Match and Mosaic Generation from Small Camera Object Separation

Paul Dickson, Al Hanson, Ed Riseman
Computer Vision Lab
Department of Computer Science
University of Massachusetts at Amherst
Amherst, MA 01003
{pauld,hanson,riseman}@cs.umass.edu

18th April 2005

Abstract

Since September 11, 2001, the threat of terrorism has become a greater concern for the United States. As concern has grown, so has the need for technological innovations tailored for tighter security. This paper describes the vision portion of a system to upgrade security at airports, sporting events, federal buildings, and other locations where there is a threat from vehicles carrying contraband into the facility. The proposed under-vehicle inspection system uses an array of cameras and mosaicing techniques to generate views of a vehicle's undercarriage from different perspectives. These views provide a pseudo three-dimensional view of the undercarriage in a graphical user interface. The paper discusses the problems inherent in matching and mosaicing images taken with small separation between cameras and object, as is found in the under-vehicle inspection system, and proposes a partial solution to the problem.

1 Introduction

1.1 Motivation

Inspection stations for vehicles at locations under terrorist threat have become a part of life, and one of the locations checked on vehicles is the undercarriage. Two main techniques are currently used to inspect the undercarriage. The first technique has an inspector slide a mirror under the vehicle [1, 2]. It has the advantage of being inexpensive and easy to set up. It's disadvantage is that the inspector cannot view all areas of the undercarriage because of the compounding problems of viewing angle, physical constraints on mirror placement, and occluded portions of the undercarriage. Typical examples of the mirrors used in these systems can be found on the Internet [1, 2]. Some work has been done

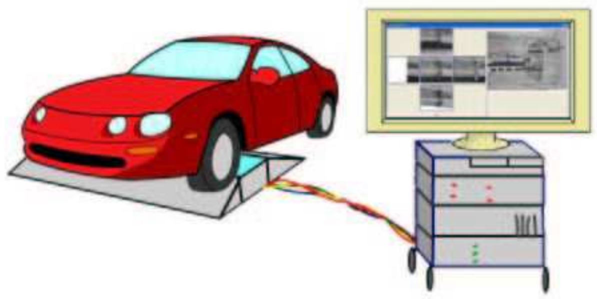


Figure 1: High-level system design

to improve this technique by replacing the mirror with a camera [3]. This will solve some of the problems of viewing angles typically found with mirrors but the other disadvantages remain.

The second technique for inspecting an undercarriage uses an inspection bay. This has the distinct advantage that the inspector can see all portions of the undercarriage and can easily focus on any suspicious portion. Disadvantages are that this both exposes the inspector to danger and requires the construction of an inspection bay.

One other inspection technique is beginning to be used but has not as yet gained the widespread acceptance of the first two methods: a vehicle is driven across a line of cameras that capture images of the undercarriage. The system proposed here is based on this technique. The only system currently available simply records the input from the cameras to videotape and allows the inspector to inspect the undercarriage by moving the tape forward and backward in a VCR[4]. Like the first method, this technique does not allow an inspector to view the entire undercarriage at one time.

1.2 System overview

The goal of this paper is to discuss the vision aspects of the Under Vehicle Inspection System (UVIS), which is designed to allow a complete inspection of the vehicle undercarriage. The overall layout of the system is similar to that of the commercial system with the line of cameras that the vehicle drives over, and is shown in Figure 1. Images are continuously taken as the vehicle drives over the cameras. These images are then mosaiced together to generate perspective mosaics of the vehicle undercarriage from five different viewing directions. These mosaics are displayed together within a GUI for visual inspection by an operator.

1.3 Prior work

The mosaic generation techniques used in UVIS were originally designed to create a compact representation of aerial video of forest canopies used for environmental monitoring [5]. The techniques were further refined for construction of mosaics from camera movement that was primarily translational [6]. The translational shift model of a stationary forest with a plane flying over it at constant height is, for all practical purposes, the same as a stationary camera set with a vehicle driving over it. In both cases only one part of the model is moving, and the movement is mostly translational.

One of the main elements of the previous work that has been adopted by UVIS is the creation of mosaics with different perspectives, or stereo mosaics [5, 6]. These techniques will be described in detail in sections 3.2.4 and 3.3.2. The stereo mosaicing techniques were originally developed for use with rotating cameras [7, 8, 9]. The original method proposed by Huang and Hung [7] used a pair of rotating cameras to create stereo mosaics. This idea was then refined by Peleg and Ben-Ezra [8] and Shum and Szeliski [9] to require only a single off-center rotating camera. As was discovered previously [5, 6], techniques based solely on rotation do not translate well to motion that is primarily translational.

The basis of stereo mosaic generation is that motion parallax can be used to create perspective images. This technique requires that depth difference between objects in the scene are small relative to the distance between the cameras and the scene. For example, the manifold projection by Peleg and Herman [10] cannot create seamless mosaics within the UVIS domain because the depth differences between different parts of the undercarriage can approach 25% of the distance between the cameras and the vehicle. A previous version of UVIS [11] included the local alignment work of Shum and Szeliski [12] and an efficient ray interpolation technique based on the local alignment techniques in the matching phase of the mosaicing process. The new system represents a marked change in methodology, shifting from local alignment techniques developed for unknown camera-to-object motion matching to a more rigid epipolar geometry-based matching approach that searches only specified regions for matches.

The previous version of UVIS [11] showed that relatively seamless mosaics can be created even by a system in which the cameras are close to the object. It successfully used simulated data to create a two-dimensional (2D) representation of the three-dimensional (3D) vehicle undercarriage. These mosaics demonstrated an effective improvement of view when compared with current inspection techniques. The simulated data was created by using a video Web server to connect to the cameras that took pictures of a model of a vehicle undercarriage. This system proved unable to mosaic more realistic data.

1.4 System goal

The goal of the new system was to take the concept proved by the previous version of UVIS [11] and implement it so that it would work with realistic data and in a manner that would allow it to function in close to real time. This paper will not describe the computation time required by the new system, but a discussion is attached in Appendix C.

The new system uses camera position information and vehicle rigid-body information to find matches between images. This new methodology should be faster than the old methodology because geometric constraints eliminate large portions of the search space within the matching problem. This new system is tailored to work with synchronized cameras likely to be used in the eventual system.

2 Vehicle inspection problem

2.1 Problem definition

The goal of the UVIS project was to create a system that would eventually replace the current systems used for inspecting vehicles undercarriages. The initial design considerations were that the system should be portable, have no moving parts, and run in close to real time. In addition, it was hoped that a system could be designed that would eventually support automatic detection of unwanted material under a vehicle. For the system to be an improvement, it had to include at least as good an inspection as was previously achieved as well as address current problems.

As mentioned above, the most common current system for inspection is to have an inspector view the undercarriage using a mirror. The advantage of an inspector with a mirror is that the inspector can view most regions of the undercarriage if enough ingenuity is used in mirror placement, but occluding objects still cause problems. There is also the issue that an inspector cannot get a complete view of the undercarriage and therefore does not see all areas in context. In addition the inspector is put within range of anything attached to the vehicle.

Any system that uses stationary cameras connected to a stationary inspection system will, by definition, remove the inspector from harm's way. The ability to give a complete representation of the undercarriage would improve upon current inspection techniques. The mechanics of the undercarriage representation would depend on the method used to view the undercarriage.

The view of an undercarriage provided by an inspection bay will not be rivaled by a camera system with anything less than complete 3D reconstruction. Because inspection bays are dangerous to the operator and require a permanent structure to be built, a system that gives close to the same information with no risk and with no construction required would be an improvement.

There are two primary ways to display the undercarriage information. One is to reconstruct the 3D scene so that the inspector can traverse a model of the undercarriage. The other is to present the data to the inspector in a manner that gives the inspector many of the angles that could be achieved with the mirror.

Of these two possibilities, the complete 3D reconstruction should give better results. A complete 3D reconstruction of a vehicle's undercarriage would allow an inspector to view any portion of the undercarriage from any angle and give visual access to the undercarriage better then that achieved with a mirror and approaching that provided by an inspection bay. These 3D models would also be easy to compare with previous models of the same vehicle or a sample vehicle of the same make and model to see whether anything had been added to a vehicle's undercarriage. There are two downsides to this technique. The first is that current 3D reconstruction techniques will quite often create visual artifacts in regions of occlusion since the reconstruction algorithm has no information from which to create the 3D model. The wrongness of these artifacts to the human eye tend to draw a viewer's attention, distracting the inspector from other regions of the undercarriage. In addition the regions where the reconstruction fails are those in which items are most likely to be hidden as they are less obvious to inspection because of their occlusion. The artifact could hinder an inspector's ability to focus on a region that might have been viewed using a mirror or a different camera technique.

A more serious concern is the time required to create the 3D reconstruction. Usually it is possible to reconstruct an object simply by taking several images from different angles, the larger the number of different views the better the reconstruction. In order to create a 3D reconstruction, multiple views of all portions of the undercarriage are required. If the undercarriage were far enough from the cameras to allow complete images of the undercarriage to be taken this would not be a problem. Because the cameras must be located under the vehicle, the maximum coverage of a camera field of view is less than a 2 foot square. Considering that cars tend to be at least 5 feet wide and 10 feet long that means is would take a minimum of 3 images across and 5 images along the length of the vehicle to get a single view of the undercarriage. This means that there would be a minimum of 15 images to get complete coverage of the undercarriage. This does not take into account that the coverage of the undercarriage will not see every portion of the undercarriage from the same view. Let us consider for the moment that these 15 images can provide one of the perspective views necessary for 3D reconstruction. Requiring 15 images per view for the reconstruction makes data processing very time consuming. If the quality of the eventual 3D construction needs to be improved either more views or more images per view would be necessary. Each would increase the time taken to integrate the images, the first linearly and the second by the square of the image count. The only way to speed this process is by using more computer resources, so reconstruction does not appear to be a viable solution. The above argument against full 3D reconstruction has not been proven, but was assumed true when the problem and solution were initially discussed.

The other solution mentioned is to present 2D images that give the inspector 3D information. The advantage of 3D over 2D is that a 3D model can be rotated and zoomed, allowing the viewer to look around objects. To give the same functionality a 2D model must contain the same information and present it in a way that an inspector can interpret. The proposed solution is to create complete views of the undercarriage from different perspectives and then present

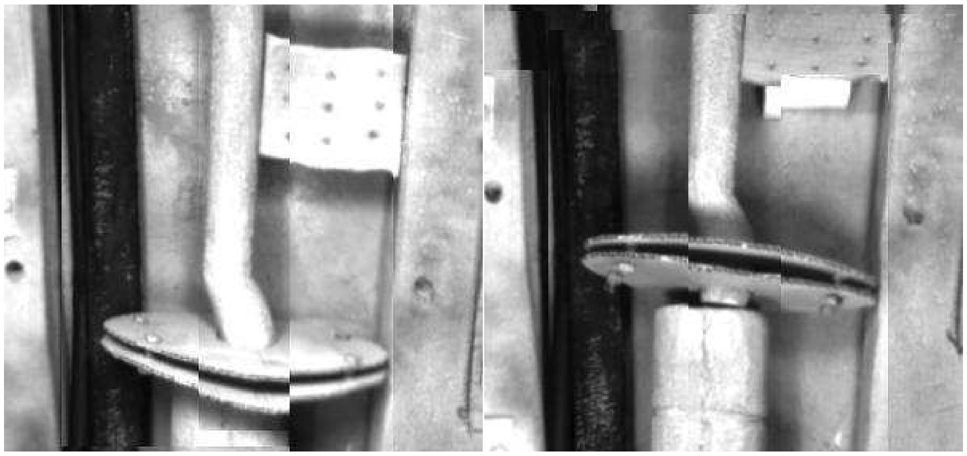


Figure 2: The two images show the same view from two different angles. These two images demonstrate that 2D images can be used to represent 3D information.

these views to the inspector. Figure 2 shows a pair of perspective views that illustrate this point. 3D information is needed to show what can be found behind occluding objects and to allow an inspector to look at an object from a different angle if something looks suspicious. Figure 2 shows how a pair of 2D images gives a view under the square in the upper right corner to reveal the object below it. It also shows how different perspective views allow one to look at the vertical object surrounding the muffler pipe from different angles, giving 3D information from 2D images.

Most of the 3D information in a scene can be presented by using a series of perspective views in this manner. An advantage of this scheme is that it should run nearly in real time because of the easy parallelization of the mosaicing techniques used to generate these perspective views. In addition, mosaicing techniques work well with large numbers of images. Disadvantages are that there will not be a complete 3D model for the inspector to view and that, as a result of irregularities in the mosaics caused by camera perspective, automatic detection will be difficult to implement.

There are three possible methods for getting the data for mosaicing. The first is to have a platform over a rectangular array of cameras that a vehicle drives onto. The second is to have a platform that the vehicle drives onto and remains stationary while a strip of cameras slides under the vehicle body, much like a flat-bed scanner. The third is to have a line of cameras stationary within a platform and to have the vehicle drive over them, like a scanner where the paper moves instead of the scanning device.

The array of cameras solution has the advantage of no moving parts. Also, the fixed positions of the cameras permits the location of each image to be known in relation to all the other images, making for easier matching. The chief disadvantage is that it requires a huge number of cameras (on the order of 2000) because of the small separation required between cameras (to be discussed later). The platform would also be difficult to transport.

The advantage of the moving camera setup is that the images can be taken at predefined locations, much like the camera array, which will help the matching process. It also allows fewer cameras to be used since the cameras can be moved to any location under the vehicle. This solution is, thus, less expensive and mosaicing is as easily implemented as for the camera array. The disadvantage is that it requires moving parts. The cameras will have to be moved under the vehicle body, which is an area where grime is likely to build up. The dirt is likely to get into the equipment that moves the cameras and breakdowns are likely to occur. This also makes the system more difficult to transport and calibrate on setup. The original guidelines called for the system to be transportable and have no moving parts.

The stationary camera-moving vehicle solution has none of the problems of the moving camera system. The lack of moving parts means that jamming is not an issue, so the camera set to be smaller and more portable. A disadvantage is that it must be able to account for drivers who cannot drive at a consistent speed over the cameras. The matching techniques must be able to accommodate data inconsistencies. It will also require more cameras than a moving camera system, because a moving camera system could theoretically use a single camera to take all of the required images. More cameras make the system more expensive than the moving camera technique, although less so than the rectangular camera array, which more cameras. A vision system for this design was chosen to be implemented and tested because of it having the best combination of probable run time, price, and mobility.

2.2 Vision problem description

Most current vision systems that match images assume either a single camera being moved or multiple cameras viewing the same object. These are both easier for mosaicing than the UVIS configuration. The advantage of matching multiple images from a single camera is that images have similar lighting and distortion characteristics even when camera movement is present. Additionally, all images taken with a single camera will have the same intrinsic parameters, lens distortion, and, when the images are taken close together in time and space, color constancy. The multiple cameras needed within UVIS eliminate these advantages for the matching process. Rectification can use the intrinsic camera parameters to eliminate most of the lens distortion, but no rectification is perfect. Working with UVIS, it has been found that matching algorithms function better on multiple images from the same camera than on multiple rectified images from different cameras. Color constancy is another major problem for matching in UVIS. When a single camera is moved 3 inches between images, it can be assumed that the colors are relatively similar between images because so much of the scene is similar. When two UVIS cameras are placed 3 inches apart color constancy cannot be assumed: no two CCDs have the same color response, and well-calibrated CCD cameras are, so far, not an option for UVIS. Regarding the latter point, well-calibrated CCD cameras are more expensive, which becomes significant when more than 30 cameras are required. In addition, the synchronization of 30 cameras at 30 Hz between multiple image acquisition machines is a significant problem. The cameras currently used in UVIS automatically synchronize, a capability not available on well-calibrated CCD cameras.

As compensation, there is typically an 80% overlap between images to be matched for mosaicing. Also, the difference in the incident angle to a common element is the scene between adjacent cameras tends to be small when the separation between camera positions is relatively small compared to the separation between camera and scene. (A typical example of this comes from aerial image mosaicing. The images of the tree cover taken from an airplane may have been taken 100 feet apart, but because the airplane is flying at over 1000 feet the incident angles to objects within the scene are similar.) The large overlap and similar viewing angle result in great similarity of images to be matched in typical mosaicing situations. Matching is, then, relatively easy despite the between-camera differences in distortion and color. The matching problem in UVIS is more difficult because the short distance between scene and cameras creates significantly different viewing angles for points in the scene as shown in Figure 3. A 3 degree incident angle difference between images may not seem to be significant but Figure 4 shows the problem. Every deep portion of the un-

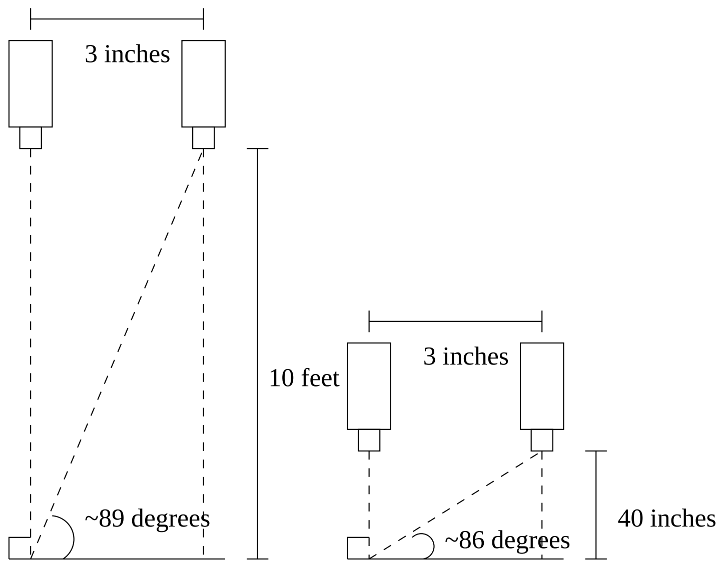


Figure 3: The figure, not drawn to scale, shows the difference in angle of incidence based on camera to image separation. The left image shows relative camera separation to camera scene separation more typical when matching and mosaicing, the right image shows the situation found within UVIS.

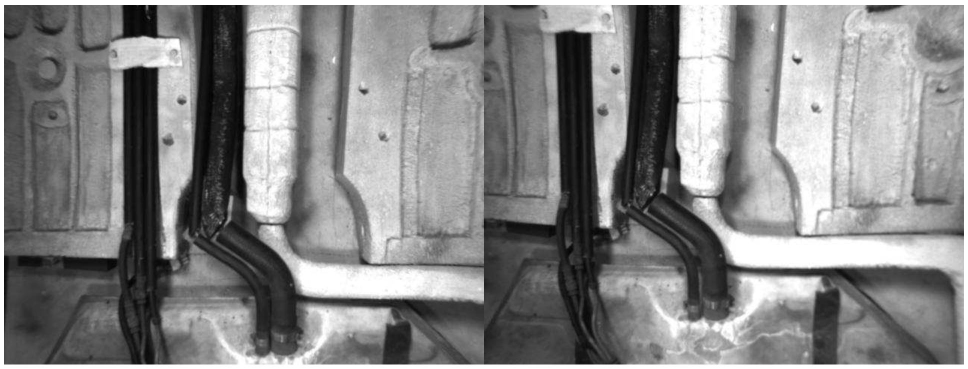


Figure 4: Two typical images of the vehicle undercarriage taken with 3 inch separation that are used to mosaic.

dercarriage looks different in the two images in Figure 4, with particularly large discrepancies occurring in the area around the muffler pipe that runs through the center of the images. The viewing angle difference showing in Figure 3 means that in UVIS a large number of side views of vertical elements will be seen. The number of these side views seen will be far larger then the then would be typical of image to mosaic. Regions with significantly different incident angles, as seen in Figure 4, appear as regions in which no match is possible because the side of a vertical region is visible in one image but appears only as a line in the other.

In addition to these matching problems, UVIS also has to match in two orthogonal directions to create images of the entire vehicle undercarriage. A typical mosaicing system matches a given image only to the images that come before and after it; that is a typical system will mosaic images taken in a line. Within UVIS we create a plane of images to give a complete view of the undercarriage. Therefore each image needs to be mosaiced not just to the images in front and back of it in relation to the motion of the vehicle, but also to those on either side of it, perpendicular to the motion of the vehicle as shown in Figure 5.

This problem is extremely difficult and time consuming to solve, as there are approximately 25 images across and 60 images down the length of a typical undercarriage, giving roughly 1500 images to be matched. An approach to this problem is to match first in one direction and then match the generated mosaics in the other direction, giving the effect of matching in two directions without requiring as much processing time.

The UVIS domain has a significant advantage in that the cameras are known to be next to each other in line, so it is easy to create epipolar lines that will significantly improve the speed of the matching process. These epipolar lines can be more easily used in the crosswise mosaicing than in the lengthwise mosaicing.

