EECE 5639 Project 2

Neel Bhalla, Eric Grimaldi

November 4 2021

1 Abstract

In this project, we explore a technique for mosaicing sequences of images captured with significant overlap between consecutive frames. Due to this overlap, common features can be correlated between the two frames, providing insight into the transformation between image planes of the two frames. We investigate the Harris Corner Feature detector algorithm for harvesting corners, normalized cross correlation for matching features across images, Random Sample Consensus (RANSAC) and least squares fitting for estimating homographies between images. Finally we explore blending techniques to resolve color differences between image frames and distortions in the final mosaic.

2 Approach

The approach to image mosaicing can be broken down in figure 1. First, two images are pre-processed to filter any small noise (using a Gaussian filter), and then corners are extracted using a Harris Corner Detector (HCD). Due to the large number of outputs from HCD, we use non-max suppression to remove any features which do not exist at a local maximum. Many of these non-max features are due to an "off-center" HCD window observing a feature that is better captured by some other window. Using the sparse set of corners from non-max suppression, corners are correlated across images by finding the maximum normalized cross correlation value between the neighborhood of corners. While this is not optimal, the error rate is low compared to the number of features provided.

Given a set of point-to-point correspondences, we estimate a homography using the least squares solution. Due to correspondence outliers skewing the solution significantly, this is done after running RANSAC and keeping inliers for some non-strict threshold.

Upon obtaining the homography, it suffices to warp the image into the base image, and merge the two images together. The size of the output mosaic is determined by transforming the corners of the input images, and choosing the mosaic size to encompass all of those corners. Bi-linear interpolation is used to estimate color values for pixels in the final mosaic which do not strictly exist in the input images due to corresponding fractional indeces. In order to account for color differences between overlapping pixels from multiple input images, feathering is applied as a blending method between the images.

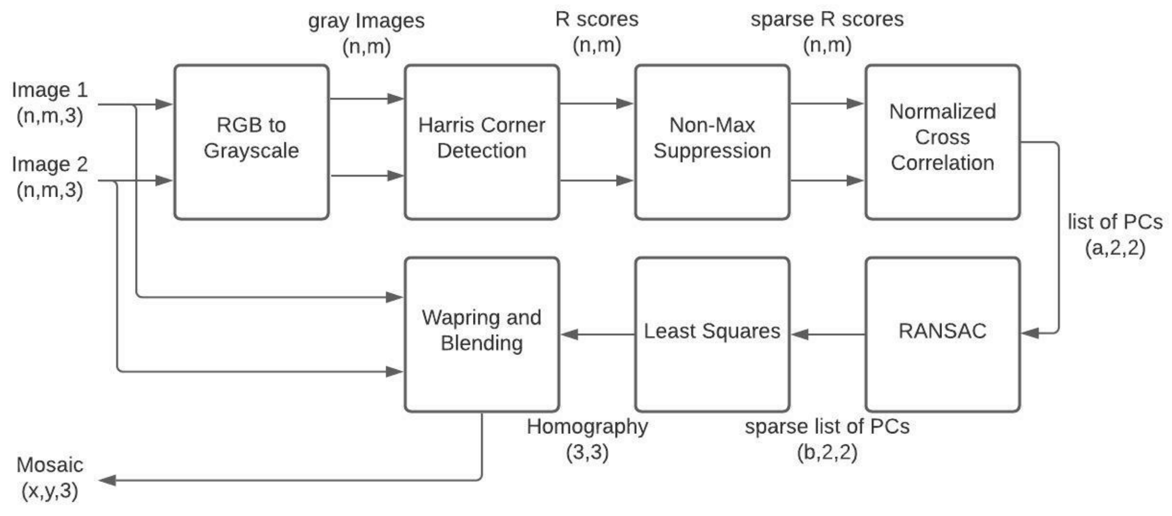


Figure 1: Block diagram of merging some `Image 1` and `Image 2` into a single `Mosaic` image.

Theory of transformation

The mosaic technique is predicated on the property that there exists an affine transformation between image frames if and only if the objects being captured lie on a common plane. As a result, there exists a transformation from such world system to both image planes. By inverting one of the transformations and composing, we get a 2D $\rightarrow$ 2D transformation between image planes.

