** the minimum number of points s needed to define the model.
2. **Fit** the model to that minimal sample exactly.
3. **Count** inliers: points within distance d of the model.
4. Repeat N times; keep the model with the most inliers.
5. **Refit** using *all* inliers of the winning model.

Step 5 matters and is often skipped. The winning minimal sample is itself noisy; refitting on all its inliers is what gives a precise answer.

RANSAC in action

Six of the 45 two-point samples, sorted by inlier count. The winner is picked by counting; step 5 refits over its 8 inliers.



Why a minimal sample?

Why fit to 2 points rather than 20?

Write e for the outlier probability of a single point; then the inlier probability is $1 - e$, and the chance that a random sample of size s is **all inliers** is $(1 - e)^s$. This falls fast with s :

- ▶ $e = 0.5, s = 2$: $(1 - e)^s = 0.25$
- ▶ $e = 0.5, s = 4$: $(1 - e)^s = 0.0625$
- ▶ $e = 0.5, s = 8$: $(1 - e)^s = 0.004$

So we sample as few points as the model allows: 2 for a line, 4 for a homography (Week 9), 8 for a fundamental matrix via the eight-point algorithm (7 is its theoretical minimum).

Precision comes from step 5, not from the sample.

How many iterations?

We want at least one all-inlier sample with probability p .

- ▶ One sample is all-inlier with probability $(1 - e)^s$.
- ▶ It is **not** with probability $1 - (1 - e)^s$.
- ▶ N independent samples are all bad with probability $(1 - (1 - e)^s)^N$.

Set that equal to $1 - p$:

$$N = \frac{\log(1 - p)}{\log(1 - (1 - e)^s)}$$

Note what is absent: the number of data points. RANSAC's cost depends on the *outlier probability* and the *model complexity*, not on n .

Aside: probability of a bad sample

Start from the single-point probabilities. An inlier has probability $1 - e$; an outlier has probability e .

Assume the s points in a sample are drawn independently. The sample is **all inliers** with probability

$$(1 - e)^s.$$

The sample is **bad** (contains at least one outlier) with probability

$$1 - (1 - e)^s.$$

Aside: solving for N

Draw N samples independently. The chance that *every one* is bad is

$$(1 - (1 - e)^s)^N,$$

so the chance of **at least one** good sample in N tries is

$$p = 1 - (1 - (1 - e)^s)^N.$$

Rearrange, take log, divide:

$$(1 - (1 - e)^s)^N = 1 - p \implies N \log(1 - (1 - e)^s) = \log(1 - p)$$

$$N = \frac{\log(1 - p)}{\log(1 - (1 - e)^s)}$$

The numbers, for $p = 0.99$

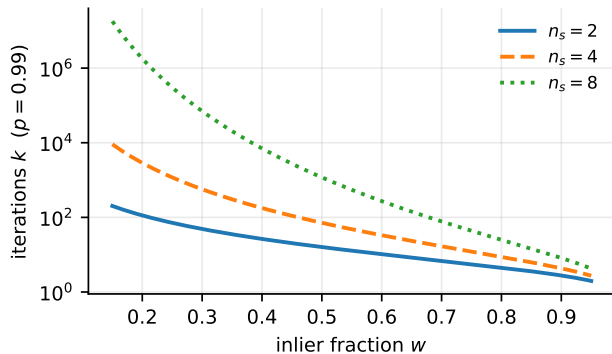
e	0.1	0.3	0.5	0.7
line ($s = 2$)	3	7	17	49
homography ($s = 4$)	5	17	72	567

- ▶ Cheap for a line even at 70% outliers.
- ▶ Two extra sample points take N from 49 to 567, an order of magnitude.

This is why **minimal solvers** matter, and why nobody runs RANSAC on a model needing 20 points.

Compute N from your **observed** outlier fraction. Quoting a default of 1000 is not an answer.

How N grows as inliers get scarce



Choosing the threshold d

d decides what counts as agreement.

- ▶ **Too small:** almost nothing is an inlier. Every model scores near zero, so the winner is essentially a random minimal sample, and the refit rests on a handful of points.
- ▶ **Too large:** everything is an inlier, including the outliers. All models score alike and the result degenerates towards plain least squares over contaminated data, exactly what we were avoiding.

If the inlier noise has standard deviation σ , take

$$d \approx 2\sigma \text{ to } 3\sigma.$$

Adaptive stopping

We do not know e in advance. But we can estimate it as we go.

1. Start with a pessimistic e and hence a large N .
2. Each time a model beats the current best, set $e = 1 - (\text{inliers})/n$ and **recompute** N .
3. Stop when the iteration count reaches the current N .