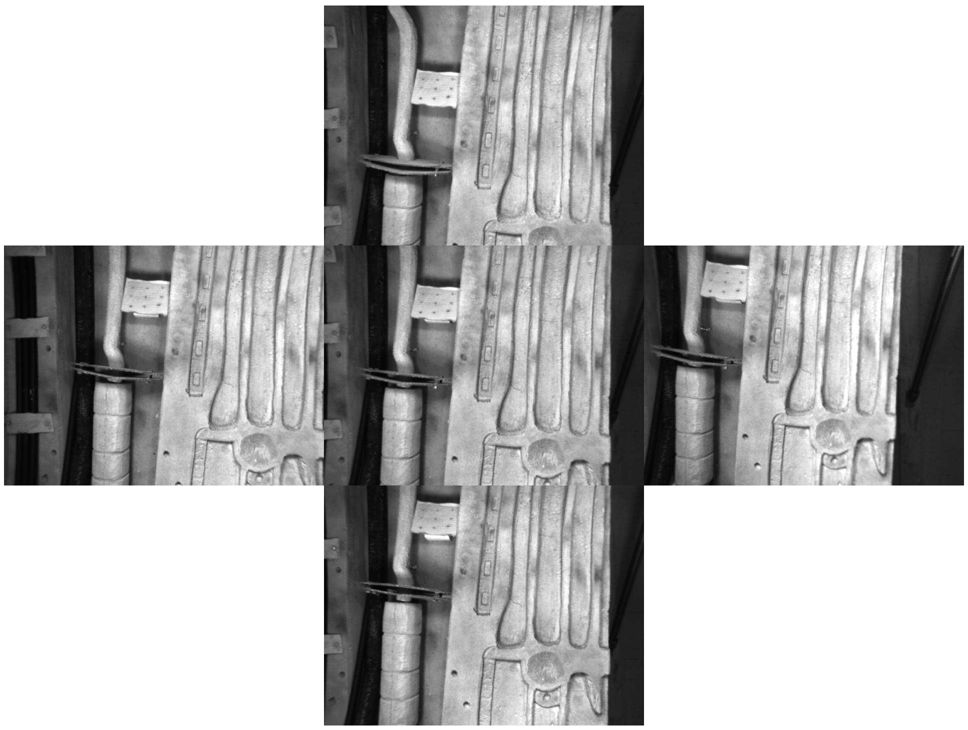


Figure 5: This figure shows the central image and the 4 surrounding images that it must be matched and mosaiced to.

2.3 Design considerations

As described previously [11], a 3-inch camera separation was empirically determined to create acceptable mosaics. Achieving comparable image separation in the direction of vehicle motion requires a vehicle speed of approximately 5 mph over the cameras at a frame rate of 30 frames per second, the maximum speed of the cameras. The system can robustly create mosaics at higher vehicle speeds but a speed of 5 mph returns good quality mosaics. The ability to handle faster and varying speeds is important because it is not realistic to believe that drivers can maintain a constant speed while driving over the cameras.

The negative effect of perspective distortion can be reduced by increasing the distance between the cameras and the undercarriage to reduce the motion parallax. Increasing this distance is possible by placing the cameras lower, in a ditch, or by elevating the platform, effectively creating a large speed bump. We suggest folding the optical path of the cameras (Figure 6) in future implementations.

The next design consideration is the camera field of view. A smaller field of view means smaller perspective distortion and motion parallax, which makes it easier to match and mosaic between images. As the field of view is increased,

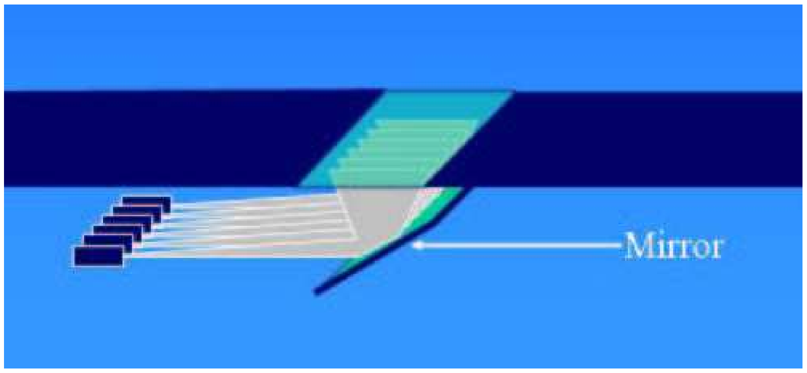


Figure 6: Camera position within the platform used to bend the optical path

the matching and mosaicing processes become more difficult but the information from the perspective mosaics increases. The information increases because the greater the field of view the greater the angle from which the perspective mosaics are generated, increasing the inspectors ability to look around occluding objects. A field of view of 72^o was chosen because it created mosaics with good perspective without creating parallax sufficient to break the matching function.

Several environmental issues are associated with an under-vehicle inspection system. The major ones are lighting the undercarriage, dealing with weather as it affects the undercarriage, and eliminating grit buildup on the surface of the camera house. These issues have not been addressed as yet and are not discussed in this paper.

2.4 Proposed Solution

The proposed solution to this problem is to have a line of cameras that continuously take images as the vehicle drives over them (Figure 1). This design appears to be the best combination of price, mobility, and robustness. The two-part mosaicing scheme of first mosaicing crosswise or perpendicular to the vehicle motion before mosaicing lengthwise or in line with the vehicle motion seems to be the only feasible solution and it can use the epipolar geometry implicit within the system.

3 System description

A high-level view of UVIS can be seen in Figure 1. The platform that the vehicles drive over appears in Figure 7. As mentioned, the camera separation is 3 inches. The transparent surface has yet to be determined but glass or Lexan is most likely to be used. Images are taken constantly as the vehicle drives over the platform. These images are taken by a series of image-acquisition machines (Figure 8; discussed in section 3.1) and transfered to a series of computers that

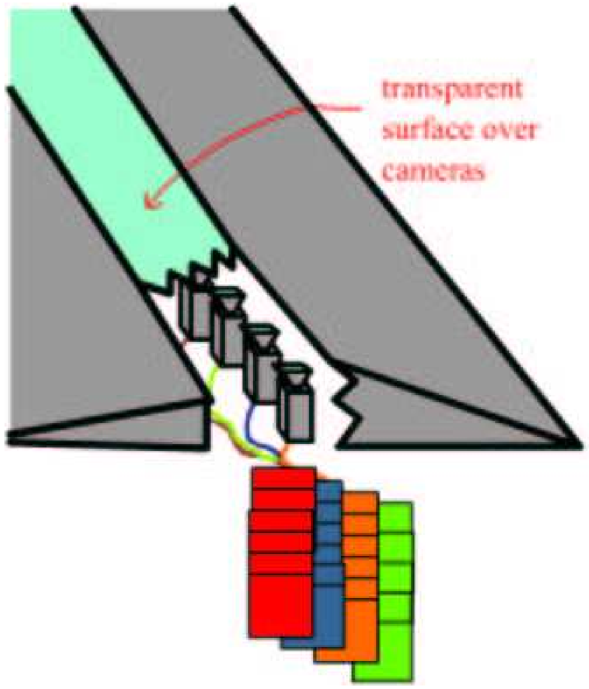


Figure 7: Camera platform; the bent optical path is not shown in order to simplify the image

perform all actions associated with the crosswise mosaicing (discussed in section 3.2.4). These crosswise images are then transfered to the lengthwise mosaicing computers (discussed in section 3.3) before being passed to the GUI (discussed in section 3.4). Figure 9 shows how the steps described in section 3 take the images and first mosaic them crosswise before mosaicing them lengthwise to create a complete single image.

Section 4 describes results from the model of the system created within the lab environment. These results are discussed in Section 5 and conclusions drawn in Section 6. Appendix A contains details of the experimental model and Appendix B contains sample crosswise mosaics generated as well as the disparity images containing the match information. Appendix C contains the description of a similar UVIS system with slightly different mosaicing algorithms that has been studied to determine a computational model capable of running it in real time.

3.1 Acquisition

Synchronized images are continuously taken as the vehicle drives over the platform. The system will eventually include more than 30 cameras, which is why the speed of the mosaicing process is important. The images are taken us-

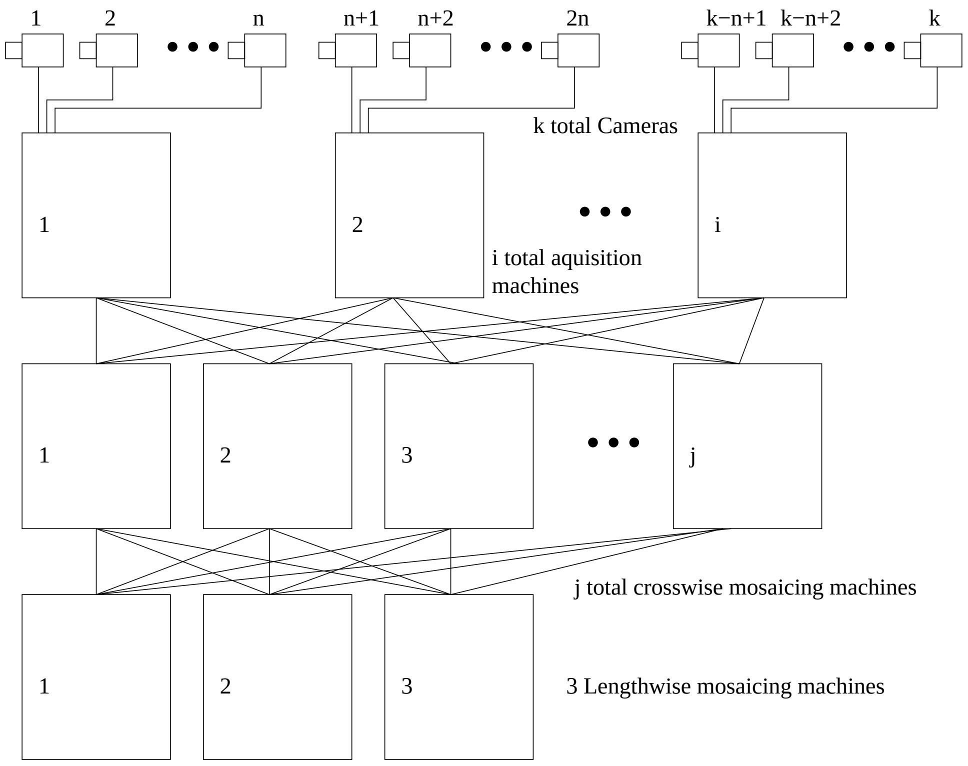

1
2
n
n+1
n+2
2n
k−n+1
k−n+2
k
k total Cameras
1
2
i
i total aquisition machines
1
2
3
j
j total crosswise mosaicing machines
1
2
3
3 Lengthwise mosaicing machines

Figure 8: Diagram of the UVIS operation

ing Point Grey's black and white DragonFly firewire cameras. These cameras were chosen because they require no frame grabbers, auto synchronize, and take 640x480 pixel 8-bit gray scale images.

3.2 Crosswise mosaicing

3.2.1 Rectification

A rectification step is performed on each image before mosaicing to geometrically correct the images by removing most of the distortion intrinsic to the camera lenses. Each camera has been calibrated to determine both intrinsic and extrinsic camera parameters. These parameters are used to modify each raw image during the rectification step. The rectification and calibration model used are those of Kovalenko [15]. The rectification process does not remove all distortion but will remove most of it, making the images similar enough to mosaic. An example of the effect of the rectification appears in Figure 10. The effect of the rectification process can be seen in the difference image in Figure 11.

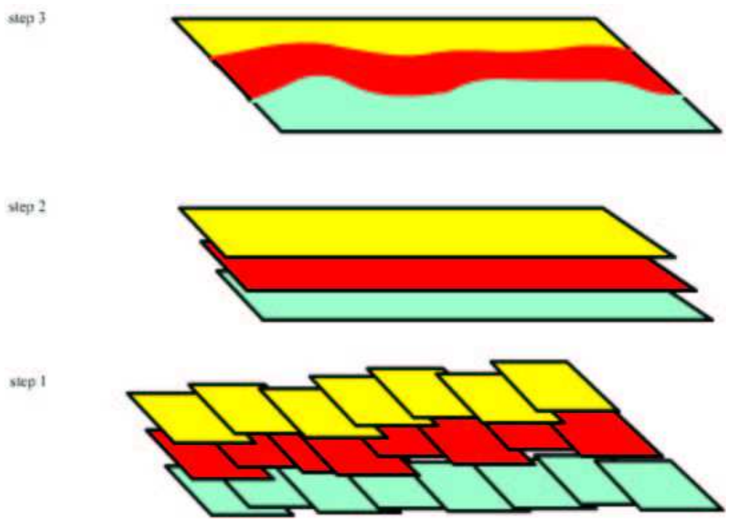


Figure 9: Mosaic building step

3.2.2 Normalized cross-correlation matching

The crosswise matching process used in UVIS relies heavily on normalized cross correlation, which is the method used to match points in our algorithm. Other matching methods, including sum of squared differences with and without image brightness normalization, were also tried, but the normalized cross-correlation matcher (NCCM) was empirically determined to return the best results.

NCCM is an established method within the field of computer vision for matching regions [16]. It uses two equal-size rectangular portions of images. The first step is to determine the average pixel value for each rectangle:

$$\overline{A} = \frac{\sum_{i=1}^{N} Image_A[i]}{N} \tag{1}$$

$$\overline{B} = \frac{\sum_{i=1}^{N} Image_B[i]}{N} \tag{2}$$

$\overline{A}$ and $\overline{B}$ are average pixel values in images A and B within the correlation window and N is the size of the correlation window. N will be defined in the following sections. $Image_A[i]$ and $Image_B[i]$ refer to specific pixels in the correlation windows within images A and B.

$\overline{A}$ and $\overline{B}$ are then used to determine the variance of each correlation window, Var_A and Var_B:

$$Var_A = \frac{\sum_{i=1}^{N}(Image_A[i] - \overline{A})^2}{N-1} \tag{3}$$

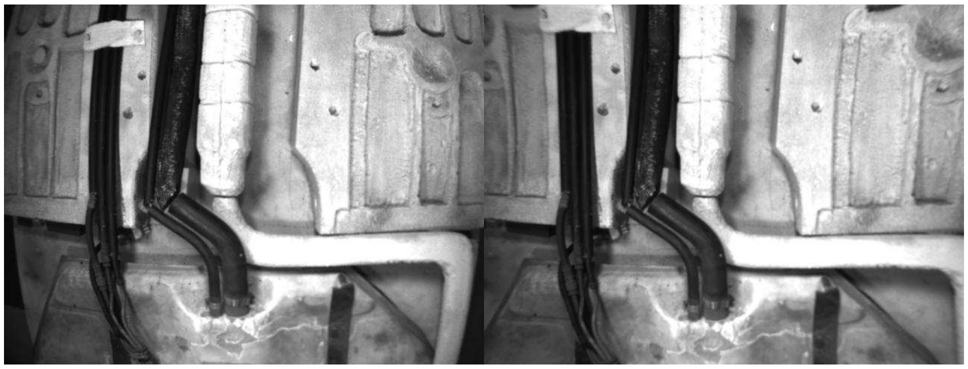


Figure 10: The image on the left is the raw image and the image on the right the same image rectified.

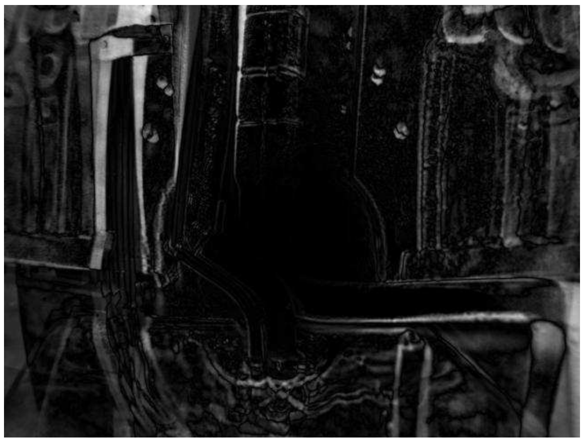


Figure 11: The image is the difference image between the rectified and unrectified images that appear in Figure 10.

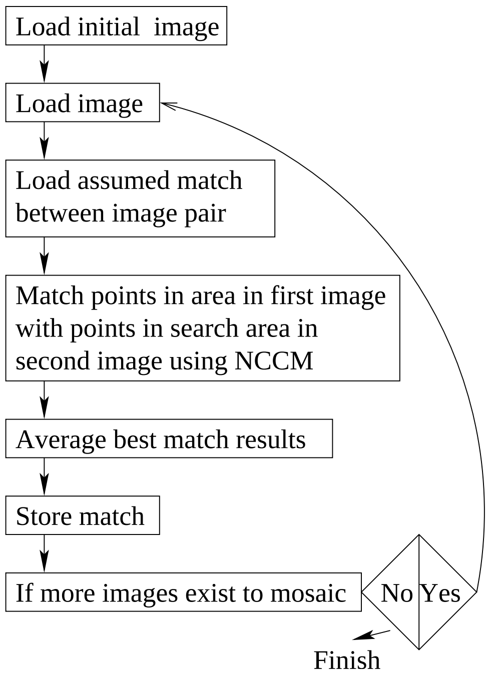

Load initial image
Load image
Load assumed match between image pair
Match points in area in first image with points in search area in second image using NCCM
Average best match results
Store match
If more images exist to mosaic
No
Yes
Finish

Figure 12: Crosswise matching algorithm

$$Var_B = \frac{\sum_{i=1}^{N}(Image_B[i] - \overline{B})^2}{N-1} \tag{4}$$

These are used to determine the match score ρ:

$$\rho = \frac{\sum_{i=1}^{N}(Image_A[i] - \overline{A}) * (Image_B[i] - \overline{B})}{(N-1) * \sqrt{Var_A * Var_B}} \tag{5}$$

As can be seen from 5, the regions with the highest variance will give the most accurate match score. The closer ρ is to 1 the better the match is.

3.2.3 Crosswise match

The algorithm for the crosswise match appears in Figure 12. The match starts at one end of the line of cameras in the platform and continues to the other end, matching images between each pair of cameras.

One of the chief improvements of this new algorithm over the old algorithm is that it uses epipolar geometry between cameras to narrow the search phase of the mosaicing process [11]. The rectification step described above rectifies the images and in doing so creates epipolar lines between the images. The idea behind epipolar lines is that an object in space will appear on a corresponding line in both images. In the context of UVIS where the camera locations are known, this means that everything that appears on one horizontal line in one image will appear on a corresponding horizontal line in the next image assuming there is no occlusion. (The match overlap between cameras is referred to as the horizontal match and the offset in the direction of the vehicle motion is referred to as vertical because when the final results are viewed by the inspector on a computer screen, the sides of the undercarriage are parallel to the sides of the monitor and the front and back of the undercarriage are parallel to the top and bottom of the screen. This makes it logical to call the assumed match between cameras the horizontal match and the offset between epipolar lines the vertical offset.)

There is also have another piece of information to use: an educated guess as to where the match will occur. The educated guess or assumed match is determined in the following manner. The average separation between cameras and undercarriage of vehicles likely to travel over the inspection system is known. Also, the separation of the cameras is 3 inches. With this information and the known focal length of the lenses we can predict where a point along an epipolar line in one image will match along an epipolar line in the other. This gives an assumed horizontal match from which the actual match will vary depending on the actual undercarriage to camera separation of the vehicle compared to the average. This match point can be determined during calibration. At the same time vertical offsets between camera pairs can be determined. Ideally, the cameras will have been aligned perfectly in the vertical direction with the epipolar lines corresponding exactly between cameras. As this will not happen in reality, the vertical offset will need to be determined during calibration by matching points. These match points should give a set of vertical offsets for matches that does not change as the epipolar lines are set, barring physical misalignment of the cameras. As stated above, the horizontal portion of the assumed match can be created for an average vehicle undercarriage to camera separation.

The advantage to using an assumed match point along with epipolar geometry is that it can speed the matching process. If the physical layout is completely unknown, then entire images must be searched in order to match points. If epipolar geometry is known then search for match points need only be made along the epipolar lines and if an assumed match point is known only a limited portion of the epipolar lines need be searched as there are only a few possible offsets from the assumed match because there is a limited range of undercarriage to camera separations.

Given the calibration step described above, the algorithm has assumed matches between the cameras in the horizontal direction. The algorithm must refine the assumed matches to find actual matches. The algorithm begins by determining an area in the first image that is completely overlapped by the second image. This area in the first image is determined by using the camera separation, vertical match, and minimum undercarriage to camera distance of a vehicle to be inspected. The algorithm then finds a match between every point within this area in the first image and the second image.

The algorithm matches using an NCCM match window. The epipolar geometry and assumed matches are employed here. Instead of matching each point from the first image with every point in the second image, which would be time consuming, we match the points only within a limited search window. The search window can be only a few pixels tall, because we know the match will occur on the epipolar line that corresponds to the epipolar line on which the pixel to be matched is located. A range of pixels above and below the epipolar line is employed in case vibration from vehicles traveling over the platform has slightly misaligned the cameras since calibration. The width of the match window is determined by the possible undercarriage to camera separations of the vehicles to be inspected. The possible clearances determine how many pixels to the left and right of the horizontal assumed match must be searched to determine the exact match between cameras. The match locations for all of the pixels from the area in the first image are then averaged to determine the exact match between images. As has just been shown, the assumed match allows a significant reduction in the amount of the second image that needs to be searched in order to determine a match. For testing purposes these matches are used to create a disparity image that is stored for future reference and for determination of algorithm match success; samples of these appear in Appendix B.

A match window of 37x37 pixels is used in NCCM. This window size was empirically determined from the images in Figure 13, which shows sample matches for different window sizes. A 37x37 pixel window was chosen as the smallest window that gave smooth results. A larger match window would have further smoothed the data, but run time depends on the size of the window squared so a smaller size is advantageous. The 37x37 pixel window size is useful for proof of concept but a smaller size might be viable within an actual system.

This cross correlation is extremely time consuming because NCCM is $O(n^2)$ and the search process described above is $O(n^4)$, making the entire process $O(n^6)$. For the eventual system the time for this process will be reduced by matching only a subset of the area in the first image, by limiting the size of the match window, and by limiting the size of the search window. The algorithm for this proposed faster system appears in Figure 14. For the present testing, none of these improvements has been made to the speed of the process and the algorithm from Figure 12 is used. The absence of speed improvements within the test runs gives to more complete results from this testing phase.

Obtaining the match by averaging the best results does not truly solve the match problem; it only accounts for translation between images and ignores rotation and scale change. Later versions of the system will be able to account for the latter based on the matches found, but for this proof-of-concept paper where it is known that no rotation or scale changes exist, this simpler rotationless match is used. The matching process stores the match offsets between images in a file that is used by the mosaicing process.