Harris Corner Detector

The Harris Corner Detector (HCD) is an algorithm that leverages the fact that corners are the intersection of two edges, and thus have gradients in the components of the edges. HCD tries to compute the sum of squared differences of neighborhoods between an original neighbor hood and one with an offset.
The derivations reveal that

$$F(\Delta x, \Delta y) \simeq \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x & \Delta y \end{bmatrix}$$

which is directly dependent on the middle matrix, known as the C matrix. In order to understand the strength / behavior of the matrix, it suffices to find its eigenvalues. If both eigenvalues are zero, then it is a flat area; if one is zero, then it is an edge; if both are non-zero, then it means that there is a corner of some sort. Furthermore, the actual values of λ_1, λ_2 must meet some minimum threshold (minimum strength of corner).

A common numerical approximation for this is to compute R scores according to

$$R = \det C - k\ (\mathrm{trace}\ C)^2$$

Since C is 2x2, det$C$ and trace$C$ are much easier to compute than eigenvalues of C. As such, we use this R score approximation for our implementation of HCD.

Non-Max Suppression

Non-max suppression is an algorithm used to find local maxima in discrete data. In the case of HCD, corners may overlap with multiple neighborhoods, yielding neighborhoods of high R-values. In order to group them all as one point, the maximum is selected via non-max suppression to represent the corner feature. The algorithm is as follows: For all pixels, if the pixel is the maximum value within in a specified window (3x3 in our implementation), then keep its value, else let it be zero. All pixels who have a larger neighbor will be forced to zero, hence only all local maxima will remain.

Normalized Cross Correlation

Normalized cross correlation is an algorithm for template matching neighborhood "templates" within another image. The goal of the algorithm is to find the All-Pairs-Similarity coefficients to figure out where the most likely template match within the target image is. The initial metric desired is a normalized sum of squared differences which is

$$\sum (f-g)^2 = \sum f^2 + g^2 - 2fg$$

Since f^2 and g^2 are equal to one (when normalized), it follows that it suffices to compute fg to determine the value of the normalized sum of squared differences. This simplified heuristic is the normalized cross correlation. The heuristic is calculated by convolving normalized f onto normalized g for all pairs.

RANSAC

RANdom SAmple Consensus (RANSAC) is an non-deterministic iterative algorithm for estimating a model on data which is prone to a lot of outliers. Instead of trying to optimize the least squares solution (mentioned below), RANSAC aims to find the best model that matches the maximum number of points within a tolerance.

For a given model, let x be the number of points required to determine all the parameters of the model (not unconstrained). At every trial, RANSAC will randomly pick x points of its data-set without replacement and construct a model. It will then retrofit the remainder points in the model and count the number that are within a tolerance of the model prediction. Possible termination criteria include: "If the number of points within tolerance are below some threshold, it will repeat this process, else it will return the model", or the best result from a maximum number of trials.

While RANSAC does not produce the best fit model, RANSAC is extremely effective at removing outliers from data. By filtering the points based on whether they satisfy the best-fit model, we can compute a least squares solution to the remainder, which is guaranteed to give a good answer as there are no significant outliers from data.

Least Squares

Given a model f which transforms $x \rightarrow \hat{y}$, The least squares solution to a problem aims to minimize the projected distance to the line for all points

For a an over-constrained problem $Ax = b$, where we need to compute $\hat{x}$, the least sum of squares problem can be seen as finding the projection of $\hat{y} = A\hat{x}$ such that it minimizes the distance to b. This can be solved by left multiplying by A^T, and then left multiplying by $(A^TA)^{-1}$ yielding

$$x = (A^TA)^{-1}(A^Tb)$$

Warping and Blending

Once the homography is obtained, it suffices to warp the first image into the domain of the second image. In order to find the bounding box of the new image, we pre-warp the corners of the images and select the size of the output mosaic to be a minimum bounding box around all 8 corners.

Next, for each pixel in the newly defined output domain, we use the inverse homography to calculate the input images' affects on the output. Note that for one of the input images, this homography is the identity, since we remain in the same perspective. In the case that neither input image has any effect on the particular output pixel under consideration, the pixel remains 0 for all channels (this corresponds to black for grayscale or RGB images). For the case that both input images affect the output pixel, we blend the two using feathering. Feathering is a weighted average computed as follows

