Webcam Image Alignment

Authors: Matthew Klein

Follow this and additional works at: https://openscholarship.wustl.edu/cse_research

Part of the Computer Engineering Commons, and the Computer Sciences Commons

Recommended Citation

https://openscholarship.wustl.edu/cse_research/59
Webcam Image Alignment

Authors: Matthew Klein

Abstract: N/A

Type of Report: MS Project Report
1. Introduction

AMOS, The Archive of Many Outdoor scenes, has been a major project at Washington University. The project focus has been collecting images from webcams all over the world. Images have been logged from thousands of webcams for over 5 years. The large datasets created by AMOS are useful for a wide variety of problems in surveillance or environmental imaging --- at Washington University this includes data sets used to improve background subtraction algorithms, and automated tools for estimating, for example, when trees become green in the spring. The caveat for quality analysis to be produced is that the camera positions remain static. Ideally, these webcams would remain in a fixed static position providing datasets of images in perfect alignment.

One limiting factor for AMOS has been that cameras used for collecting images are uncontrolled. For a variety of reasons, these camera move, they blow in the wind, they can be physically relocated, a user can control orientation of the camera, etc. As a result, the images produced from these cameras which are not static cannot be used for long term image analysis. As state before the long term analysis is predicated on having images collected from a static camera.

Thus, we consider the problem of a large scale image alignment. The image datasets produced by the AMOS cameras are very large. The entire collection has over 90,000,000 images in total from 1300 cameras. Individual cameras can be sampled as frequently as two times per hour producing up to 48 images per day. The collection period for some of these cameras have been 2000 days. For one single camera, we can have nearly 100,000 images.
The specific goal of this project was to create a system that takes as input a set of 100,000 images from an unstable camera and produces the warping parameters to remove the camera jitter. The problem is a unique one because of the sheer size of the datasets. Images are commonly aligned to generate panoramas, however those images captured over the course of a few seconds and only a small number of images are aligned. In addition to dealing with the large size of image sets, there can be significant changes between images since they are taken every 30 minutes. The images below are samples of consecutive images taken from the same camera. Notice the large variations in the scene in the thirty minute period (where the dominant, highest contrast feature in the scene, the sun and its glare, moves relative to the scene, and changes its appearance).

In addition for determining the best mechanism for aligning images, a smart strategy is needed to handle the large number images in the datasets. The brute force technique of computing all possible n^2 image alignments is not feasible with the size of the datasets. There are a number of existing image alignment techniques and commercial of the shelf products available. Section 2 describes the background and notation for this problem, and some of the tools that were built upon. The evaluation of the image alignment techniques is detailed in Section 3. Section 4 gives an overview of how multiple images will tied together. Section 5 discusses methods for scoring an image alignment. Finally, Section 6 proposes strategies to link together alignments between multiple images. Finally, section 7 highlights some of the larger lessons learned with the selected toolset along with some discussion of what worked or more importantly what did not work.
2. Background Section
The fundamental piece for aligning these large datasets is performing an image alignment of a pair of images. Aligning a pair of images is a very well studied problem, and our goal is to build on top of this rather than devise a new method for image alignment. In this section we first define the problem of aligning pairs of images, and then current methods for alignment. Work in this area was performed to select the best performing method or best available COTS product which produces the best performance while being tolerant to large scene changes or handling little overlap in images. The output from these different alignment mechanisms is common.

That common output is a Homography Matrix. This matrix defines a transformation from one image to another. This transformation describes exactly where one point in one image falls in the other image. Since all points in one image can be transformed by the Homography matrix to another image, it is possible to align all pixels in one image to another image using this matrix.

The first methods tried used feature matching. The features are interesting points in an image which also have a description about themselves. Features can be extracted from any image. The feature descriptions or descriptors are comprised of pixels surrounding the feature point. Descriptors extracted from a pair of images can be compared to find matches. This approach was used with a variety of methods for determining features to use and how the match was performed.

A popular version of this feature detection and description method is an algorithm known as SIFT, Scale-Invariant Feature Transform [Lowe1999]. SIFT features are used quite extensively in the areas of image stitching, object recognition, and many other areas in computer vision. SIFT features were first proposed by David Lowe’s paper, Object Recognition from Local Scale-Invariant Features. What makes
these features and their descriptions so useful is that they are tolerant to changes in scale, orientation, partially invariant to illumination changes, and affine projection changes.