3.2.4 Crosswise mosaicing

The mosaicing process takes the output match scores from the matching program and uses these to construct three crosswise mosaics. These mosaics form

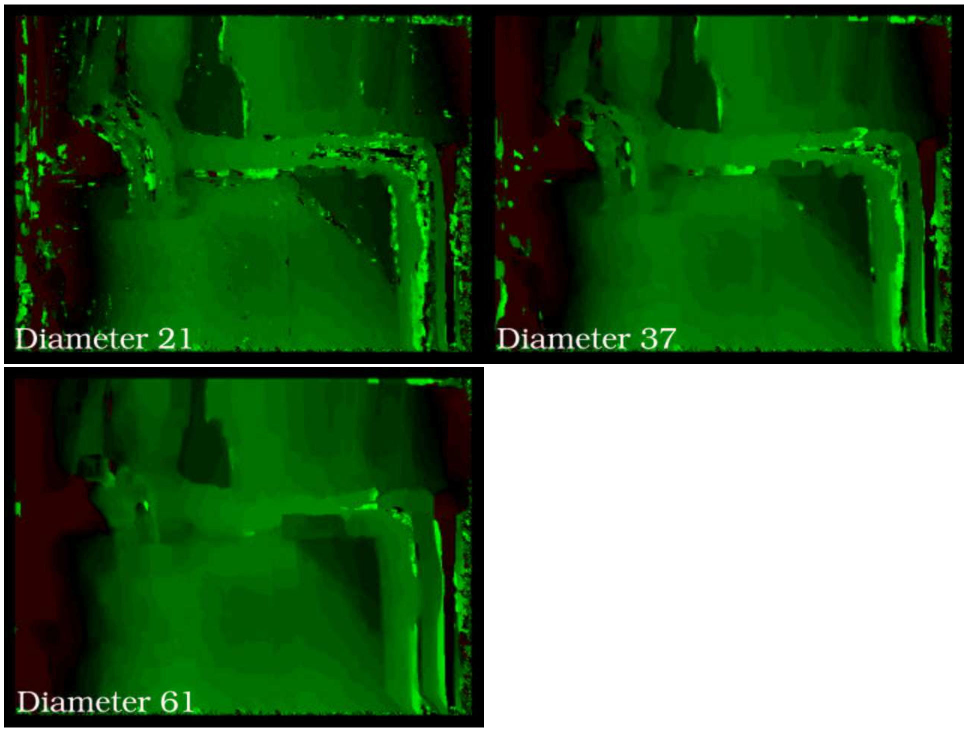


Figure 13: Disparity maps generated with square match windows of diameter 21, 37, and 61. These disparity images only contain the X-direction disparity as the Y-direction should not be a factor in the actual system. The disparity values are expanded to fill the entire brightness spectrum so that the data may be viewed

the basis of the perspective mosaics used to give the 3D information. A mosaic is created from individual images that are stitched together to create a continuous image. Controlling where the stitched portions of the images are located in the original image determines the perspective view of the created mosaic. Figure 15 shows how the slices from the original images can be taken to create left, center, and right mosaics. The left and right mosaics are both created by stitching together the regions from the extreme edge of the original images. The center mosaic is created by stitching together the regions from the center portion of each image.

The mosaicing process used currently does not allow for any rotation between images to be stitched as was present in the previous version of this system used to test the concept [11]. Just as rotation will be added to the matching process later, the rotation and scale shifts necessary for seamless mosaics will be added later to the mosaicing algorithm.

After completion, the crosswise mosaics are transfered to a computer that

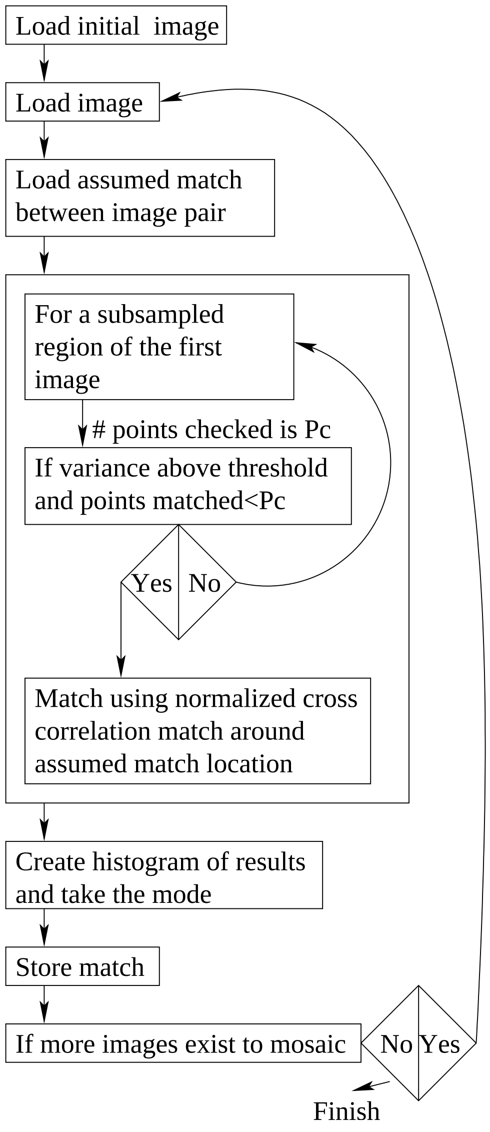

Load initial image
Load image
Load assumed match between image pair
For a subsampled region of the first image
# points checked is Pc
If variance above threshold and points matched<Pc
Yes
No
Match using normalized cross correlation match around assumed match location
Create histogram of results and take the mode
Store match
If more images exist to mosaic
No
Yes
Finish

Figure 14: Crosswise matching algorithm with speed improvement

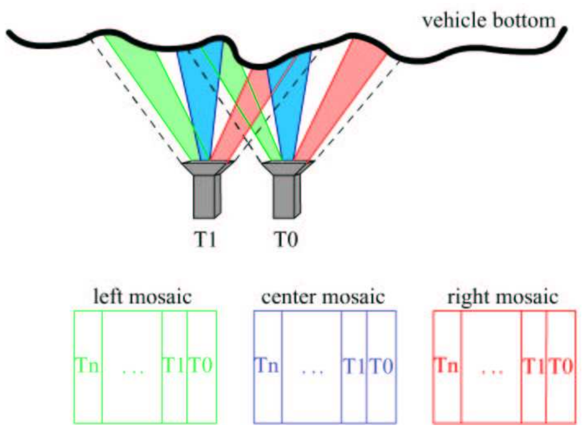


Figure 15: Example of how slices are taken from the original images to create perspective mosaics

will perform the lengthwise mosaics.

3.3 Lengthwise mosaics

3.3.1 Lengthwise matching

The lengthwise matching function uses the same basic principles as used for crosswise mosaicing, the main difference being that there are no assumed matches in the lengthwise direction. Lengthwise assumed matches cannot be generated because they would require the speed of the vehicle over the platform to be constant, which cannot be assumed.

For the purpose of testing, the lengthwise matching process has been implemented in the following slow but comprehensive manner. A series of match windows are obtained from the first image, and each of the windows is matched across a corresponding strip in the next image as shown in Figure 16. NCCM is again used to determine the match scores between windows. The best matches from each of these searches are then averaged to determine the match offsets between images. As with crosswise matching, the lengthwise matching process ignores possible rotation and scale change information while matching between images. Again, a later version of this system will require the implementation of these features, but this is merely a test of the concept on real data and these features have not yet been implemented. For testing purposes, disparity images are also generated from this match information and stored for future reference. This process is shown in Figure 17. Although it is slow because of the $O(n^6)$ run time, it returns detailed match results.

As the lengthwise match algorithm described above is $O(n^6)$, a faster method of match determination is proposed. This lengthwise match algorithm has the same basic methodology as the crosswise match, as can be seen by comparing

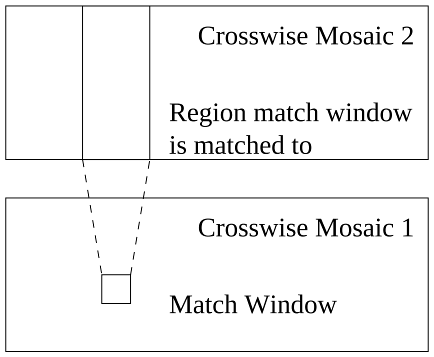


Figure 16: The match process takes each section of the first crosswise mosaic and matches it within a strip of the second crosswise mosaic.

Figure 14 with Figure 18. Both algorithms function similarly to find the exact match location; they differ in that the lengthwise match must find an initial match instead of loading it from a file as the crosswise match does with the assumed match. The faster algorithm finds its assumed matches by initially matching a few select points in depth as described above and as shown in Figure 16. It then uses these matches in the same manner as the assumed matches from the crosswise algorithm as a basis for determining the true match between images.

Separate matching functions are performed on each of the three crosswise perspective mosaics created. This separate matching will ensure that each of the final perspective mosaics generated has good continuity within itself because the images in each will have been matched specifically to those stitched around them.

3.3.2 Lengthwise mosaicing

The lengthwise mosaicing process functions the same as the crosswise mosaicing process. Only center mosaics are generated from the left and right crosswise mosaics which creates complete left and right perspective views of the undercarriage as shown in Figure 19. As both mosaics are created in the same manner Figure 19 shows only the creation of the left perspective mosaic. Three separate lengthwise mosaics are generated from the center crosswise mosaics. The perspective mosaics are from the top, center, and bottom to create the three remaining perspective views as seen in Figure 20. The same mosaicing function used to generate center mosaics is used to generate center and perspective mosaics together with just a parameter change. As above, the mosaicing function will need to be altered in the next version of the algorithm so that it can handle more than simple translation. The generated mosaics are then transfered to a computer that runs the GUI to display them.

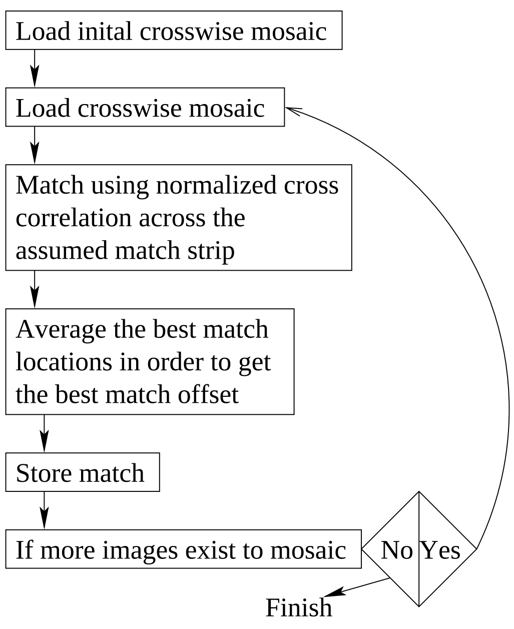

Load inital crosswise mosaic
Load crosswise mosaic
Match using normalized cross correlation across the assumed match strip
Average the best match locations in order to get the best match offset
Store match
If more images exist to mosaic
No
Yes
Finish

Figure 17: Lengthwise matching algorithm

3.4 GUI

3.4.1 GUI design

The GUI is the heart of UVIS. It allows the inspector to view the five perspective views in concert to present the 3D information. An image of the GUI being used appears in Figure 21. Six small images are displayed on the left and the view of the entire undercarriage is on the right, which enables the operator to inspect the complete undercarriage. As the operator uses a cursor to move a box around the right-side image, five corresponding images appear as a cross on the left side. These five images are windows into the five perspective mosaics. By moving this box, which is red in the Figure 3.4, the operator is able to look around occluding objects within the undercarriage and get 3D information from the 2D representation. Because this inspection does not always yield satisfactory results, it is also possible for the operator to view the original images from which the mosaics were generated. The operator can do this by selecting a region on the right image for display of the original image. This original image is displayed in the upper left hand corner of the GUI. As this upper left hand corner of the GUI is not large enough to display the entire image, only a portion is displayed, but the inspector can use the mouse to scroll around and view the entire image. The location from which the original image is taken from is displayed as a green box in the image on the right of the GUI. A sample of the GUI can be downloaded from the project web page [14].

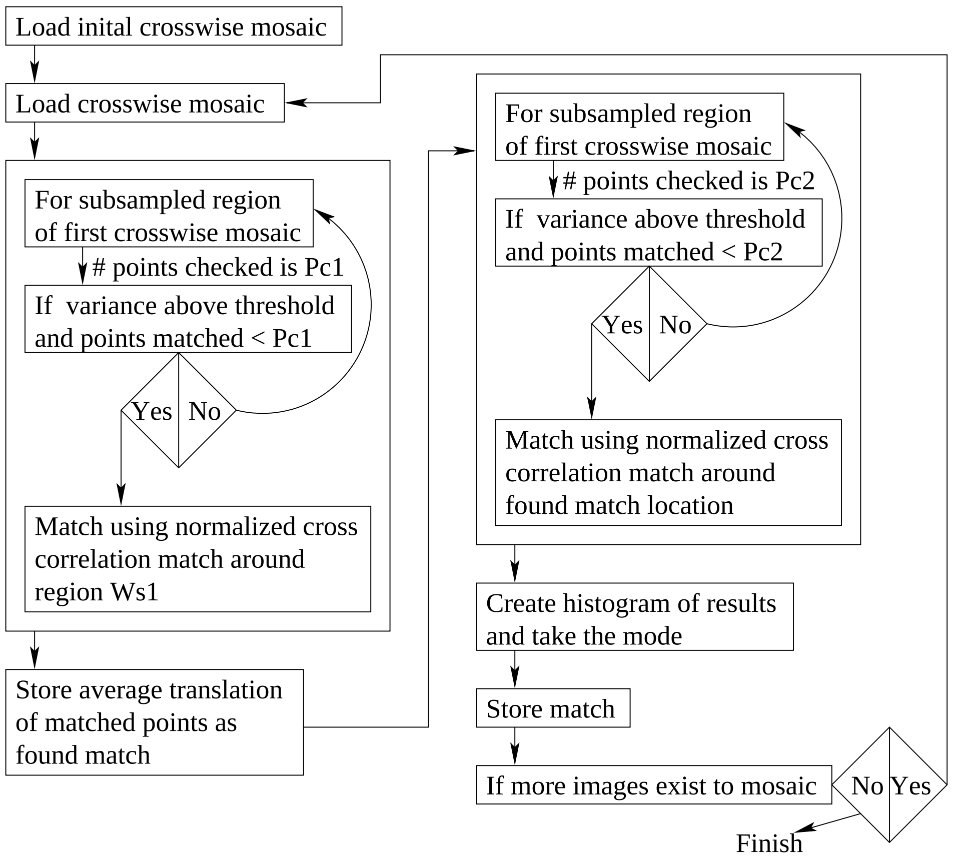

Load inital crosswise mosaic
Load crosswise mosaic
For subsampled region of first crosswise mosaic
# points checked is Pc1
If variance above threshold and points matched < Pc1
Yes
No
Match using normalized cross correlation match around region Ws1
Store average translation of matched points as found match
For subsampled region of first crosswise mosaic
# points checked is Pc2
If variance above threshold and points matched < Pc2
Yes
No
Match using normalized cross correlation match around found match location
Create histogram of results and take the mode
Store match
If more images exist to mosaic
No
Yes
Finish

Figure 18: Lengthwise matching algorithm with speed improvements

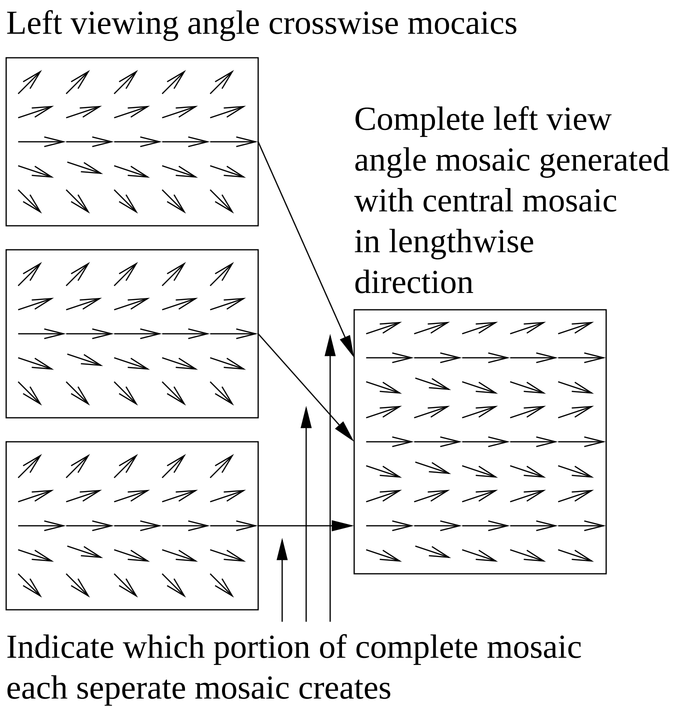

Left viewing angle crosswise mocaics
Complete left view angle mosaic generated with central mosaic in lengthwise direction
Indicate which portion of complete mosaic each seperate mosaic creates

Figure 19: Left perspective crosswise mosaics being mosaiced into the left perspective mosaic

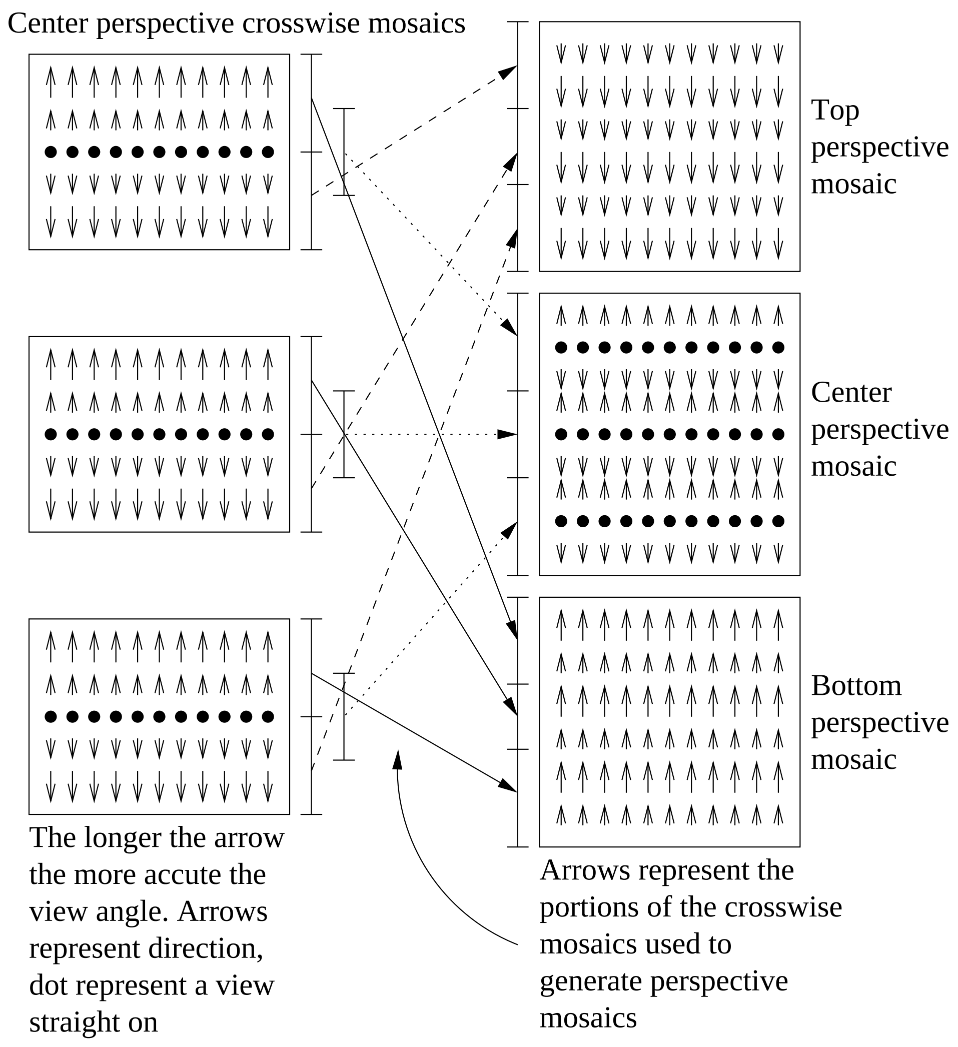

Center perspective crosswise mosaics
Top perspective mosaic
Center perspective mosaic
Bottom perspective mosaic
The longer the arrow the more accute the view angle. Arrows represent direction, dot represent a view straight on
Arrows represent the portions of the crosswise mosaics used to generate perspective mosaics

Figure 20: Center perspective crosswise mosaics being mosaiced into top, center, and bottom perspective mosaics

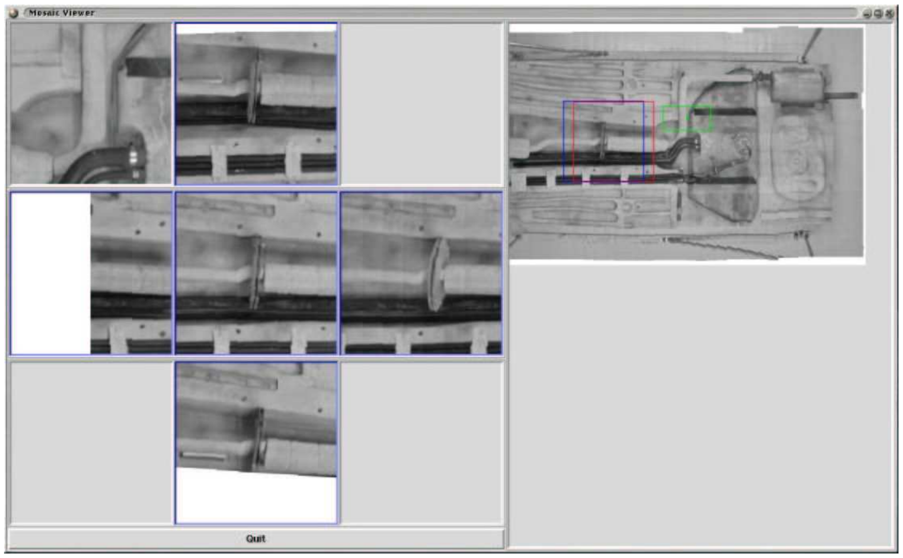


Figure 21: The GUI used to display the generated mosaics

4 Results

The results of this study fall into two categories: the results of the crosswise mosaic generation and the results of the lengthwise mosaic generation that builds on them. Only the images associated with the center mosaic are discussed because the problems found with the center images are the same as those for the perspective mosaics. Though the errors are more pronounced in the perspective mosaics, the concept can be proved or disproved within the confines of the center mosaic.

