CSCI 5561: Assignment #5

Stereo Reconstruction

1 Submission

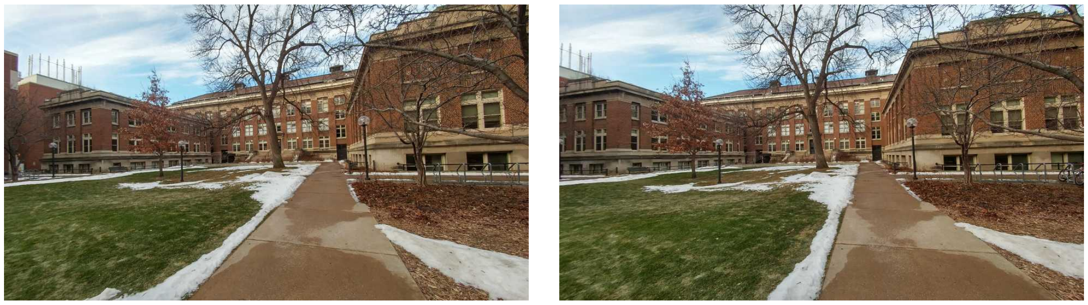
- Assignment due: Tuesday April 23, 2024 (11:59:59pm)
- Individual assignment
- Up to 2 page summary write-up with resulting visualization (more than 2 page assignment will be automatically returned.).
- Submission through Canvas.
- List functions to submission:
  - `compute_F`
  - `triangulation`
  - `disambiguate_pose`
  - `compute_rectification`
  - `dense_match`
- DO NOT SUBMIT THE PROVIDED IMAGE AND DATA
- The function that does not comply with its specification will not be graded.
- The code must be run with Python 3 interpreter.
- You are not allowed to use computer vision related package functions unless explicitly mentioned here. Please consult with TA if you are not sure about the list of allowed functions.

CSCI 5561: Assignment #5

Stereo Reconstruction

2 Overview

In this assignment, you will implement a stereo reconstruction algorithm given two view images.



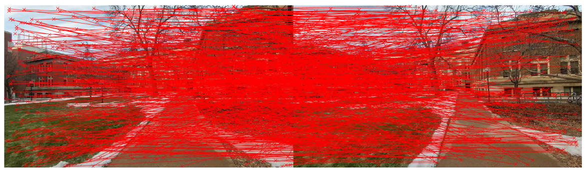
(a) Left image (b) Right image

Figure 1: In this assignment, you will implement a stereo reconstruction algorithm given two images.

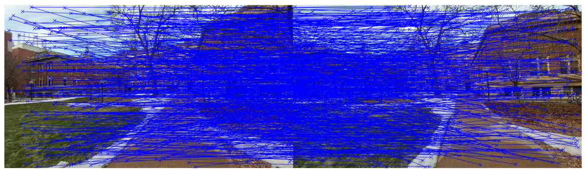
You can download the images and data from the homework 5 page on Canvas.

You will fill each function to submit such that the skeletal code in `Stereo_Reconstruction` can run through and produces a stereo disparity map.

