Solution 1

(a) To start, we look at the probability of a single sample being in the set of inliers, S_{in}, which can be expressed as the number of inlier points in S_{in}, $N - J$, divided by the total points, N.

$$P[n_1 \in S_{in}] = \frac{N - J}{N} \tag{1}$$

Since we are sampling with removal, it follows that the probability that the $i + 1$th sample is in the inlier set is

$$P[n_i \in S_{in}] = \frac{N - J - i}{N - i} \tag{2}$$

from this we can conclude that the probability of K samples $\{n_1,\ n_2,\ \dots\ ,\ n_K\}$ all being inliers is the product of the probability that each ith sample is an inlier

$$P[\forall\ i \in \{1,\ \dots\ ,\ K\},\ n_i \in S_{in}] = \prod_{i=0}^{K} \frac{N - J - i}{N - i} \tag{3}$$

(b) If we let the solution for part (a) in equation 3 be p, we know that the probability of any given sampled set being outlier free is p. We then use use a binomial distribution to calculate how many iterations are required to get confidence P. The PDF of a binomial distribution is defined by

$$P[X = k] = \binom{n}{k} p^k (1 - p)^{n-k} \tag{4}$$

where n is the number of trials, and k is the number of successes. In our case, we want to solve for n, with $k = 0$. This gives us the probability that every trial has at least one sample in the outlier set. Therefore, we can write the probability of at least one trial has no samples in the outlier set,

$$1 - P = (1 - p)^n \tag{5}$$

which we can take the log of to solve for n,

$$n \geq \frac{\log(1 - P)}{\log(1 - p)} = \frac{\log(1 - P)}{\log\left(1 - \prod_{i=0}^{K}\left(\frac{N - J - i}{N - i}\right)\right)} \tag{6}$$

Figure 1: some caption

(c) We start by defining I_1 and I_2 as

$$I_1 = N - J - I_2 \tag{7}$$

$$I_2 = N - J - I_1 \tag{8}$$

as the total number of inlier points on each plane. We then consider each case separately: the probability of getting all K points as inliers of I_1 and I_2 individually. We can use the same equation as in equation 3, replacing $N - J$ with I_1 and I_2:

$$P_{I_1} = P[\forall\ i \in \{1,\ \dots\ ,\ K\},\ n_i \in S_{I_1}] = \prod_{i=0}^{K} \frac{I_1 - i}{N - i} \tag{9}$$

$$P_{I_2} = P[\forall\ i \in \{1,\ \dots\ ,\ K\},\ n_i \in S_{I_2}] = \prod_{i=0}^{K} \frac{I_2 - i}{N - i} \tag{10}$$

We then consider the probability of either of these happening,

$$P[A \text{ or } B] = P[A] + P[B] + P[A \text{ and } B] \tag{11}$$

Since it is impossible for both events to happen at the same time, the "and" probability is 0 and we just get the sum of the individual events

$$P[\forall\, i \in \{1,\, \ldots\, ,\, K\},\, n_i \in (S_{I_1} \text{ or } S_{I_2})] = P_{I_1} + P_{I_2} \tag{12}$$

Solution 2

(a) In this problem we implement a robust version of least squares line fitting. The algorithm iteratively computes a least squares fit of a reduced data set that is designed to exclude outliers. Our error function is

$$E_i(h) = (y_i - h(x_i))^2, \tag{13}$$

$$h(x) = mx + b \tag{14}$$

and our goal is to minimize the clipped error,

$$h = \arg\min_h \sum_{i \in C} \min(\epsilon, E_i(h)) \tag{15}$$

where C is the sample set. To turn this into a least squares minimization problem, we observe that for points where $E_i(h) > \epsilon$ the gradient is 0, so we can just ignore them and use normal least-square to solve for the leftover points, the inlier set. Therefore, for each iteration of the algorithm we compute the inlier set of the current hypothesis space, and then run least-squares on that hypothesis set.

