CSC 420: Final Project report

Alexander Kondratskiy g9sash 996366285
Rohan Chandra g9chandr 996274142
Jiwoong Im g0d 996835732

Component 1:

Our algorithm functions in three major phases that enable us to assign and maintain flower labels across a pair of successive frames in a video sequence. To discover the location of potential flowers, we determine regions of yellow and brown and use the intersection of these regions to create potential locations of flowers. We fit circles to these intersected regions and use these circles as the locations of the flowers in a given frame. To find the correspondence of flowers detected in this manner between two frames, epipolar geometry is used to narrow the region in which a flower may have moved from the first frame to the later frame. Within this region various heuristics are then used to create a correspondence of flowers between frames.

- **Step 1 - determining potential flower locations:**

To find potential areas where a flower may reside we first detect regions of yellow. Our method uses the starter code provided to generate a set of points where the yellow slab detector was triggered. A connected components method is then run on these points to group sets of nearby points into a set of components.

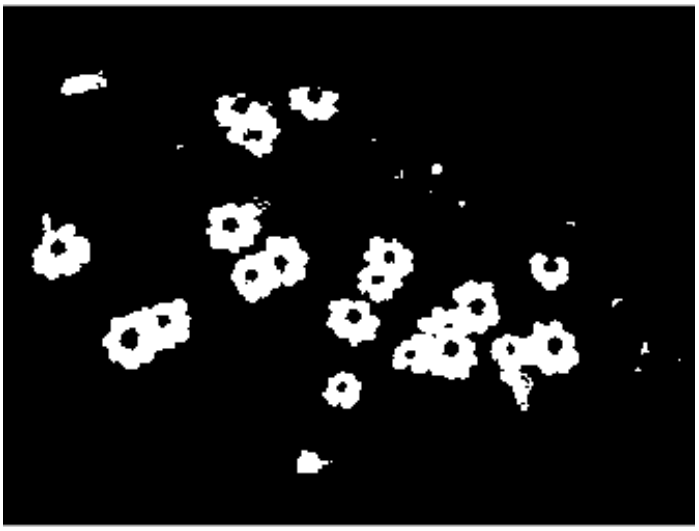

figure 1: the connected components generated by the points found by the yellow slab detector

However, simply using the yellow slab detectors response is not enough to be a sufficient indication of the presence of a flower, as many other objects, such as leaves, may generate a reasonably sized component.

It was noted that many of the erroneously detected components lacked a dark brown center, as would be expected of a typical flower. Thus, a secondary set of components corresponding to these brown centers were generated using the provided starter code for matching areas of similar mean and covariance across the image.

figure 2: the connected components generated by the points found by brown covariance detector

These two sets of connected components could then be combined in order to find the regions that were most likely to be a flower. In both sets, the connected components were expanded by using a form of gaussian blurring and then clipping the values to 0 or 1 below and above a given threshold. Thus edgels found on the border of regions of yellow and brown should have significant overlap when the intersection of these expanded masks of brown and yellow regions was computed. This gave us edgels that were mainly around the flower centers to which we could then fit circles.

figure 3: The final set of edgels produced via this method from the frame seen on the right

- **Step 2 - Refining determined components into labelled flowers:**

Given the final set of edgels produced in the previous section, it is possible that a group of edgels in the same connected component may belong to a different flower center than another group of edgels, or that some of the edgels in the cluster may have been erroneously produced. To resolve the initial issue of clustered flowers and misleading edgels, we fit circles to the flower centres in the following way. Components that have too few supporting edgels are removed and, for each remaining component, circles are fit to the component's supporting edgels.This circle fitting is accomplished via the same method as was used in assignment two, with some tweaking to the algorithm to fit the situation. In particular, any proposed circles that do not contain enough brown pixels within, and yellow pixels on the outside are rejected. This combats erroneous circles that fit to the petals, instead of the center.

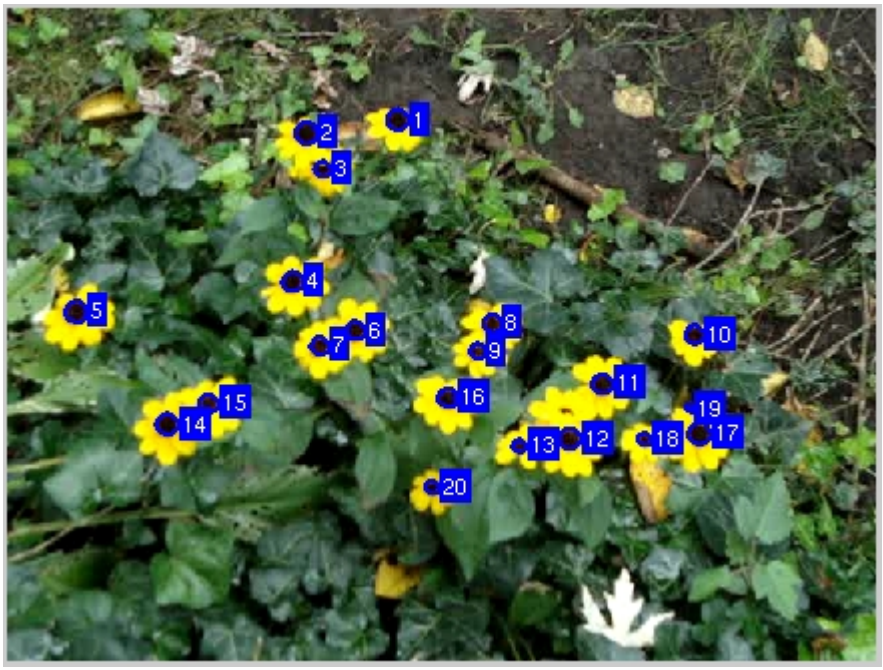

figure 4: The final set of detected flowers for the given frame.