To use the SIFT algorithm for an image pair alignment, features were computed for each image and their respective descriptors were extracted. The SIFT descriptors were then matched using a brute force approach. In other words, all possible SIFT descriptor pairs, $n^2$, were considered. The best match for each SIFT descriptor in the basis image was selected as its correspondence. Using these correspondences produced from the descriptor matching, the feature point locations were used to compute a Homography transformation matrix. This approach performed well for the images that were similar in scene. However, SIFT Feature mapping struggled when the scene underwent large changes.

SURF, Speeded up Robust Features, was introduced in order to make SIFT more robust. [Bay2008] This approach varies from SIFT in both how the features are computed and what their descriptors are comprised of. They are similar in the sense that both use features and descriptor vectors which can be compared between points. Bay’s paper claims SURF outperforms SIFT, however for our difficult image alignment cases it was not the case. In fact, in nearly all cases SURF was found to not perform as well as SIFT and at best it matched the performance.

The “Lucas Kanade” method is typically used for computing optic flow. [Lucas1981] Using the Lucas Kandae approach to perform image alignment involved selecting points of interest in the images. These points of image used the optic flow method to find matches of these selected points between the two images. Using the matches between the points, a homography matrix can be computed. The optic flow method is suited for pixel changes which occur over a small distance. This method will not be tolerant to large movements. In addition, to using the selected points for pixel matching, all points in the images were used to calculate the transformation matrix. In this case, better performance was achieved
compared to the select pixel matches. However, computing the optic flow for all pixels in an image is slow. In general this method, did not perform as well as using SIFT feature matching.

Examining the feature matches produced by each of the methods attempted, the correspondence pairs contain many mismatches. RANSAC, RANdom Sampling and Consensus, is a method that estimates a solution from a dataset while ignoring outliers. RANSAC can reduce the number of outliers used in the homography calculation. This is done by choosing a subset of image pairs, computing the homography with those pairs, then evaluating how well the rest of the matches fit to that homography. This is an iterative approach where different pairs are tried. The number of pair sets tried is a fixed number. This method improved the performance of each of the methods.

Another attempt was made to reduce the number of mismatched correspondence pairs. This time the descriptor match step was modified. Rather than use a Brute Force approach comparing all correspondence pairs, a nearest neighbor search approach was used. FLANN, Fast Library for Approximate Nearest Neighbors, was the specific search method used.[Muja2009] This approach was successful in reducing the number of outliers produced in the match step.

Using SIFT, FLANN, and RANSAC offered the best performance. However, it still did not appear tolerant to the scene changes that occur in the AMOS datasets. Another attempt to improve performance was to first compute the edge map of the images prior to performing the feature matching step. The goal was to produce an edge map image that isolates objects in the image pairs that remain constant through various lighting scene and seasonal changes. Unfortunately, this method did perform well either. In most cases, the alignments produced were worse than their counterpart without the edge maps computed.
Another paper “Registration of challenging image pairs: initialization, estimation, and decision” written by Gehua Yang et al., proposed a solution specifically tailored for aligning difficult image pairs. The documented aligned image pairs appeared promising. A free software implementation of their algorithm was available. The gdb-icp software accepts an image pair. The software produces an aligned image and a transformation file containing the calculated homography. Running gdb-icp on the test data demonstrated gdb-icp was most tolerant to the difficult scene changes. It also handled images with minimal overlap so image pairs with large motions could also be aligned.

In addition, to the free offering of gdb-icp a commercial version was also available, i2k Align. i2k Align was created by DualAlign. It offered all of the benefits of gdb-icp, but also offered additional improvements. Extra input parameters and additional output information were added. This application provide to be best option of all those considered. It was selected as the method for performing the image alignment.

