Introduction to Computer Vision Assignment #3

Due May. 20 (Mon)

1. Least Square Line Fitting

- The n 2D points data $\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, i = 1 \ldots n$ is provided. Visualize the data distribution.
- Calculate the second moment of the data and find its eigenvalues and eigenvectors.

$$\mathbf{U}^T\mathbf{U} = \begin{bmatrix} \sum_{i=1}^{n}(x_i - \bar{x})^2 & \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n}(y_i - \bar{y})^2 \end{bmatrix},$$

where $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_n^T \end{bmatrix} = \begin{bmatrix} (\mathbf{x}_1 - \boldsymbol{\mu})^T \\ \vdots \\ (\mathbf{x}_n - \boldsymbol{\mu})^T \end{bmatrix} = \begin{bmatrix} x_1 - \tilde{x} & y_1 - \tilde{y} \\ \vdots & \vdots \\ x_n - \tilde{x} & y_n - \tilde{y} \end{bmatrix}$, and $\boldsymbol{\mu} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n} x_i \\ \frac{1}{n}\sum_{i=1}^{n} y_i \end{bmatrix}$.

- Visualize the geometric interpretation of the eigenvalues and eigenvectors w.r.t the data distribution.
- Find the optimal line $\mathbf{l}$: $ax + by = d$ that fits the data in the total least square sense, $E = \sum_{i=1}^{n}(ax_i + by_i - d)^2$. $\mathbf{n} = \begin{bmatrix} a \\ b \end{bmatrix}$ is the normal vector of the line $\mathbf{l}$. Visualize the optimal line on the data distribution.

2. Another Interpretation of Least Square Line fitting

- The least-square fitting can be interpreted as finding the optimal projection line **l** that maximizes the variance of the projected samples of the original data on **l**.
- Consider the mean-zero data set $\mathbf{u_i} = \mathbf{x}_i - \boldsymbol{\mu} = \begin{bmatrix} x_i - \tilde{x} \\ y_i - \tilde{y} \end{bmatrix}$. If we project $\mathbf{u_i}$ onto a vector **v,** it becomes $\mathbf{v}^T\mathbf{u_i}$. Then, the variance of the projected samples of $\mathbf{u}_{i,i=1\ldots n}$ on **v** is

$$var(\mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}^T\mathbf{u_i}(\mathbf{v}^T\mathbf{u_i})^T = \mathbf{v}^T\boldsymbol{\Sigma}\mathbf{v} \text{ , where } \boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{u_i}(\mathbf{u_i})^T.$$

- Show that the vector **v** that maximizes the variance of the projected samples of $\mathbf{u_i}$ is the principal component, i.e., the eigenvector associated with the largest eigenvalues of $\boldsymbol{\Sigma}$.
- What the relationship between $\boldsymbol{\Sigma}$ and the second moment matrix $\mathbf{U}^T\mathbf{U}$?
- What is the relationship between **v** and the line normal vector $\mathbf{n}$ that you obtained from the second moment matrix?

3. Homography and Image Stitching

You will implement an image stitching algorithm that automatically stitches multiple images of a scene into a single panoramic image using image warping and homographies.

Implementation

- **Taking panoramic pictures**
  - Obtain your own images (3 images, I_1, I_2, I_3) by taking pictures of a scene in different views by rotating your smart phone camera. Be sure to maintain the same center of projection while taking pictures and adequate overlap between neighboring views are required. To avoid lens distortion, do not use wide-angle lens or mode of your camera.
  - Resize your images into 256 x 256.

- **Feature Extraction**
  - Detect feature points in each image.
  - You need to convert color images to grey images (`rgb2gray`)
  - You can use SIFT feature and its, OpenCV version, Python version, C++ version, or its MATLAB version vl_sift (set the threshold to extract about 200-300 points in each image).
  - You have to make the feature points evenly distributed over each image and avoid multiple detections at the same point. (Think about how to get features extracted evenly across an image).

- **Feature Matching**
  - Find the putative matches between images I_i and I_{i+1}.
  - For measuring the uniqueness of the correspondences, use the ratio of the distance between the best matching keypoint and the distance to the second-best one. You can use the code; Brute-Force Matching with SIFT Descriptors and Ratio Test (OpenCV), or vl_ubcmatch (MATLAB).
  - Display the detected feature correspondences using lines or vectors overlaid on the images.