- **Step 3 - Generating flower correspondence between two frames.**

Given two separate frames in which flowers have been detected in this manner, we can then begin to generate a correspondence of the identified flowers. We use a two stage method where we first iteratively try to find correspondences between flowers using epipolar geometry, and then in the second stage we try to find correspondences in past frames for any unmatched flowers using homographies.

1 - Iterative initial correspondence.

Details of the implementation of our epipolar geometry method, specifically the determination of the F matrix, can be found in the report of component 3. Once the F matrix has been discovered using SIFT points, we use it to find the corresponding epipolar line in the second frame of each flower in the first frame.

figure 5: frame 100 shown on the left, with the corresponding epipolar lines of each flower center overlayed onto frame 150 shown on the right.

New flower centers are then detected in the new frame (frame 150 in the above case) as described previously, and the initial correspondence is determined by the following heuristics. We compare the perpendicular distance of every flower discovered in the new frame to the epipolar line of a flower from the previous frame. We then compare the straight line euclidean distance of every new flower to the flower from the previous frame, under the assumption that, within only 20-50 frames, a single flower cannot move very far from its starting position. This is to help with cases where two or more flowers lie on the same epipolar line, and would thus have a very similar perpendicular distance. Initial correspondence is determined as the min of the sum of these two distances with more importance given to distance to the epipolar line (by scaling up the value). Giving more importance to epipolar distance is very beneficial, as can be seen in the following figure:

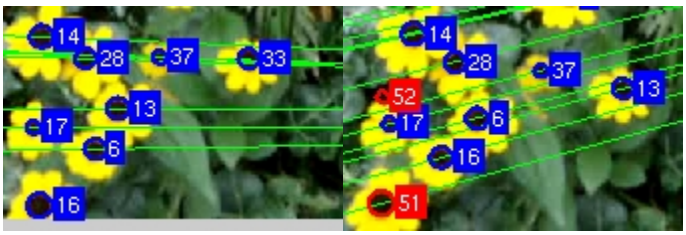

*figure 6: Two successive frames (541 and 561) where a very bad mislabeling occurs, when running the matching algorthim **without giving more weight to epipolar distance**. Due to the specific camera movement, wrong flower pairs line up at close pixel positions between the two frames. Since epipolar distance is not given more weight, euclidean distance takes precedence and 33 is mislabeled as 13, 13 becomes 6, and 6 becomes 16.*

However, at the end of this initial matching, it is possible that newly discovered flowers have no sufficient match from the previous frame. We use the matched pairs to calculate an approximate homography (assuming most matches are correct), between the two frames to better inform the next iteration of the process. We then repeat the whole stage from scratch, but with euclidean distance now measured between flowers in the first frame, and flowers from the second frame that have been warped by this homography. Euclidean distances for correct matches should shrink given the homography, and more flowers are matched as a result. The benefit of this method can be seen in the following two figures:

*figure 7: two successive frames with labeled flowers, **without using homographies** and iterative matching. Flowers labeled blue in the frame on the right denote flowers that got matched to the ones from the previous frame, while those labeled red were unmatched. As you can see, the flowers labeled 20 and 24 in frame 61, were mislableled as 19 and 11 in frame 81 respectively. This is because all 4 flowers lay on the same line, and taking the euclidian distance between the two frames without a homography resulted in a mismatch. Due to the wrong labeling of these two flowers, new labels were introduced for the flowers that should have been matched.*

*figure 8: two successive frames with labeled flowers, **using homographies** and iterative matching. As can be seen, all flowers are matched correctly after the last iteration of matching. The first iteration of matching would essentially be used to inform the homography, and any successive ones will refine it and produce a better matching.*

Still, at the end of the iterative procedure, unmatched flowers may still be left over. It is possible that these flowers are new entities that were never tracked before, or that they are flowers that were tracked at some point in the past, but, at some point in the sequence, became occluded.

To detect the latter case, we try to look at the previous frames to find possible matches.

2 - resolving unmatched flowers

As we find matches between successive frames, a homography is also found and stored for that pair of frames (using the matched flower centers as points). We are then able to use this homography to look at a neighbourhood around the unmatched point in previous frames to see if there is an older flower with no corresponding flower in the current frame. We continue to repeat this process of matching as the sequence progresses, and are thus able to maintain labels of flowers throughout the sequence even through occlusion or misdetection.

For a given proposed flower, we find the closest match out of the previous 4 frames, and do not accept the match if the distance is too large (the proposed flower has never been seen). The distances are also cumulatively amplified the further back we look, so that matches from more recent frames are favoured over those from older frames. In the following figures, a few notable successes of our algorithm are detailed.

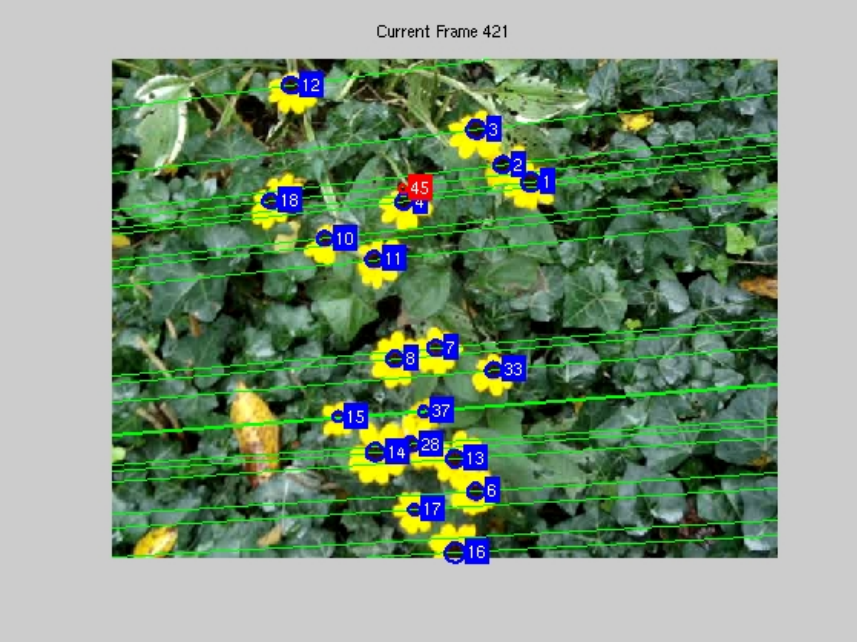

figure 9: in this frame, note the position of flower 12.