A project with a similar project domain exists, Microsoft’s Photosynth. Photosynth takes a set of images taken of a particular structure or location and recreates it 3-d reconstruction from the images. Photosynth supports images taken from different sources or different time periods. However, it is acceptable for Photosynth to discard many images and select only those images it can reliably match. As a result the 3d scene that is recreated primarily contains images which are all similar in nature. The Webcam Image Alignment project cannot make this exception. It needs to align all images regardless of time or image similarity.
3. Image Alignment Selection

Introduction
Already discussed were a number of mechanisms for image alignment. Regardless of how the image alignment is performed, this procedure becomes the fundamental building block for this project. Of all the methods investigated, i2k Align offered the best performance.

To evaluate the correctness of the homographies a number of small test datasets were created. These datasets were produced from images collected from AMOS webcams. To produce these test datasets, selected images were taken from webcams whose images that were already in alignment. To simulate the misalignment problem, a random transformation Homography was created for each image. These Matrices were then used to transform each of the test images. When evaluating the correctness of the Homography, these matrices were used as the ground truth. An ideal image aligner would reproduce the inverse of these matrices. The inverse matrices would transform the warped test images back to their original aligned position. The test images covered a range of seasons to simulate images taken over the course of a year. Some examples are shown here:

SIFT
Using SIFT features, when the image scenes were very similar, taken from the same time of day and a similar day of the year, the image alignment performance was acceptable. The discerning factor in how well the alignment performed was the number, correctness, and coverage of feature matches in the image pairs. An ideal pair would have a large number of correspondences with few mismatches and the correspondences would cover a majority of the images.
The first example shows an image pair of the test data. The first image was taken on 2006-07-04 at 6:00pm and the second was taken the following day, 2006-07-05 at 6:02p. The images appear to be reasonably similar. The largest variations appear to be in the contrast of the image.

The images below show the same two images along with their SIFT features. The circles on the image show the SIFT features found in each image. The color of the circle indicates how well it fit on the calculated homography. The green color indicates zero error and red indicates large error. Essentially this can be interpreted that the green features were proper matches and the red features were mismatches. In this case, there are a large number of correct matches, green circles. There is also good coverage of the image area. The result is a properly aligned image.
In contrast, the same image used above on the left was matched to an image taken on 2007-12-16 at 6:00 pm. This image was taken in December. Comparing and contrasting the images shows the tower and buildings remain very similar, while the snow covered ground is different. The sky has changed significantly in color from white to blue. The trees have also shed their leaves.

Examining the feature correspondences, there are a great number of features detected in each image. However, there are very few correct matches, green circles. In addition the matches are confined to the tower and building areas. The rest of the image is left with no proper matches.

The transformed image is show below. It is an obvious failure. This is likely due to the poor coverage area of the matches.
**SURF**
The SURF feature matching performed similarly to the SIFT feature matching. In general, it under performed though. In this example, the same images from the SIFT feature match are shown. In this case there are fewer green circles. The feature matches are still focused on the tower and the building and in similar areas of those structures. Overall, there are fewer matches. The result is a transformation that’s worse the SIFT counterpart.

**Lucas Kanade**
Lucas Kanade performed well with the test dataset with small variations in motion. However, as expected it struggled with large motions. In the example shown below, the image pair was collected from an AMOS camera. This was not an induced motion, but rather real motion in the camera. The images appear relatively similar, but have a large motion between the two of them.
The result below shows how the Lucas Kanade method was unable to align this pair.

![Original Image](image1.jpg) ![Edge Map Image](image2.jpg)

**Edge Map**
The use of edge map images were performed in hopes of isolating the aspects of the image that would remain similar regardless of seasonal impacts, show as snow or leaves of trees. The images below show the original image and an edge map computed for that image. Notice the images on the left appear very different with the snow covered ground and lack of leaves on the trees. Comparing the images on the right, the strong edges of the images are now highlighted. Areas such as the tower and the building appear very similar. The steps also seem similar while they appear very different with the snow on the left.
SIFT Features were computed on each image and matched in the same fashion as the normal SIFT matching. Inspection of the feature matches shows an expanded coverage area for the images. But, there are too few matches to produce a proper alignment.
The transformed result is show below. It was a failed alignment.

The first example shows the same images from the test data. In this case, there are an even larger number of feature matches on the building and tower. In addition to having more feature matches in these areas, there are additional matches in the areas on the steps and on the ground. This results in a much more robust coverage area and a properly aligned image.