3 SIFT Feature Matching

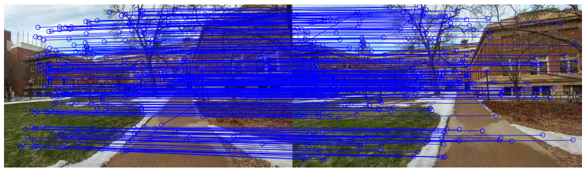


(a) Matching from $\mathcal{I}_1$ to $\mathcal{I}_2$

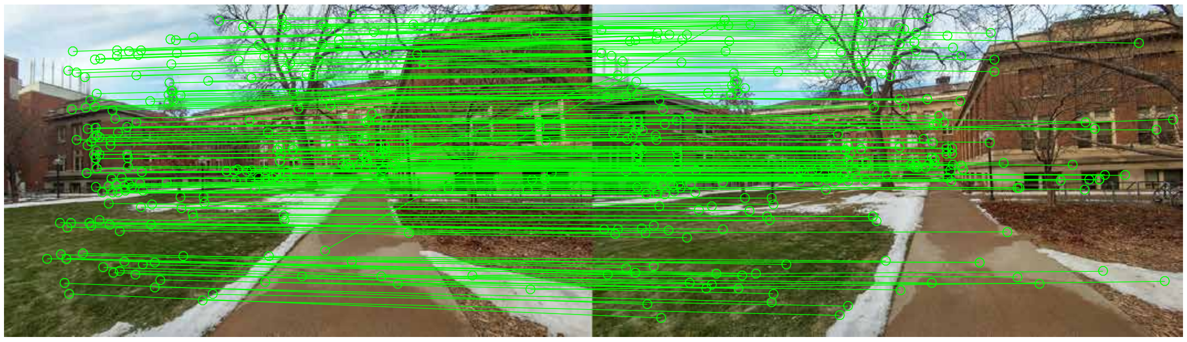


(b) Matching from $\mathcal{I}_2$ to $\mathcal{I}_1$

(c) Matching from $\mathcal{I}_1$ to $\mathcal{I}_2$ after ratio test



(d) Matching from $\mathcal{I}_2$ to $\mathcal{I}_1$ after ratio test



(e) Bidirectional matching between $\mathcal{I}_1$ and $\mathcal{I}_2$

Figure 2: You will match points between $\mathcal{I}_1$ and $\mathcal{I}_2$ using SIFT features.

For this homework, we provide SIFT feature matching results with both ratio test and bidirectional consistency check stored in `correspondence.npz`. You can use the provided `visualize_find_match` function to visualize the correspondences.

(Note) It is not required but feel free to find correspondences on your own.

CSCI 5561: Assignment #5

Stereo Reconstruction

4 Fundamental Matrix Computation

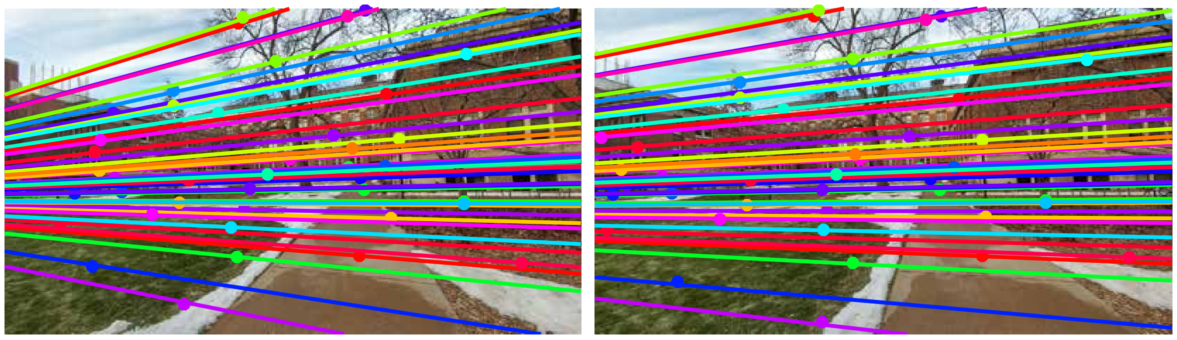


Figure 3: Given matches, you will compute a fundamental matrix to draw epipolar lines.

```
def compute_F(pts1, pts2):
  ...
  return F
```

Input: `pts1` and `pts2` are $n \times 2$ matrices that specify the correspondence.
Output: `F` $\in \mathbb{R}^{3\times 3}$ is the fundamental matrix.
Description: `F` is robustly computed by the 8-point algorithm within RANSAC. Note that the rank of the fundamental matrix needs to be 2 (SVD clean-up should be applied.). You can verify the validity of fundamental matrix by visualizing epipolar line as shown in Figure 3.

CSCI 5561: Assignment #5

Stereo Reconstruction

(Note) Given the fundamental matrix, you can get camera poses using the PROVIDED function:

```
def compute_camera_pose(F, K)
  ...
  return Rs, Cs
```

This function computes the four sets of camera poses given the fundamental matrix where `Rs, Cs` are python lists of rotation matrices (3×3) and camera centers (3×1) respectively (represented in the world coordinate system) and `K`$\in \mathbb{R}^{3\times3}$ is the PROVIDED intrinsic parameter. These four configurations can be visualized in 3D using PROVIDED function `visualize_camera_poses` as shown in Figure 4.

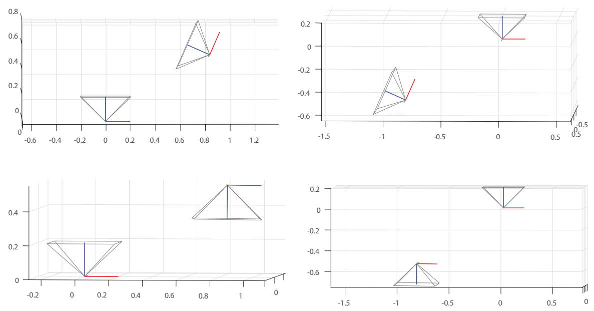


Figure 4: Four configurations of camera pose from a fundamental matrix.