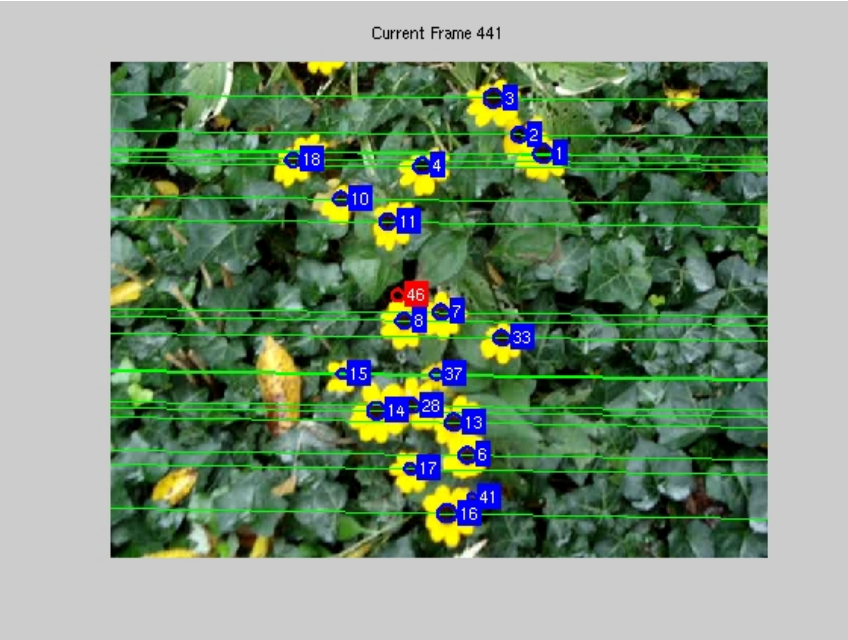

figure 10: Due to movement of the camera, the flower labelled 12 in the previous frame is no longer visible

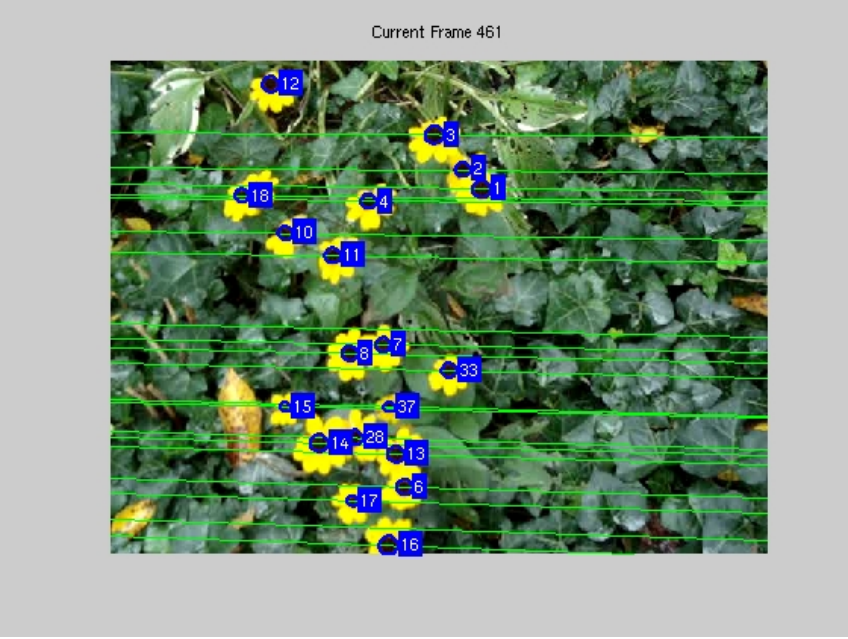

figure 11: flower 12 then reappears in frame 461. As can be seen, the correct flower label is recovered by looking into past frames.

Here is the first and last frame of tracking through the whole sequence, each image 20 frames apart:

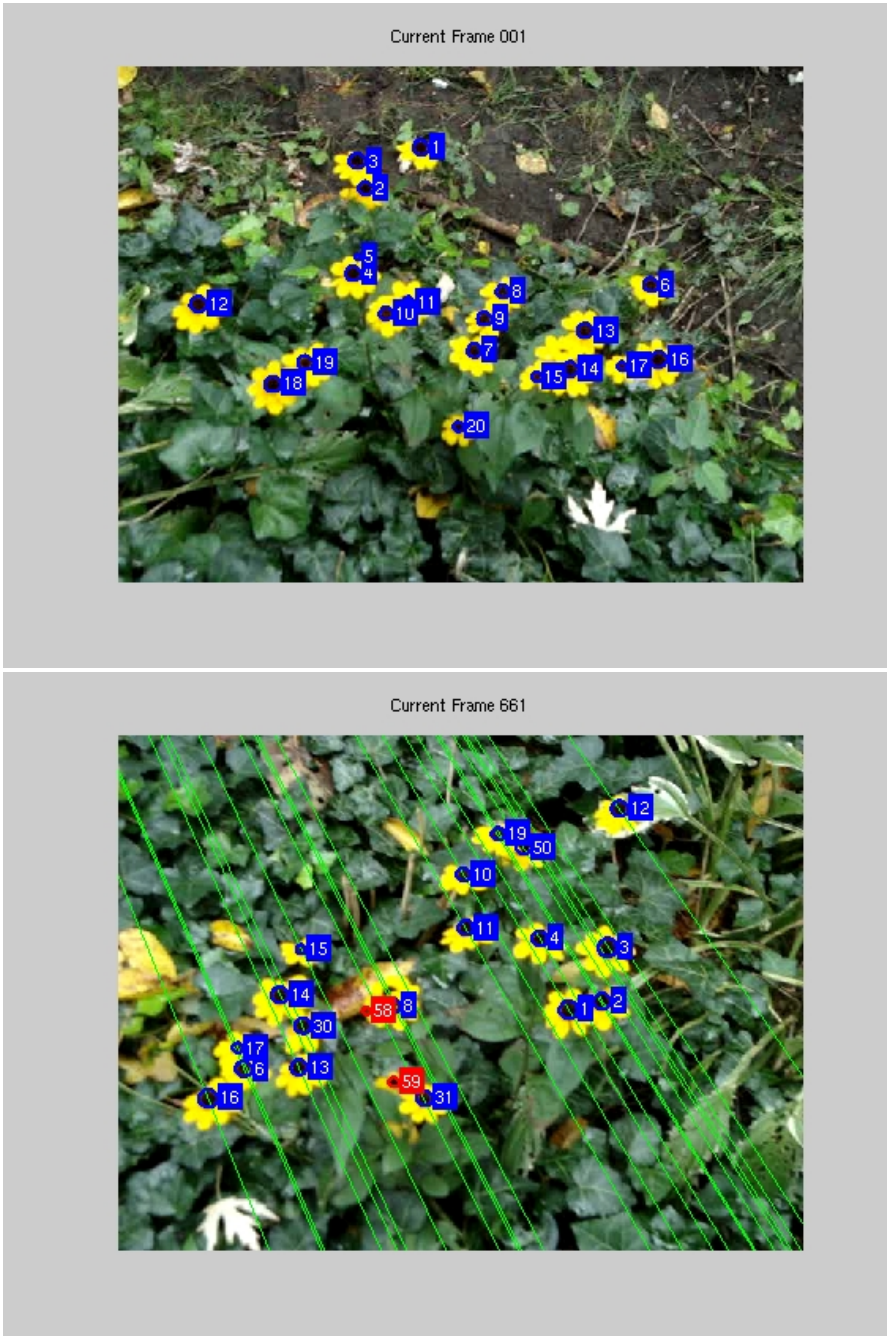

figure 12: An example of tracking through 660 frames.

Most of the flowers seen in frame one have been tracked through to the last. Notably, flower 6 has moved quite a bit relative to other, but has been labelled correctly. This is due to the fact that the flower was not occluded during any part of the sequence. Flower 31 was actually 9 at the start of the sequence, but was occluded near the start, and moved a great distance relative to its original position.

Limitations:

- When initially determining where the flowers reside in an image, our algorithm may generate a false positive if a patch of light brown ground lies directly underneath the yellow petals of the flower. This false positive can be seen in figure 4, point 19.

- Our implementation is very specific to a range of yellow and brown as being the key components of a flower. One would need to recalibrate these metrics if a new set of sequence with drastically different appearing flowers.

- If the first stage mislabels flowers, the second stage cannot correct such errors, as seen in the figures below

figure 13: Three frames are shown around the location of the same flowers: frames 141, 151, and 261 respectively. Flower 9 gets occluded by 7 and then emerges on the other side. Our algorithm mislabels this flower as a new one, since its movement does not follow the approximate homography of all other flowers. When it reappears in frame 261 it is treated as a new flower.

Improvements:

- In some frames, the brown flower center is not detected as those centers are brighter than the rest. A better mean and covariance of brown could be used to improve results.

- In some cases weighing epipolar distances too much, results in mismatches when a lot of flowers lie along a single line. Conversely putting too little emphasis results in errors shown in figure 6. Better or even dynamic calibration of this parameter would give better results.

- We do not utilize any of the information that could be gained from the frames that occur between the frames we analyze at 20 or 50 frame intervals. We might be able to improve our results by using the motion tracking code developed in part 2, specifically by attaching a patch to each flower and tracking its movement between each frame we analyze. This additional information gained by the motion tracking would help us further narrow down the region along the epipolar line where the corresponding flower is found.

- For better tracking of occluded flowers, a motion vector could be stored per flower, in relation to the approximate plane of all the matched flowers (found by the homography). The positions of currently invisible flowers could then be extrapolated, and matched to any “new” flowers.

Component 2:

We begin by selecting the patches we will track by randomly assigning patch centers across the image, with some equal distribution in each portion of the image. The number of patches is then reduced by only retaining those patches who have a high cumulative Harris operator value across their patch comparative to other patches in their section of the image. These patches are then tracked using similarity transformations, assuming brightness constancy. Finally, the tracked patches are validated, to detect patches that have drifted from what they were originally tracking, by comparing several hogs within the patch in the original frame compared to several hogs in the same location (relative to the patches movement) in the new patch.

- **Step 1 - Patch generation:**

In order to avoid having a build up of proposed patches in a small region of the image, we divide the image into subsections and randomly assign an equal number of patch centers to each of these subsections. We then calculate the average value of the R value of each pixel in the given patch, where R = det(H) - (Trace(H)).^2**.** This value is used to determine which patches are most suitable for tracking, as a high average R value tends to be indicative of a patch that contains a set of gradients that are not near parallel and can be used to determine changes in scale, rotation and translation between frames. Subsequently, of those patches that were not culled due to a comparatively low R value, patches that significantly overlap with another patch are removed. The effects of the culling can be seen in the figure below.

figure 14: 200 patches were randomly placed in this image, those patches seen in red would be retained by our algorithm after low R value culling, those in blue would be removed. Significantly intersecting patches would be removed afterwards.