**i2k Align**

As previously stated i2kAlign offered the best performance of all the evaluated methods. The examples shown below highlight the effectiveness of the application compared to other alignment approaches.
The next sample shows the image that the Lucas Kanade struggled with. I2kalgan finds many matches throughout the terrain and over the buildings in the images. There are also a few matches in the sky. Most of the few mismatches that do occur are in the sky, however.

The aligned image without the features is shown below.
Here is sampling of three images aligned by i2k Align. These images were collected from a AMOS webcam. The images collected from the camera were not in alignment, no test transformation was applied to the images. The images go through large scene changes. The middle image is taken at night. The orientation of the street sign shifts from image to image. The trees are full of leaves in the right most image, bare in the left image and not visible in the middle. There is also a large motion in the middle image. I2k Align impressively handles all of these variations and large motions and properly aligns the images.

4. Multiple Image Alignment
Building on the ability to align image pairs, it becomes possible to align large sets of images. Large sets of images can be aligned by “chaining” the alignments together. For example, consider a case with 3 images: A, B, and C. Image A has overlapping image data with Image B, but not Image C. Image B has overlapping image data with both B and C. It is possible to align Image A to Image B and then align Image B to Image C. Each of these alignments produces a transformation homography. One property of the homographies is that a set of homographies can be chained together. The chain processes involves the multiplication of each of the homographies in the set. The result is a homography that when applied to an image would have the same effect of transforming the by the first homography and then transforming that image by the 2nd homography. Suppose image A has a homography that transforms A to B, $H_{a\rightarrow b}$ and image B has a homography that transforms B to C, $H_{b\rightarrow c}$. The transformation of $H_{a\rightarrow c}$ can be computed by multiplying $H_{a\rightarrow b} \ast H_{b\rightarrow c}$.
\[ H_{a:b} = H_{a:b} \ast H_{b:c} \]

**i2k Align**

Although i2k Align’s performs well with difficult image alignments, unfortunately it is not able to align all image pairs. As stated earlier, it is possible to chain images together. As a result, the desired goal of aligning all images in a large dataset can be achieved without the need to align each pair. The i2k Align software does support aligning multiple image sets not just individual pairs. End to end running time tests were performed with i2k Align with a variety of image set sizes. These timing tests showed that the running time grew exponentially. This led to the conclusion that i2k Align is comparing all \( n^2 \) image pairs to perform the alignment. Since all possible pairs are considered it’s not practical to use this method for large datasets. The example figure shows the running time for different image set sizes.

![Running time graph](image)

The running time for a naïve solution to align a dataset is \( O(n^2) \). This is naïve solution involves comparing each possible image pair in the set, which it shown that i2kAlign does. The absolute minimum number of alignments required in an ideal solution would be \( n-1 \). However that requires an assumption that all \( n-1 \) alignments are successful. With the large datasets in the AMOS database, that
assumption is unreasonable. Thus, a method must be devised for strategically selecting image pairs for alignment.

5. Alignment Evaluation

**i2k Align binary output**
For this method to work correctly, a reliable method for classifying an alignment as a success or failure is needed. It is also important that no false positives are added to the successful alignment list. If this should occur, that false positive could be used as basis in the subsequent passes, thereby causing all images that align to that basis to be incorrect. Fortunately, i2k Align provides output data in addition to homography and the translated image. This output includes a binary output that indicates whether or not the alignment failed. I2k Align misclassifies a very high percentage of alignments as false positives. As a result, it’s not sufficient to simply use that as the means for evaluating the alignment.

