GENERAL

- Last hybrid class. Online only from next week.
  - Office hours (separate Zoom link) with me Tu, Th 1-1:30pm.
  - Recitation tomorrow for PSET 2 (will be recorded).
  - PSET 2 due Tuesday.

PROJECT

- Option 1: Read and analyze a computer vision paper.
  - Recent or classic from ICCV, ECCV, CVPR. (Suggestions posted)
  - Either implement, or if implementation available, modify / analyze.
  - Key is to demonstrate you understood method, and why it was needed.
- Option 2: Apply what you’ve learned in class to a problem you care about.
  - Read up on most relevant related work.
  - Implement adapted method for your problem.
  - Analyze results. Did it work? If so, how well. If not, why not.
  - (Should not involve work that is submitted to another course for credit.)
- Should be roughly 2x the effort of a problem set. (Let’s say, problem set 2)
- Avoid anything that requires training neural networks!

PROJECT

Read through the project section on the course website

Grading

- 25 points Report
  - Abstract: 3 pts (One paragraph succinct summary)
  - Introduction/Motivation: 5 pts
  - Why is this problem important, what is the vision task, prelude to rest of the report.
  - Related work: 4 pts
  - How have other people solved it? What are other similar problems? Read, describe.
  - Description / Experiments / Technical Correctness: 10 pts
  - Conclusion: 3 pts

2-3 Paragraph Proposal is due 11:59pm November 3rd.
Submit as a text file through git (repo will be created shortly).
LAST TIME

- Discussed Homogeneous Co-ordinates
- Book-keeping trick that makes non-linear projection linear.
- Talks about lines and points in 2D in homogeneous co-ordinates.
- Talked about expressing various 2D transformations as matrix multiplications with homogeneous co-ordinates.
- Homographies are the most general form of transformations:
  - Any 3x3 invertible matrix (defined upto scale)
  - Doesn’t preserve area, length, angles, orientation.
  - Straight lines stay straight: but can make parallel lines intersect.
- The transformation between two images of the same scene taken by different cameras is a homography if:
  - Either, the scene (or the part of the scene you’re applying this to) is entirely planar.
  - Or, the two cameras had the same camera center, and only differed in viewing direction.

ESTIMATION

I know a bunch of pairs of points \((p'_i, p_i)\), and want to find \(H\) such that:

\[ p'_i \sim Hp_i, \quad \forall i \]

- How many unknowns?
- How many equations for four points?

\[ u \times v = \|u\|\|v\| \sin \theta \hat{n}, \quad \text{where} \quad \hat{n}^T u = \hat{n}^T v = 0 \]
\[ p'_i \times (Hp_i) = 0 \]

Recall: \(u \times v = [(u_2v_3 - u_3v_2), (u_3v_1 - u_1v_3), (u_1v_2 - u_2v_1)]^T\)
How to re-write equations as linear systems in elements of $H$

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \quad \Rightarrow \quad h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$$

$p_i' \times (Hp_i) = 0 \quad \Rightarrow \quad A_i h = 0$

---

How to re-write equations as linear systems in elements of $H$

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \quad \Rightarrow \quad h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$$

$p_i' \times (Hp_i) = 0 \quad \Rightarrow \quad A_i h = 0$

Let $p_i = [p_{ix}, p_{iy}, p_{iz}]^T$: 
How to re-write equations as linear systems in elements of $H$

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \Rightarrow h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$$

$$p_i' \times (Hp_i) = 0 \Rightarrow A_i h = 0$$

Let $p_i = [p_{ix}, p_{iy}, p_{iz}]^T$:

$$Hp_i = \begin{bmatrix} h_1 p_{ix} + h_2 p_{iy} + h_3 p_{iz} \\ h_4 p_{ix} + h_5 p_{iy} + h_6 p_{iz} \\ h_7 p_{ix} + h_8 p_{iy} + h_9 p_{iz} \end{bmatrix}$$

