1. How do we find distinctive / easy to locate features? *(Harris/Laplacian of Gaussian)*
2. How do we describe the regions around them? *(Normalize window, use histogram of gradient orientations)*
Earlier I promised

Solving for a Transformation

3: Solve for transformation $T$ (e.g. such that $p_1 \equiv T p_2$) that fits the matches well
Before Anything Else, Remember

You, with your gigantic brain, see:

You should expect **noise** (not at quite the right pixel) and **outliers** (random matches)

The computer sees:
Today

• How do we fit **models** (i.e., a parameteric representation of data that’s smaller than the data) to data?

• How do we handle:
  • **Noise** – least squares / total least squares
  • **Outliers** – RANSAC (random sample consensus)
  • **Multiple models** – Hough Transform (can also make RANSAC handle this with some effort)
Working Example: Lines

• We’ll handle lines as our models today since you are more familiar with them than others
• Next class will cover more complex models. I promise we’ll eventually stitch images together
• You can apply today’s techniques on next class’s models
Model Fitting

Need three ingredients

Data: what data are we trying to explain with a model?

Model: what’s the compressed, parametric form of the data?

Objective function: given a prediction, how do we evaluate how correct it is?
Example: Least-Squares

Fitting a line to data

Data: \((x_1,y_1), (x_2,y_2), \ldots, (x_k,y_k)\)

Model: \((m,b)\) \(y_i = mx_i + b\)

Or \((w)\) \(y_i = w^T x_i\)

Objective function:
\((y_i - w^T x_i)^2\)
Least-Squares Setup

\[ \sum_{i=1}^{k} (y_i - w^T x_i)^2 \rightarrow \|Y - Xw\|_2^2 \]

\[
Y = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \quad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_k & 1 \end{bmatrix} \quad w = \begin{bmatrix} m \\ b \end{bmatrix}
\]
Solving Least-Squares

$$\|Y - Xw\|^2_2$$

Where can I find derivatives + matrix expressions and matrix identities?

**WolframAlpha**

**The Matrix Cookbook**

Kaare Brandt Petersen
Michael Syskind Pedersen

**VERSION: NOVEMBER 15, 2012**
Solving Least-Squares

\[ \|Y - Xw\|_2^2 \]

\[ \frac{\partial}{\partial w} \|Y - Xw\|_2^2 = 2X^TXw - 2X^TY \]

Recall: derivative is 0 at a maximum / minimum. Same is true about gradients.

\[ 0 = 2X^TXw - 2X^TY \]

\[ X^TXw = X^TY \]

\[ w = (X^TX)^{-1}X^TY \]

Aside: \( \mathbf{0} \) is a vector of 0s. \( \mathbf{1} \) is a vector of 1s.
Two Solutions to Getting $W$

In One Go

Implicit form (normal equations)

$$X^T X w = X^T Y$$

Explicit form (don’t do this)

$$w = (X^T X)^{-1} X^T Y$$

Iteratively

Recall: gradient is also direction that makes function go up the most.

What could we do?

$$w_0 = 0$$

$$w_{i+1} = w_i - \gamma \left( \frac{\partial}{\partial w} \| Y - Xw \|^2_2 \right)$$
What’s The Problem?

- Vertical lines impossible!
- Not rotationally invariant: the line will change depending on orientation of points
Alternate Formulation

Recall: \( ax + by + c = 0 \)

\( l^T p = 0 \)

\( l \equiv [a, b, c] \quad p \equiv [x, y, 1] \)

Can always rescale \( l \).

Pick another \( a, b, d \) so

\( \|n\|_2^2 = \|[a, b]\|_2^2 = 1 \)

\( d = -c \)
Alternate Formulation

Important part: Any line can be framed in terms of normal \( n \) and offset \( d \)

Now: \[ ax + by - d = 0 \]
\[ n^T [x, y] - d = 0 \]

Point to line distance:
\[
\frac{n^T [x, y] - d}{\|n\|_2^2} = n^T [x, y] - d
\]

\[ \| [a, b] \|_2^2 = 1 \]
Total Least-Squares

Fitting a line to data

Data: \((x_1,y_1), (x_2,y_2), \ldots, (x_k,y_k)\)