In practice this terminates far sooner than the worst-case bound.

What RANSAC does not do

- ▶ It does not give the **globally optimal** fit: it is randomised, and two runs can differ.
- ▶ It finds **one** model. A road has at least two boundaries; a scene may contain many lines.
- ▶ It needs a **minimal solver** and a sensible distance function.
- ▶ It degrades badly when the inlier fraction is very low, or when the outliers themselves happen to be consistent with a wrong model.

What RANSAC does not do

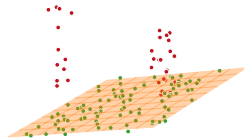
- ▶ It does not give the **globally optimal** fit: it is randomised, and two runs can differ.
- ▶ It finds **one** model. A road has at least two boundaries; a scene may contain many lines.
- ▶ It needs a **minimal solver** and a sensible distance function.
- ▶ It degrades badly when the inlier fraction is very low, or when the outliers themselves happen to be consistent with a wrong model.

The second limitation motivates the second half of the week.

RANSAC in the wild

- **Image stitching (Week 9).** RANSAC picks the homography $\mathbf{H}$ that most SIFT matches agree with, discarding the 30–50,% that are wrong.
- **Ground-plane / road (Lab 9).** Fit the plane with the most inliers; everything above it is an obstacle.

ground-plane fit on a LiDAR sweep



ground (inliers)
obstacles (outliers)

Same pattern: noisy correspondences, minimal solver, count inliers.

Lecture 1 recap

- ▶ Squared cost is unbounded; robust costs saturate; RANSAC uses a 0/1 cost.
- ▶ Hypothesise from a **minimal** sample, verify by **counting** inliers, then **refit** on the inliers.
- ▶ $N = \log(1 - p) / \log(1 - (1 - e)^s)$, independent of n .
- ▶ $d \approx 2-3\sigma$; both extremes fail, in different ways.
- ▶ RANSAC finds one model, not many.

Lecture 2: The Hough transform

A little history

Paul V. C. Hough, 1959:

P. V. C. Hough, "Machine Analysis of Bubble Chamber Pictures," Proc. 2nd International Conference on High-Energy Accelerators and Instrumentation, CERN, Geneva, pp. 554–558, 1959.

Hough was looking at photographs of particle tracks from bubble-chamber experiments and needed to fit **straight lines** to thousands of dots per frame. He filed a US patent (US 3,069,654) in 1962; the parameterisation you will use today, in (ρ, θ) rather than (m, c) , is due to

R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," Communications of the ACM, 15(1):11–15, January 1972.

The extension to arbitrary shapes, using the R-table trick you will meet shortly, is due to *D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," Pattern Recognition, 13(2):111–122, 1981.*

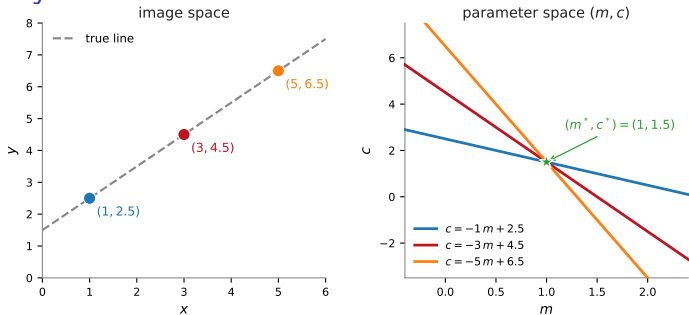
A different idea: let the data vote

Instead of hypothesising models and testing them, let every measurement **vote** for all the models it is consistent with, and look for models with many votes.

A single edge pixel is consistent with **infinitely many lines**, all the lines through it. So it votes for all of them.

Lines that many pixels agree on accumulate many votes.

The (m, c) duality



Rewrite $y = mx + c$ as $c = -xm + y$. Fixing (x, y) makes that a **line** in (m, c) space. Each image point contributes one such line. n collinear image points give n parameter-space lines that all pass through the single point (m^*, c^*) : the parameters of their common line.

Finding a line in the image reduces to finding a **crossing** in parameter space.

Why not $y = mx + c$?

The obvious parameter space is (m, c) . It fails:

- ▶ **Vertical lines need** $m = \infty$. The space is unbounded, so it cannot be discretised into a finite array.
- ▶ Even for finite m , orientations are sampled very unevenly: a huge range of m covers near-vertical lines and a tiny range covers near-horizontal ones.

We need a bounded, uniform parameterisation.

