Junseo Kim

Assignment 4

Summer 2023

The objective of this course is to learn how to distinguish inliers from out-liers in statistical inference. For more information, refer to 'Ch 5.3 Handling Outliers' of your textbook.

Given the source and destination 2d points below, compute a homography that maps the source points to the destination. a minimum of 10 inliers needed tobe detected by the RANSAC algorithm. The re-projection error of the inliers need to be less than 0.005.

Report your normalized homography transformation. Write your code in Python, do not use OpenCV. Plot a scatter plot, showing inliers (marked by o)and outliers (marked by x), where inliers of source and destination are connected via a line. Include the plot in your report. You should only use the following modules:
import numpy as npimport
random
import matplotlib.pyplot as plt

In the first part, homography should be defined based on the source points and destination points.

Assuming the source points are in (x,y) form and the destination points are in (u,v), the following equation is considered for the homography:

$$A = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -u*x & -u*y & -u \\ 0 & 0 & 0 & x & y & 1 & -v*x & -v*y & -v \end{bmatrix}$$

Once matrix A is constructed, Singular Value Decomposition (SVD) was used for the system. This will generate matrix V.
The last row of the matrix V will give the solution of the system. This is reshaped into 3x3 matrix for the final homography matrix.

RANSAC

1. Get four random points

As homography is defined by 8 parameters, with each point pair providing 2 equations, we need at least 4 point pairs to solve for 8 unknowns.

2. Compute H using DLT

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Assuming H is the homography matrix, (x,y) are the coordinates for source points and (x', y') are the coordinates for projected points, projected points are converted to inhomogeneous coordinates by

dividing by w’

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} \frac{x'}{w'} \\ \frac{y'}{w'} \end{bmatrix}$$

3. Inlier detection

Euclidean distance between the projected points and destination points are calculated. As the threshold is set as 0.005 from the requirement, if the error is less than the threshold, the point pair will be considered as an inlier.

$$error = \sqrt{(u - x'')^2 + (v - y'')^2}$$

4. Update

If the number of inliers for the model is greater than the number of inliers for the best model, the function will update the best model and the maximum number of inliers.

```
Homography Transformation:
[[0.71377842 0.42900386 0.12835417]
 [0.28547466 0.17162952 0.42849705]
 [0.15697198 0.01441864 1.        ]]
```

Figure 1: Best model using RANSAC

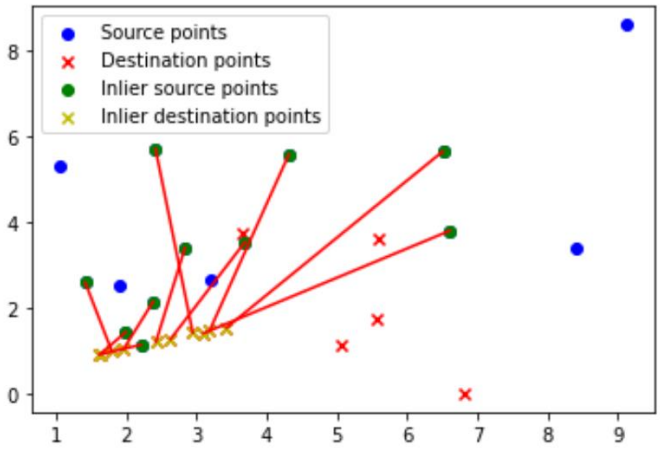

Figure 2: RANSAC example

From figure 2, 10 inliers are detected with 15 source points and destination points

Detailed code can be found from
https://github.com/JunseoKim19/State_estimation/blob/main/RANSAC.py

GNC (Graduated Non-Convexity)

Graduated Non-Convexity (GNC) algorithm is a robust estimation method to optimize non-convex due to the presence of outliers.

In addition to functions used in RANSAC, GNC algorithm refines the Homography until the residuals are minimized

The errors between the observed data points and the model's predictions will be considered:

$$Projected\ points = H\ \cdot source\ points$$

Once the projected points are normalized by their third coordinate and converted into inhomogeneous coordinates, Euclidean distance (L2 norm) can be computed
The residual can be computed as the following:

$$residual = \left\| d - \frac{H_s}{(H_s)_3} \right\|_{[1]}$$

where d is the destination points and H is the homography

Then, weights for each residual will be calculated using GNC algorithm

Weight can be described as the following:

$$w = \frac{1}{r + c}$$

where r is the residual and c is the constant
A new estimate of the homography matrix H is then computed with least squares problem to find the most minimized residual. The parameter "c" will be halved every iteration to control the influence of the residuals on the weights. Therefore, as "c" decreases, the weights will become more sensitive to the residuals and makes the system to focus more on the inliers.