Model: \((n,d), \|n\|^2 = 1\)
\[n^T[x_i,y_i]-d=0\]

Objective function:
\[(n^T[x_i,y_i]-d)^2\]

\(n = [a, b]\)
\[\|[a, b]\|^2_2 = 1\]
Total Least Squares Setup

Figure out objective first, then figure out $\|n\|=1$

$$\sum_{i=1}^{k} \left( n^T [x, y] - d \right)^2 \rightarrow \|Xn - 1d\|_2^2$$

$$X = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_k & y_k \end{bmatrix} \quad 1 = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad n = \begin{bmatrix} a \\ b \end{bmatrix} \quad \mu = \frac{1}{k} 1^T X$$

The mean / center of mass of the points: np.sum(X,axis=0). We’ll use it later.
Total Least Squares Setup

Want to make sure that the following is minimized:

$$\|Xn - 1d\|^2_2$$

Won’t derive, but can show that whenever you find the $n$, and $d$ that minimize the objective, $d = \mu n$.

(at back of slides if you’re curious.)

$$X = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_k & y_k \end{bmatrix} \quad 1 = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad n = \begin{bmatrix} a \\ b \end{bmatrix} \quad \mu = \frac{1}{k}1^T X$$

The mean / center of mass of the points:

np.sum(X,axis=0). We’ll use it later
Solving Total Least-Squares

\[ \|Xn - 1d\|_2^2 = \|Xn - 1\mu n\|_2^2 \quad d = \mu n \]

\[ = \|(X - 1\mu) n\|_2^2 \]

**Objective is then:**

\[ \arg \min_{\|n\| = 1} \|(X - 1\mu) n\|_2^2 \]

The thing that makes the expression smallest
Homogeneous Least Squares

\[ \arg \min_{v} \|Av\|_2^2 \quad \rightarrow \quad \text{Eigenvector corresponding to smallest eigenvalue of } A^TA \]

Why do we need \( \|v\|^2 = 1 \) or some other constraint?

Applying it in our case:

\[ n = \text{smallest}_\text{eigenvec}((X - 1\mu)^T(X - 1\mu)) \]

Note: technically homogeneous only refers to \( ||Av||=0 \) but it’s common shorthand in computer vision to refer to the specific problem of \( ||v||=1 \).
Details For ML-People

Matrix we take the eigenvector of looks like:

\[(X - 1\mu)^T(X - 1\mu) = \begin{bmatrix}
\sum_i (x_i - \mu_x)^2 & \sum_i (x_i - \mu_x)(y_i - \mu_y) \\
\sum_i (x_i - \mu_x)(y_i - \mu_y) & \sum_i (y_i - \mu_y)^2
\end{bmatrix}\]

This is a scatter matrix or scalar multiple of the covariance matrix. We’re doing PCA, but taking the least principal component to get the normal.

Note: If you don’t know PCA, just ignore this slide; it’s to help build connections to people with a background in data science/ML.
Running Least-Squares
Running Least-Squares
Ruining Least Squares
Ruining Least Squares
Ruining Least Squares

Way to think of it #1:
\[ \| Y - Xw \|_2^2 \]

100^2 >> 10^2: least-squares prefers having no large errors, even if the model is useless overall

Way to think of it #2:
\[ w = (X^T X)^{-1} X^T Y \]

Weights are a linear transformation of the output variable: can manipulate W by manipulating Y.
Outliers in Computer Vision

Single outlier: \textit{rare}

Many outliers: \textit{common}
Ruining Least Squares Continued
Ruining Least Squares Continued
A Simple, Yet Clever Idea

• *What we really want:* model explains *many* points “well”

• *Least Squares:* model makes as few big mistakes as possible over the entire dataset

• *New objective:* find model for which error is “small” for as many data points as possible

• *Method:* RANSAC (*RA*ndom *SA*mple *C*onsensus)

RANSAC For Lines

bestLine, bestCount = None, -1
for trial in range(numTrials):
    subset = pickPairOfPoints(data)
    line = totalLeastSquares(subset)
    E = linePointDistance(data,line)
    inliers = E < threshold
    if #inliers > bestCount:
        bestLine, bestCount = line, #inliers
Running RANSAC

Lots of outliers!