3 x 9

How to re-write equations as linear systems in elements of $H$

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \Rightarrow h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$$

$$p_i' \times (Hp_i) = 0 \Rightarrow A_i h = 0$$

Let $p_i = [p_{ix}, p_{iy}, p_{iz}]^T$:

$$Hp_i = \begin{bmatrix} h_1 p_{ix} + h_2 p_{iy} + h_3 p_{iz} \\ h_4 p_{ix} + h_5 p_{iy} + h_6 p_{iz} \\ h_7 p_{ix} + h_8 p_{iy} + h_9 p_{iz} \end{bmatrix}$$

3 x 9
ESTIMATION

How to re-write equations as linear systems in elements of H

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \Rightarrow \quad h = [h_1, h_2, h_3, h_4, h_5, h_5, h_7, h_8, h_9]^T$$

$$p_i' \times (H_{pi}) = 0 \quad \Rightarrow \quad A_i h = 0$$

Let $$p_i = [p_{ix}, p_{iy}, p_{iz}]^T$$:

$$H_{pi} = \begin{bmatrix} h_1 p_{ix} + h_2 p_{iy} + h_3 p_{iz} \\ h_4 p_{ix} + h_5 p_{iy} + h_6 p_{iz} \\ h_7 p_{ix} + h_8 p_{iy} + h_9 p_{iz} \end{bmatrix} = \begin{bmatrix} p_{ix} & p_{iy} & p_{iz} \\ 0 & 0 & p_{ix} \\ 0 & 0 & p_{ix} \\ 0 & 0 & 0 \end{bmatrix} h$$

$$3 \times 9$$

ESTIMATION

How to re-write equations as linear systems in elements of H

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \Rightarrow \quad h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$$

$$p_i' \times (H_{pi}) = 0 \quad \Rightarrow \quad A_i h = 0$$

Let $$p_i = [p_{ix}, p_{iy}, p_{iz}]^T$$:

$$H_{pi} = \begin{bmatrix} h_1 p_{ix} + h_2 p_{iy} + h_3 p_{iz} \\ h_4 p_{ix} + h_5 p_{iy} + h_6 p_{iz} \\ h_7 p_{ix} + h_8 p_{iy} + h_9 p_{iz} \end{bmatrix} = \begin{bmatrix} p_{ix} & p_{iy} & p_{iz} \\ 0 & 0 & p_{ix} \\ 0 & 0 & p_{ix} \\ 0 & 0 & 0 \end{bmatrix} h$$

Cross product of two vectors as Matrix vector Multiply

$$u \times v = \begin{bmatrix} u_y v_x - u_x v_y \\ u_x v_y - u_y v_x \\ u_z v_x - u_x v_z \\ u_x v_z - u_z v_x \end{bmatrix}$$

$$3 \times 3$$
How to re-write equations as linear systems in elements of $H$

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \Rightarrow h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$$

$$p_i' \times (H p_i) = 0 \Rightarrow A_i h = 0$$

Let $p_i = [p_{ix}, p_{iy}, p_{iz}]^T$:

$$H p_i = \begin{bmatrix} h_1 p_{ix} + h_2 p_{iy} + h_3 p_{iz} \\ h_4 p_{ix} + h_5 p_{iy} + h_6 p_{iz} \\ h_7 p_{ix} + h_8 p_{iy} + h_9 p_{iz} \end{bmatrix} = \begin{bmatrix} p_{ix} & p_{iy} & p_{iz} \\ 0 & 0 & p_{ix} \\ 0 & 0 & 0 \end{bmatrix} h$$

Cross product of two vectors as Matrix vector Multiply

$$u \times v = \begin{bmatrix} u_y v_z - u_z v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{bmatrix} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} v$$

How to re-write equations as linear systems in elements of $H$

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \Rightarrow h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$$

$$p_i' \times (H p_i) = 0 \Rightarrow A_i h = 0$$

Let $p_i = [p_{ix}, p_{iy}, p_{iz}]^T$:

$$H p_i = \begin{bmatrix} h_1 p_{ix} + h_2 p_{iy} + h_3 p_{iz} \\ h_4 p_{ix} + h_5 p_{iy} + h_6 p_{iz} \\ h_7 p_{ix} + h_8 p_{iy} + h_9 p_{iz} \end{bmatrix} = \begin{bmatrix} p_{ix} & p_{iy} & p_{iz} \\ 0 & 0 & p_{ix} \\ 0 & 0 & 0 \end{bmatrix} h$$