**Per Pixel Comparison**
A method is needed to evaluate the correctness of an alignment. In most cases, it is acceptable to accept images that are not perfectly aligned. Ideally, the evaluation mechanism provides a score that rates how well the image was aligned. For example, a transformation that maybe one pixel off compared with one that is ten pixels off would have a different score. It would not simply be a binary result, pass or fail. One possible approach would be computing the per pixel difference from the transformed image with the original. In this case, each overlapping pixel between the two images would be compared. The pixels falling only in one image would be ignored. This results of this approach are not always indicative of how closely aligned the images are. Consider an image that with a picket fence. If this image is only slightly misaligned with the basis image, the per pixel differences will be significant. As a result, this image certainly will be discarded. This approach will result in far too many false negatives.
**Number of Feature Matches**
Another approach involves the use of the output correspondence file produced by i2k Align. This file is an important piece of data and will be considered in all subsequent approaches tried. This simple method involves counting the number of correspondences in the file. As stated earlier, in the paper the correctness of the alignment in feature based matching is typically dependent on the number of matches, the coverage of those matches in image, and the correctness of the matches. For this method, the number of matches is considered. This could give an indication of how well the images are aligned. However, some images simply do not have many features. Also, an image pair can be very successfully aligned, but there is a large occlusion in one of the images. The large occlusion would prevent a higher of feature matches from being counted. These consequences cause the feature match approach to permit a number of false negatives to be calculated. In addition, a high match count doesn’t really consider the correctness of the matches and could be producing false positives.

**Summed Weight Score**

The first method that was implemented for evaluating the matches again used the correspondence file produced by i2k Align as the source of data. In addition to listing the feature correspondences between the two images, this method also included a weight associated with the match. The weight assigned to the correspondence indicates the influence the i2k Align algorithm gave to the correspondence when computing the homography. [The i2k Align and i2k Align Retina Toolkits: Correspondences and Transformations, Charles V. Stewart]

This method using these weights was a simple and reasonably effective means for evaluating the matches. As stated above, each correspondence listed in the file had associated weight value. The weight is a value between 0 and 1. Since the higher value indicated more emphasis from i2k Align the algorithm, the assumption was made that this weight varied directly with the correctness of the match.
The simple method used was to sum the values of the weights for all correspondences in the file. The i2k Align documentation states that the number of correspondences in the file could vary however in practice in every alignment the number of correspondences was always fixed at 800. The fixed count of 800 allowed for a simple sum of the weights to be performed. This summed total could then be compared with a threshold to determine if the alignment was successful or not.

To show the effectiveness of the summed weights, the following charts show the summed weight and the error of the transformation produced by i2k Align. The error plot on the left was calculated the same way and is the same plot depicted above. The right plot shows the confidence score that was calculated by summing the weights of the correspondences. There is a reasonably strong correlation between the ground truth error and the sum of the weights.

**Homography Cross Validation**

Another attempt to validate the alignment was to use a cross validation matrix. This test was performed by first aligning all images to one basis image. The image alignment with the strongest confidence, computed by using the sum of the weights, was then selected as the new basis image. In this pass, all images were aligned back to the newly selected basis. Homography matrices were computed by
multiplying the homography produced from the transformation to the original basis and the transformation to the new basis to the current image. Remembering the homography matrices can be chained, it is possible to multiply the homography from the original basis to the new basis with the homography from the new basis to a selected image in the dataset. This result from this chaining process should be the same as the homography from the original basis to the selected image in the dataset, assuming all the homographies computed represent proper transformations. To evaluate how similar these matrices match the Frobenius norm was calculated.

\[ H_{\text{original basis}} \times H_{\text{new basis}} \times H_{\text{new basis}} = H_{\text{original basis}} \times H_{\text{selected image}} \]

The assumption was that if the images were properly aligned the original homography would match the computed homography. In practice, this succeeded in removing nearly all of the false positives, unfortunately many false negatives were computed. Because of the great number of false negatives, this was not an effective mechanism for evaluating alignments.

Future work is possible to improve this method. A variation on which images are used to compute the cross validation could be applied. It was learned in general Images who are similar in time are able to be aligned well. This is evident by the ground truth error plot shown above. Rather than use the same two basis images for the cross validation step in a current pass, a different basis could be used for each image in the pass. The image selected as the basis for the cross validation step should be a neighboring image. The manner in which the cross validation score is computed would be the same, only one of the basis images would be different.

**Correspondence Match Error**

Another possible mechanism to validate the alignment is to use the correspondence file again. In this case, the locations correspondence points produced will be considered to evaluate the alignment. In the correspondence file for each feature match, there two points listed. The first point is the center of
the feature in basis image and the second point is the center of the feature in the image being aligned.
The points listed for the image being aligned are in the transformed coordinate system. So, if all the
listed correspondences are perfect matches each feature match would have the same two points listed.
This almost always not the case, however, and would likely only happen if the images being aligned are
identical. So, for all other cases the matching points can be compared. Using the points for each
correspondence the Euclidean distance is computed in pixels.