- **Homography Estimation using RANSAC**
  - Using the detected putative correspondences between I_i and I_j, you can estimate the 3x3 Homography matrix H_{ij} between two images using RANSAC (Algorithm 1) and DLT method (Algorithm 2).

Objective

Compute homography between two images

Algorithm 1 **Automatic Estimation of H using RANSAC**

(i) **Interest points:** Compute interest points in each image
(ii) **Putative correspondences:** Compute a set of interest point matches based on some similarity measure
(iii) **RANSAC robust estimation:** Repeat for N samples
  (a) Select 4 correspondences and compute **H** using **DLT** method
  (b) Calculate the distance $d_\perp$ for each putative match
  (c) Compute the number of inliers consistent with **H** ($d_\perp < t$)
  Choose **H** with most inliers
(iv) (Option) **Optimal estimation:** re-estimate **H** from all inliers by minimizing ML cost function with DLT or Levenberg-Marquardt Algorithm
(v) (Option) Guided matching: Determine more matches using prediction by computed **H**

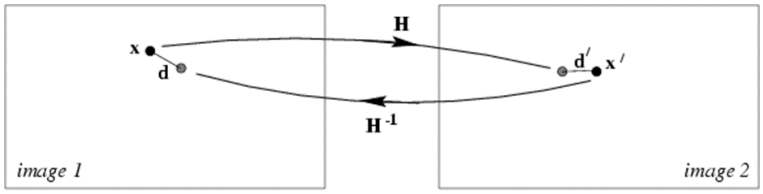
Optionally iterate last two steps until convergence

Objective

Algorithm 3 Adaptive determination of the # of samples for **RANSAC**

(i) $N = \infty$, $sample\_count = 0$.
(ii) While $N > smaple\_count$ Repeat
  (a) Choose a sample and count the number of inliers.
  (b) Set $\varepsilon = 1 - \frac{number\ of\ inliers}{total\ number\ of\ points}$
  (c) Set N from ε and $N = \log(1-p)/\log\left(1-(1-\varepsilon)^s\right)$ with $s = 4$, and $p = 0.99$.
  (d) Increment the *sample_count* by 1.
(iii) Terminate

- Implement a Homography estimation function using RANSAC,
  `Hij = HbyRANSAC(fi,fj)` for two vectors of corresponding features in `Ii` and `Ij`, respectively (four point features each).
- Use the symmetric transfer error for the distance measure $d_{\perp} = d(\mathbf{x}, \mathbf{H}^{-1}\mathbf{x}')^2 + d(\mathbf{x}', \mathbf{H}\mathbf{x})^2$



- Use DLT for determining *H* for a sampled 4 correspondences. For randomly sampling matches, you can use the `randperm` or `randsample` functions. The solution to the homogeneous least squares system $\boldsymbol{Ah} = 0$ is obtained from the SVD of $\mathbf{A}_{nx9}$ by the singular vector corresponding to the smallest singular value:
  ```
  [U,S,V]=svd(A); X = V(:,end);
  ```
- Set the RANSAC parameter values: *t=1.25, p=0.99*
- Use the adaptive determination method for the # of samples *N*.
- Determine the homographies H_{12}, H_{32} using this function.
- **Reference**: CH4. of "Multiple View Geometry in Computer Vision" by Richard Hartley and Andrew Zisserman, 2nd Ed., Cambridge University Press, March 2004

- **Warping Images**
  - Implement a warping function that transform image I_1 and I_3 to the center image I_2 using the homography H_{12} and H_{32}, respectively.
  - Mosaic the three images by warping other images to the plane of I_2. (Backward warping using inverse homography matrix is recommended for better visualization).
  - For color images, you can use the same homography for each color channel, and warp each RGB channel separately and then stack together to form the output.

- **Result image**
  - Report the final homography estimation results and the total symmetric transfer error for each `H`.
  - Display the intermediate pairwise matching results, and the final mosaiced image.
  - Discuss any issues or artifacts that may be evident in your output.

Submission instructions: what to hand in:

- Upload the electronic file that includes the report, source code, and data in a single zip format with the name "**ICV_assignment#3_yourname.zip**" on the ETL class homepage.
- The report should include the brief description of the problems, results, and discussions

Note: All works should be individual-based. Copy or using existing codes is not allowed.