To solve the least squares problem, we define our minimization function, and set its gradient equal to 0

$$h = \arg\min_h \sum_{i \in S_{in}} (y_i - mx_i - b)^2 \tag{16}$$

$$\nabla(h) = \sum_{i \in S_{in}} 2 \begin{pmatrix} (y_i - mx_i - b)(-x_i) \\ (y_i - h_m x_i - h_b)(-1) \end{pmatrix} = \mathbf{0} \tag{17}$$

we can then rewrite this in matrix form,

$$Y = mX + b\mathbf{1} \tag{18}$$

$$Y = [X \mid \mathbf{1}] [m, b]^T \tag{19}$$

where $\mathbf{1}$ is a column vector of ones. We can then compute the pseudo-inverse of $[X \mid \mathbf{1}]$ to get the linear coefficients of the hypothesis h_m, h_b.

$$[m, b]^T = \text{pinv}([X \mid \mathbf{1}]) Y \tag{20}$$

Once we have our least squares hypothesis, we test each data point to see if it is an inlier or outlier,

$$\text{sign}(\epsilon - E_i(h)) \tag{21}$$

where inlier points are positive and outlier points are negative. We then compute a least squares fit on the new inlier set using equation 20, and repeat until we have completed the required iterations or there are too few inlier points to compute a best fit (less than 2 inlier points).

The result of this algorithm, compared to a standard least fit is shown in Figure 2. The top left is a standard fit of a data set with no outliers. Since there is no noise, the fit is perfect. The top right is the standard fit with 10% outliers, and the fit is no longer perfect. The bottom right is the same imperfect data set as the top right, but using the robust algorithm; we can see that the fit is nearly perfect here, even though there are some large outliers. The bottom left is a robust fit of a data set with 50% outliers. It converged to a local minimum, but did not perfectly match the data set. This is one of the downsides of this robust algorithm: it is not guaranteed to find the global minimum. The next part of this problem gives a method for randomly choosing an initial hypothesis (and therefore initial inlier set), to allow for a better chance of finding a global minimum, or close to global minimum.

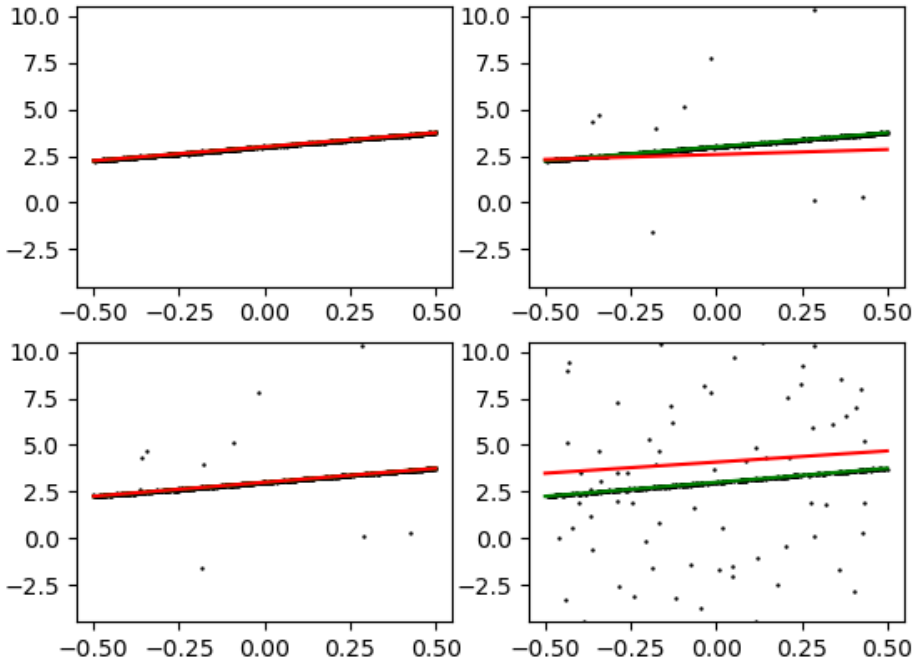

Figure 2: Robust Line Fitting

(b) In this problem we implement a RANSAC algorithm to choose an initial hypothesis for the robust line fitting algorithm described in part (a). In this algorithm, we randomly select K samples of the data set and compute a least-squares hypothesis for it, and repeat this N times, selecting the hypothesis in the set that satisfies equation 15 among the computed hypotheses. The results of this algorithm are shown for different values of K and N for a data set with 50% outliers in Figure 3. The figure shows the resulting hypotheses for each value of K, N for 7 trial to show which values consistently perform the best. In the top left, we have a large K and small N, which is very inconsistent. This makes sense because each randomly selected data set is very likely to contain outliers with large K and there aren't enough trials, N, to make it likely for a data set to not contain any outliers. The bottom left is a large K, but a much larger N as well, this is still inaccurate, but much more consistent than before. This makes sense since more trials makes it more likely for to find a sample set with no outliers, or at least fewer outliers. The top and bottom right subplots show the algorithm with small K, with small and large N, respectively. These are the best results, a small K means it is likely for any given trial to not sample any outliers, and a large N, like the bottom right means that you are likely to hit at least one such sample set. The top right sometimes returned a perfect fit, but also sometimes was inaccurate.