The translated pixel locations were computed by i2k Align by applying the calculated homography to
those points. The pixel error for each correspondence shows how well that correspondence fit to the
homography. These error values should give an indication of how many correct correspondences were
computed and severity of the mismatches.

To generate a confidence score the expected value was computed on the all of the error values in the
set. With this confidence score there is now ability to threshold these scores at some point to classify
the alignments as successes or failures. For the AMOS datasets, it is preferred to take a conservative
approach. In other words, it false negatives are favored over false positives. To determine the proper
threshold one year’s worth of data was selected from multiple AMOS cameras. i2k Align was used to
compute the alignment for all images to a selected basis. The correspondence file was saved for each
image. After the alignment, a chart was created showing the expected value for all of the alignments in
the dataset. A video was also produced of the transformed images. Careful inspection of the video and
the chart produced a useable threshold value. The process was repeated for the different cameras to
generate a robust number that worked in the general case.
6. Alignment Techniques
This section discusses the methods used to take our image pair alignment tool and use it to align large
datasets of images. This is done by choosing the image pairs to align and then using the property of
homography chaining to combine these alignments. A brute force approach would align all possible
image pairs. This approach is not practical for the large datasets considered in this project. The two
methods detailed below propose methods for aligning the large dataset without having to consider all
possible image pairs. The tree structure is the first approach considered followed by the greedy
algorithm.

Tree Structure
The first method involved breaking the images into a tree structure was used. Next, i2k Align would be
used to perform image alignments on smaller subsets of images in the tree. The basic concept was that
image pair alignments perform better with similar images.

Sample test data was created to have a quantitative mechanism for evaluating the alignments produced
by the algorithm presented in this paper with the i2k Align image alignment tool. The data was
produced by taking one week’s worth of images from a particular webcam. These images were
captured already were in alignment. A random homography which contained a rotation only
transformation was generated for each image. This homography was stored and applied to each of the
images. The transformations produced by i2k Align could then be benchmarked against these
homographies.

To show the effectiveness of the summed weights, the following charts show the summed weight and
the error of the transformation produced by i2align. The error is computed by warping the four corners
of the image by the ground truth homography and the i2aligned homography. The ground truth points
are diffed with the calculated points. The plots were created by taking one day of images from the
weeks’ worth of test data. Each pass one image was selected from the image set and all images in the
set were aligned back to that image. This was performed for each image in the set. The result was a set
of alignments from each image in the set to each image in the set. For each of the alignments, the
certainty score was produced by summing the weights of the correspondences and a ground truth
error was calculated. The error was calculated in the same fashion described previously, using a corner
of the image with the largest error. The plots below show the results. There is a reasonably strong
correlation between the ground truth error and the sum of the weights.

The plot below shows error compared to the ground truth data for one day’s worth of images. This plot
was created by performing the image alignment on all possible image pairs. The error is computed by
warping the four corners of the image by the ground truth homography and the i2k Aligned
homography. The ground truth points are diffed with the calculated points. The maximum error in
pixels for one of the translated images compared to the ground truth image was the mechanism for
calculating the error value. The plot shows that the images which are similar in time perform better for
image alignment.
Using what was learned about similarly timed images, the images from the dataset were grouped by hour of the day. These image groups were then broken into sub groups of size 10. These subgroups made up the bottom level of the tree. The sub group size of 10 was selected to reduce the number of image alignments required by i2k Align. For the subgroup, one image would be defined as the basis, the image that all other images in the subgroup align to. The next level of the tree would contain the basis images from level below. Again, these basis images would be divided by group and then placed into subgroups of 10. This process would be repeated until for each hour there was one basis image. The single hour images would be divided into 3 subgroups of 8 images, where hours 0-7, 8-15, and 16-23 are
together. Finally, the basis images from these last 3 subgroups were aligned to complete the top node of the tree.