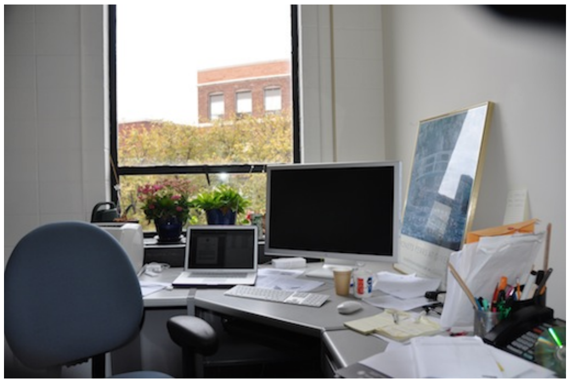
$$p = \frac{w_1 p_1 + w_2 p_2}{w_1 + w_2}$$

where w_1 and w_2 are the distances of pixels p_1 and p_2 to the nearest boarder of their respective image. This gives a smooth transition from one input image to the next when traversing the overlapping areas.

Another issue is dealing with partial coordinates. Applying the inverse homography on a integer indexed point yields a floating point index; simply casting these indeces as floats will result in some inn accuracy. To solve this we implement bi-linear interpolation, which is a weighted sum of the 4 pixels overlapped by the hypothetical source pixel with floating point indeces. Weights are computed according to the proportion of the pixel overlapped by the hypothetical source pixel. This produces a "weighted blend" of the pixels that is more representative of what the true homography point is.

3 Experiments

In this project, there were only 2 small variations 1 experiment conducted. Namely we constructed an image mosaic from 2 photo sets; one of an office and one of a hallway. Two pictures were chosen from each data set to mosaic. Results can be seen in figure 4.


(a) left office frame

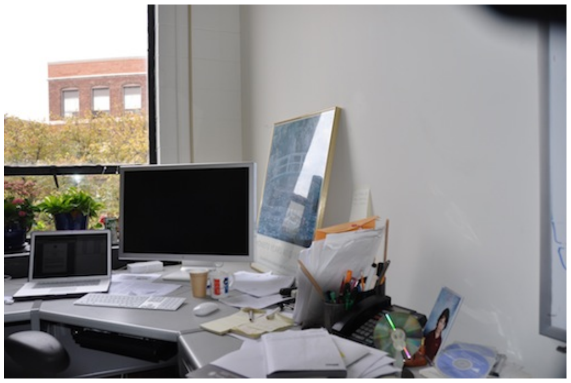
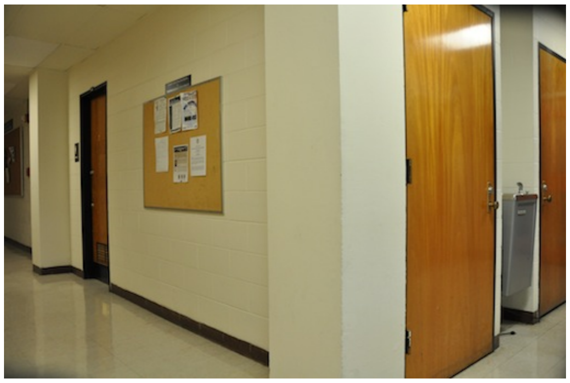

(b) right office frame

Figure 2: Office frames to mosaic.


(a) left hallway frame

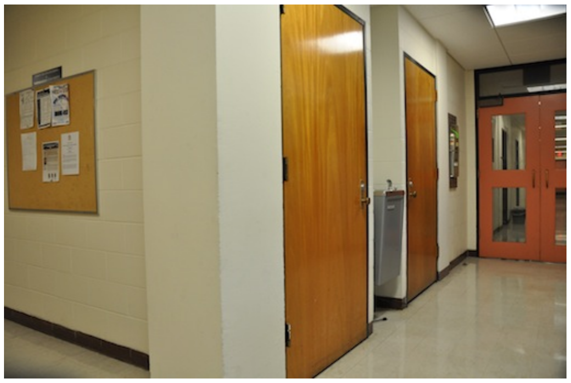

(b) right hallway frame

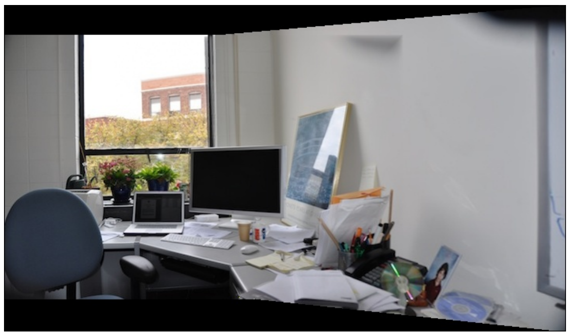
Figure 3: Hallway frames to mosaic.