Trial #1

Best Model: None

Best Count: -1
Running RANSAC

Trial #1

Fit line to 2 random points

Best Model: None

Best Count: -1
Running RANSAC

| Trial #1 | Best Model: None | Best Count: -1 |

Point/line distance $|n^T[x,y] - d|$
Running RANSAC

Distance < threshold
14 points satisfy this

Trial #1

Best Model: None

Best Count: -1
Running RANSAC

Distance < threshold
14 points

Best Model:

Best Count: 14

Trial #1
Running RANSAC

Distance < threshold
22 points

Trial #2

Best Model:

Best Count: 14
Running RANSAC

Distance < threshold
22 points

Trial #2

Best Model:

Best Count: 22
Running RANSAC

Distance < threshold
10

Trial #3

Best Model:

Best Count: 22
Running RANSAC

Trial #3

Best Model:

Best Count: 22
Running RANSAC

Distance < threshold
76

Best Model:

Best Count:
22
Running RANSAC

Trial #9

Distance < threshold
76

Best Model:

Best Count:
76
Running RANSAC

Trial #9

Best Model:

Best Count: 76
Running RANSAC

Trial #100

Distance < threshold

22

Best Model:

Best Count: 85
Running RANSAC

Final Output of RANSAC: Best Model
RANSAC In General

best, bestCount = None, -1
for trial in range(NUM_TRIALS):
    subset = pickSubset(data, SUBSET_SIZE)
    model = fitModel(subset)
    E = computeError(data, line)
    inliers = E < THRESHOLD
    if #(inliers) > bestCount:
        best, bestCount = model, #(inliers)
    (often refit on the inliers for best model)
Parameters – Num Trials

\( r \) is the fraction of outliers (e.g., 80%)
Suppose we pick \( s \) points (e.g., 2)
we run RANSAC \( N \) times (e.g., 500)

What’s the probability of picking a sample set with no outliers?
\[ \approx (1 - r)^s \]  
(4%)

What’s the probability of picking a sample set with any outliers?
\[ 1 - (1 - r)^s \]  
(96%)
Parameters – Num Trials

$r$ is the fraction of outliers (e.g., 80%)

Suppose we pick $s$ points (e.g., 2)
we run RANSAC $N$ times (e.g., 500)

What’s the probability of picking a sample set with any outliers?

$$1 - (1 - r)^s$$

What’s the probability of picking only sample sets with outliers?

$$(1 - (1 - r)^s)^N$$

What’s the probability of picking any set with inliers?

$$1 - (1 - (1 - r)^s)^N$$
Parameters – Num Trials

Death by vending machine

RANSAC fails to fit a line with 80% outliers after trying only 500 times

P($157M Jackpot): 1 / 302,575,350

P(Death): ≈1 / 112,000,000

P(Failure): 1 / 731,784,961

Odds/Jackpot amount from 2/7/2019 megamillions.com, unfortunate demise odds from livescience.com
Parameters – Num Trials

$r$ is the fraction of outliers (e.g., 80%)
Suppose we pick $s$ points (e.g., 2)
we run RANSAC $N$ times (e.g., 500)
Parameters – Subset Size

• Always the smallest possible set for fitting the model.
• Minimum number for lines: 2 data points
• Minimum number for 3D planes: how many?

• Why the minimum intuitively?
• You’ll find out more precisely in homework 3.
Parameters – Threshold

- No magical threshold
# RANSAC Pros and Cons

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ridiculously simple</td>
<td>1. Have to tune parameters</td>
</tr>
<tr>
<td>2. Ridiculously effective</td>
<td>2. No theory (so can’t derive parameters via theory)</td>
</tr>
<tr>
<td>3. Works in general</td>
<td>3. Not magic, especially with lots of outliers</td>
</tr>
</tbody>
</table>

List credit: S. Lazebnik
Hough Transform

Image credit: S. Lazebnik
Hough Transform

1. Discretize space of parametric models
2. Each pixel votes for all compatible models
3. Find models compatible with many pixels


Slide design credit: S. Lazebnik
Hough Transform

Line in image = point in parameter space

\[ y = m_0 x + b_0 \]

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real
Hough Transform

Point in image = line in parameter space

All lines through the point: \( b = x_0 m + y_0 \)

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real
Hough Transform

Point in image = line in parameter space

All lines through the point: \( b = x_1 m + y_1 \)

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real
Hough Transform

Point in image = line in parameter space
All lines through the point: \( b = x_1 m + y_1 \)

If a point is compatible with a line of model parameters, what do two points correspond to?