4.1 Crosswise mosaic generation results

The complete set of crosswise mosaics generated are given in Appendix B. In this section we will discuss the results by looking at specific regions cropped from the mosaics that appear in Appendix B.

The first result to discuss is a breakdown in the matching algorithm, an illustration of which can be seen in Figure 22. Inspection of Figure 22 makes it clear that the match found by the algorithm is not correct and that greater overlap of images would have created a better mosaic. No system will ever function perfectly and Figure 22 shows an instance in which failure occurs within this system.

Another problem found in the resulting mosaics is caused by the perspective differences between camera views to be mosaiced. In Figure 23 the top arrow points to a region that is aligned correctly but that does not create a smooth mosaic due to the perspective differences within the images. The different angles

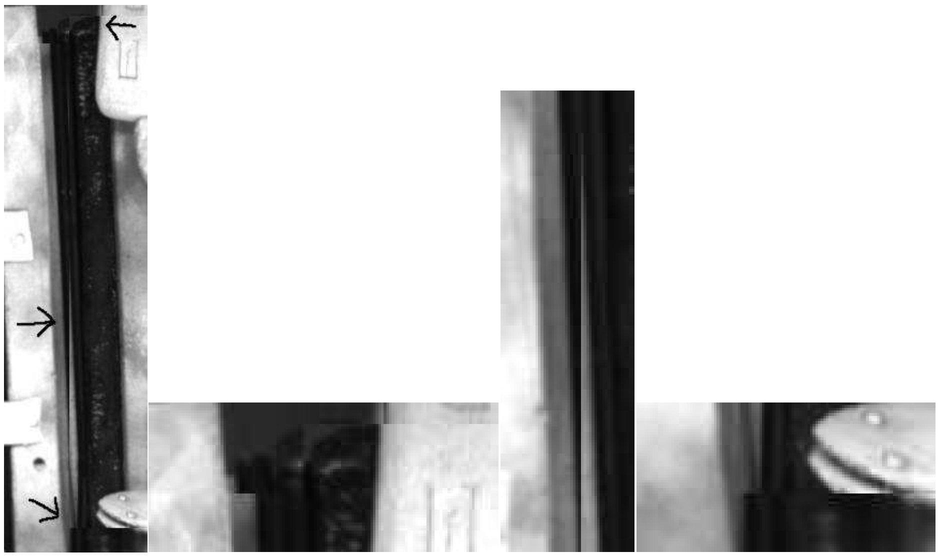


Figure 22: This portion of one of the crosswise mosaics illustrates failure of the matching algorithm to find a correct match. The arrows point to specific regions where the human eye can see that the images should have been overlapped more in order to get a correct match. Enlargements of the arrowed regions appear on on the right.

into the trench in the vehicle undercarriage show different amounts of the side wall of the trench. There is no way to make a completely successful match because the images contain different data within the same region. The specific error shown by the top arrow in Figure 23 will not occur in an actual system because the vertical misalignment of the cameras will not be as great, it only occurs here because the cameras were misaligned within the system test bed. The error identified by the bottom two arrows is not as clear, but is caused by the same perspective shift as the top arrow error. The left portion of the image in Figure 23 comes from an image that shows a large portion of the side of the trench. The right portion from an image that shows very little of the side of the trench. When they are mosaiced together a line appears where the different perspectives meet. No information is lost but the mosaic created will not be as clear as if there was no perspective difference.

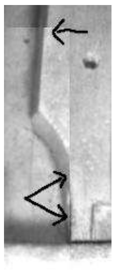


Figure 23: The arrows show where perspective differences within the original images cause a misalignment of portions of the resulting mosaic.

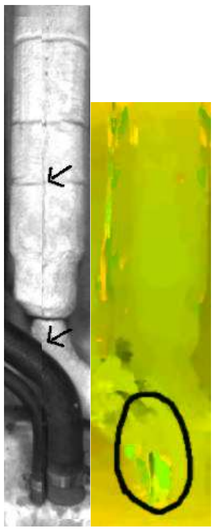


Figure 24: The image on the left shows a cut line within a mosaic. The top arrow points to a location with a match and the bottom arrow points to a location with a mismatch. The image on the right displays the disparity map of the matches used to create this portion of the mosaic. The dark vertical region in the circled portion corresponds to the lower arrow in the left image.

Errors were also found that resulted from a match between images that was correct for part, but not all, of the images to be mosaiced. Figure 24 shows a mosaic that was created by a match that is not correct down the entire cut line. The top arrow in the left image in Figure 24 shows a region where the match is correct, but the bottom arrow shows a region where there is a mismatch, as shown by the lack of alignment of the pipes across the cut line. The disparity image on the right side of Figure 24 tells the whole story. The area corresponding to the simulated muffler is a relatively uniform color in the disparity image, indicating that the match is the same through that region in the two original images. The circled region at the bottom of the disparity image shows a location where the match is nonuniform, as illustrated by the sharp, localized color change. This color change, or different match, means that the two images will not align along the same line whereas the regions at the top will. There is no solution to this problem short of warping the images, which both introduces errors and lengthens run time.

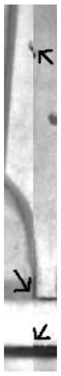


Figure 25: The arrows illustrate where the errors occur due to a difference in scale between the original images.

There were also errors that occurred because of differences in scale between the original images. Figure 25 shows a cut line between a left image that is more zoomed than the right image. The middle arrow points to alignment while the top and bottom arrows point to misalignment. The manner of misalignment, the left image being higher than the right at the top arrows and lower than the right at the bottom arrows, shows that the left image is larger, that is, more zoomed in. This type of error has occurred within the results primarily because the cameras were only hand aligned with visual focusing and positioning. In an actual system the alignment and focusing of the cameras would be more precise, eliminating most of this type of error. As this will not eliminate all such errors, an implementation of a scaling algorithm into the mosaicing process will be necessary. The ability to handle scale changes will make the algorithm more similar to that found in [11].

Another type of error can be created by the rectification process. Figure 26 shows a mosaic in which the top and bottom arrows point to misalignment and the middle arrow points to alignment. Unlike what is seen in Figure 25, the top and bottom arrows in Figure 26 both show the image on the right of the cut line to be higher than that on the left of the cut line. This suggests that some warping of one of the two original images has occurred that has changed the ratio of the distance from the top to middle arrow to the distance from the middle to bottom arrow enough so that proper alignment cannot occur even with scaling. The only warping that has been performed on these images has been the warping within the rectification process, so the errors seen in Figure 26

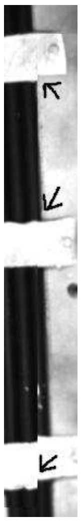


Figure 26: The arrows in this mosaic show misalignment at the top and bottom of the mosaic while there is alignment at the middle.

suggest that the calibration done to in preparation for image rectification was not sufficiently exact and that errors occurred. This suggests that errors of this type could be eliminated with a proper calibration in the eventual system.

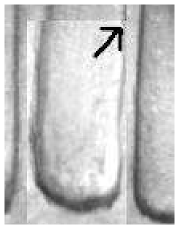


Figure 27: This mosaic section shows a region where the cut lines are visible due to different gray scales between the original images.

The region of a mosaic that appears in Figure 27 demonstrates the problem mentioned above of color inconsistency between cameras. The arrow points out a region of the cut line showing a correct match, and a visual inspection of the entire image will confirm this match. The problem is that the apparent difference in illumination as seen by the two cameras leads to an apparent discontinuity between the two stitched images. This situation might be improved through calibration, but most likely can only be solved by using, in combination, more expensive cameras that would be cost prohibitive for UVIS and better, more even lighting.

Figure 28 shows a region where the match was successful. The stitch line is

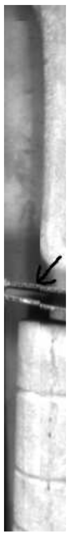


Figure 28: This mosaic portion illustrates a successful match and mosaic of images.

only barely visible along this line except in the region where the arrow points to the horizontal drive shaft. This problem with the drive shaft can be eliminated by simply aligning the cameras better vertically within the eventual system.

Overall, the crosswise mosaics appear to give a good representation of the undercarriage with generally only minor errors that can be solved through better calibration and alignment. These results show that the design is feasible for the crosswise mosaicing process and that the results are acceptable. Appendix B shows the complete results of the crosswise mosaicing process, including the disparity images used to create on the the crosswise mosaics.

4.2 Lengthwise mosaic generation results

Figure 29 shows the complete center perspective view of the vehicle undercarriage model used for testing. The image shows many areas of breakdown, too many to discuss at once, so specific problems will be addressed by referencing figures that show specific regions of breakdown.

Figure 30 shows a region where the match worked correctly. The arrows point to certain locations where the horizontal cut line is slightly more visible. Even in these locations the mosaic is relatively seamless. Note that the two images which have been mosaiced together in order to create Figure 30 look relatively similar, with the same vertical lines alignments in each. Though it cannot be seen, the depths of the model surface at the two locations stitched together are similar. Equally important to the seamlessness of the mosaic is that the two original images from which it is created have the same width, since the algorithm has no scaling factor to handle differences in width.

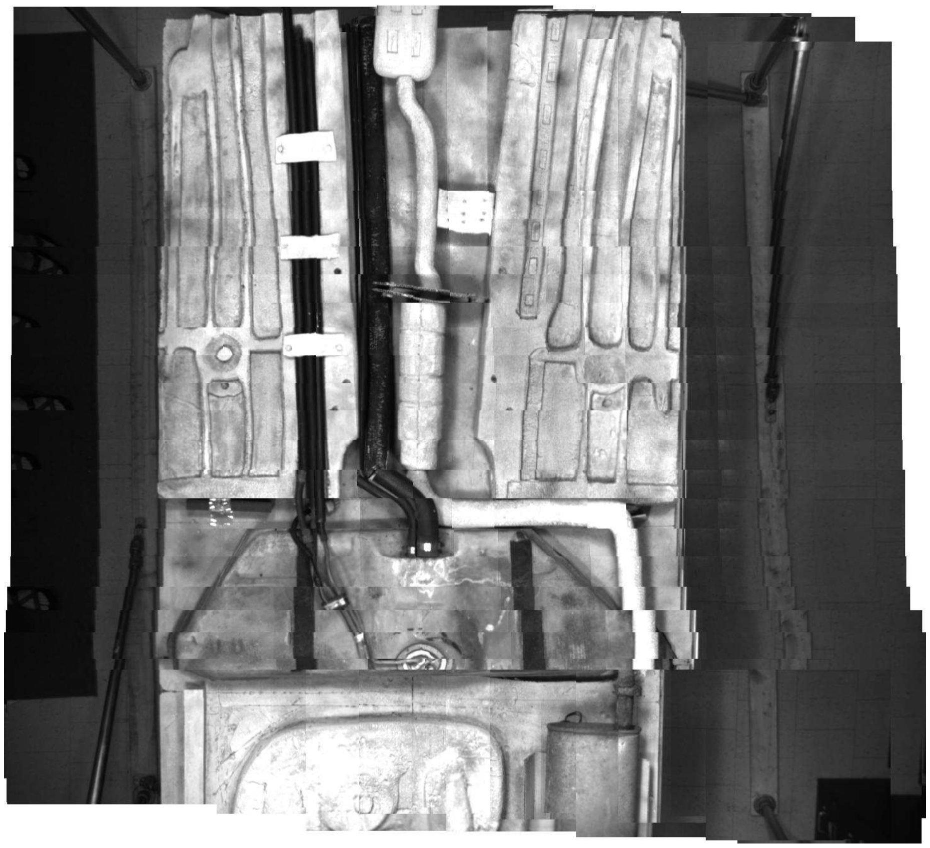


Figure 29: Complete center image of undercarriage

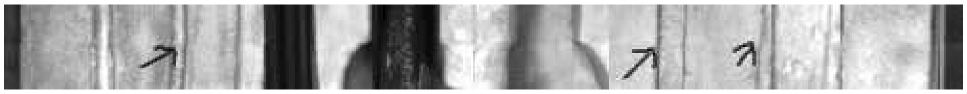


Figure 30: Section of the center mosaic where the model height is relatively uniform

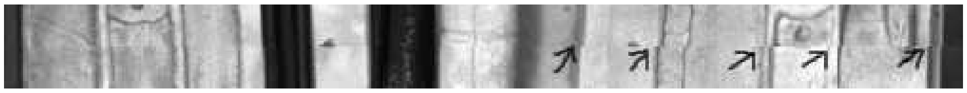


Figure 31: Section of the center mosaic with breakdown due to crosswise mosaic width difference.

Problems arise when the widths of the crosswise mosaics to be mosaiced lengthwise are not the same. In the strip of the center mosaic that appears in Figure 31, the vertical lines in the left part of the image connect and those in the right part with the arrows pointing to them are all offset by the same amount. The problem results from perspective differences and is illustrated in Figure 3. The depth of the model at a given location changes the viewing angle and hence where the match is located within the images; Figure 32 shows the situation geometrically. The 8-inch depth difference shown in Figure 32 translates to a 5-pixel match location shift that causes the broken vertical lines shown in Figure 31. These depth changes add up with each crosswise mosaic, creating the image in Figure 29 in which every vertical line is broken at least once. It is worth noting that the problems found in Figure 31 are not matching problems but instead reflect differences in the widths of the images used to create the mosaics.

The underlying problems in Figure 31 are reflected in the corresponding disparity map, Figure 33, of the match. The extreme breakdown in the match over the left-most 10% and the right-most 20% (which has been removed from Figures 30 and 31) can be ignored because it results from an attempt to match regions outside of the model of the undercarriage. These regions of extreme breakdown can seen on the left and right sides of Figure 29. The uniformity of the match on the left side of Figure 33 shows where the clean match was made, whereas the right side shows the breakdown in the match. This breakdown could be fairly easily handled by adding warping to the mosaicing process. This warping could expand the middle of the image, shifting the right side so that it would match like the left side, creating a smooth output image. This is fairly easy to do in this case as there is only one depth change causing the match to break. Warping might not be feasible in general where there could be multiple depth changes. Another problem with adding warping is that warping will change the sizes and shapes of parts of the car, making it harder for an inspector to identify any contraband.

Figure 33 shows a simple case of mismatched widths; a more representative case as shown in Figure 34. This section of Figure 29 is generated from a large

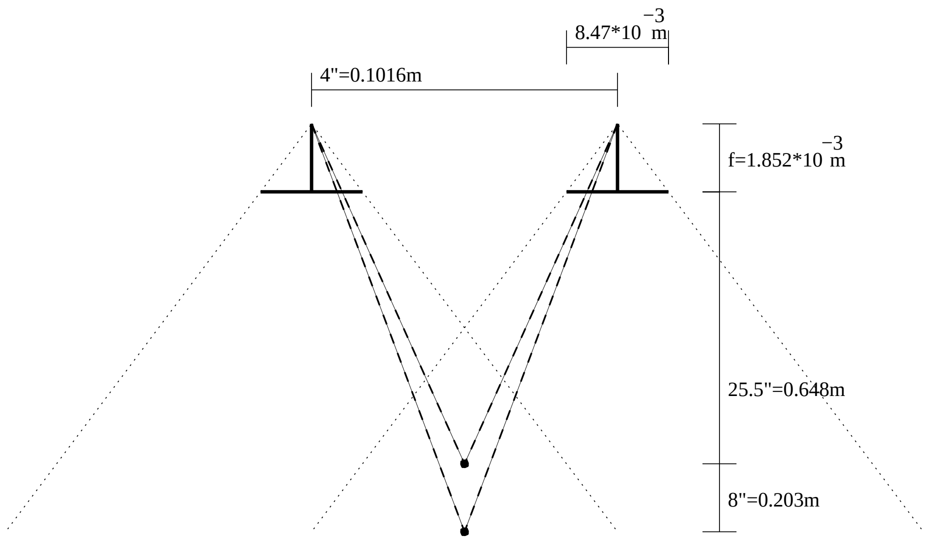


Figure 32: Diagram of match location shifts due to height differences

contoured section of undercarriage typical of a real vehicle and shows widespread misalignments as are found over most of Figure 29. Figure 35 shows one of the original images of this contoured section of the undercarriage. The extent of the problems with the match can be seen in Figures 36 and 37, which show the X and Y components of the disparity map, respectively. Considering Figure 36, again ignoring the edges outside the model, it is evident that, unlike Figure 33 where the X offset fit into two distinct regions, the X offset shifts continually across the model. This constant shifting would require warping across the entire image to get it to match the previous image and eliminate the breaks in vertical lines seen in Figure 34. This total image warping would add large areas of distortion, calling into question anything observed by the inspector viewing the end results.

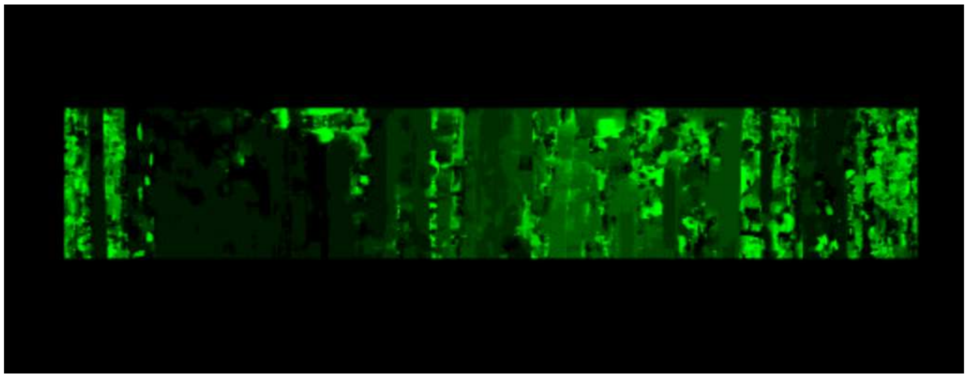


Figure 33: The X component of the disparity map between the images used to create the Figure 31 section of Figure 29. The values within this disparity map are expanded fill the range from black to green for easier viewing. The brighter the color the greater the disparity.

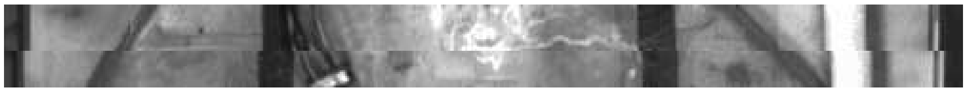


Figure 34: Representative case of most of the breakdown regions present in the lengthwise mosaicing process

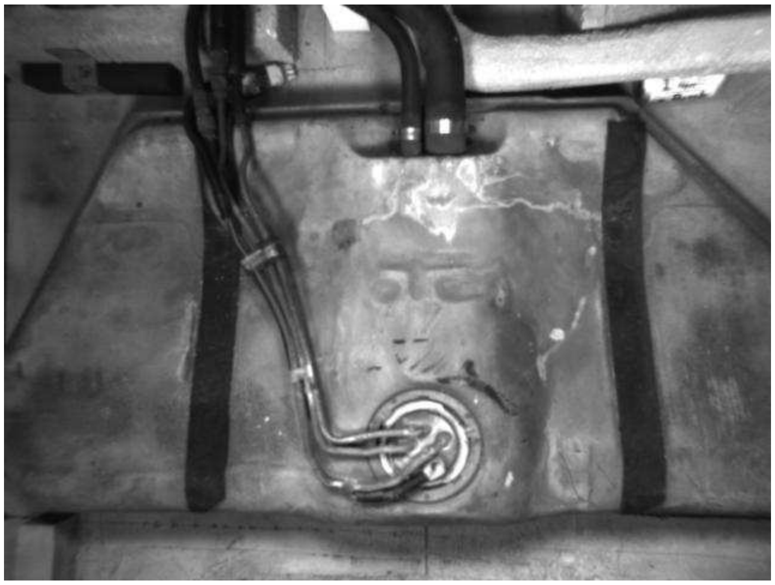


Figure 35: Original image used to generate undercarriage mosaics

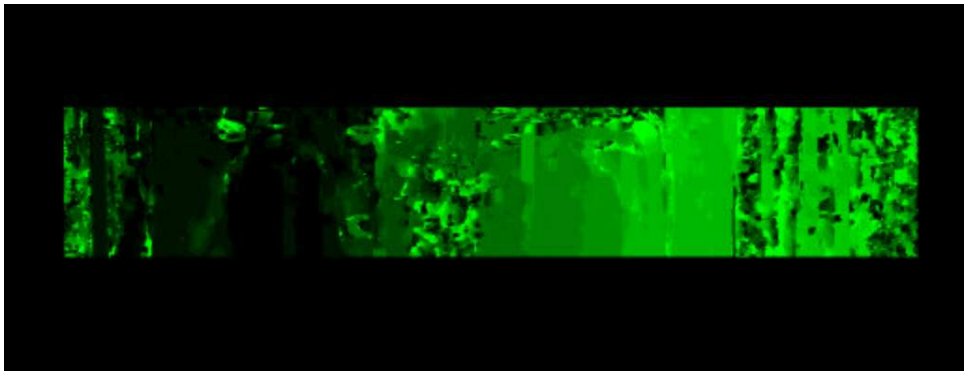


Figure 36: The X portion of the disparity map created when matching Figure 34

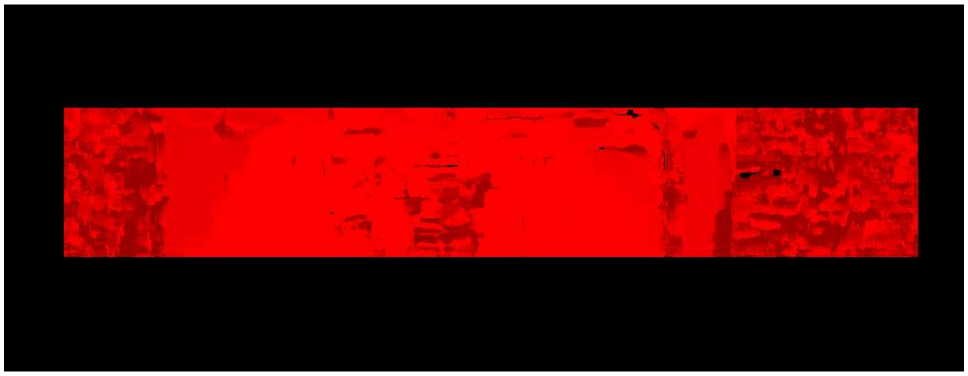


Figure 37: The Y portion of the disparity map created when matching Figure 34

Figure 37 shows evidence of another breakdown, this one caused by depth differences found in the Y direction of mosaicing. The differences in shades of color closer to the edges of the model show that the gas tank will not match vertically around the edges at the same point as it does vertically in the center. This suggests a need to warp crosswise mosaics in the Y direction in addition to warping in the X direction to make them fit seamlessly, possibly creating more artifacts.