The (ρ, θ) parameterisation

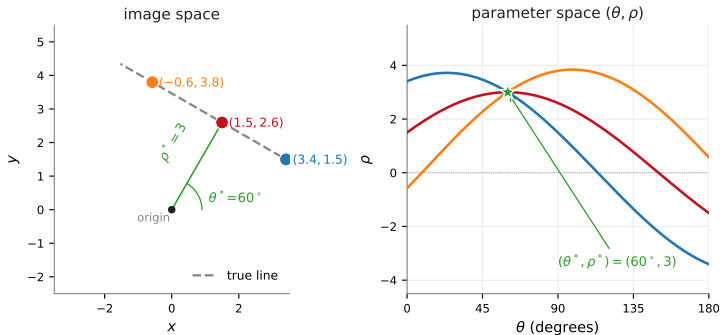
$$\rho = x \cos \theta + y \sin \theta$$

- ▶ $\theta \in [0, 180^\circ)$, the orientation of the line's normal.
- ▶ ρ , the signed perpendicular distance from the origin.
- ▶ $|\rho| \leq \sqrt{H^2 + W^2}$, the image diagonal. **Bounded.**
- ▶ Every orientation, including vertical, is represented equally.

Sizing the accumulator as $\pm W$ instead of $\pm\sqrt{H^2 + W^2}$ causes index errors on diagonal lines. Common bug in Lab 6.

What one pixel votes for

Fix (x_0, y_0) and sweep θ : the equation $\rho(\theta) = x_0 \cos \theta + y_0 \sin \theta$ traces a **sinusoid** in the (θ, ρ) plane. Collinear image points give sinusoids that all pass through one common cell: the parameters of their common line.



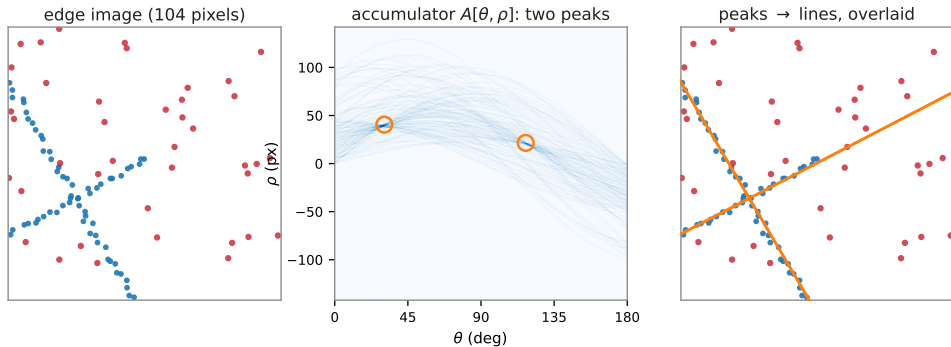
The algorithm

1. Allocate an accumulator array $A[\rho][\theta]$, zeroed.
2. For each edge pixel (x, y) : for each θ , compute $\rho = x \cos \theta + y \sin \theta$ and increment $A[\rho][\theta]$.
3. Find local maxima above a threshold.
4. Each maximum gives (ρ, θ) , a line.

Cost: (number of edge pixels) $\times$ (number of θ bins). With 180 bins and 40,000 edge pixels that is 7.2 million votes: fine vectorised, painful in a Python loop.

Vectorise the voting. Build an array of θ values and broadcast.

The accumulator on a toy image



Two synthetic lines plus scattered noise (104 pixels total). Every pixel votes for a sinusoid; the two crossings show up as clear peaks at the true (θ, ρ) . Decode each peak back into a line and overlay.

Accumulator resolution

A real trade-off, in both directions.

Bins too coarse: distinct lines fall into the same cell and merge. You detect one line where there are two.

Bins too fine: the votes for a single line, which is never perfectly straight and whose pixels are noisy, spread across several neighbouring cells. No cell reaches the threshold and you detect nothing.

Starting point: 1 pixel in ρ , 1° in θ . Then look at the accumulator itself. It is an image, and it will tell you which way to move.

Using the gradient direction

We already computed gradient orientation in Week 4.

At an edge pixel, the line through it is (approximately) **perpendicular to the gradient**, so θ is essentially known.

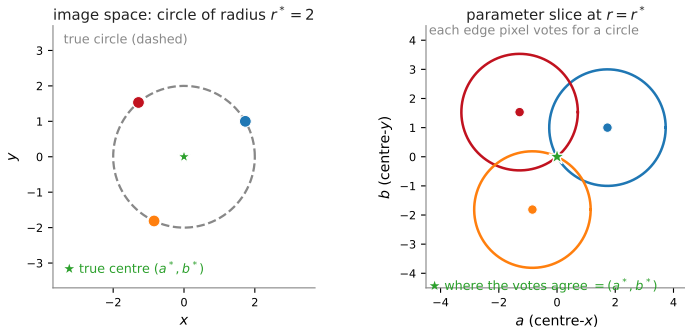
Instead of voting for all 180 orientations, vote only for a few degrees either side of the predicted θ .