4 Discussion

In this pipeline there were many different parameters to be tuned. Tuning these parameters not only had impact on runtime, but also on quality of results and final mosaics.

Harris Corner Window and Threshold

The first parameter we encountered was the size of the window used to compute the Harris Corner matrix C. While increasing the window will increase the quality of corners, making


(a) Office frames mosaiced

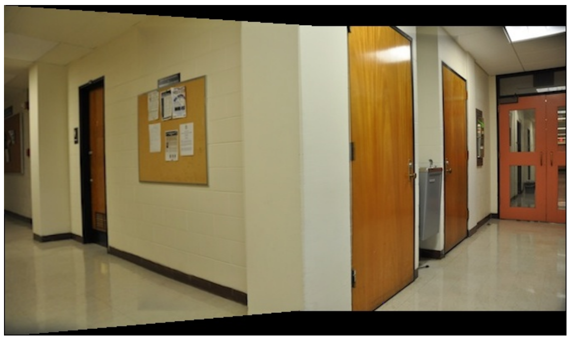
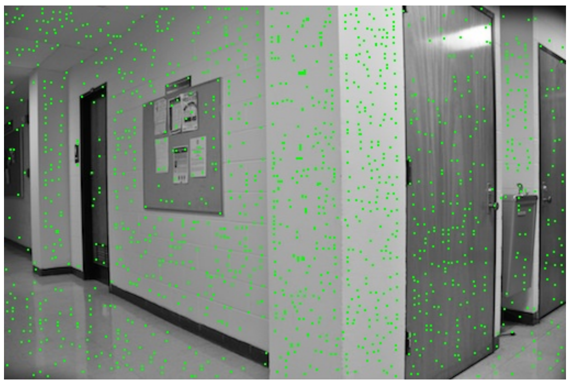

(b) Hallway frames mosaiced

Figure 4: Mosaic Results

it too big will result in small features being ignored. Furthermore the accuracy of the actual corner location decreases with window size (non-max suppression is less accurate as the same corner is likely to appear fully in several window locations). After our testing, we found that a 9x9 window was reasonable for finding corners in out images.

Finding the correct minimum R value is also a parameter to tune. Should the R value be too small, there would be too many detections, possibly resulting in a large number of outliers increasing the run time of RANSAC or worsening out homography. In figure 5, it is clear that choosing $R \geq 10^3$ yields very noisy features, virtually the majority of surfaces, corners or not, have a R value at least 1000. Since the feature set is so dense, it is unlikely that RANSAC will find the correct homography. Increasing the R value to 10^5 yielded similar, but cleaner results.


(a) hallway post-nms with $R \geq 10^3$

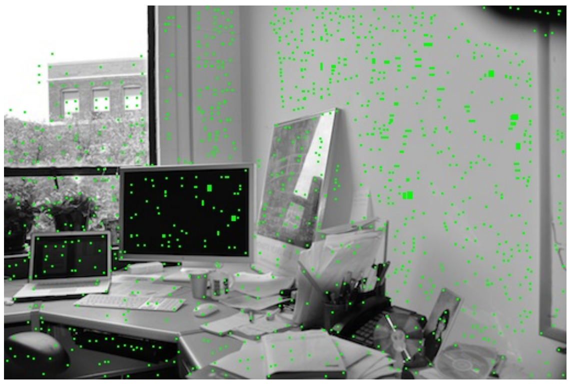
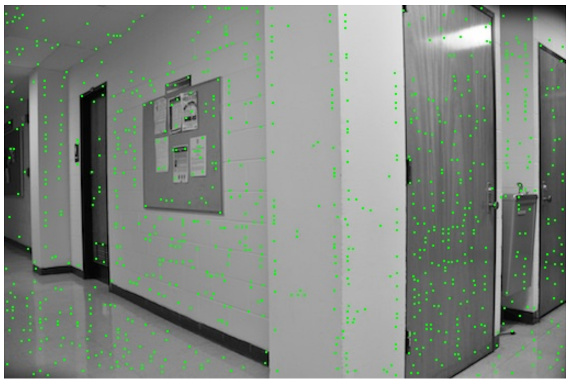