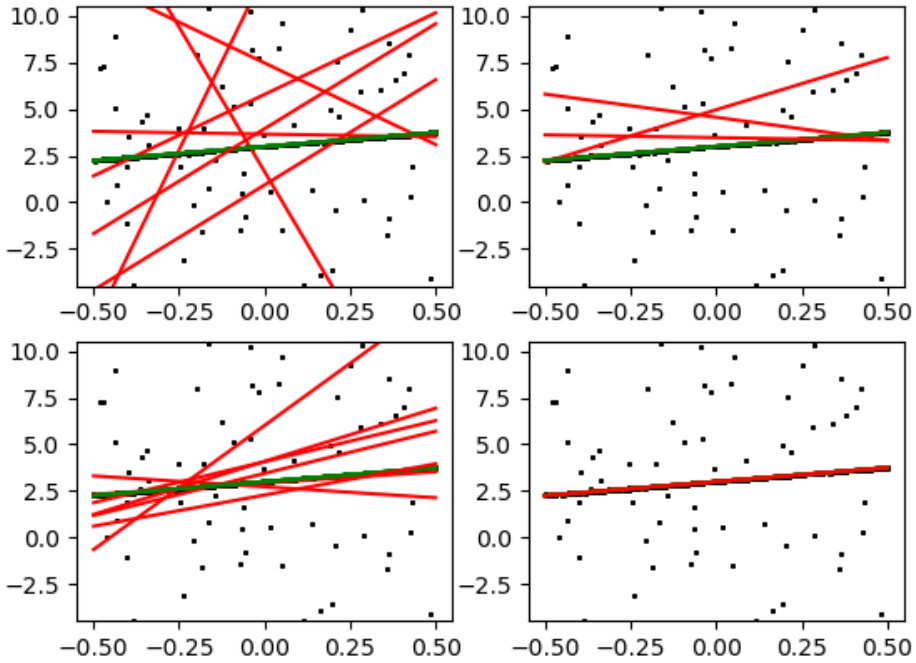

Figure 3: RANSAC Hypothesis Initialization

Solution 3

(a) If we let

$$K_i = \begin{pmatrix} f_i & 0 & W/2 \\ 0 & f_i & H/2 \\ 0 & 0 & 1 \end{pmatrix} \tag{22}$$

then we can define a point p_i from the ith camera mapping from a 3D homogeneous point $p' = [\alpha x,\ \alpha y,\ \alpha z,\ \alpha]^T$

$$p_i = [K_i \mid 0]p' \tag{23}$$

maps to the 2D homogeneous point p_i,

$$p_i = \left[\left(f_i \frac{x}{z} + W/2\right)\alpha z,\ \left(f_i \frac{y}{z} + H/2\right)\alpha z,\ \alpha z\right]^T \tag{24}$$

Then, we define a 3×3 homography, A s.t. $p_1 = Ap_2$, and we get the following relationship

$$\begin{bmatrix} \left(f_1 \frac{x}{z} + W/2\right) \\ \left(f_1 \frac{y}{z} + H/2\right) \\ 1 \end{bmatrix} = A \begin{bmatrix} \left(f_2 \frac{x}{z} + W/2\right) \\ \left(f_2 \frac{y}{z} + H/2\right) \\ 1 \end{bmatrix} \tag{25}$$

If we define the point $p_i = [p_i^1, p_i^2, p_i^3]^T$, we note that we must find elements of A that satisfy the three following equations:

$$p_1^1 = A_{11}p_2^1 + A_{12}p_2^2 + A_{13}p_2^3 \tag{26}$$

$$p_1^2 = A_{21}p_2^1 + A_{22}p_2^2 + A_{13}p_2^3 \tag{27}$$

$$p_1^3 = A_{31}p_2^1 + A_{32}p_2^2 + A_{33}p_2^3 \tag{28}$$

to get the solution

$$A = \begin{pmatrix} f_1/f_2 & 0 & W/2(1 - f_1/f_2) \\ 0 & f_1/f_2 & H/2(1 - f_1/f_2) \\ 0 & 0 & 1 \end{pmatrix} \tag{29}$$

Since A is a triangular matrix with main diagonal terms not equal to zero (assuming $f_1, f_2 \neq 0$), we know it is invertable, and therefore is a valid homography.

