CSE 559A: Computer Vision

Fall 2018: T-R: 11:30-1pm @ Lopata 101
Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Course Staff: Zhihao Xia, Charlie Wu, Han Liu
http://www.cse.wustl.edu/~ayan/courses/cse559a/
Oct 2, 2018

GENERAL

- Grace Hopper Attendees
- Last week’s lectures were recorded. Check Piazza for link to videos.
- Monday Office Hours in Jolley 217.
- Recitation this Friday, 10:30-Noon in Jolley 309.
- Typo in PSET 2 code comments: ntod should return HxW array (not HxWx3)
- In response to requests, slide numbers to help you take notes
  - Note that the posted slide PDFs have a subset of slides (avoid redundant transitions).
  - Slide numbers will stay the same, but won’t be contiguous in PDFs.
  - Posted HTML slides are exactly those presented in class.

LAST TIME: TRANSFORMATIONS

Most general form is a Homography:

$$p' = Hp$$

where $H$ is a general invertible $3 \times 3$ matrix.

- Defined upto scale. So 8 degrees of freedom.
- Defines mapping of co-ordinates of corresponding points in two images taken from different views:
  - If all corresponding points lie on a plane in the world.
  - If only the camera orientation has changed in two views (center is at the same place)

ESTIMATION

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

$$p'_i = Hp_i, \quad \forall i$$
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? 8 (defined up to scale)
- How many equations for four points? 8 (2 x 4)

But how do we write these equations for equality up to scale?

Recall:

\[
(p'_i, p_i)^T \sim (H p_i)^T, \quad \forall \ i \ p'_i \times (H p_i) = 0
\]

Recall: \(u \times v = [(u_2 v_3 - u_3 v_2), (u_3 v_1 - u_1 v_3), (u_1 v_2 - u_2 v_1)]^T\)

\[
p'_i \times (H p_i) = 0 \Rightarrow A_i h = 0
\]

Cross product gives us 3 equations, so \(A_i\) has rank 2.

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:

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

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

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

- Trivial solution is $\hat{h} = 0$. But want to avoid this.
- Cast as finding $A\hat{h} = 0$ such that $\|\hat{h}\| = 1$.
- Since $A$ is exactly rank 8, there exists such a solution and it is unique (up to sign).
- Can find using eigen-decomposition / SVD.
  
  $A = UDV^T$ where $D$ is diagonal with last element 0. $\hat{h}$ is the last column of $V$.

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

- Over-determined case. Want to find "best" solution.
  
  $\hat{h} = \text{arg min}_h \|A\hat{h}\|^2, \quad \|\hat{h}\| = 1$
  
  Same solution, except that instead of taking 0 singular value, we take minimum singular value.
  
  $\|A\hat{h}\|^2 = (Ah)^T(Ah) = \hat{h}^T(A^TA)\hat{h}$

- Minimized by unit vector corresponding to lowest eigenvalue of $A^TA$, or lowest singular value of $A$.

---

Estimation from Lines

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

- 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 $H$.

Other approaches:

- Instead of measuring $\|A\hat{h}\|^2$, 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**

Application:
- Blending / Compositing Images

*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.
If multiple pixels map to same place, should average.

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

While true Hom is inverse of Hom estimate of inverse not inverse of estimate.
SAMPLING

Say you want to take part of source, and warp it and place / blend 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:
\[
\{(x_1,y_1) = \begin{cases} x & \text{if } 0 \leq x < w_x \\ w_x & \text{if } w_x \leq x < 2w_x \\ 2w_x & \text{if } 2w_x \leq x < pw_x \\ pw_x & \text{if } pw_x \leq x < 2pw_x \\ \text{else} & \end{cases}, \\
\{(y_1,y_2) = \begin{cases} y & \text{if } 0 \leq y < w_y \\ w_y & \text{if } w_y \leq y < 2w_y \\ 2w_y & \text{if } 2w_y \leq y < pw_y \\ pw_y & \text{if } pw_y \leq y < 2pw_y \\ \text{else} & \end{cases} \}
\]

If out of bounds, leave at zero / gray.

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.

Remember, we only needed 4 points to fit H anyway. But need to know which 4.

Small perturbations in correspondences leads to small perturbations in estimate.

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:

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

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

So if I knew which $i \in C$ would have $E_i > c$ for the correct $h$,

$$h = \arg \min_{h} \sum_{i \in C^*} E_i(h)$$

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

ROBUST FITTING

Iterative Version:

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

1. Fit the best $h$ to all samples in full set $C$.
2. Given the current estimate of $h$, compute the inlier set $C^* = i : E_i(h) \leq c$
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 $\min(c, E_i(h))$. 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 $> c$.
- Step 3 finds the value of $h$ with minimum cost over inlier set. So error can only decrease over inliner set.
- Step 3 can increase or decrease error over outlier set. But increased error doesn’t hurt us, since it was already $> c$ before step 3.

RANSAC

Random Sampling and Consensus

Lots of different variants.

1. Randomly select $k$ points (correspondences) as my inlier set. Choice of $k$ can vary: has to be at least 4 for computing homographies.
2. Fit $h$ to these $k$ points.
3. Store $h$ and a measure of how good a fit $h$ is to all points. This measure can either be the thresholded robust cost, or the number of outliers.

Repeat this $N$ times to get $N$ different estimates of $h$ and associated costs.

Choose the $h$ with the lowest cost, and then refine using iterative algorithm.

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