Figure 38: Section of the complete undercarriage from Figure 29 where complete breakdown of matching occurs

Finally, matching may break down entirely. The best example of this occurs at the bottom of Figure 29, which appears in Figure 38. This can easily be seen to be a complete breakdown where no sections of the undercarriage line up smoothly. Figure 39 shows the complete break in the matching process that causes this broken mosaic to occur. The disparity map clearly shows, by the complete lack of uniform color and hence match regions, that no overall match was found between the crosswise mosaic strips. The breakdown most probably occurs because of the contours present in the model. The resulting depth changes create differences in views between cameras that cannot be handled by the matcher. No amount of warping will fix this problem. Although no definite solution is known, it has been hypothesized that a smaller match window might lead to a greater ability to match contoured regions because less of the contour would appear in any single match window. Appendix B contains all of the disparity maps used to generate the complete center image of which Figure 39 is an example.

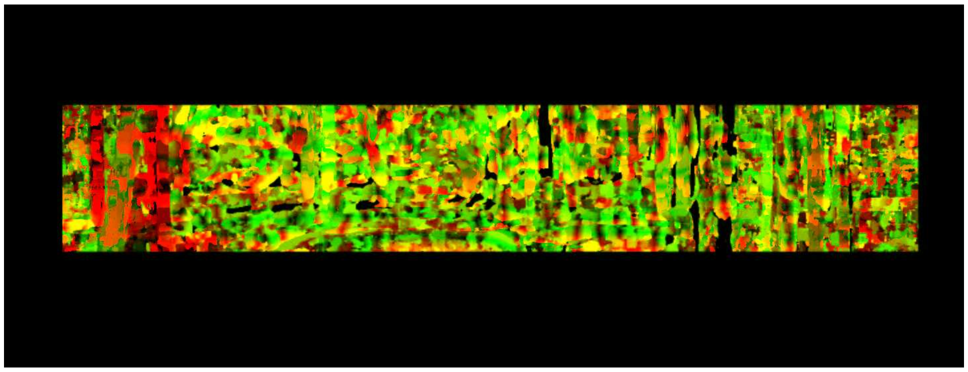


Figure 39: Complete disparity map between the crosswise mosaics used to create the section of Figure 29 shown in Figure 38

5 Discussion

The results show that the system as currently designed gives overall results that are not error free. While the system does give a complete view of the undercarriage, the image is not smooth enough to equal the quality of what the inspector sees with a mirror. The quality issue is illustrated by Figure 40 which shows the gas tank from the complete center mosaic. While the inspector will have a complete view of the undercarriage, the broken lines and missing sections of the image will severely limit his ability to identify objects.

Some of the problems listed in the results section that lead to what is seen in Figure 40 could be eliminated with the introduction of warping. The warping would smooth the results and create more seamless cut lines that would significantly improve the overall view and ease of inspection. There are three problems

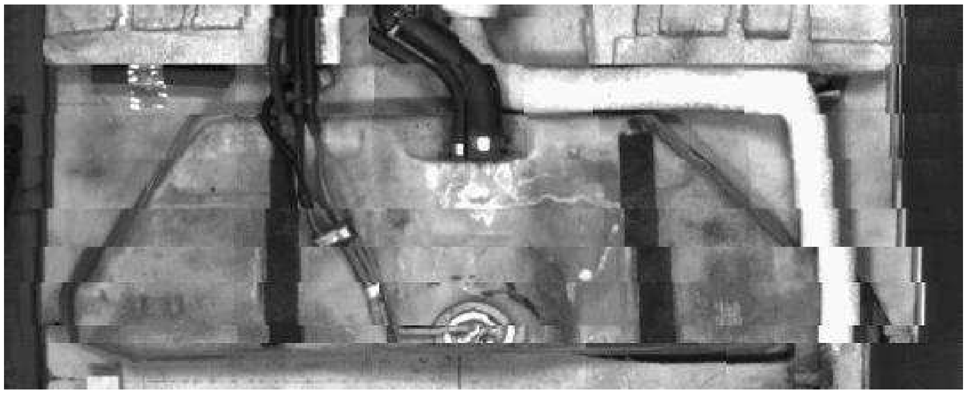


Figure 40: This image is the bottom of the complete undercarriage view.

with this, the first of which is that warping alters the images. The alterations could add or eliminate details about the undercarriage by either stretching or compressing parts of the image to look like something they are not. Every modification of the original images adds a degree of uncertainty and possible error to inspection. Taking the original images and mosaicing them is one degree of modification, taking all of the original data, cutting it down, and fitting it together to give a complete view. This modification adds an overall view of the undercarriage but does so at the expense of some of the data from the original images that will appear in none of the mosaics. Any warping performed on the original images will add a second modification to that data already modified by mosaicing.

The second problem with warping the images is the time it will add to the mosaicing process. As described above, the algorithms to match are $O(n^6)$, which makes close to real-time operation extremely difficult. A study of the feasibility and timing of a similar system to this which appears in Appendix C determined that a system using 320x240 pixel images instead of the 640x480 pixel images used here might just be able to generate and display results within 30 seconds, the run time goal of this system. However, the timing model used did not include a warping process, suggesting that adding warping will make the process take too long and therefore be unfeasible in the real world.

The final issue with adding warping is not a factor now but will be in the future. It was hoped that this project would create a system that could eventually automatically detect contraband. A system to auto detect would probably begin by comparing of the undercarriage of the vehicle being inspected to the undercarriage of a vehicle of the same make and model. If the element of warping is added, there is little chance that the undercarriage will look similar from one run to the next as the images will likely be warped differently from one vehicle run to the next.

Another possibility for improving the results would be to implement it using flat field cameras. Flat field cameras would eliminate errors added by the rectification process and would give more uniform images that would be easier for the system to mosaic. The better quality original images would lessen errors caused by scale changes and color consistency by eliminating any noise added by rectification. The problems with shifting to flat field cameras is that they will add to the price, and would create a need to add software to automatically synchronize the cameras to the system.

6 Conclusion

The current method of generating mosaics does not work sufficiently well or quickly enough to make it an improved method for under-vehicle inspection. The methodology used was chosen to be fast, accurate, and inexpensive. When the system was developed in concept, these three factors caused the idea of a full 3D reconstruction to be eliminated from consideration even though it has a possibility for better results. The work done on the timing model of a similar system (see Appendix C) suggests that the system in its current format is not fast enough for real time under-vehicle inspection. The research also suggests that the computation power required to improve the system to run close to real time would be expensive and hard to transport because of the sheer number of processors required. This suggests that the proposed system was almost too slow and expensive before the addition of the warping required to make usable results. As the present results indicate that it is improbably that the system will be able to automatically detect contraband.

Thus, the conclusion is that the system is not a solution to the problem of under-vehicle inspection in its current form, and that other methodologies must be explored in order to find a feasible solution. It is the opinion of the researcher that a full 3D reconstruction should be the next method to try as it would probably allow true inspection and, later, automatic detection. As 3D reconstruction is probably not currently feasible in real time within the constraints of UVIS, it seems unlikely that the problem can be solved with current technology and without a breakthrough in either mosaicing or multi-camera reconstruction.

7 Acknowledgments

This work was partially supported by Atlantic Coast Technologies, Inc. (ACTI), the National Science Foundation under Grant No. 0105272, and the Army Research Office under grants DAAD19-99-1-0016 and DAAD19-02-1-0133.

References

[1] "Security mirrors \ observation systems", available at http://www.insight-security.com/pf16.htm#Vehicle%20inspection%20mirrors, last checked 6/26/02.

[2] "Inspection Mirrors", available at http://www.acrilconvex.com/inspection/default.htm, last checked 6/26/02.

[3] "Search Systems Incorporated: Technical Search & Surveillance Equipment", available at http://www.searchsystems.com/contraband.html, last checked 6/26/02.

[4] "UVI - Under Vehicle Inspection System", available at http://www.uvisystems.com/ last checked 6/26/02.

[5] Z. Zhu, A. R. Hanson, H. Schultz, F. Stolle, E. M. Riseman, "Stereo Mosaics from a Moving Video Camera for Environmental Monitoring", First International Workshop on Digital and Computational Video, December 10, 1999, Tampa, Florida, USA, pp. 45-54

[6] Z. Zhu, E. M. Riseman, A. R. Hanson, "Parallel-Perspective Stereo Mosaics", The Eighth IEEE International Conference on Computer Vision, Vancouver, Canada, July 4, 2001, vol I, 345-354.

[7] H.-C. Huang, Y.-P. Hung, "Panoramic stereo imaging system with automatic disparity warping and seaming", Graphical Models and Image Process., 60(3): 196-408, 1998.

[8] S. Peleg, M. Ben-Ezra, "Stereo panorama with a single camera", CVPR'99: 395-401

[9] H. -Y. Shum, R. Szeliski, "Stereo reconstruction from multiperspective panoramas", ICCV99, 14-41, 1999.

[10] S. Peleg, J. Herman, "Panoramic mosaics by manifold projection", CVPR'97: 338-343.

[11] P. Dickson, J. Li, Z. Zhu,A. R. Hanson, E. M. Riseman, H. Sabrin, H. Schultz, G. Whitten, "Mosaic Generation for Under Vehicle Inspection", ACV'02:251-256.

[12] H. -Y. Shum, R. Szeliski, "Construction and refinement of panoramic mosaics with global and local alignment", ICCV'98: 953-958.

[13] R. Kumar, H. Sawhney, J. Asmuth, J. Pope, S. Hsu, "Registration of video to geo-referenced imagery", ICPR98,vol.4: 1393-1400.

[14] "Vision Lab at UMass - UVIS", available at http://vis-www.cs.umass.edu/projects/uvis/index.html, last checked 6/26/02.

[15] Sergei Kovalenko, Camera Calibration for Generating Mosaics, Master's Thesis, University of Massachusetts, Amherst, Massachusetts, 2002.

[16] D Tsai,C Lin, "Fast norlmalized Cross Correlation for Defect Detection", Pattern Recognition Letters, vol 24(15):2625-2631,November 2003.

Appendix A

Experimental setup

Because it is not feasible to build a scale working prototype of the system, one had to be simulated in the lab environment. To this end, a scale model of a vehicle undercarriage was created as shown in Figure 41. The model is a combination of carved and painted Styrofoam and actual car parts.

The cameras are mounted on a frame around the undercarriage as shown in Figure 42. The idea behind the model is that moving the cameras above a stationary model is equivalent to having a moving undercarriage over stationary cameras. The camera-to-undercarriage movement will be the same either way. As can be seen from Figure 42, the frame allows the cameras to be set at any location above the simulated vehicle undercarriage.

Figure 43 shows the actual cameras as they are attached to the frame pictured in Figure 42. The cameras are attached to circuit board material with wires. This setup allows for easy camera movement to test different separations of the cameras. However the model does not lend itself to a stable platform or careful camera alignment. In the eventual system the cameras will have to be rigidly mounted in place and their fields of view will have to be controlled for a consistent overlap. Because the cameras are hand mounted and their placement is suspect a mathematical derivation of the assumed match locations cannot be derived as it would be in a actual system, as such the assumed matches were instead created by hand matching points between the four camera images.

The cameras themselves sit on a sled (Figure 42) that rides down the frame on rail runners designed for this project (Figure 44). These runners allow the camera sled to be pulled down the frame at varying speeds to simulate the actions of a vehicle traversing the platform with an uneven speed.

Camera setup

The Point Grey DragonFly cameras used were calibrated and rectified using techniques and software provided by Kovalenko [15]. Figures 45 and 46 show, respectively, a raw input image and a rectified image. The radial distortion shown in Figure 45 demonstrates the need for image rectification. Without removal of the distortion, no possibility exists for matching the images.

Experimental model

Only 4 cameras are available, so it is not possible to test the system with the 30+ cameras the final system will have. Because four points of data are not enough to reasonably extrapolate to a 30+ camera, systems behavior from more cameras had to be simulated. A 16-camera system was simulated by taking the 4 cameras and taking images from them at four different locations across the vehicle undercarriage. This was done by shifting the longitudinal pipes to which the cameras are attached (Figure 42). These 16 cameras had to have the same

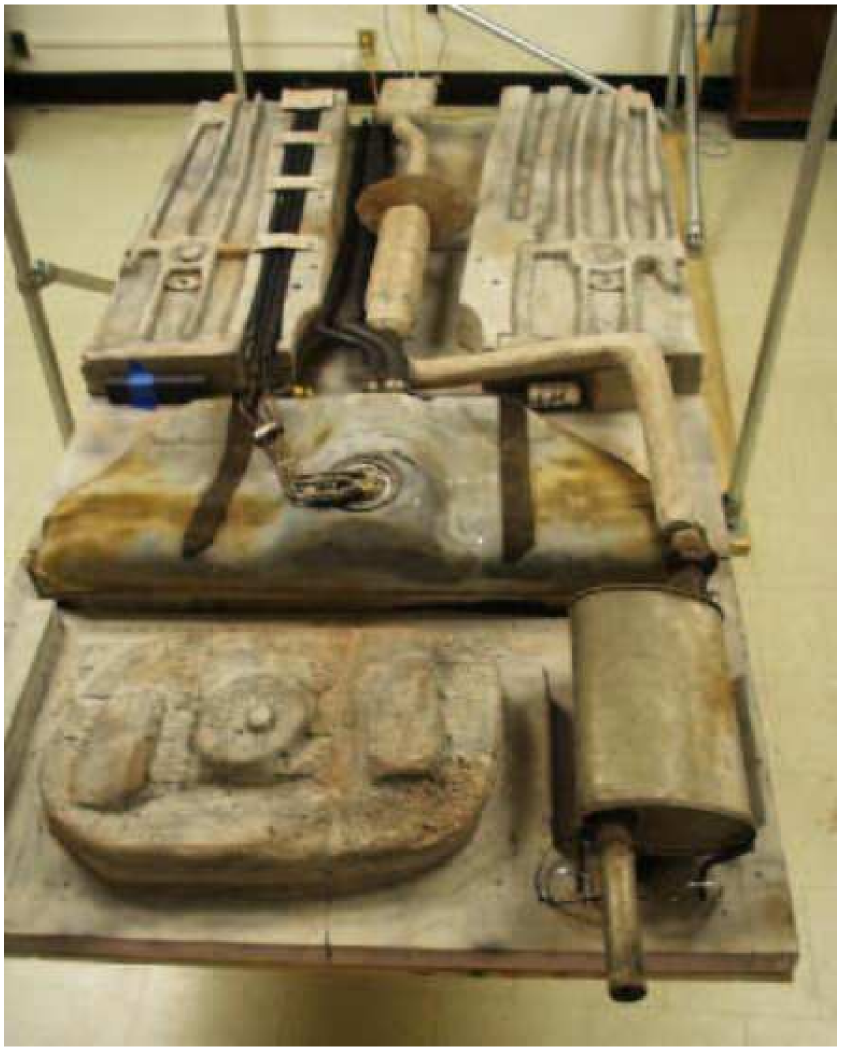


Figure 41: Image of scale vehicle undercarriage model

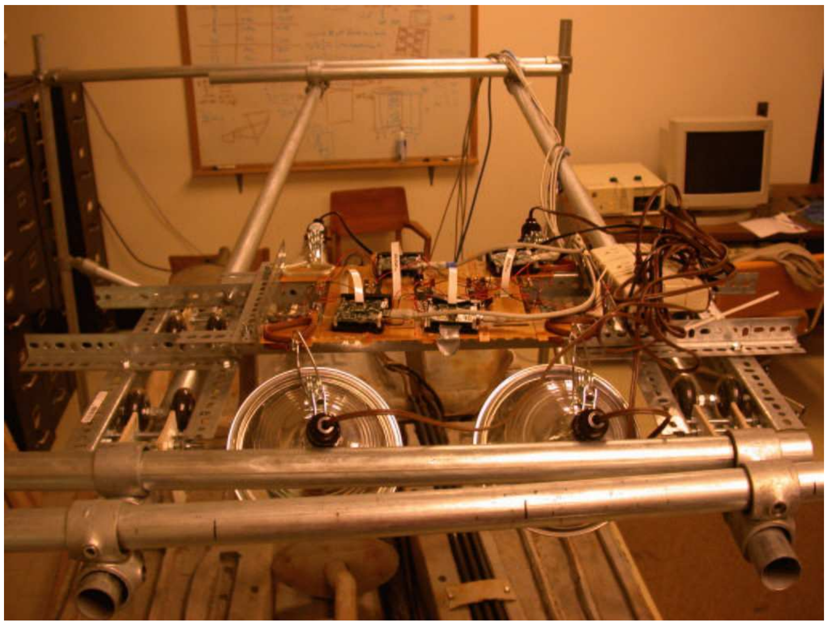


Figure 42: The frame on which the cameras are mounted above the undercarriage model

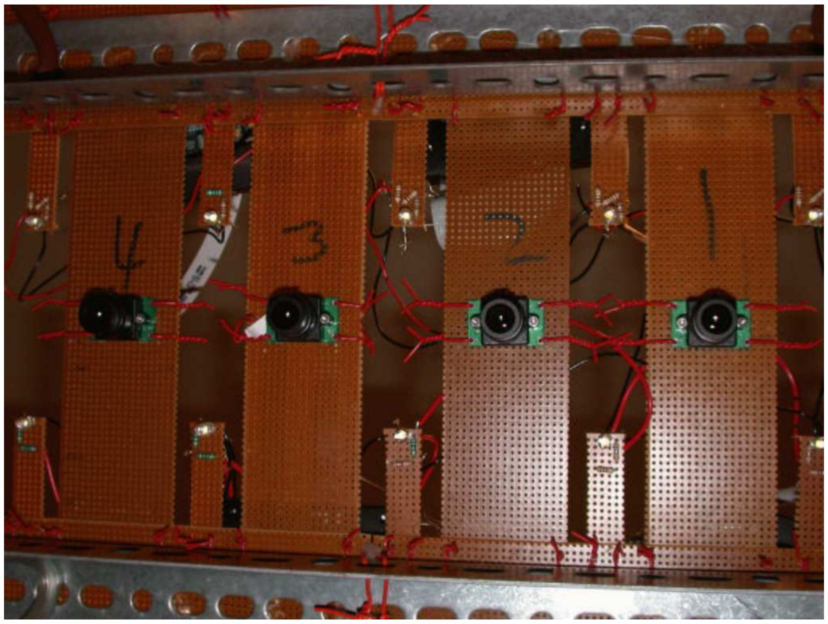


Figure 43: View from under cameras

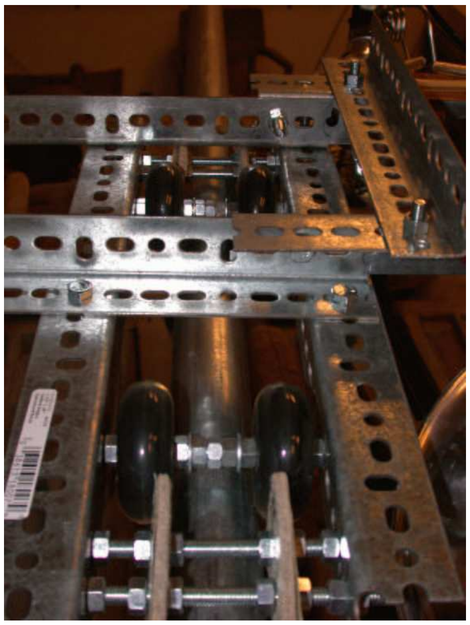


Figure 44: Camera platform runners

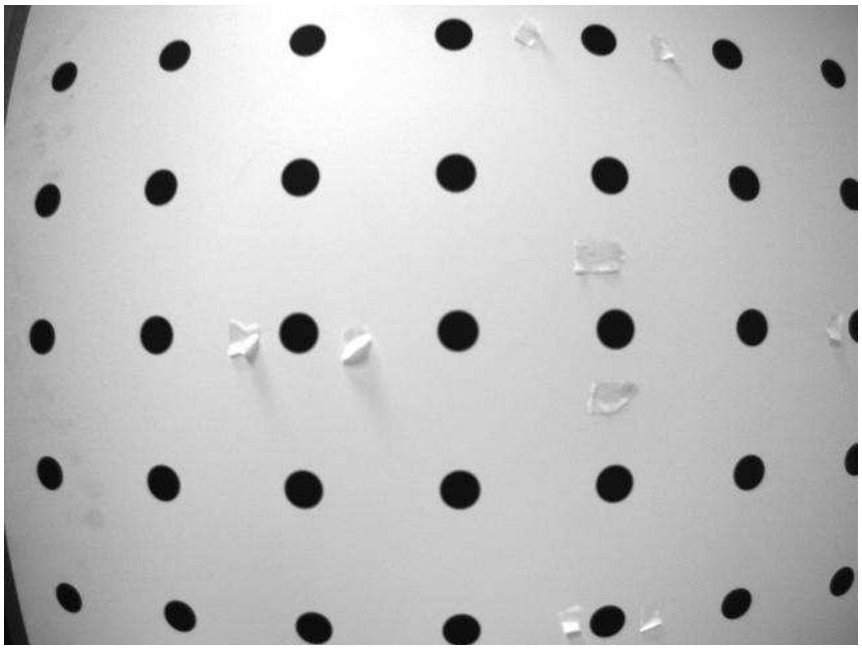


Figure 45: A raw input image from one of the cameras

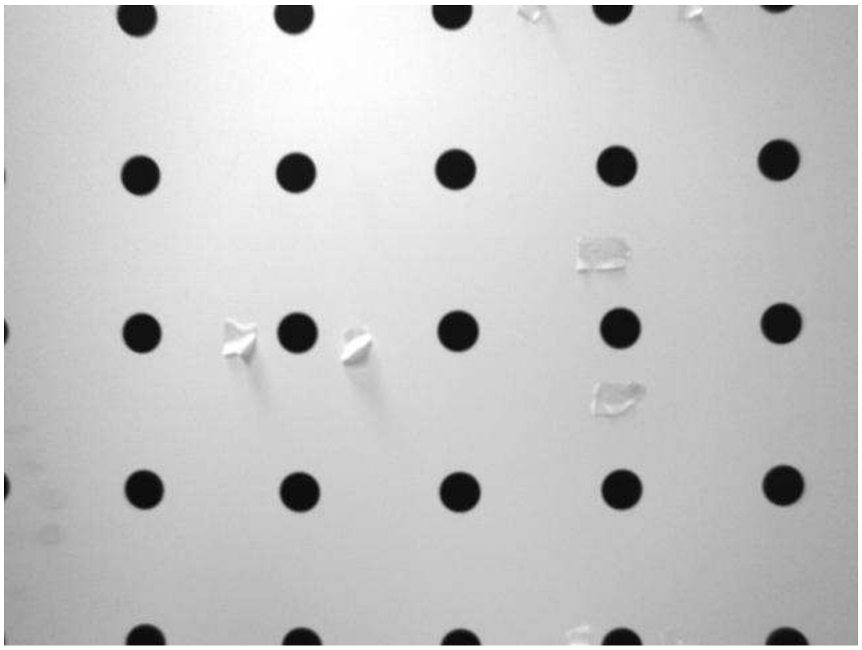


Figure 46: The same image as in Figure 45 rectified

simulated movement in the direction of simulated car movement; therefore, the cameras were drawn down the length of the model, taking images every 3 inches for each horizontal location of the cameras.