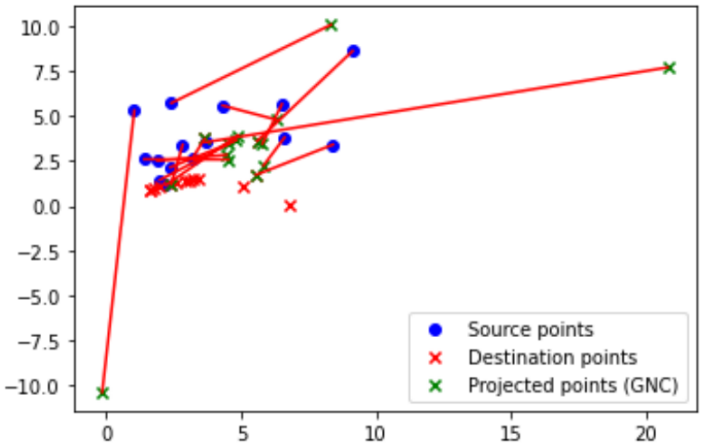

Figure 3: GNC example [1]
Detailed code can be found from
https://github.com/JunseoKim19/State_estimation/blob/main/GNC.py

ADAPT (Adaptive Trimming)

Adaptive Trimming (ADAPT) algorithm works very similar to the GNC algorithm.
The main difference happens in the convergence criterion.

Algorithm 1: Adaptive Trimming (ADAPT).

Input: Measurements $\boldsymbol{y}_i$, $\forall i \in \mathcal{M}$; thresholds τ, θ;
MaxIterations, *SamplesToConverg* > 0;
ThrDiscount $\in (0, 1)$.

Output: Estimate of $\boldsymbol{x}^\circ$ and corresponding inliers.

$\mathcal{I}^{(0)} = \mathcal{M}$; $\boldsymbol{x}^{(0)} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(0)}}, \boldsymbol{x}) \|_2^2$;
$\varepsilon^{(0)} = \mathit{ThrDiscount} \cdot \max_{i \in \mathcal{I}^{(0)}} r(\boldsymbol{y}_i, \boldsymbol{x}^{(0)})$; $j = 0$;
for $t = 1, \ldots,$ *MaxIterations* **do**
  $\mathcal{I}^{(t)} = \{i \in \mathcal{M} \text{ s.t. } r(\boldsymbol{y}_i, \boldsymbol{x}^{(t-1)}) \leq \varepsilon^{(t-1)}\}$;
  $\boldsymbol{x}^{(t)} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t)}}, \boldsymbol{x}) \|_2^2$;
  if $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t)}}, \boldsymbol{x}) \|_\ell < \tau$
  and $|\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t)}}, \boldsymbol{x}) \|_2^2 - \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t-1)}}, \boldsymbol{x}) \|_2^2| <$
   $\theta(|\mathcal{I}^{(t-1)}|, |\mathcal{I}^{(t)}|)$ **then**
    $j++$;
  else
    $j = 0$;
  end
  if $j =$ *SamplesToConverg* **then**
    break;
  end
  $\varepsilon^{(t)} = \mathit{ThrDiscount} \cdot \max_{i \in \mathcal{I}^{(t)}} r(\boldsymbol{y}_i, \boldsymbol{x}^{(t)})$;
end
return $(\boldsymbol{x}^{(t)}, \mathcal{I}^{(t)})$.

Figure 4: ADAPT algorithm [2]

From the GNC algorithm, the system will stop computing when the maximum residual is below the threshold which is 0.005 in this case. However, this can be very unstable and ADAPT algorithm therefore, refines the homography until it becomes stable. Some residuals may be little bit above the threshold, but will be considered to make the homography stable

$$\|H_{new} - H\| < threshold$$

where || || is in Frobenius norm.

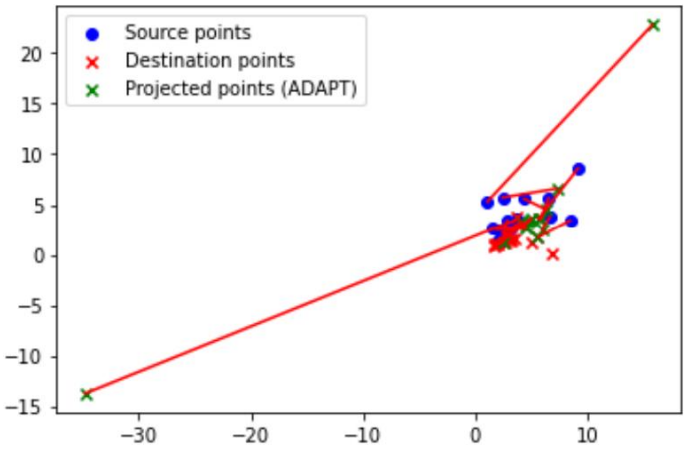

Figure 5: ADAPT example [2]

Detailed Code can be found from
https://github.com/JunseoKim19/State_estimation/blob/main/ADAPT.py

However, this is very basic implementation of GNC and ADAPT algorithm and therefore, more steps will be required in complex cases.

Reference

[1] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, “Graduated non-convexity for robust spatial perception: From non-minimal solvers to Global Outlier rejection,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1127–1134, 2020. doi:10.1109/lra.2020.2965893

[2] P. Antonante, V. Tzoumas, H. Yang, and L. Carlone, “Outlier-robust estimation: Hardness, minimally tuned algorithms, and applications,” *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 281–301, 2022. doi:10.1109/tro.2021.3094984