- **Step 2 - determining the similarity transformation:**

To track the individual patches, from one frame to the next, we use a function that is a modified version of the motionTutorial, that does similarity transforms. Given two successive frames (*im0* and *im1*) and a patch to track in the first image, we perform the following actions:

- Since the patch may be rotated and scaled, we crop it to its original size and orientation, by using a similarity transformation.

- The patch is blurred, and a gradient image is found.

- Given a predicted translation from the previous tracking, we crop the patch from the predicted location in *im1* to the same size and orientation, and blur it. To clarify, we do not also use predicted scale and rotation, so we reuse these parameters from the first patch. In effect, we always crop using a patch with the same size and rotation, just from two different locations in *im0* and *im1*.

- Having both cropped patches (which should be the same size after cropping), we can solve for a least squares approximation of the similarity transform between them. (Details to follow)

- Obtaining this approximation, we perform several iterations, each time readjusting the position where the second patch is cropped (but not scale and rotation).

- Finally, we apply this newly found transformation to the polygon defining the patch in the previous frame, and store this new patch for the new frame.

Given the transformation relative to an origin:

$$\vec{\boldsymbol{u}}(\vec{x}) = \alpha \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \vec{x} + \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$$

Constraints are:

$$0 = (\vec{\boldsymbol{u}}(\vec{x}) - \vec{x})^T \cdot \vec{\nabla} f(\vec{x}, t) + f_t(\vec{x}, t)$$

$$0 = \vec{\nabla} f(\vec{x}, t)^T \cdot (\boldsymbol{A}(\vec{x})\vec{\boldsymbol{u}} - \vec{x}) + f_t(\vec{x}, t)$$

Least Squares error function:

$$E = \sum_{\vec{x}} \left(\vec{\nabla} f(\vec{x}, t)^T \cdot (\boldsymbol{A}(\vec{x})\vec{\boldsymbol{u}} - \vec{x}) + f_t(\vec{x}, t) \right)^2$$

Normal Equations become:

$$\frac{\partial E}{\partial \vec{a}} = 0$$

$$\frac{\partial E}{\partial \vec{a}} = \sum_{\vec{x}} g(\vec{x})(\vec{\nabla} f^T \boldsymbol{A})^T \cdot \left(\vec{\nabla} f^T \cdot (\boldsymbol{A}\vec{\boldsymbol{a}} - \vec{x}) + f_t\right)$$

$$\frac{\partial E}{\partial \vec{a}} = \sum_{\vec{x}} g(\vec{x})(\boldsymbol{A}^{\boldsymbol{T}} \vec{\nabla} f) \cdot \left(\vec{\nabla} f^T \cdot (\boldsymbol{A}\vec{\boldsymbol{a}} - \vec{x}) + f_t\right)$$

$$\frac{\partial E}{\partial \vec{a}} = \sum_{\vec{x}} g(\vec{x})\boldsymbol{A}^{\boldsymbol{T}} \vec{\nabla} f \cdot \left(\vec{\nabla}^T \boldsymbol{A}\vec{\boldsymbol{a}} - \vec{\nabla} f^T \vec{x} + f_t\right)$$

$$0 = \sum_{\vec{x}} g(\vec{x})\boldsymbol{A}^{\boldsymbol{T}} \vec{\nabla} f \vec{\nabla} f^T \boldsymbol{A}\vec{\boldsymbol{a}} + g(\vec{x})\boldsymbol{A}^{\boldsymbol{T}} \vec{\nabla} f (f_t - \vec{\nabla} f^T \vec{x})$$

$$\sum_{\vec{x}} g(\vec{x})\boldsymbol{A}^{\boldsymbol{T}} \vec{\nabla} f \vec{\nabla} f^T \boldsymbol{A}\vec{\boldsymbol{a}} = \sum_{\vec{x}} g(\vec{x})\boldsymbol{A}^{\boldsymbol{T}} \vec{\nabla} f (\vec{\nabla} f^T \vec{x} - f_t)$$

Finally giving the formulas that we used in the similarity transform code:

$$\hat{\boldsymbol{a}} = \boldsymbol{M}^{-1}\vec{\boldsymbol{b}}$$

$$\boldsymbol{M} = \sum_{\vec{x}} g(\vec{x})\boldsymbol{A}^{\boldsymbol{T}} \vec{\nabla} f \vec{\nabla} f^T \boldsymbol{A}$$

$$\vec{\boldsymbol{b}} = \sum_{\vec{x}} g(\vec{x})\boldsymbol{A}^{\boldsymbol{T}} \vec{\nabla} f (\vec{\nabla} f^T \vec{x} - f_t)$$

- **Step 3 - Validation of patches:**

As we track these patches across the sequence, it is possible that patches have begun to drift away from the region they were originally tracking, due to representational errors by the similarity transformation, i.e numerical issues or inability to represent skew, or due to non uniform motion within the patch. We validate patches by assigning a set of HoGs to them, typically 9 hogs within the patch. We calculate the orientation and magnitude of the constituents of each HoG of the patch in the first frame of the sequence, when tracking of the patch began. These values are then compared to those gained from identically placed hogs of the patch in the current frame. If these differ to too great of a degree, the patch is considered to be erroneous and is culled. Additionally we check the difference in each pixel as it appears in the current and first instance of the patch. If the difference at the pixel location between the two patches is too large we consider it be erroneous. If the ratio of erroneous pixels to normal pixels in the patch is too large, we again flag the patch as having deviated and should be culled. We combine these two methods, as the HoG method should capture a larger view of structural changes in a local neighbor hood around the HoG centre, but may not notice uniform changes of pixel intensity in its region. Conversely, the ratio of bad pixels informs us of very small, pixel granular, changes in the patch, but may not be sensitive enough to capture larger structural changes if these changes are isolated enough to not meet the ratio threshold. Examples of when this would take effect is shown below.

figure 15: A patch that has become erroneous due to the back ground moving in a different way than the foreground.