Once the tree structure was created, each node of the tree defined had a set of images as children and had a defined basis image from those children. These groups of children along with the basis image were written into lists which i2k Align could accept as input. i2k Align would then be performed on all the image input lists in the tree. The result for each node with children was a set a homographies that defined transformation for its children to the basis image. To compute the homography for any image to the basis image for the dataset, the child image would multiply it’s homography times the parent homography until it reaches the root node of the tree.

The bottom up approach performed well, but still required a great number of alignments. For example, consider one year’s worth of images were there are two images per hour. So, for one particular hour we have 730 images in the dataset. This breaks into 73 different subgroups. Each subgroup requires \(10^2\) alignments. As a result we have 73*10^3 image alignments required for each hour for the base level. That yields 73 basis images, which break up into 8 groups. 73*10^2+7*10^2+3^2=8079. So for the all 24 hours in the day we have 8079*24 or 192,216 alignments. In addition to aligning the different hour groups, the basis images from each hour must be aligned. Those images were broken into groups of 8, 3*8^2, and finally those basis images were aligned, 3^2. In total the minimum number of alignments required was 192,216 + 3*8^2 + 3^2 = 192,432. This is a significant reduction compared to the worst case of \((365*24*2)^2\) or 306,950,400 image alignments for the year. However this is still a significant number of alignments.

In addition, the assumption was made that an image to be aligned would have a high probability of success in its subgroup. As a result, to deal with failed alignments, the image was discarded, since the image had already been compared with 9 other possible candidates for a match. This method may have
not been acceptable for the overall goal of the project, but it was successful in keeping the tree structure intact, with one caveat. That caveat being the failed image must not be the basis for the group. If it is, the tree needs to be restructured selecting a new basis. At this point, the difficult handling of failed alignments along with the high number of minimum alignments required, it was decided to investigate another approach.

**Greedy Algorithm**
The three structure discussed above was built using a bottom-up approach. The next solution is a much simpler top-down approach. In this case, a greedy algorithm was used. First, a basis image is selected. Next, an attempt is made to align all other images in the dataset to that basis image. The successful alignments are saved and removed from future processing. Any failed alignments are saved for consideration in future passes. In subsequent passes, a new basis image is selected from the list of successful alignments. Then, the images in the failed alignments are then aligned to the new basis image. This same process is repeated until there are no images left in the failed alignment list (all images have been successfully aligned), or all images in the success set have been used as a basis image. In the case, where all basis images have been considered, the remaining images in the failed alignment list could not be aligned. Future work could be done, to explore a possible separate connected component(s) in the failed subset.

**Basis Selection**
Now, that the image alignment method is selected and there is a suitable alignment evaluation method, the final piece to the greedy algorithm is the basis selection. This selection is critical in how many image alignments are required to align a dataset or how quickly the entire alignment can be run. In general the main factors that determine whether or not an alignment will be successful are the time the image was taken, day of year, and how much overlap exists with the basis image. For example, one week
worth of data was aligned using an image taken at 12:00pm as the basis. The histogram below shows the number of failures per hour.

![Histogram showing number of failures per hour](image1)

The chart below shows a plot of the expected error value for all image alignments in one year’s worth of data. The basis image selected was taken around 12:00pm on January 1\textsuperscript{st}. The dataset was stripped down to only contain images taken in 10:00am to 3:00pm hours. The chart shows a trend where the expected value gets worse towards the middle of the plot, this area represents the summer months. It then starts to improve in the fall months.

![Chart showing expected error value](image2)
The last factor mentioned was the amount of overlap between the basis image and the image to be aligned. This case is specific to cameras that exhibit large motions. In fact, some of the cameras move so much that there is no overlap with a particular image and the basis. In this case, an intermediate image or images must be selected to bridge the gap between the two images. For these types of cameras with large motions, it’s critical to choose a new basis image that expands the field of view covered by the aligned images. In fact it is necessary for some of the images, because there can be no overlap with the original basis. The follow images show an example of how much some cameras in the AMOS dataset can move from one image to the next.

The way the basis is selected for both the day and time approach is very similar. As the alignments are evaluated, the number of failures for each hour are counted along with the number of failures for each month. When selecting the next basis, these histograms are examined to find the month or hour with the largest number of failures. Next a successful image alignment that was closest in terms of month or hour is selected as the new basis. The confidence score is used as a tie breaker, so the best alignment from that month or time is used.