- ▶ Roughly an **order of magnitude** fewer votes.
- ▶ The accumulator is also cleaner, because pixels no longer vote for orientations they clearly do not support.
- ▶ Risk: too tight a tolerance and you lose real lines, since the gradient estimate is noisy.

Remember the 90°: gradient direction and edge direction are perpendicular. Confusing them makes every detected line wrong.

Beyond lines: circle Hough

A circle has three parameters, (a, b, r) : centre and radius. Every image pixel that could lie on a circle of radius r constrains the centre to a circle of the same radius around itself.



Sweeping r turns the vote into a **cone** in (a, b, r) space. Peaks give circles. Complexity is now $n_{\text{pixels}} \times n_{\theta} \times n_r$, so keep the radius range tight.

Beyond circles: the generalised Hough transform

For an arbitrary shape given as a template, Ballard (1981) stores an **R-table** of offsets from a reference point. Each edge pixel of the target image looks up its gradient orientation in the table and votes for every reference-point location consistent with it.

- ▶ Handles shapes that have no analytic parameterisation.
- ▶ Accumulator dimension = pose parameters (translation, rotation, scale).
- ▶ Underlies OCR of small logos, industrial part inspection, template detection.

Hough and deep learning

Voting is **differentiable**: an accumulator is a sum of soft assignments, which back-propagates through. Drop a Hough layer into a network and it learns *which features* should vote for *which models*.

- ▶ **Implicit Shape Model** (Leibe, Leonardis, Schiele, ECCV workshop 2004; IJCV 2008): visual words vote for object centres. The pre-deep-learning ancestor.
- ▶ **VoteNet** (Qi, Litany, He, Guibas, ICCV 2019): each PointNet++ feature votes for a 3D object centre; clustered votes are the detections. A workhorse for indoor 3D object detection on ScanNet and SUN RGB-D.
- ▶ **Deep Hough transform** for line detection (Zhao, Han, Zhang, Xu, Cheng, ECCV 2020 / TPAMI 2021; Lin, Pinteá, van Gemert, ECCV 2020): CNN feature map → polar voting → semantic lines in wireframe and architectural scenes.

The pattern is unchanged: **measurements vote, peaks are detections**. What is new is that the “measurements” are learned features, not raw edge pixels, and everything trains end-to-end.

Hough versus RANSAC

	Hough	RANSAC
Finds	many models at once	one model
Cost grows with	parameter-space size	model complexity s
Practical for	2–3 parameters	any minimal solver
Output	quantised peaks	exact fit on inliers

Hough versus RANSAC

	Hough	RANSAC
Finds	many models at once	one model
Cost grows with	parameter-space size	model complexity s
Practical for	2–3 parameters	any minimal solver
Output	quantised peaks	exact fit on inliers

For a road with two boundaries and several lane markings, Hough is the natural first pass, but its peaks are quantised, so a common recipe is **Hough to find candidates, then least squares or RANSAC to refine each one.**

The hard part is not the fitting

In Lab 6 you will extract lines from a KITTI edge map and find you have 20–35 plausible ones per image.

Only two of them are the road boundary.

Deciding **which** is the real work: split by slope sign, require the pair to meet near the vanishing point (Week 2), weight by actual pixel support, reject lines that cross the road region.

And you will find one scene category where the model itself is wrong. There, the strongest line on each side is a lane marking, not the road edge. When that happens, the answer is not to tune the threshold.

Week 7 recap

- ▶ Robust costs saturate; RANSAC's is 0/1.
- ▶ Minimal sample $\rightarrow$ count inliers $\rightarrow$ refit on inliers.
- ▶ $N = \log(1 - p) / \log(1 - (1 - e)^s)$; independent of n ; explodes with s .
- ▶ Hough votes in a bounded (ρ, θ) space; peaks are lines.
- ▶ One point votes along a sinusoid; collinear points share a cell.
- ▶ Bin size trades merged peaks against smeared ones.
- ▶ Gradient direction cuts the voting cost by about $10\times$.

This week's work

- ▶ **Reading:** Szeliski Sec. 4.1.3, 8.1.4, 7.4.2.
- ▶ **Exercise:** w07a.
- ▶ **Lab 6:** least squares, TLS, RANSAC and your own Hough accumulator, then back to the road images from Lab 4 to fit actual boundaries and score them.

Next week: what should we be measuring in the first place? Corners, descriptors, and motion.

Copyright and License

©Faisal Z. Qureshi



This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.