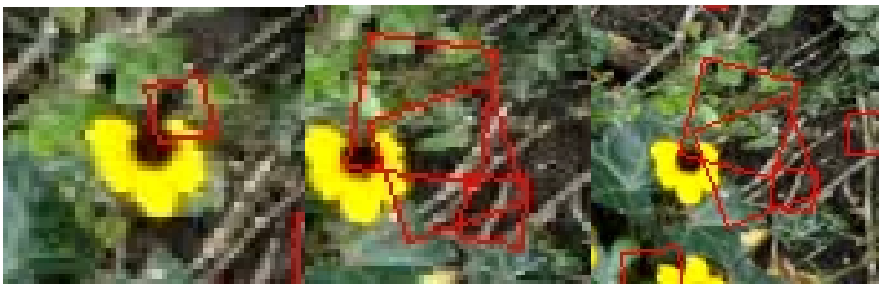

figure 16: Another patch that has become erroneous due to the back ground moving in a different way than the foreground.

Limitations

- Our patch selection favors patches that lie on regions where flowers or leaves border the background. These regions have strong opposing gradients that in theory would enable good tracking, but do not move uniformly and produce poor patches for tracking.

- The validation code tends to be overly harsh for small deviations in patches if the patch contains weak gradients (i.e is located entirely on a bckground leaf). However, weaker thresholds didn't prove to be adequate to handle the case in which results from the similarity transformation encouraged the patch to rotate and expand, likely due to the inability of representing patch skew.

- When new patches are added, we do not check if the region previously held a patch that was culled for deviating from its original place. In practice, we tend to have a few locations of where the foreground meets the background where patches are continuously added and removed (at a frequency of about 20-30 frames) throughout the sequence.

Improvements

- Implement affine transformation to replace similarity transformation and enable our algorithm to track skew.

- Store locations of patches that were previously culled and factor this into the generation of new patches, to avoid putting patches on regions that are not suitable for tracking, such as an area overlapping the background and foreground.

Component 3

Our algorithm functions in three major phases that enable us to solve F matrix to compute the epipolar line of a given point, and estimate H matrix to warp the images. Given two frames, sift points are determined in each image and the correspondence between points is then found. Using this correspondence, it is possible to begin ransack trials to estimate the F matrix of the two images. Additionally this correspondence can be used to perform a series of RANSAC trials in order to determine the homography that warps from one frame to the other.

Step 1. Detecting and matching SIFT points between two images.

To solve the F and H matrix, we need a certain number of corresponding points to have the minimum constraints necessary to compute our estimates. To figure out these points and there corresponding points in the other image, SIFT points are utilized, as it is possible to generate a reasonably large number of corresponding points between the images. The Vl_Feat sift modules are used to identify sift points from the two image separately, and subsequently match corresponding points that appear in both images.

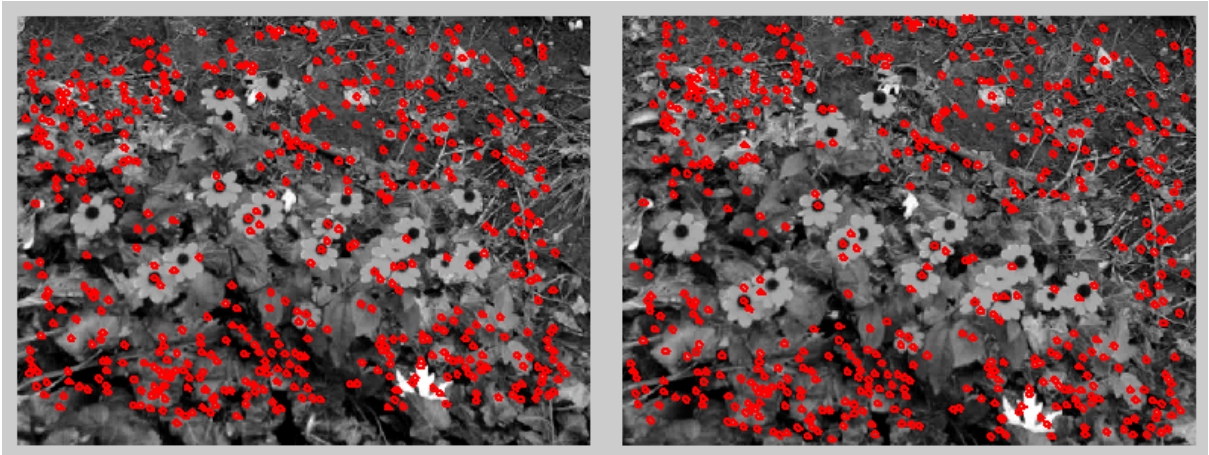

figure 17: These red circles are the SIFT matched points from two images.

Step 2 Estimate the F matrix for the image pair.
To estimate the F matrix, we perform the following steps:

1) Random pick 8 points from the SIFT point correspondence previously extracted. Solve the F matrix using those 8 points

We need a minimum of eight points to estimate the F matrix. A worse estimation may produced if more than eight points are selected, as we are basing our estimate on the assumption that the eight points are perfect inliers. However, if more than eight points are chosen then the probability that all of the points are actually inliers decreases proportionally to the number of points and the F matrix generated will have a larger degree of error.

2) Compute the error of epipolar line

To find the error of epipolar line, we take the perpendicular error value of the warped point to the epipolar line it originally generated.

3) Given a certain threshold, determine the inliners based on this distance.

4)Store the inliners, and repeat this for several trials,

We want to repeat this several times, as our randomly selected points may not all be inliers, producing errors in the estimate of the F Matrix. Thus, we retain all the inliners that we found in the preceeding steps.