CSCI 5561: Assignment #5

Stereo Reconstruction

5 Triangulation

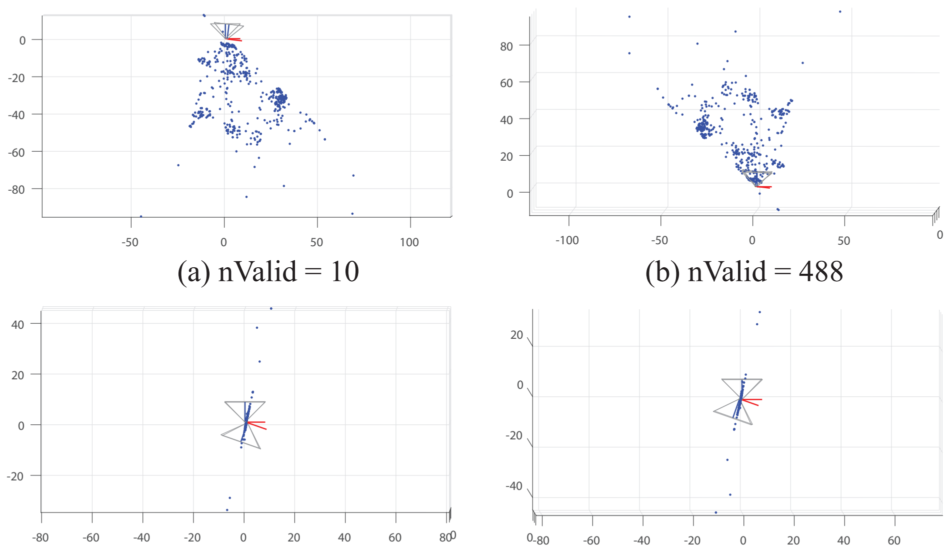
Given camera pose and correspondences, you will triangulate to reconstruct 3D points.

```
def triangulation(P1, P2, pts1, pts2)
  ...
  return pts3D
```

Input: `P1` and `P2` are two camera projection matrices ($\mathbb{R}^{3\times4}$). `pts1` and `pts2` are $n \times 2$ matrices that specify the correspondence.
Output: `pts3D` is $n \times 3$ where each row specifies the 3D reconstructed point.
Description: You can use a linear triangulation method, i.e.,

$$\begin{bmatrix} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix}_{\times} \mathbf{P}_1 \\ \begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix}_{\times} \mathbf{P}_2 \end{bmatrix} \begin{bmatrix} \mathbf{pts3D} \\ 1 \end{bmatrix} = \mathbf{0}$$

(Note) Use PROVIDED function `visualize_camera_poses_with_pts` to visualize 4 sets of camera poses with reconstructed 3D point cloud as shown in Figure 5.



(a) nValid = 10

(b) nValid = 488

(c) nValid = 0

(d) nValid = 0

Figure 5: You can visualize four camera pose configurations with point cloud.

6 Pose Disambiguation

Given four configurations of relative camera pose and reconstructed points, you will find the best camera pose by verifying through 3D point triangulation.

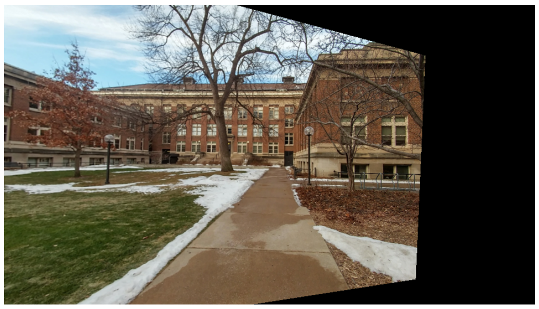
```
def disambiguate_pose(Rs, Cs, pts3Ds)
  ...
  return R, C, pts3D
```

Input: `Rs, Cs, pts3Ds` are python lists of rotation matrices, camera centers and 3D reconstructed points respectively
Output: `R`, `C`, `pts3D` are the best camera rotation, center, and 3D reconstructed points.
Description: The 3D point must lie in front of the both cameras. In Figure 5, nValid means the number of points that are in front of both cameras. (b) configuration produces the maximum number of valid points, and therefore it is the best configuration.