For the datasets that exhibit large motions where there can be minimal overlap with the basis image, the transformations are considered. The transformations for the successful alignments are examined to find the homographies that causes the largest transformation. To determine the largest transformation, the corner pixels of the aligned images are compared before and after the transformation. The alignment which has the largest corner Euclidean distance in pixels is selected as the next basis.
Since the efficiency of the greedy algorithm is directly dependent on the selection of the basis image, to achieve optimal performance it’s critical to have an effective method for selecting the basis. Future work is required to put this in practice. Currently, each of these methods has been implemented on their own, but has not put together in a complete algorithm.

**Homography Computation**

Finally tying it all together requires computing the homographies for each image back to the original basis image. In each pass of the greedy algorithm, a basis image is recorded and a list of successfully aligned images with their associated homography matrix. This data along with the property of chaining homographies is sufficient to compute a homography which aligns the image with the original basis. The structure is essentially that of a tree. The root node is a basis image and its children are the images that were successfully aligned to that basis. The children who were not used as a basis image in subsequent passes do not have children below them. Images that were used a basis image have a set of child images that were successfully aligned to them. To compute the homography for each image, the tree is traversed while multiplied the homography at each level. This multiplication process chains the homographies together to produce the final homography for each image.

### 7. Lessons Learned

One major limiting factor this project was devising an effective means of evaluating an alignment. The three main techniques tried were the weighted sum score, the homography cross-validation and the correspondence pixel error. None of these methods achieved 100% accuracy in classifying successful alignments. This section will detail how each of these methods were actually performed and some of the shortcomings.
As stated earlier the weighed sum score summed the weight associated with each correspondence listed in an output file produced by i2k Align. This file is a text file that is produced after i2k Align aligns an image pair. The file header lists the images being aligned and the number of matches found. It appears that this value could be variable, however in each and every image pair considered throughout the course of this project that number was always fixed at 800. Following the header, the correspondences are listed. Each correspondence has 17 values. The weight given to that particular correspondence followed by 8 values from the feature in image 1 and another 8 for the feature in image 2. In order to compute the score, this text file was parsed and the weights were summed. Since, the weight count was fixed at 800 for all alignments, it was sufficient to simply sum the weights. It’s not clear what this weight represents, the i2k align documentation states the weight relates to the influence that particular correspondence had on the algorithm. Regardless of what this weight exactly meant, there seemed to be a strong correlation between this score and how well the images were aligned. However, this method was still prone to false positives.

The next method was the homography cross validation. In addition to the correspondence file, i2k Align produces a transformation file. This file details the homography that i2k align computed and then used to warp the image. To extract this homography, the text file was parsed. In addition to using the homography extracted for cross validation, this homography was used to warp the images. The alignment images produced by i2k were not used, instead the homography was used. The cross validation seemed promising as it was highly successful reducing the false positives permitted. But, this method produced a great number of false negatives. It may be useful to revisit this area though and attempt to perform the cross-validation method using images that are similar. This project used two basis images to perform the cross-validation. However, a better approach may be to use one basis and then select two images that are similar to complete the triangle of images.
To compute the correspondence pixel error, the correspondence output file again was parsed. In this case, the focus was the location of the point pairs. Values 6, 7, 14, and 15 represent the x, y coordinates in both images. For the image being warped, the point location is given in the warped coordinate system. In other words the homography has been applied and transformed the point vector. So, to compute the correspondence pixel error it was simple enough to compute the Euclidean difference between the two point vectors. Finally, the expected value was computed using all of these error values. Unfortunately, this method did not accurately identify image alignments that failed. It was prone to false positives. Also, the threshold used to classify the alignment, did not seem consistent for different datasets. For example, a value of 1.2 may be needed for one value and 1.9 for another. That was fairly sizeable margin.

In generally, i2k align performed well on daytime scenes. Not surprisingly, it struggled with night or dark scenes. Images similar in time and season generally were able to be aligned. For some of the night scenes, there is not a lot of valuable information visible in the images. Many of the pixels may just be black.

References