5) With all of the inliers, recompute the F matrix again

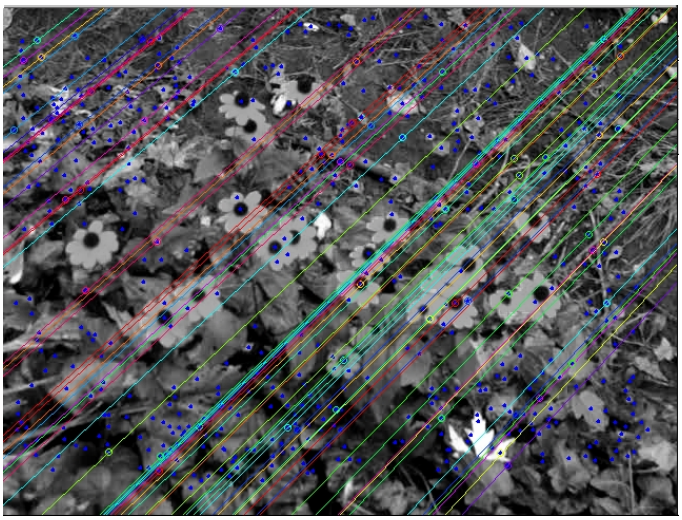

figure 18: Epipolar lines of applying F matrix on points in the second frame overlayed over the first frame.

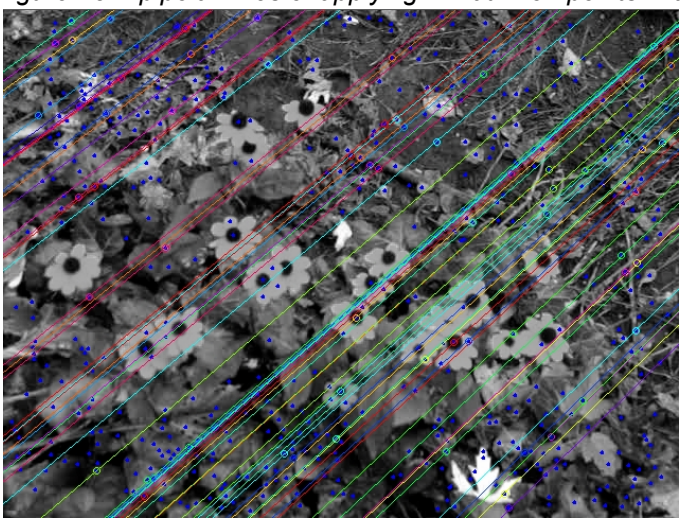

figure 19: Epipolar lines of applying F matrix on points in the first frame overlayed over the second frame.

Step 3a. Estimate a homography H which accounts for a "dominant" or reference plane in the scene

1) Find four pairs of corresponding SIFT points in order to determine the homography.

We need minimum of four points to estimate the
H matrix. The reason why we do not take more than four pairs is the same reason as was the case in estimating the F matrix.

2) Compute the error distance value of each SIFT points

To find the error distance value, we compute use a pair of corresponding sift points and the homogrpahy that relates the two images. We use the homography to warp one of the corresponding points to the other image, then measure the distance from the warped point to the point it originally corresponded to.

3) Determine inliners by comparing error value to a given threshold

4) Store the inliners, and repeat this for several trials.

As was the case in the F matrix, our results may have been generated from randomly selected points that were not entirely made up of inliers. Thus we can further refine our estimate in subsequent trials.

5) Recompute the H matrix with all the inliners.

Step3b. Rewarp one of these images by the homography H, so that the dominant plane is stationary in the resulting image pair.

figure 20: The first frame shown on the left, a frame 50 frames ahead show on the right

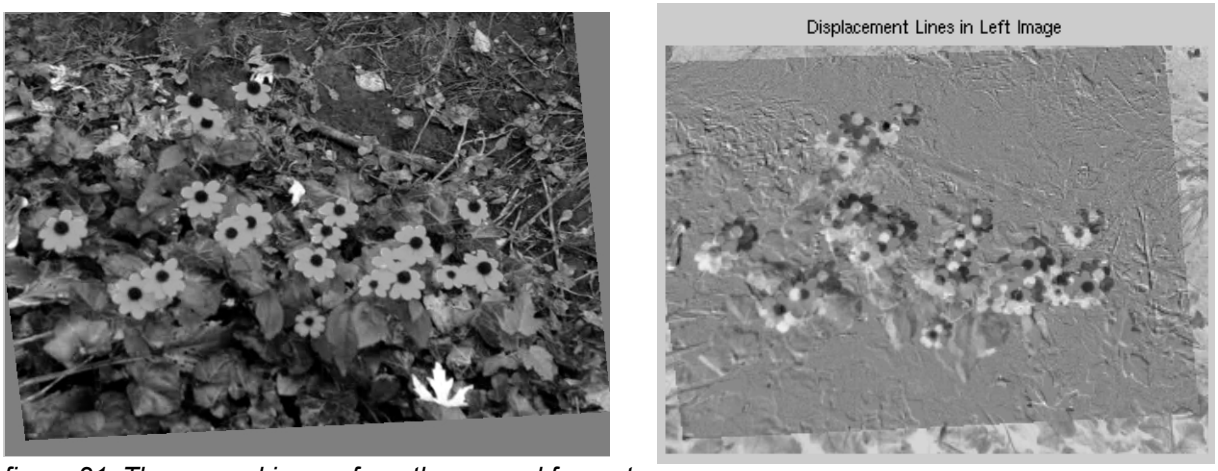

figure 21: The warped image from the second frame to the first first frame on the Left. The right image shows the displacement of the warping.

Step 4. Describe precisely the epipolar lines, and the displacement of corresponding points, for this new pair of images.