CSCI 5561: Assignment #5

Stereo Reconstruction

7 Stereo


(a) Rectified image 1

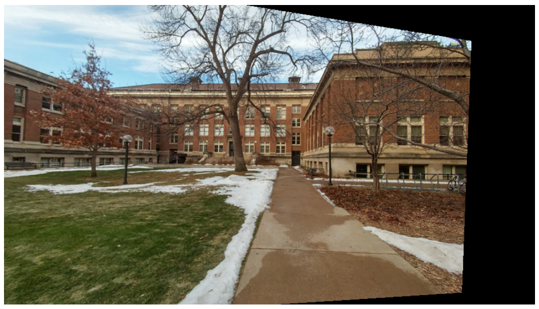

(b) Rectified image 2

Figure 6: Stereo rectification.

Given the disambiguated camera pose, you will implement dense stereo matching between two views based on dense SIFT.

```
def compute_rectification(K, R, C)
  ...
  return H1, H2
```

Input: The relative camera pose (`R` and `C`) and intrinsic parameter `K`.
Output: `H1` and `H2` are homographies that rectify the left and right images such that the epipoles are at infinity.
Description: Given the disambiguated camera pose, you can find the rectification rotation matrix, $\mathbf{R}_{\text{rect}}$ such that the x-axis of the images aligns with the baseline. Find the rectification homography $\mathbf{H} = \mathbf{K}\mathbf{R}_{\text{rect}}\mathbf{R}^{\mathsf{T}}\mathbf{K}^{-1}$ where $\mathbf{R}$ is the rotation matrix of the camera. The rectified images are shown in Figure 6. This rectification sends the epipoles to infinity where the epipolar line becomes horizontal.

As shown in the skeletal code, function `cv2.warpPerspective` is then applied to warp original images to get rectified image pair.

CSCI 5561: Assignment #5

Stereo Reconstruction

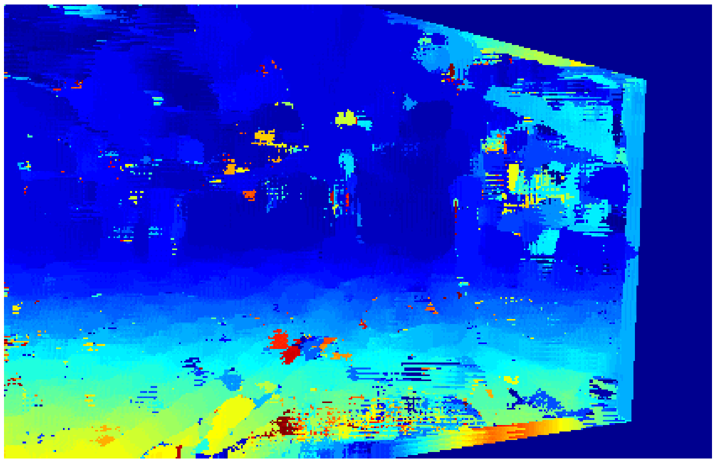


Figure 7: Visualization of stereo match.

`function [disparity] = dense_match(img1, img2, desp1, desp2)`
Input: two gray-scale rectified images in `uint8` format and their dense SIFT descriptors. `img1, img2` $\in \mathbb{R}^{H\times W}$, `desp1, desp2` $\in \mathbb{R}^{H\times W\times 128}$, where H and W are the image height and width.
Output: disparity map `disparity` $\in \mathbb{R}^{H\times W}$ where H and W are the image height and width.
Description: Compute the dense matches across all pixels. Given a pixel, $\mathbf{u}$ in the left image, sweep along its epipolar line, $\mathbf{l_u}$, and find the disparity, d, that produces the best match, i.e.,

$$d = \arg\min_i \|\mathbf{d}^1_{\mathbf{u}} - \mathbf{d}^2_{\mathbf{u}+(i,0)}\|^2 \quad \forall i = 0, 1, \cdots, N$$

where $\mathbf{d}^1_{\mathbf{u}}$ is the dense SIFT descriptor at $\mathbf{u}$ on the left image and $\mathbf{d}^2_{\mathbf{u}+(i,0)}$ is the SIFT descriptor at $\mathbf{u}+(i,0)$ (i pixel displaced along the x-axis) on the right image. Visualize the disparity map as shown in Figure 7.

(Note) We provide pre-computed dense SIFT descriptor stored in `dsift_descriptors.npz`. It is not required but feel free to extract dense SIFT descriptors on your own.