Because the 4 cameras needed to simulate the 16 cameras with known assumed offsets, the 16 simulated cameras were hand matched for each of the 19 locations down the model length. This 16-by-19 grid of images is the set of images from which all timing model information is determined. The images appear in Figure 47.

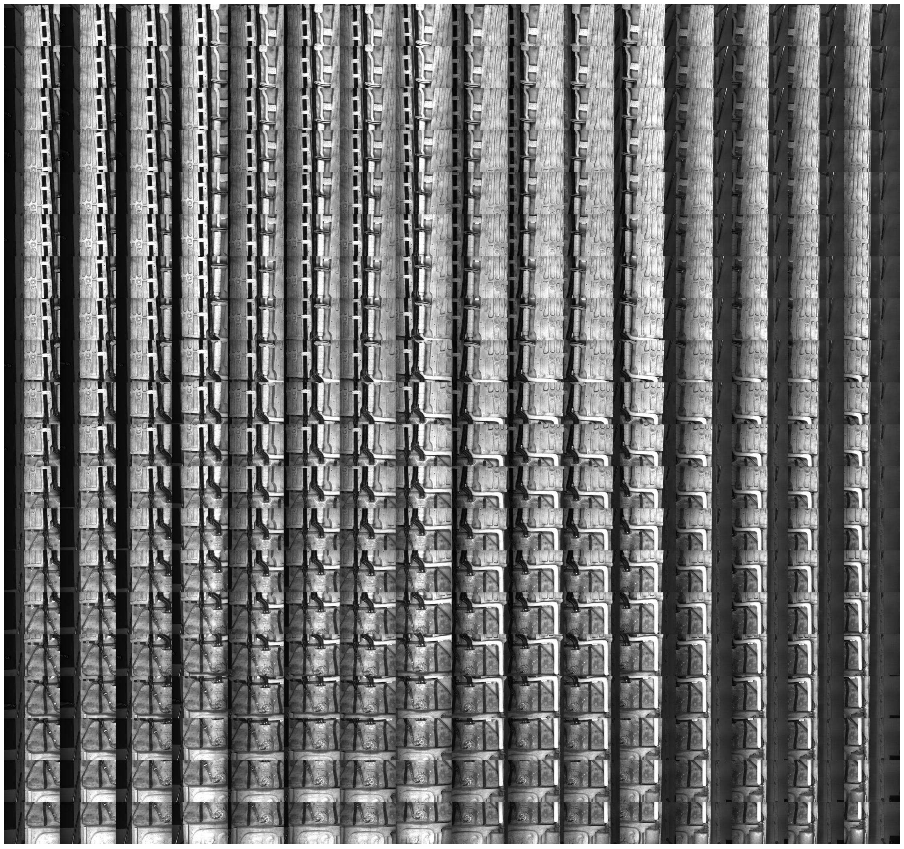
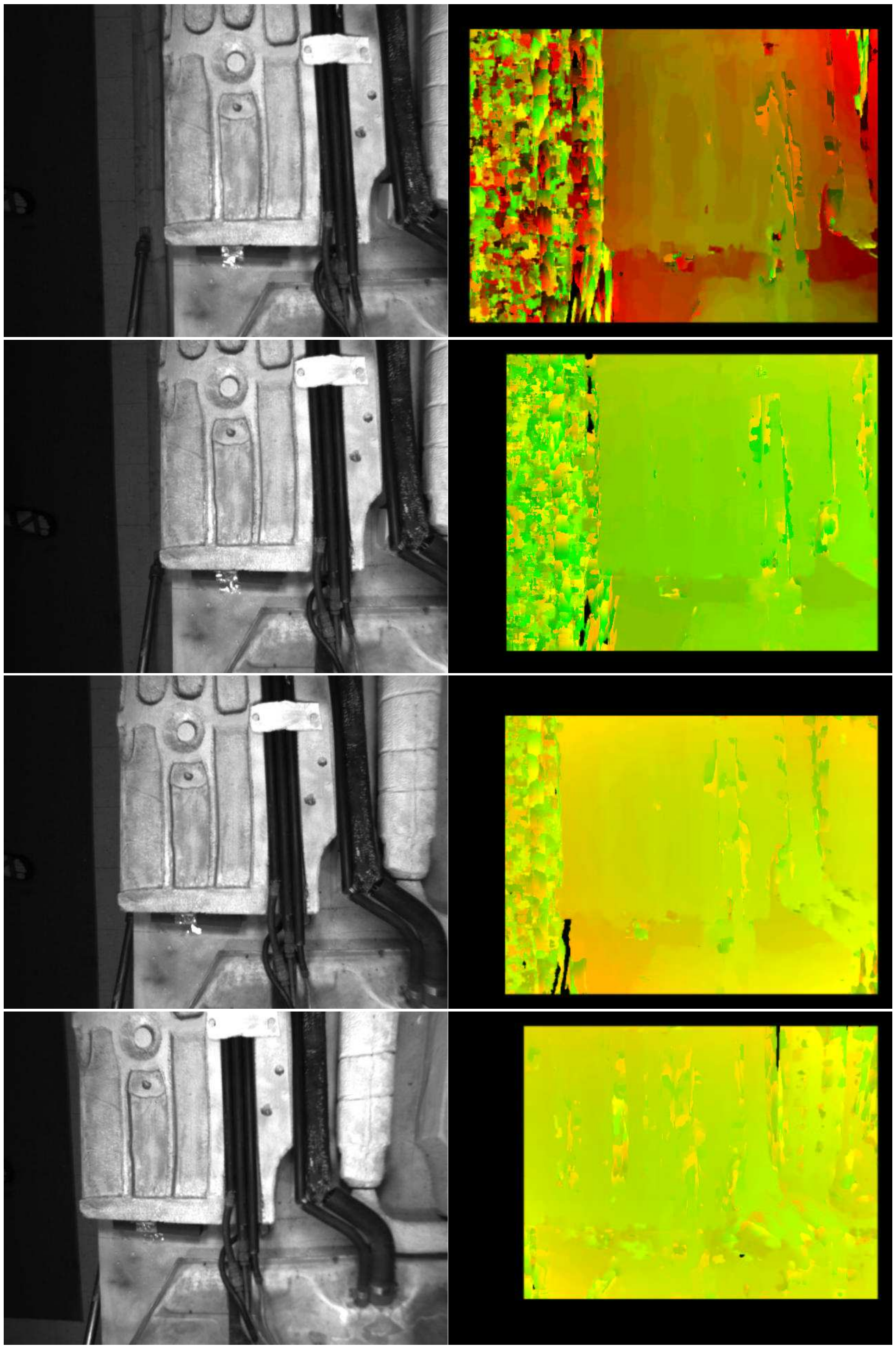
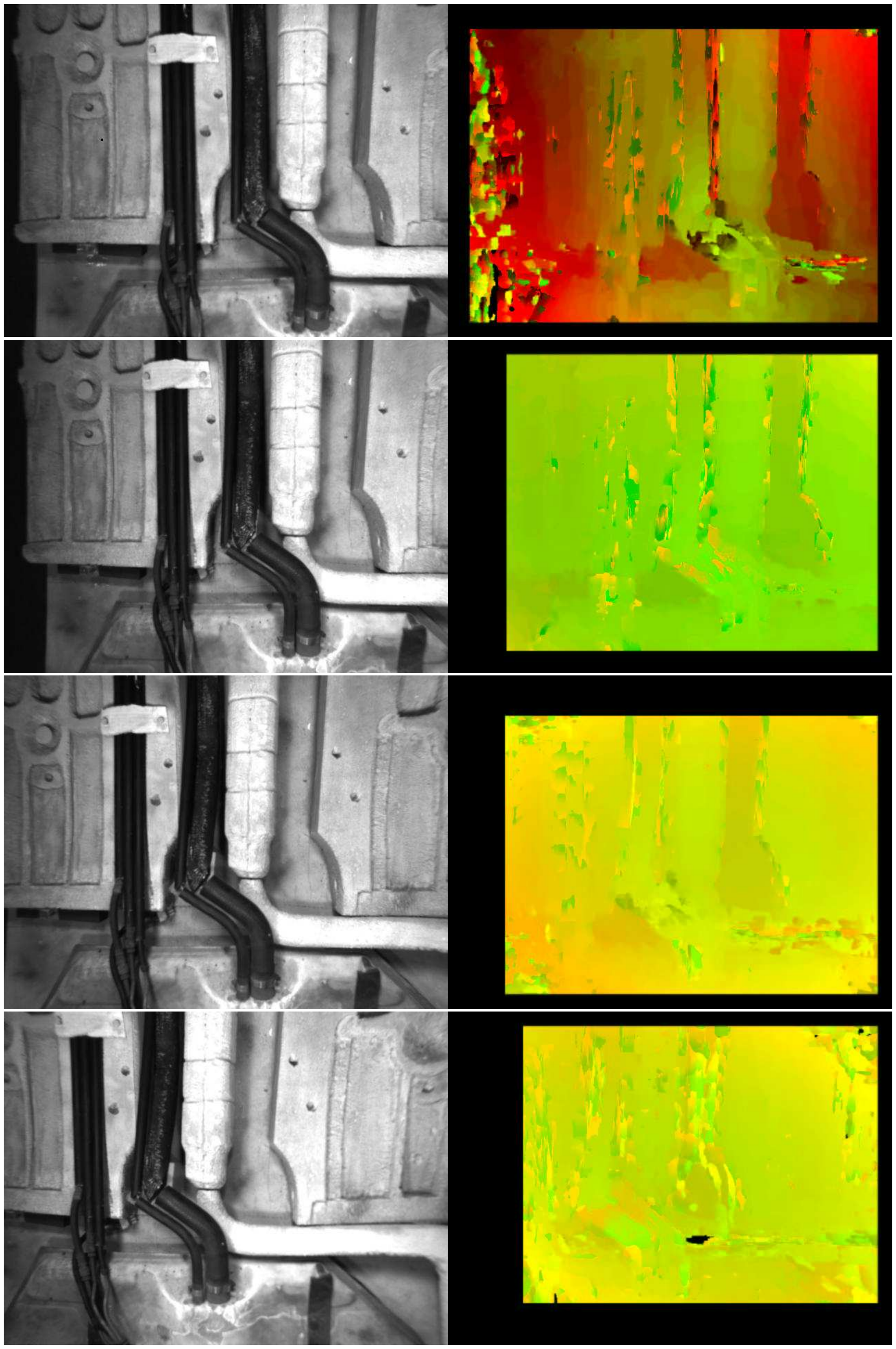


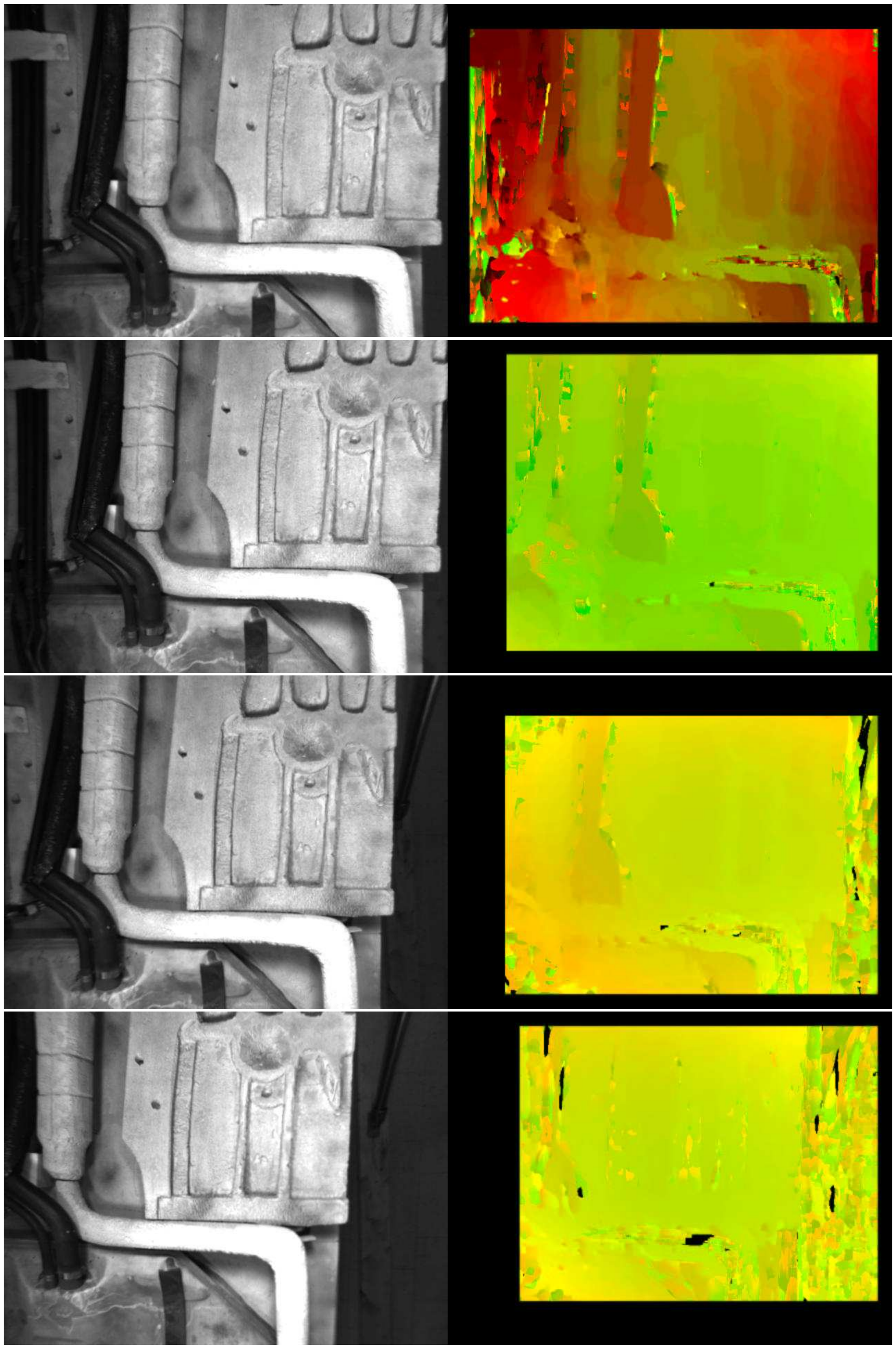
Figure 47: Complete set of images used for timing model

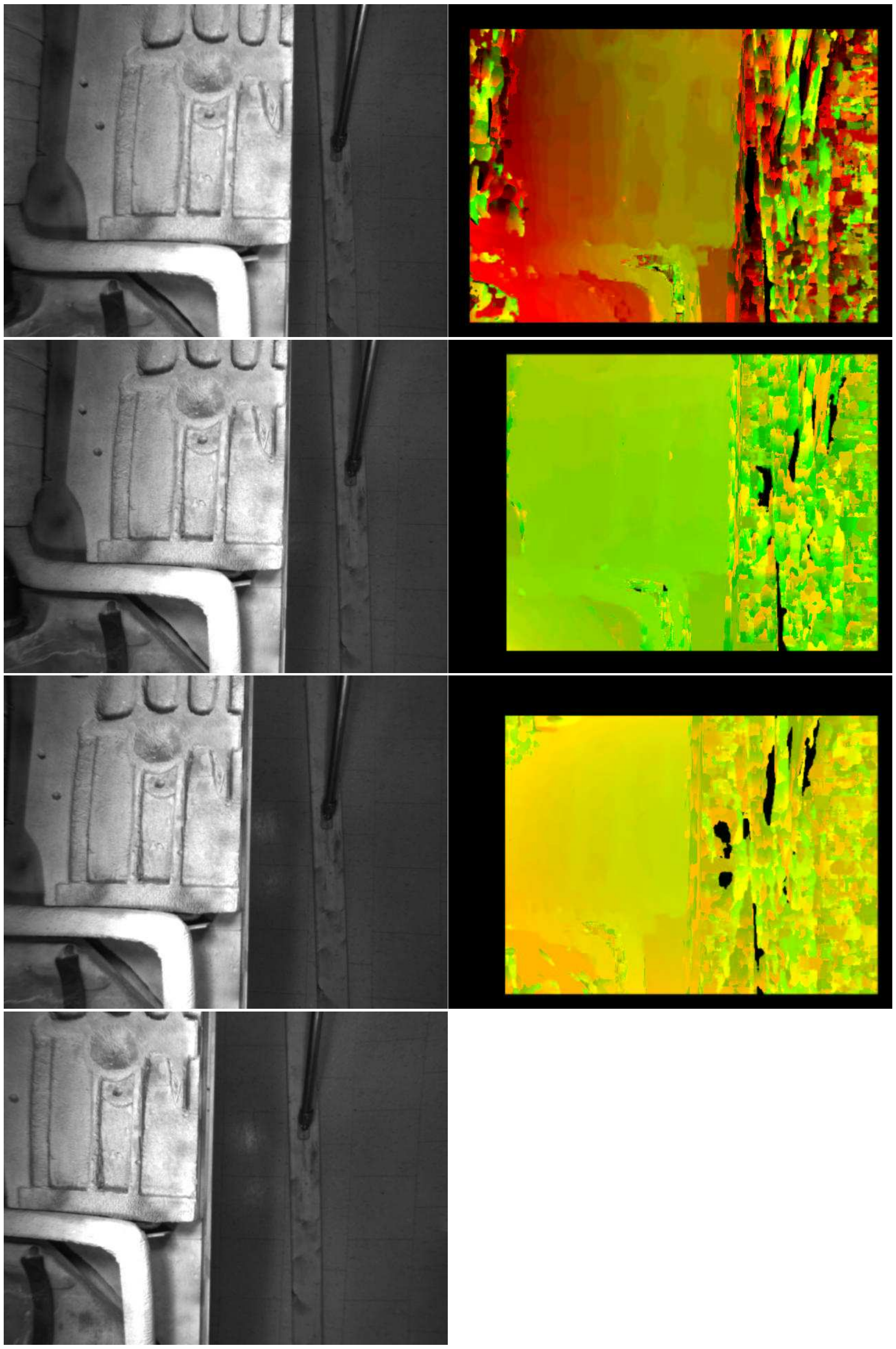
Appendix B

The following images are alternating rectified original image and disparity images. The alternation is such that each line contains a rectified original image followed by the disparity image that contains that match information between the rectified image on that line and the following line. The disparity images vary in overall color scheme because of the hand alignment of the cameras. The inexactness of the hand alignment means that the X and Y displacement, showed in green and red respectively, vary greatly between camera pairs. If the cameras had been better aligned then all of the disparity images would have looked similar in that they would have had similar shades of green present and almost no red. It is the fact that there are four cameras maintain their inexact alignment that makes the disparity images rotate between the four different general color combinations. The crosswise mosaic generated from these images appears after all of the images that create it.