(b) office post-nms with $R \geq 10^3$

Figure 5: Harris with R threshold of 10^3


(a) Harris with R threshold of $R \geq 10^5$

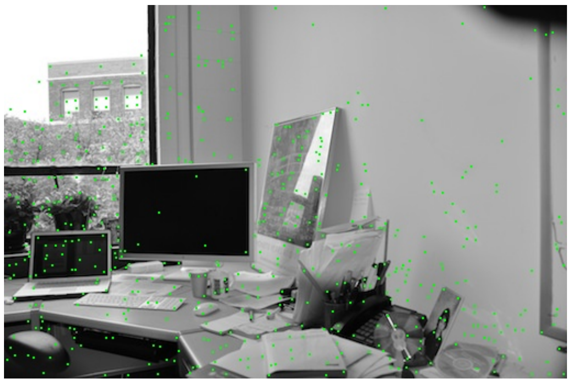
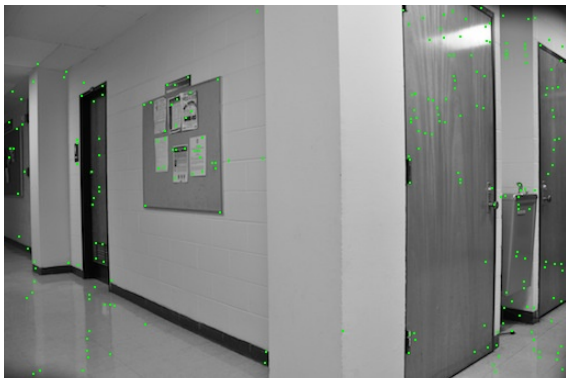

(b) Harris with R threshold of $R \geq 10^5$

Figure 6: Hallway frames to mosaic.


(a) hallway post-nms with $R \geq 10^7$

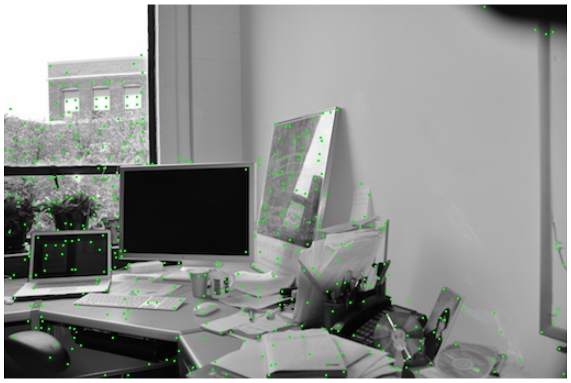
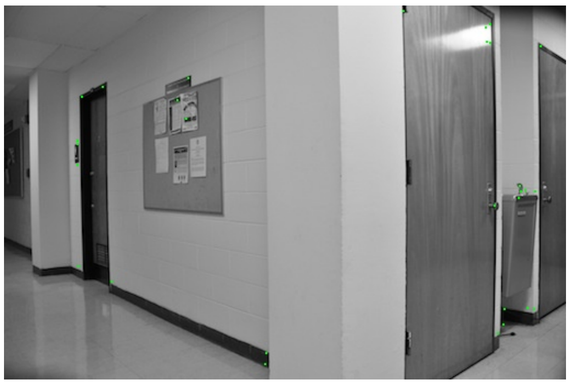

(b) office post-nms with $R \geq 10^7$

Figure 7: Harris with R threshold of $R \geq 10^7$


(a) hallway post-nms with $R \geq 10^{11}$

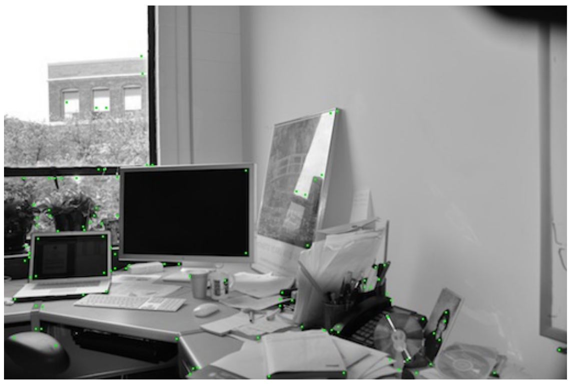

(b) office post-nms with $R \geq 10^{11}$

Figure 8: Harris with R threshold of $R \geq 10^{11}$

By Increasing the Threshold to 10^7, the number of features dropped to a sparse enough set to work with. Increasing the threshold too high, such as $R \geq 10^{11}$ produces too little features to work with. While they might be individually very good corners, there is not enough redundancy across overlapping images to get useful output from normalized cross correlation.