Cross product of two vectors as Matrix vector Multiply

$$u \times v = \begin{bmatrix} u_y v_x - u_x v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{bmatrix} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} v$$
How to re-write equations as linear systems in elements of $H$

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \Rightarrow h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$$

$$p_i' \times (H p_i) = 0 \Rightarrow A_i h = 0$$

Let $p_i = [p_{ix} p_{iy} p_{iz}]^T$:

$$H p_i = \begin{bmatrix} h_1 p_{ix} + h_2 p_{iy} + h_3 p_{iz} \\ h_4 p_{ix} + h_5 p_{iy} + h_6 p_{iz} \\ h_7 p_{ix} + h_8 p_{iy} + h_9 p_{iz} \end{bmatrix} = \begin{bmatrix} p_{ix} & p_{iy} & 0 \\ 0 & 0 & p_{ix} \\ 0 & 0 & 0 \end{bmatrix} h$$

Cross product of two vectors as Matrix vector Multiply

$$u \times v = \begin{bmatrix} u_y v_z - u_z v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{bmatrix} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} v$$

The cross product gives us 3 equations, so $A_i$ is $3 \times 9$.

But, one of the rows of $A_i$ is a linear combination of the other ($A_i$ has rank 2). Can choose to keep only two rows, or all three.

Stacking all the $A_i$ matrices for all different correspondences, we get:

$$A h = 0$$

$A$ is $2n \times 9$ or $3n \times 9$ matrix, where $n$ is number of correspondences. Rank($A$) is at most $2n$.

Rank exactly equal to $2n$ if no three points are collinear.
So we have $Ah = 0$ and want to find $h$ up to scale. $A$ has rank $2n$ and $h$ has 9 elements.

Case 1: $n = 4$ non-collinear points.

- Trivial solution is $h = 0$. But want to avoid this.
- Cast as finding $Ah = 0$ such that $\|h\| = 1$.
- Since $A$ is exactly rank 8, there exists such a solution and it is unique (up to sign).
- Can find using eigen-decomposition / SVD.

$A = USV^T$ where $S$ is “diagonal” with last element 0. $h$ is the last column of $V$.

- $V$ will be $9 \times 9$ (call `np.linalg.svd` with `full_matrices=True`).

Case 2: $n > 4$ non-collinear points.

- Over-determined case. Want to find “best” solution.
- $h = \text{arg min}_h \|Ah\|^2$, $\|h\| = 1$
- Same solution, except that instead of taking 0 singular value, we take minimum singular value.
- $\|Ah\|^2 = (Ah)^T(Ah) = h^T (A^T A) h$
- Minimized by unit vector corresponding to lowest eigenvalue of $A^T A$, or lowest singular value of $A$.

Estimation from Lines

- How does a homography transform a line:
  $l^T p = 0 \leftrightarrow l'^T p' = 0$
  $l^T H^{-1} H p = 0 \Rightarrow (H^{-T} l)^T (H p) = 0$
  $l' = H^{-T} l \Rightarrow l = H^T l'$

- If we find four pairs of corresponding lines, we can get a similar set of equations for $l_i = H^T l'_i$ as for points.
- Get equations from $l_i \times (H^T l'_i) = 0$ for elements of $H$. 

Matching with point correspondences

Matching with line correspondences
Matching with line correspondences

Homography $H$

Degeneracy constraints?

- No $l_i$ vector a linear combination of other two
- No more than two parallel lines

$L_i \times H^T l_i' = 0$

Degeneracy constraints?

- No $l_i$ vector a linear combination of other two
- No more than two parallel lines

$L_i \times H^T l_i' = 0$

Not really surprising; can always find intersections of lines and match those.
**ESTIMATION**

Matching with line correspondences

\[
\begin{bmatrix}
H
\end{bmatrix}
\]

Homography

Degeneracy constraints?
- No 1 vector a linear combination of other two
- No more than two parallel lines
- No more than two lines intersecting at same point.

\[
l_i \times H^T l'_i = 0
\]

Not really surprising: can always find intersections of lines and match those.