Given a homography H that maps ground points from *im0* into ground points in *im1*, and the fundamental matrix F that relates $(x_L F)\, x_R = 0$, we can then calculate a new fundamental matrix to display epipolar lines in the warped image:

$$(x_L F)\, x_R = 0$$
$$(x_L F)(H^{-1} H)\, x_R = 0$$
$$(x_L F H^{-1})(H\, x_R) = 0$$
$$(x_L F H^{-1})(x_{R warped}) = 0$$

Where $x_{R warped}$ are image coordinates in the warped right image. Thus the new fundamental matrix is $F H^{-1}$.

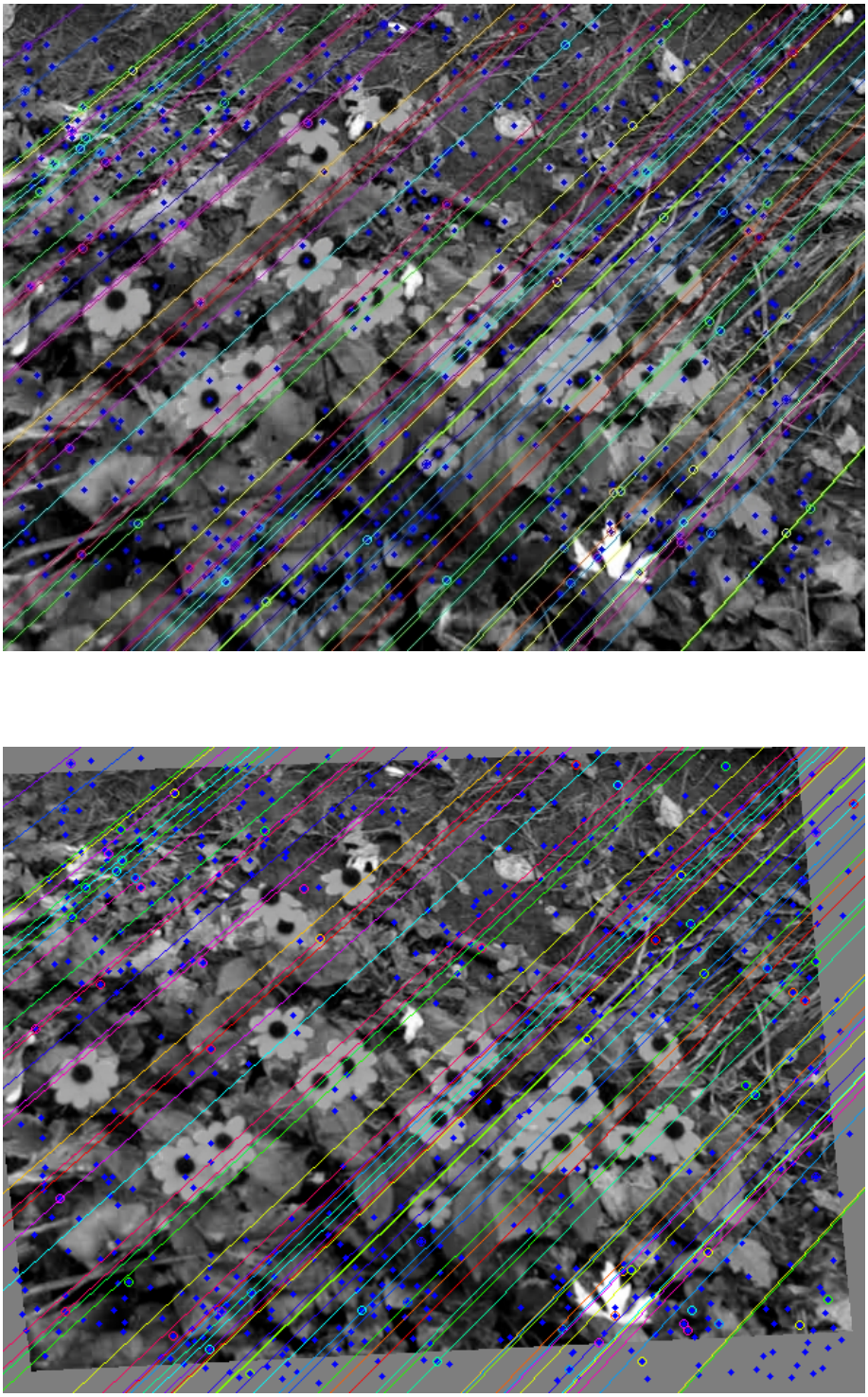

figure 22: given two frames im0 and im1, and homography H that warps from im1 to im0 such that the dominant plane is stationary, the top image shows the epipolar lines between that warped image and im0, whereas the bottom is the epipolar lines displayed on the warping of im1.

Instructions for running on a different image sequence:
To run on a different image set, modify the “infile” parameter that appears at the top of each component.

Other relevant controls, such as the start and end frame, number of frames to batch, number of patches to track, etc should be located at the top of each component.

Group responsibilities:

Alexander Kondratskiy:

- Flower database framework, circle fitting (reusing my code from A2), resolving flower matches using rohan's scores, and looking up flowers in previous frames for component 1.
- Similarity transform solving code, patch database framework, and cropping for component 2.
- Refactoring tutorial code into functions for component 3.
- Various helper function refactoring for the utils folder.

Jiwoong Im:

- Generating the random initial patches, and evaluating those random patches with R values per patch. Implemented harris operator matrix
- Dividing image into sections to keep track the location of losing patches, and responsible for adding new patches onto the corresponding image sections.
- Validating patches by counting number of pixels that has certain amount of difference in their intensity.
- Implementing similarity matrix
- Connected component for separating flower clusters.
- Detecting brown colors to get the mask for flower centers.

Rohan Chandra:

- Averaged R value calculation, patch comparison by hogs and scoring, patch database first and current frame storing, bbox soft cropping.
- Sift points and correspondence for initial homography estimate
- connected components and intersection to obtain final edgels
- circle fitting evaluation
- obtaining epipolar line in other image given a point and distance of points between frames in euclidean distance