(b) If we define a point p_i in terms of a 3×4 projection matrix P_i,

$$p_i = P_i p' \tag{30}$$

where

$$P_i = [K_i\ \ 0] \begin{bmatrix} R_i & t \\ 0 & 1 \end{bmatrix} \tag{31}$$

finding a homography between two points is impossible unless we assume the point p' lies along a plane. We define a world-plane such that $z = 0$ for all p' to get

$$p' = [\alpha x,\ \alpha y,\ 0]^T \tag{32}$$

We can assume this without loss of generality, since we can define the world-plane however we want, so if we know that such a plane exists, we the translations and rotations of our cameras with respect to the world-plane. We can then write P_i as

$$p_i = K_i(R_i p' + t_i) \tag{33}$$

and can relate two points $p_1,\ p_2$ with a homography A. We then try to construct an A s.t.

$$p_1 = Ap_2 \tag{34}$$

$$K_1(R_1 p' + t_1) = AK_2(R_2 p' + t_2) \tag{35}$$

We begin by noting that

$$A = K_1 A^* K_2^{-1} \tag{36}$$

where A^* satisfies the simplified equation

$$(R_1 p' + t_1) = A^*(R_2 p' + t_2) \tag{37}$$

We can then satisfy two two addition parts individually and make sure they don't interfere with each other

$$R_1 p' = A^* P_2 p' \tag{38}$$

has solution

$$A^* = R_1 R_2^{-1} \tag{39}$$

for any p', but since $p' = [x',\ y',\ 0]^T$, the third column is free. Therefore we can write

$$A^* = R_1 R_2^{-1} + \begin{pmatrix} 0 & 0 & \alpha \end{pmatrix} \tag{40}$$

where α is a 3×1 vector and the 0s being 3×1 zero vectors. We know this α also has to satisfy the second part of the addition,

$$t_1 = \begin{pmatrix} 0 & 0 & \alpha \end{pmatrix} t_2 \tag{41}$$

$$t_1 = \alpha t_{2z} \tag{42}$$

with t_{2z} as the z component of t_2, so

$$\alpha = t_1 / t_{2z} \tag{43}$$

We can now construct our complete homography,

$$A = K_1 \left(R_1 R_2^{-1} + \begin{pmatrix} 0 & 0 & t_1/t_{2z} \end{pmatrix} \right) K_2^{-1} \tag{44}$$

This is guaranteed to be invertible since we know K_1 and R_1 are invertible, and we can follow the same process swaping p_1 and p_2 to find an inverse transform

$$A^{-1} p_1 = p_2 \tag{45}$$

Thus, A is a valid homography.

Solution 4

(a) In this problem I implemented an algorithm using SVD to find a homography that maps a set of points to a coordinate space of another set of coordinates. The algorithm requires at least 4 non-colinear points, but can take more. If more are included, it will return the best homography of the overdetermined system. I solved for H by defining an $8(+) \times 9$ matrix multiplied by a vectorized version of H, h.

$$Ah = 0 \tag{46}$$

This problem can be solved using SVD, and returning the singular vector associated with a singular value of 0, or the smallest singular value if there isn't a 0 in the overdetermined case. A is constructed by considering that for any given homography, the following holds:

$$p' \times Hp = 0 \tag{47}$$

for a point p being transformed into the coordinates of p'. This is true because Hp must be a scaled version of p', and therefore the crossproduct is 0. We then write the crossproduct as a matrix multiplication with matrix $C_{p'}$ and rewrite $Hp = Ph$ to get

$$\underbrace{\begin{pmatrix} 0 & -p'_z & -p'_y \\ p'_z & 0 & -p'_x \\ -p'_y & p'_x & 0 \end{pmatrix} \begin{pmatrix} p_x & p_y & p_z & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_x & p_y & p_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_x & p_y & p_z \end{pmatrix}}_{A} h = 0 \tag{48}$$

Each A_i has rank ≤ 2, which is why we need at least 4 pairs of points to solve for the homography, to get rank 8 (one paramater of h is free from scaling).