We ultimately decided on a threshold of $R \geq 2 * 10^7$.

Non-Max Suppression Window

Another parameter to consider is the Non-max suppression window. This window dictates the size of the neighborhood that is argmaxed over to check for local maxima. As a result, this also dictates the minimum distance between adjacent features (lower bound). By choosing a lower window, we may get multiple corners adjacent, but choosing a bigger window may exclude good features from our image. By choosing a large enough R floor, the "too many features" issue can be compensated for, thus we opted to choose a 5x5 kernel and a large R value.

Normalized Cross Correlation Window

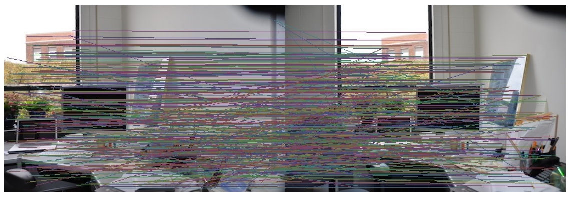
The normalized cross correlation method also requires a window parameter. In essence, larger windows effectively require larger areas of each image to "match" in order be considered a point correspondence. If the window is too large, the transformation of the image due to the movement of the camera may result in true matches failing to result in a correspondence. If the window is too small, detail surrounding similar but distinct features may be excluded from consideration, resulting in a false correspondence that RANSAC would need to filter out.

RANSAC Tolerance Radius and Maximum Trials

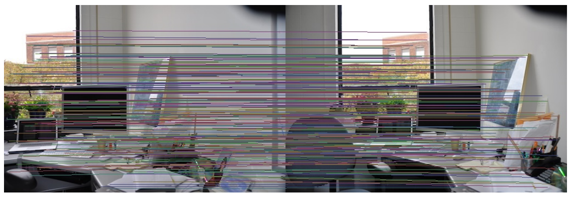
The RANSAC procedure features a tolerance radius and a max trials parameter for estimating the efficacy of proposed homographies. Tuning the parameter too low will pessimistically score homographies, requiring more trials to find a good match. If the parameter is over-tuned, it is likely that the 4 points originally used to estimate the homography will be the only ones that match that actual output, yielding a nonsense output from the algorithm. Similarly, if the radius is too loose, nonsense homographies that include outliers and other mismatches will still be accepted, defeating the purpose of RANSAC all-together.

Our approach for this was a 2 step process: Run RANSAC twice on decrementing radii. By first Running RANSAC with a tolerance of 20 pixels, mismatches that were close, but still in the correct direction of homography would be included, in order to get rid of the true outlier correspondences (from a possible mismatch during the normalized cross correlation). Next RANSAC is run on the inliers of the prior run but now with a tight bound (ex: $r = 4$), accepting some noise, but requiring that the homography accurately capture the transformation for as many correspondences as possible up to a noise factor.

The maximum trials parameter is directly proportional from the theorized proportion of outliers in the data. When there is a large proportion of outliers, the probability of drawing

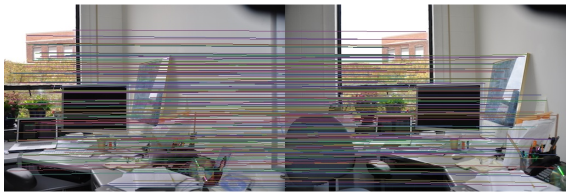


(a) Unfiltered point correspondences from normalized cross correlation

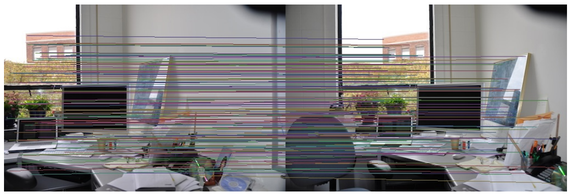


(b) Filtered point correspondences on best solution from RANSAC with $R = 20$

Figure 9: RANSAC Procedure part 1



(a) Filtered point correspondences on best solution from RANSAC with $R = 4$



(b) Least Squares Solution (correspondences satisfied)

Figure 10: RANSAC Procedure part 2

4 "inlier" points is very small, thus we need a very large number of trials to somewhat guarantee a legitimate solution. Similarly, when there are minimal outliers, very few trials are needed to guarantee that we have draw 4 inlier points (to some confidence value).