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real
Hough Transform

Line through two points in image = intersection of two lines in parameter space (i.e., solutions to both equations)

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real
Hough Transform

Line through two points in image = intersection of two lines in parameter space (i.e., solutions to both equations)

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real
Hough Transform

- *Recall*: m, b space is awful
- ax+by+c=0 is better, but *unbounded*
- Trick: write lines using angle + offset (normally a mediocre way, but makes things bounded)

\[ x \cos(\theta) + y \sin(\theta) = \rho \]

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real.
Hough Transform Algorithm

Remember: \( x \cos(\theta) + y \sin(\theta) = \rho \)

Accumulator \( H = \text{zeros}(?,?) \)

For \( x,y \) in detected_points:
  
  For \( \theta \) in range(0, 180, ?):
    
    \[
    \rho = x \cos(\theta) + y \sin(\theta)
    \]
    
    \( H[\theta, \rho] += 1 \)

any local maxima \((\theta, \rho)\) of \( H \) is a line

of the form \( \rho = x \cos(\theta) + y \sin(\theta) \)

Diagram is remake of S. Seitz slides
Example

Points \((x, y) \rightarrow\) sinusoids

Image Space  Parameter Space

Peak corresponding to the line

Few votes

Slide Credit: S. Lazebnik
## Hough Transform Pros / Cons

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Handles <strong>multiple</strong> models</td>
<td>1. Have to bin <strong>ALL</strong> parameters: exponential in <strong>#params</strong></td>
</tr>
<tr>
<td>2. Some robustness to noise</td>
<td>2. Have to parameterize your space nicely</td>
</tr>
<tr>
<td>3. In principle, general</td>
<td>3. Details really, really important (a working version requires a lot more than what I showed you)</td>
</tr>
</tbody>
</table>
Next Time

• What happens with fitting more complex transformations?
Details for the Curious
Least Squares
Derivation for the Curious

\[ \|Y - Xw\|^2_2 = (Y - Xw)^T (Y - Xw) \]
\[ = Y^T Y - 2w^T X^T Y + (Xw)^T Xw \]

\[ \frac{\partial}{\partial w} (Xw)^T (Xw) = 2 \left( \frac{\partial}{\partial w} Xw^T \right) Xw = 2X^T Xw \]

\[ \frac{\partial}{\partial w} \|Y - Xw\|^2_2 = 0 - 2X^T Y + 2X^T Xw \]
\[ = 2X^T Xw - 2X^T Y \]
Total Least Squares

• In the interest of less material better, I’m giving that $d = \mu n$.
• This can be derived by solving for $d$ at the optimum in terms of the other variables.
Solving Total Least-Squares

\[\|X_n - 1d\|_2^2 = (X_n - 1d)^T(X_n - 1d)\]
\[= (X_n)^T(X_n) - 2d1^TXn + d^21^T1\]

First solve for \(d\) at optimum (set to 0)

\[\frac{\partial}{\partial d} \|X_n - 1d\|_2^2 = 0 - 21^TXn + 2dk\]

\[0 = -21^TXn + 2dk \quad \Rightarrow \quad 0 = -1^TXn + dk\]

\[\rightarrow d = \frac{1}{k}1^TXn = \mu n\]
Common Fixes

Replace Least-Squares objective

Let \( E = Y - XW \)

LS/L2/MSE: \( E_i^2 \)

L1: \( |E_i| \)

Huber:

\[
|E_i| \leq \delta: \quad \frac{1}{2}E_i^2
\]

\[
|E_i| > \delta: \quad \delta(|E_i| - \frac{\delta}{2})
\]
Issues with Common Fixes

• Usually complicated to optimize:
  • Often no closed form solution
  • Typically not something you could write yourself
  • Sometimes not convex (local optimum is not necessarily a global optimum)

• Not simple to extend more complex objectives to things like total-least squares

• Typically don’t handle a ton of outliers (e.g., 80% outliers)