CSE 559A: Computer Vision

GENERAL

- Recitation Tomorrow.
- Project proposal deadline pushed back: will post details next week.

RECAP

- Homogeneous Co-ordinates in 2D
  - Points and Lines and Cross Products
  - Different kinds of Transforms
- Estimating Transforms: Key Ideas
  - Write down constraint \( p' \sim Hp \), use \( p' \times (Hp) = 0 \)
  - Can solve equations of the form \( Ah = 0, ||h|| = 1 \) with SVD of \( A \).

\[ p'_i \times Hp_i = 0 \]
RECAP

Matching with line correspondences

$H$

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

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

RECAP

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

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

Cross product of two vectors as Matrix vector Multiply

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

RECAP

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

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

SAMPLING

Application:
- Blending / Compositing Images
SAMPLING

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.

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:
\[(x2, y2) = 0.75(x2, y1) + 0.25(x2+1, y1) + 0.25(x2, y1+1) + 0.75(x2+1, y1+1)\]

If out of bounds, leave at zero / gray.

ROBUST FITTING

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.

Given a set of Correspondences:

How do we fix this?

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

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.

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

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 \( C' = i : E_i(h) \leq c \)
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 \( \min(e, E_i(h)) \). 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 \( > c \).
- Step 3 find 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.

ROBUST FITTING

For a bunch of samples denoted by set \( C \):

\[ h = \arg \min_h \sum_{i \in C} E_i(h), \quad \text{for some error function } E \text{ from sample } i \]

Robust Version:

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

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

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:

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

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

Will this converge? Yes. Cost will never increase.
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 \( > c \), 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.
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 \( h \) these \( k \) points.
3. Store \( h \) and a cost. This can either be the 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.

CAMERA PROJECTION
3D Homogeneous Co-ordinates
- Four dimensional vector defined up to scale: \( p = [\alpha x, \alpha y, \alpha z, \alpha]^T \)
- If \( l \) is a four-dimensional vector, what does \( l^T p = 0 \) represent? A plane.
- How do we represent a line? \( L^T p = 0 \) where \( L \) is a \( 4 \times 2 \) matrix.
- Interpret line as intersection of two planes.

CAMERA PROJECTION
3D Transformations
Represented by \( 4 \times 4 \) matrices.
- Translation

\[
p' = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} p
\]

- Rotation

\[
p' = \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} p
\]

Where \( R \) is now a \( 3 \times 3 \) matrix with \( R^T R = I \).
Also covers reflection. For rotation only, \( R = R_x(\theta_1)R_y(\theta_2)R_z(\theta_3) \)

\[
R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}, \quad R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

Corresponds to rotation around each axis. Not commutative.
3D Transformations

General Euclidean Transformation

\[ p' = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} p \]

Now \(R\) is a \(3 \times 3\) rotation matrix, and \(t\) is a \(3 \times 1\) translation vector.