We then take the SVD of the stacked A matrices, and use the least singular vector reshaped to get H.

(b) In this problem we use the algorithm from part (a) to splice an image into a location in destination image. Since a homography can map for any set of points once defined, we only need to define the corners of the quadrilateral we want to splice into, and the size of the source image as the points for our homography algorithm. This gives us a homography H which we can use to transform points from the destination image into the source image. We define a meshgrid of points that includes the set of points we want to splice, and transform them into the source's coordinates. any points that is withing the bounds of the source image after the transform is deemed 'valid', ie we want to include it. After this we just replace all valid points from the meshgrid with the point they landed on in the source image. The result is shown in figure 4

Figure 4: Homography Splicing of Two Images

Solution 5

(a) In this problem we design a census transform, a representation of a local neighborhood of a pixel. The census transform encodes the number of pixels that are of lower intensity than the center pixel, with indices outside the boundary being considered 0 intensity.

Since we are making a 5×5 census transform, ie ± 2 pixels from the center, I padded the image with 2 zeros on each edge to avoid indexing problems. Then, I just loop through each pixel in the image, flattening the neighboring 24 pixels, and comparing them to the center. I used broadcasting of an array of 2^i multiplied by the boolean array of which pixels are less intensity to the center to get the census transform of the center pixel.

(b) In this problem we use the census transform coded in the previous to calculate an elementwise distance (hamming distance) between points on a horizontal line. By minimizing the hamming distance of between points in a left and right image we can get an estimate of depth in a pair of images.

To do this we loop through every pixel, and test the hamming distance between $C_{left}[i, j]$ and $C_{right}[i, j - k]$ where $k = \{0, \ldots, D_{max}\}$ and D_{max} is the maximum horizontal distance we check. We then return the disparity of each pixel, this is shown in Figure 5.

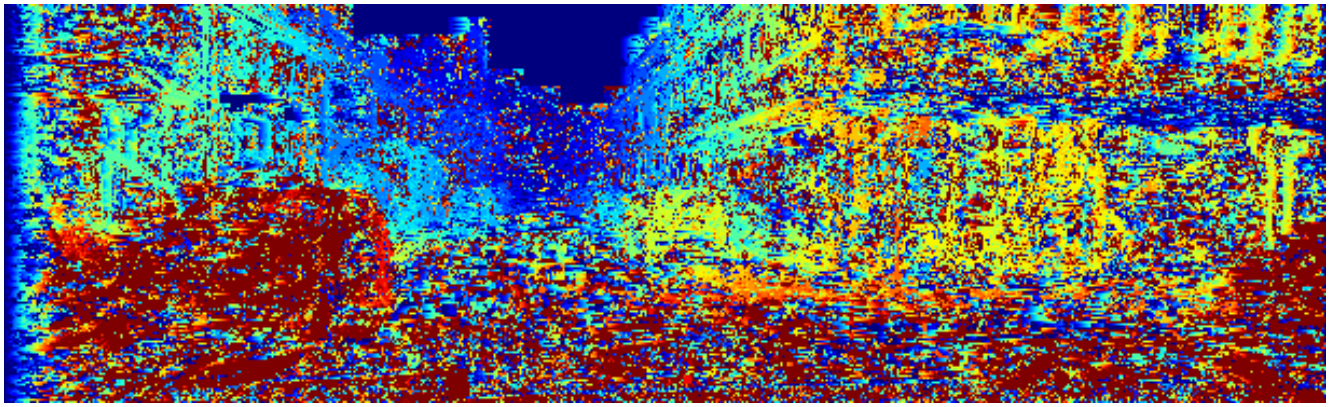

Figure 5: Disparity Map of a Left and Right Image

Information

This problem set took approximately 20 hours of effort.

I discussed this problem set with no one.

I also got hints from the following sources:

- Wikipedia article on binomial distribution at https://en.wikipedia.org/wiki/Binomial_distribution