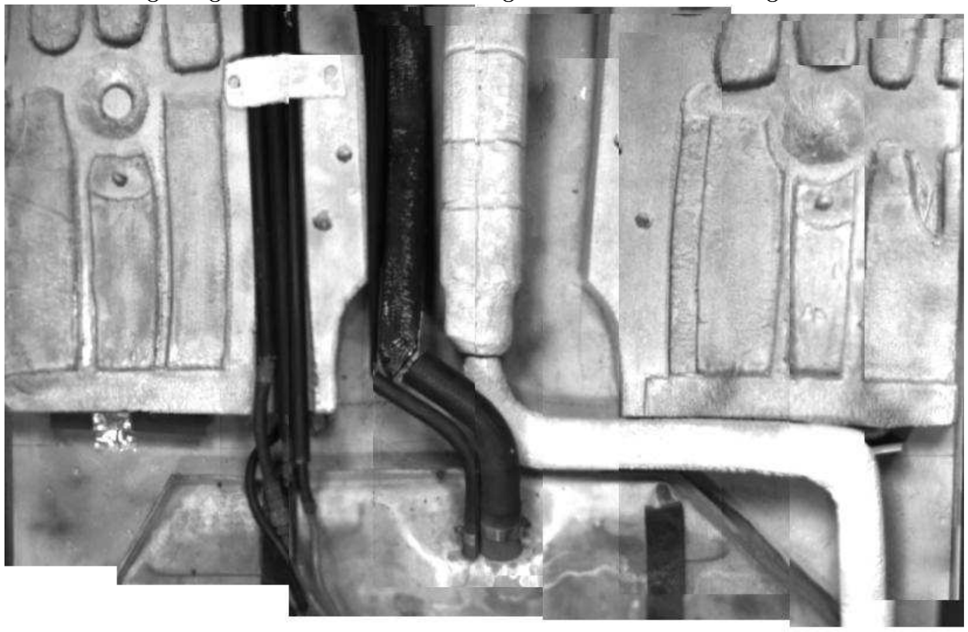




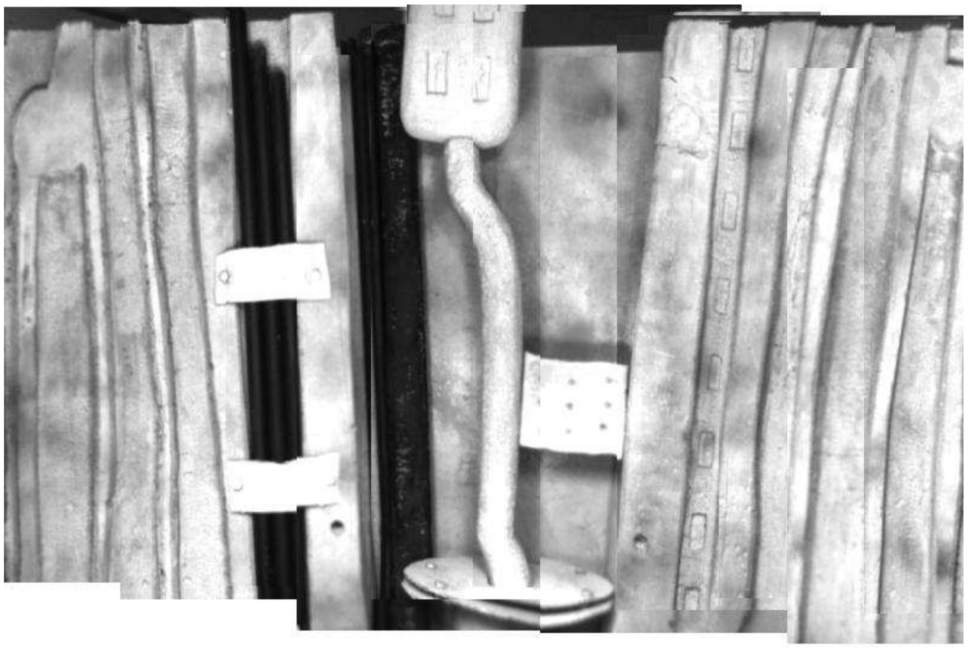


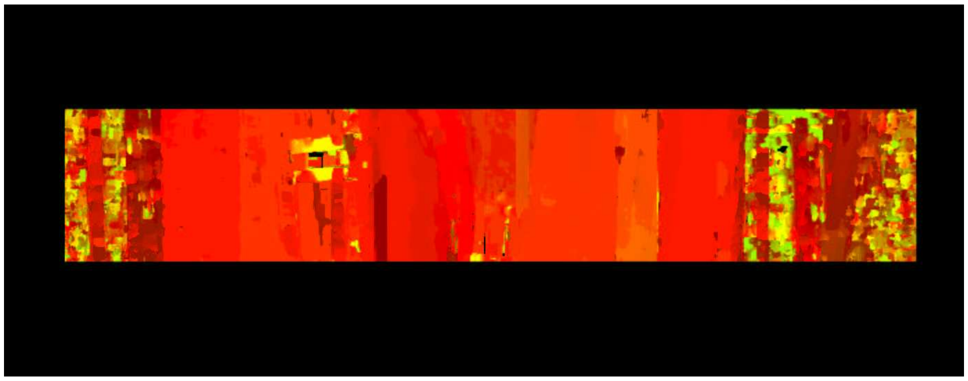


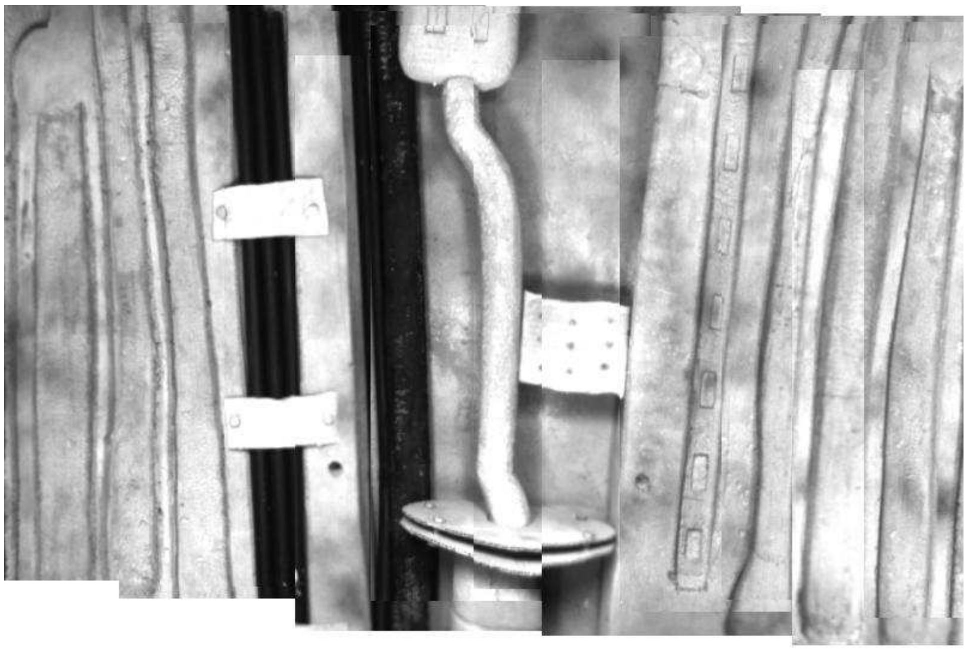
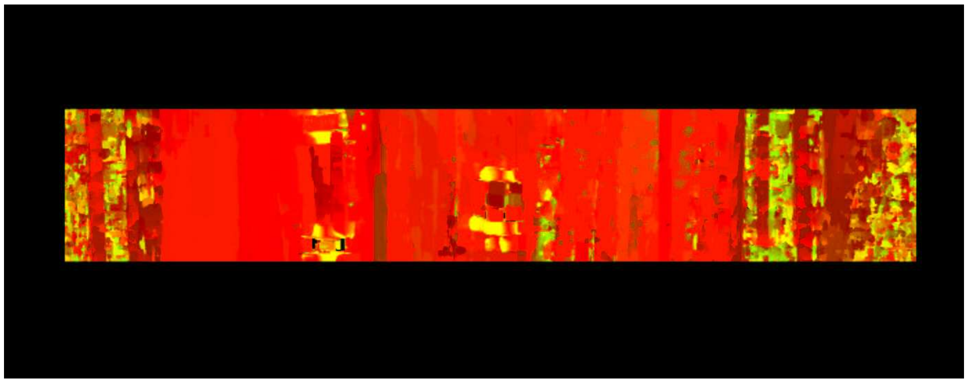
The following image is the crosswise mosaic generated from the images above.

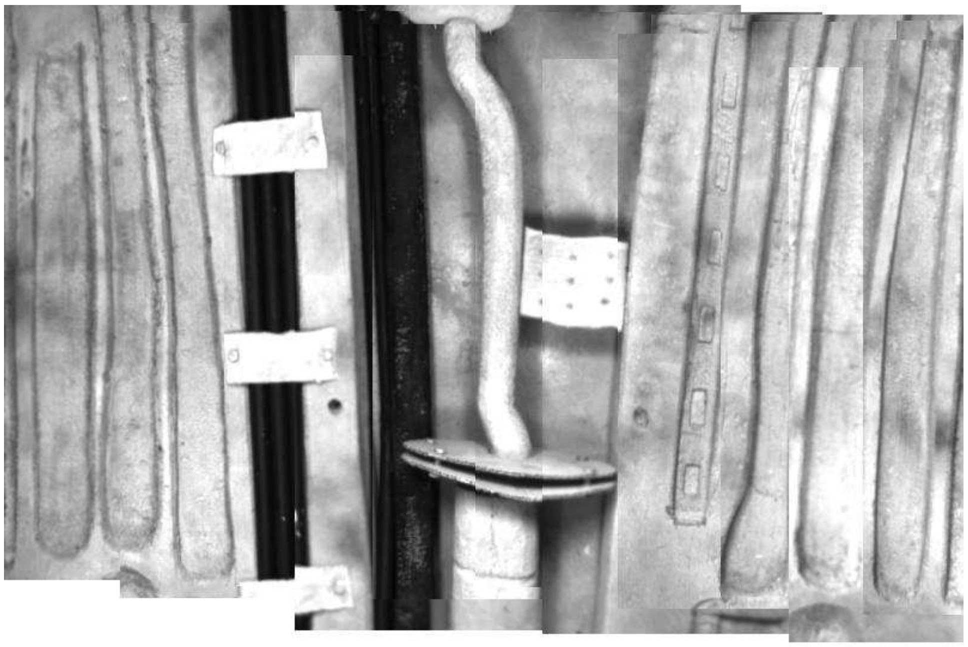


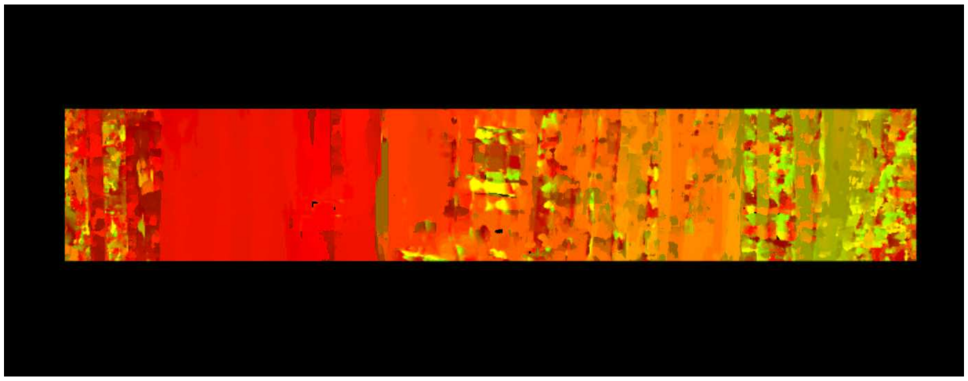
The following images alternate crosswise mosaics and disparity images ordered so that the disparity image appears between the two mosaics. The mosaics have been cropped to remove the area outside of the model; the disparity images have not been cropped. As with the crosswise disparity images, the red and green channels have been expanded to fill the range from 0 to 255 to make the disparities easier to see. The red channel contains the Y offset and the green channel contains the X offset.

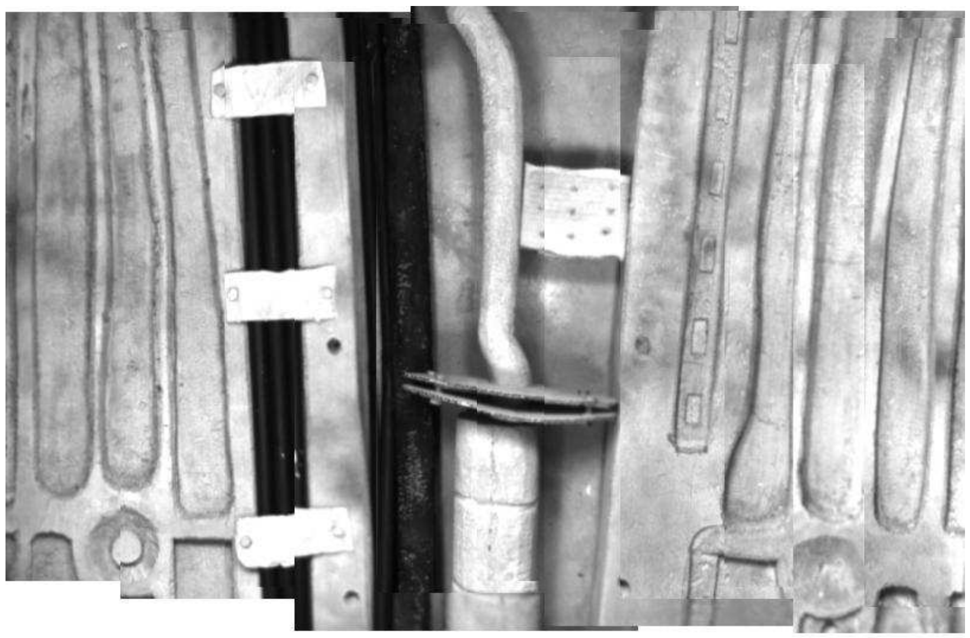


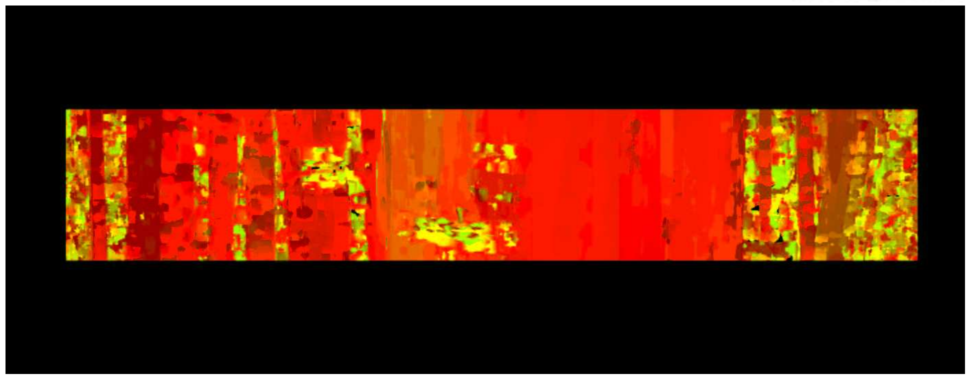


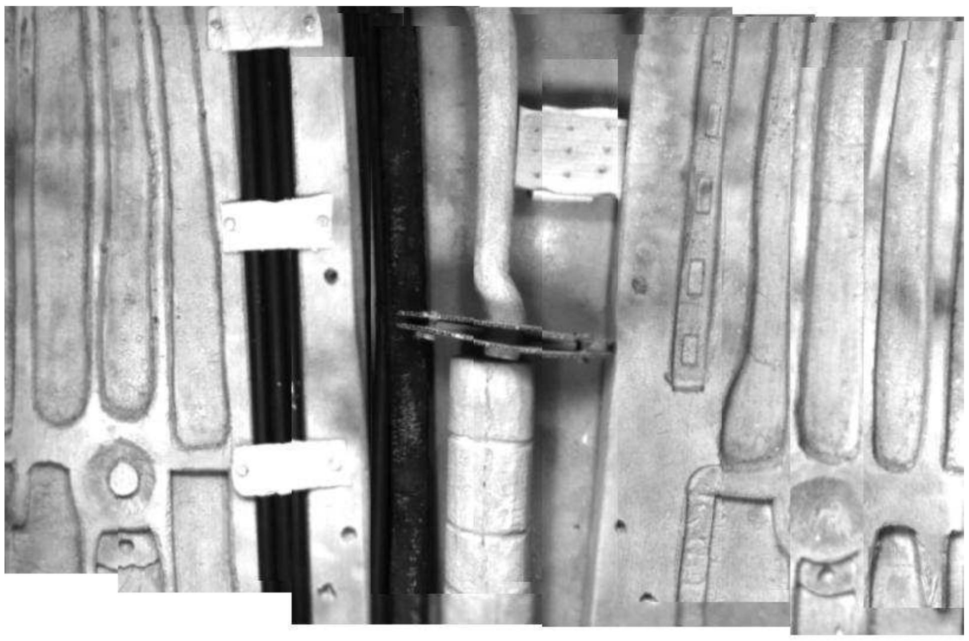



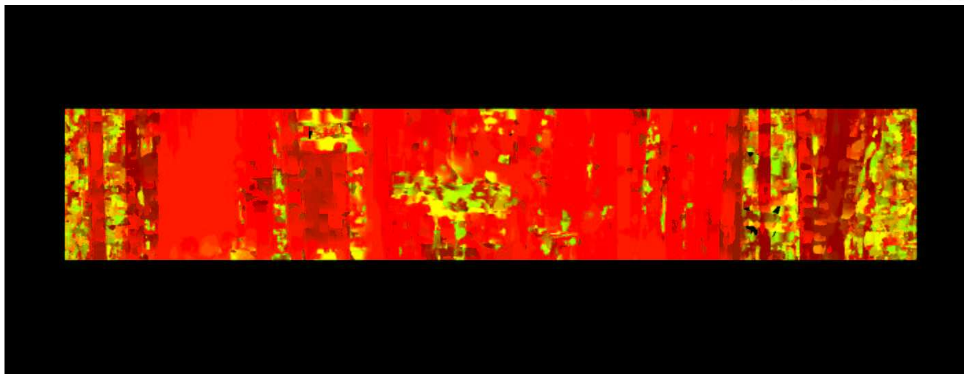


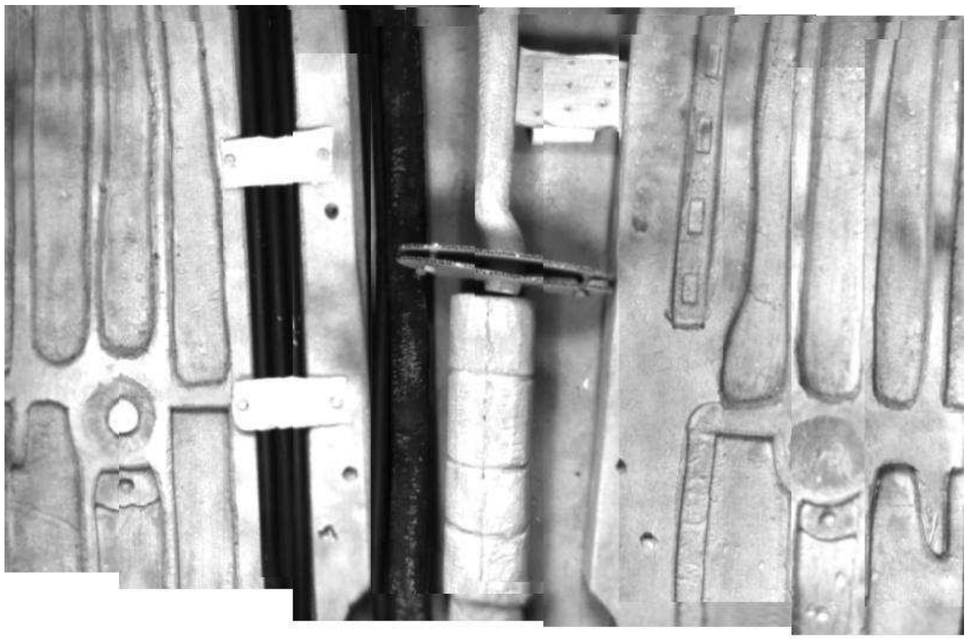


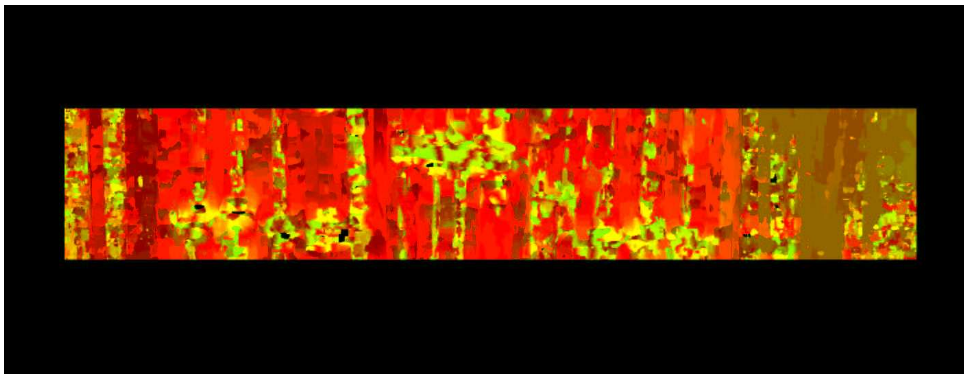


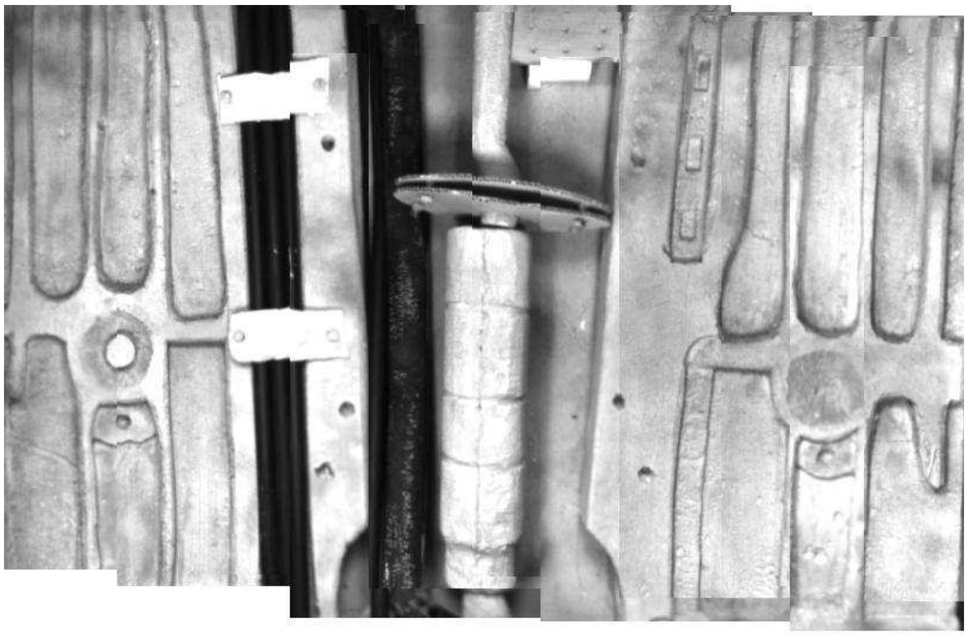
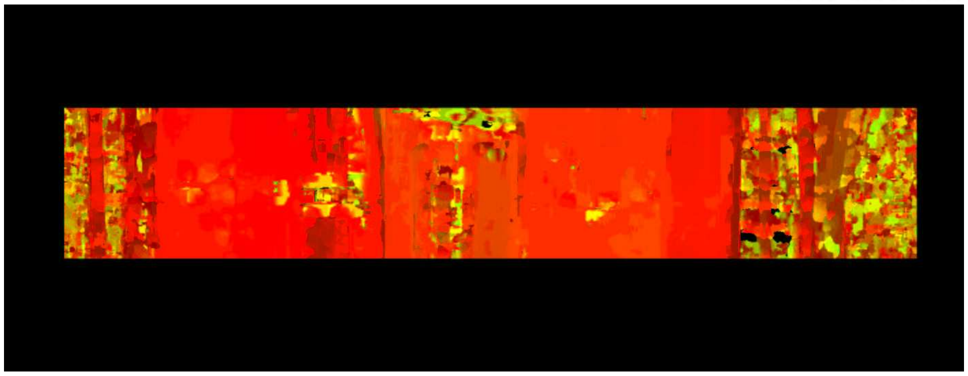