**OTHER APPROACHES**

- Instead of measuring \([Ah]\)^2, might want to measure explicit geometric distance.
- Minimize distance in mapped cartesian co-ordinates (re-projection error).
- Involves division, no longer linear in \(H\). Iterative methods.
- See “Multiple View Geometry in Computer Vision,” Hartley & Zisserman: Section 4.2 (or really, the whole book for a thorough discussion of geometry)

**SAMPLING**

Say you want to take part of source, and warp it and place / blend it in destination:

Application:
- Blending / Compositing Images
**Sampling**

Say you want to take part of source, and warp it and place / blend it in destination:

![Source](image1)

![Destination](image2)

Given a set of Correspondences:

**Option 1**
- Estimate Homography from Source to Destination
- For every pixel in the input, apply homography to determine position in the output, and copy the pixel there.

---

**Sampling**

Say you want to take part of source, and warp it and place / blend it in destination:

![Source](image3)

![Destination](image4)

Given a set of Correspondences:

**Option 1**
- Estimate Homography from Source to Destination
- For every pixel in the input, apply homography to determine position in the output, and copy the pixel there.
SAMPLING

Say you want to take part of source, and warp it and place / blend it in destination:

Given a set of Correspondences:

Option 1
- Estimate Homography from Source to Destination
- For every pixel in the input, apply homography to determine position in the output, and copy the pixel there.

Round pixel location

SAMPLING

Say you want to take part of source, and warp it and place / blend it in destination:

Given a set of Correspondences:

Option 1
- Estimate Homography from Source to Destination
- For every pixel in the input, apply homography to determine position in the output, and copy the pixel there.

Round pixel location

SAMPLING

Say you want to take part of source, and warp it and place / blend it in destination:

Given a set of Correspondences:

Option 1
- Estimate Homography from Source to Destination
- For every pixel in the input, apply homography to determine position in the output, and copy the pixel there.

Round pixel location

SAMPLING

Say you want to take part of source, and warp it and place / blend it in destination:

Given a set of Correspondences:

Option 1
- Estimate Homography from Source to Destination
- For every pixel in the input, apply homography to determine position in the output, and copy the pixel there.

Round pixel location

Leaves gaps if there's scaling.
Say you want to take part of source, and warp it and place / blend it in destination:

**Option 1**
- Estimate Homography from Source to Destination
- For every pixel in the input, apply homography to determine position in the output, and copy the pixel there.

Round pixel location
Leaves gaps if there's scaling.
If multiple pixels map to same place, should average.

**Option 2**
- Estimate Homography from Destination to Source
SAMPLING

Say you want to take part of source, and warp it and place / blend it in destination:

Given a set of Correspondences:

Option 2
- Estimate Homography from Destination to Source
- For each pixel in destination, compute location in source, and copy from there.

ROBUST FITTING

Say you want to take part of source, and warp it and place / blend it in destination:

Given a set of Correspondences:

Option 2
- Estimate Homography from Destination to Source
- For each pixel in destination, compute location in source, and copy from there.

Bi-linear Interpolation:
\[ (x_0, y_0, w_0) = 0.7 \cdot (x_{10}, y_{10}) + 0.3 \cdot (x_{20}, y_{20}) \]
\[ 0.3 \cdot (x_{10}, y_{10}) + 0.7 \cdot (x_{20}, y_{20}) \]

If out of bounds, leave at zero / gray.

In reality, correspondences will be noisy:
- Automatic detection isn’t perfect
- Neither are user clicks!
**ROBUST FITTING**

In reality, correspondences will be noisy:
- Automatic detection isn’t perfect
- Neither are user clicks!

**ROBUST FITTING**

In reality, correspondences will be noisy:
- Automatic detection isn’t perfect
- Neither are user clicks!

Small perturbations in correspondences leads to small perturbations in estimate.

**ROBUST FITTING**

But sometimes, you have outliers: one correspondence that is completely wrong.