5 Bonus: Warping Images into frames

The task of warping images into frames is a subset of the pipelines described above in this report. Since the user is responsible for supplying the correspondences, the homography is fully defined and it suffices to place the image into the frame using our warping techniques described above. In figure 12 we offer a picture of a Pikachu warped into the office.

6 Appendix: Code

Please see our git repository for copies of our code.

$$https://github.com/EAGrimaldi/EECE5639\_Project2$$

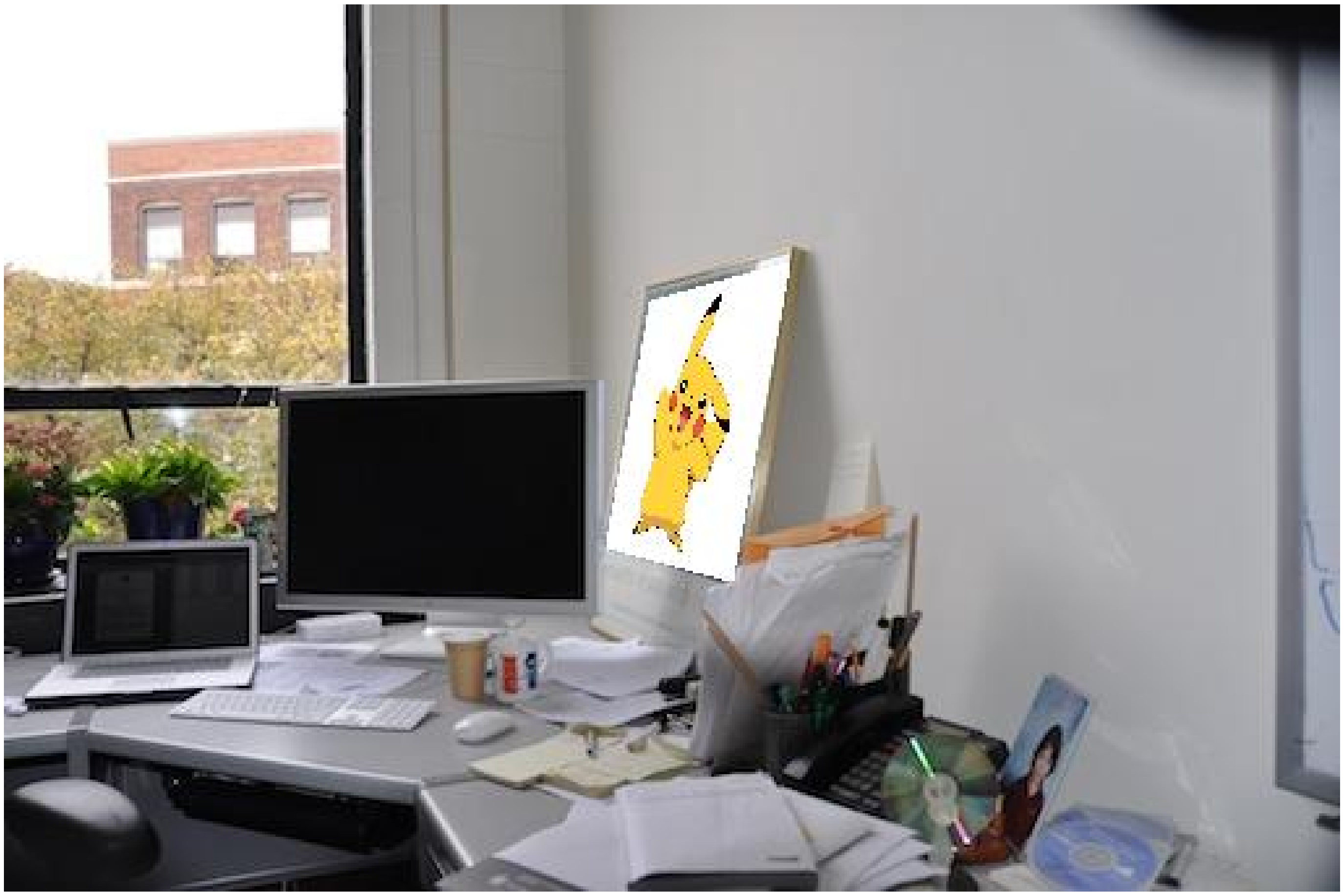


Figure 11: Pikachu warped into the frame of a picture

Figure 12: What a gamer!