**ROBUST FITTING**

But sometimes, you have outliers: one correspondence that is completely wrong.

In reality, correspondences will be noisy:
- Automatic detection isn’t perfect
- Neither are user clicks!

Small perturbations in correspondences leads to small perturbations in estimate.
But sometimes, you have outliers: one correspondence that is completely wrong.

In reality, correspondences will be noisy:
- Automatic detection isn’t perfect
- Neither are user clicks!

Because our estimator tries to minimize the average error across all correspondences, the estimate is significantly skewed.

But sometimes, you have outliers: one correspondence that is completely wrong.

In reality, correspondences will be noisy:
- Automatic detection isn’t perfect
- Neither are user clicks!

Because our estimator tries to minimize the average error across all correspondences, the estimate is significantly skewed.

One large error is as bad as multiple medium-scale errors

Remember, we only needed 4 points to fit H anyway.

In reality, correspondences will be noisy:
- Automatic detection isn’t perfect
- Neither are user clicks!

Because our estimator tries to minimize the average error across all correspondences, the estimate is significantly skewed.

One large error is as bad as multiple medium-scale errors
ROBUST FITTING

For a bunch of samples denoted by set $C$:

$$h = \arg \min_{h} \sum_{i \in C} E_i(h), \text{ for some error function } E \text{ from sample } i$$

Robust Version:

$$h = \arg \min_{h} \sum_{i \in C} \min(\epsilon, E_i(h))$$

- Limits the extent to which an erroneous sample can hurt
- If a specific $E_i > \epsilon$, what is its gradient with respect to $h$?

ROBUST FITTING

For a bunch of samples denoted by set $C$:

$$h = \arg \min_{h} \sum_{i \in C} E_i(h), \text{ for some error function } E \text{ from sample } i$$

Robust Version:

$$h = \arg \min_{h} \sum_{i \in C} \min(\epsilon, E_i(h))$$

- Limits the extent to which an erroneous sample can hurt
- If a specific $E_i > \epsilon$, what is its gradient with respect to $h$? 0

So if I knew which $i \in C$ would have $E_i > \epsilon$,

$$h = \arg \min_{h} \sum_{i \in C \leq \epsilon} E_i(h)$$

Drop those samples, and solve the normal way (SVD, etc.)

ROBUST FITTING

Iterative Version:

$$h = \arg \min_{h} \sum_{i \in C} \min(\epsilon, E_i(h))$$

1. Fit the best $h$ to all samples in full set $C$.
2. Given the current estimate of $h$, compute the inlier set $C' = \{i : E_i(h) \leq \epsilon\}$
3. Update estimate of $h$ by minimizing error over only the inlier set $C'$
4. Goto step 2

Will this converge?

Consider the original robust cost $\min(\epsilon, E_i(h))$. Can step 3 ever increase the cost?
**ROBUST FITTING**

Iterative Version:

\[ h = \arg \min_h \sum_{i \in C} \min(e, E_i(h)) \]

1. Fit the best \( h \) to all samples in full set \( C \).
2. Given the current estimate of \( h \), compute the inlier set \( C' = \{ i : E_i(h) \leq \epsilon \} \).
3. Update estimate of \( h \) by minimizing error over only the inlier set \( C' \).

Will this converge?

Consider the original robust cost \( \min(e, E_i(h)) \). Can step 3 ever increase the cost?

- Before step 3, \( h \) had some cost over the inlier set, and cost over each outlier sample was \( \succ \epsilon \).
- Step 3 finds \( h \) with minimum cost over inlier set. So error can only decrease over inlier set.
- Step 3 can increase or decrease error over outlier set. But increased error doesn't hurt us, since it was already \( \succ \epsilon \) before step 3.

Stop when it stops decreasing. (Might oscillate between two solutions with same cost).

So method converges to some solution. Is it the global minimum?

No. It's possible that if I made one more point an outlier, that would increase its error to \( \succ \epsilon \), but reduce other errors by a lot.

Fundamentally a combinatorial problem. Only way to solve exactly is to consider all possible sub-sets of \( C \) as outlier sets.