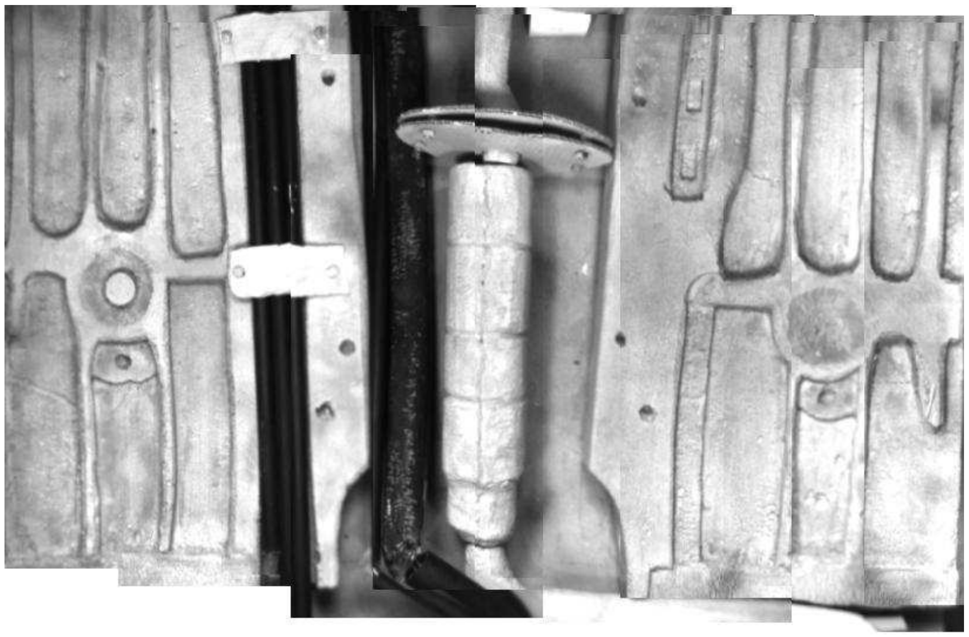
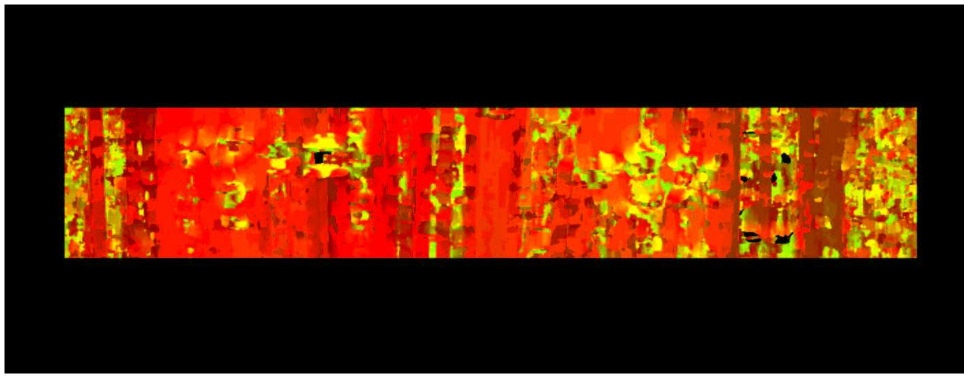


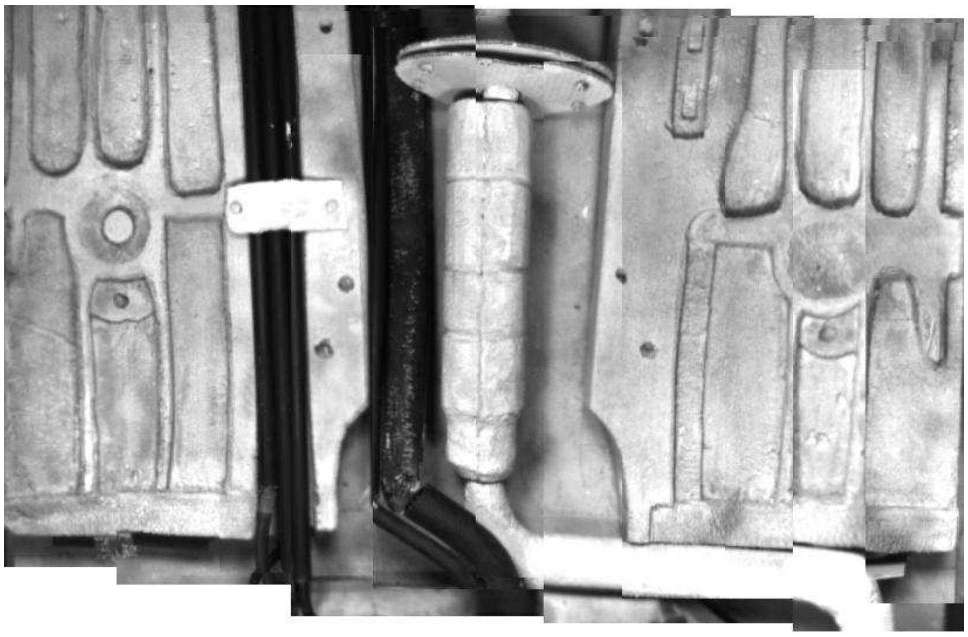


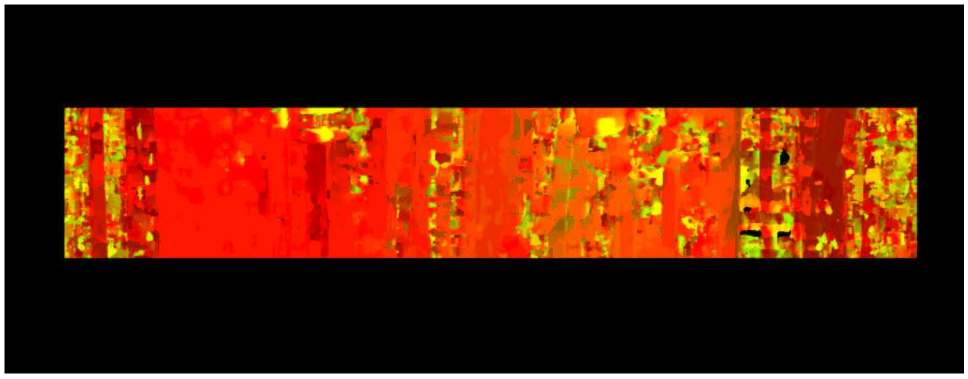


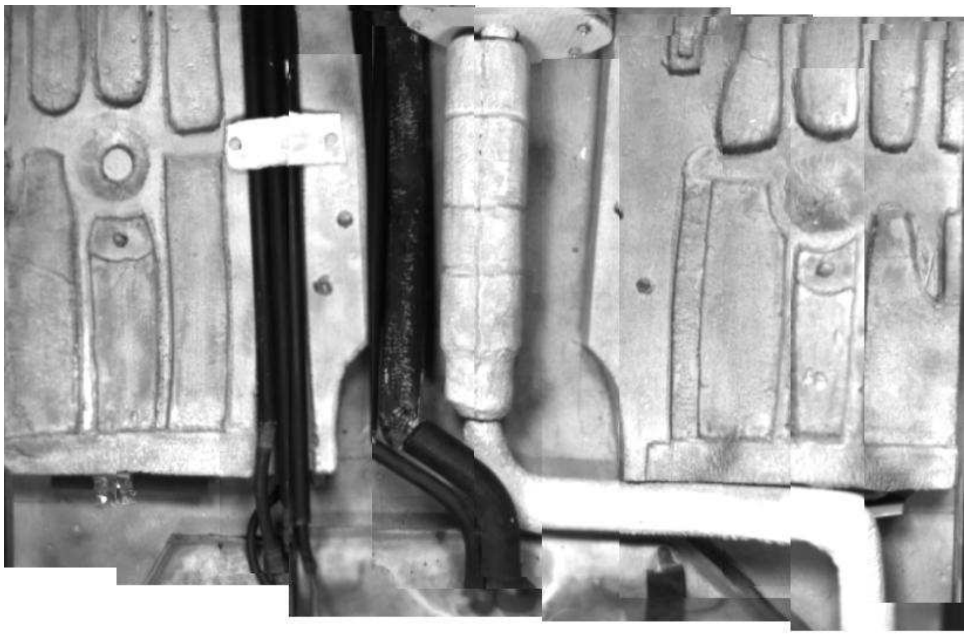


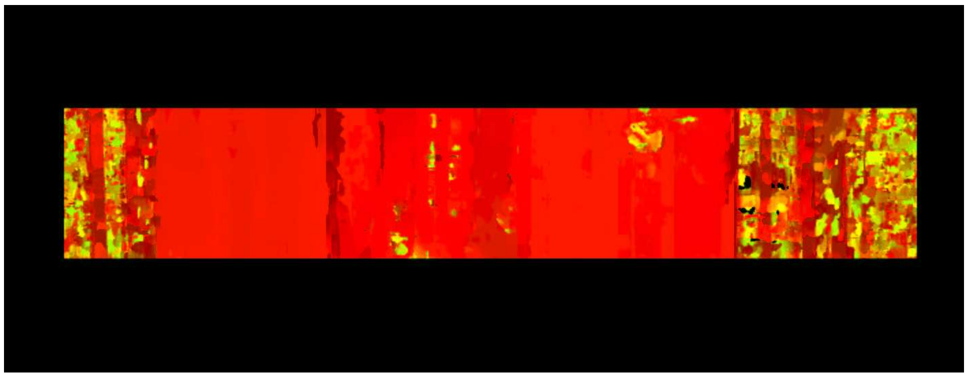



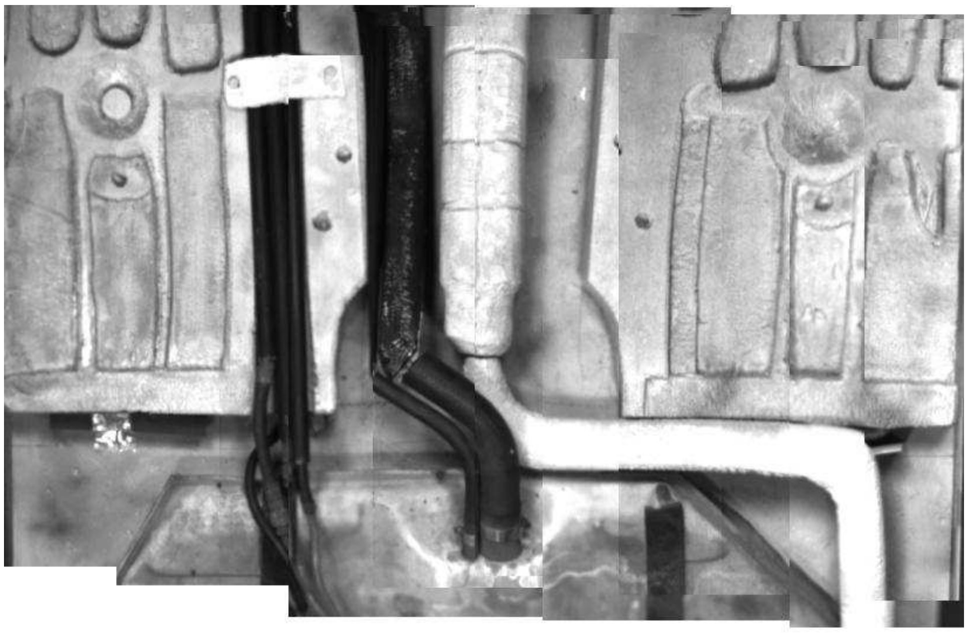



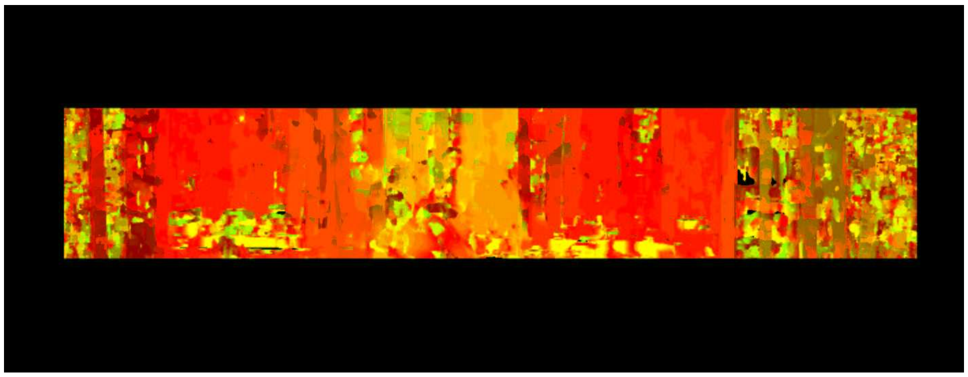


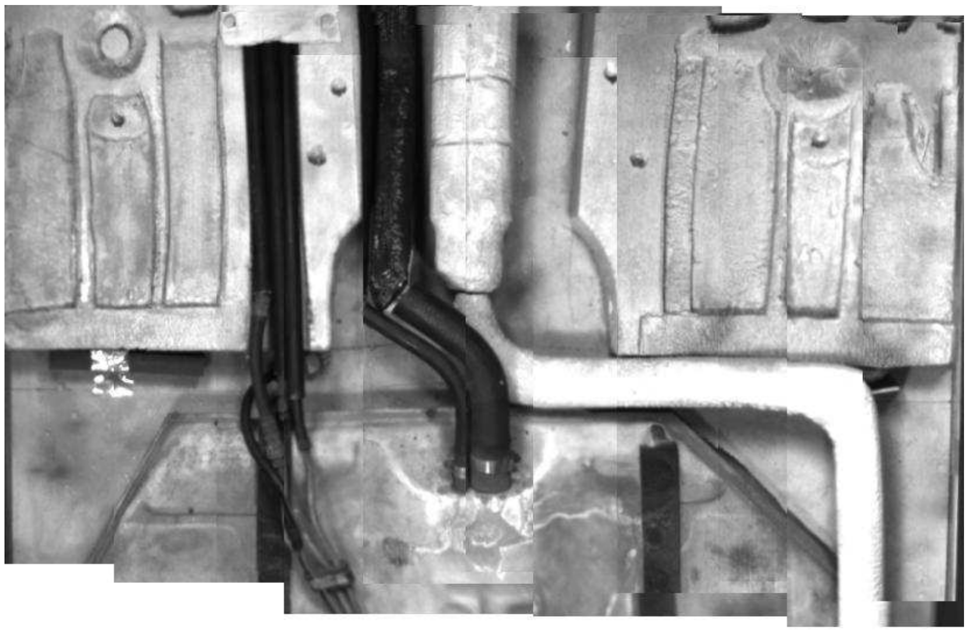


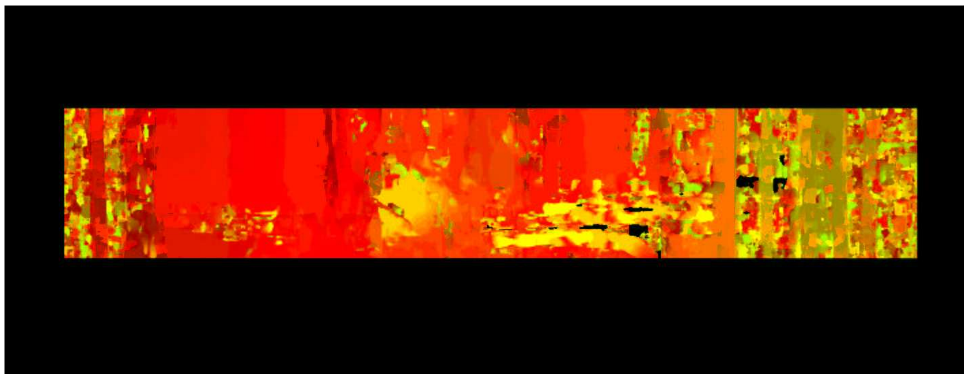


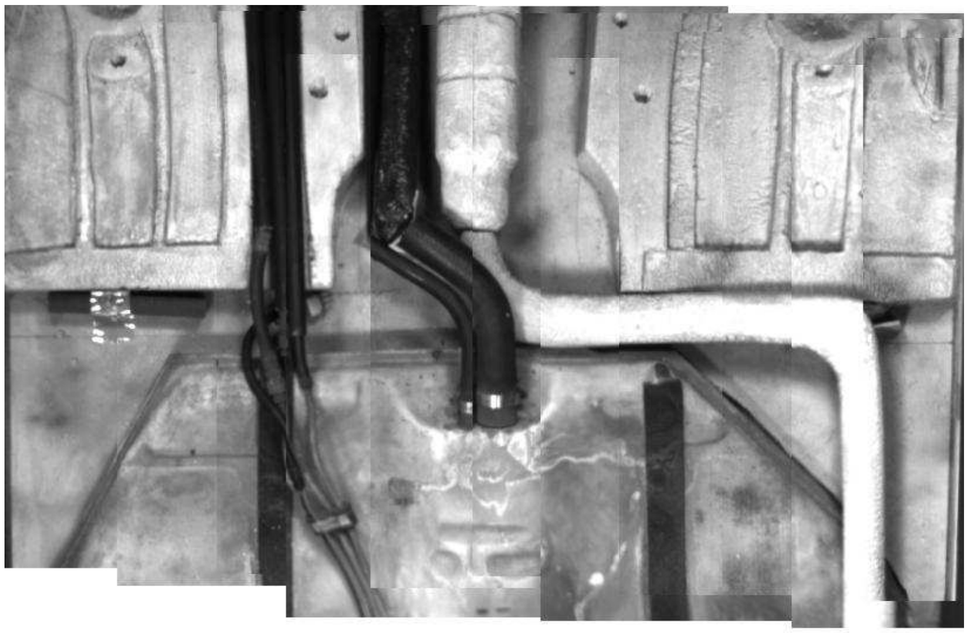
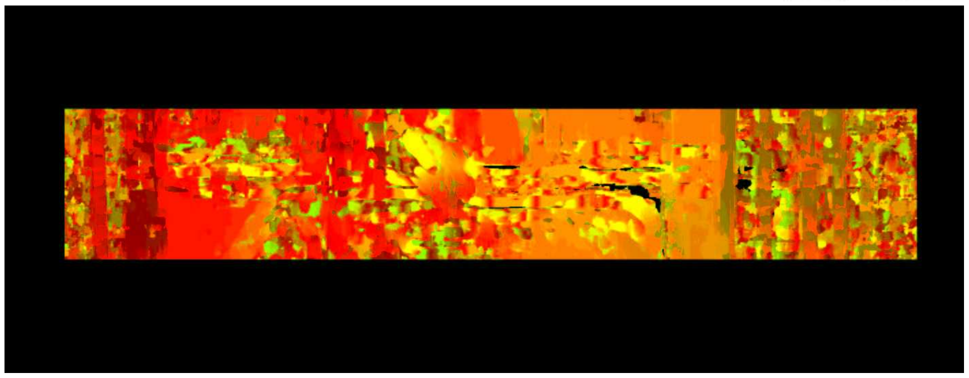


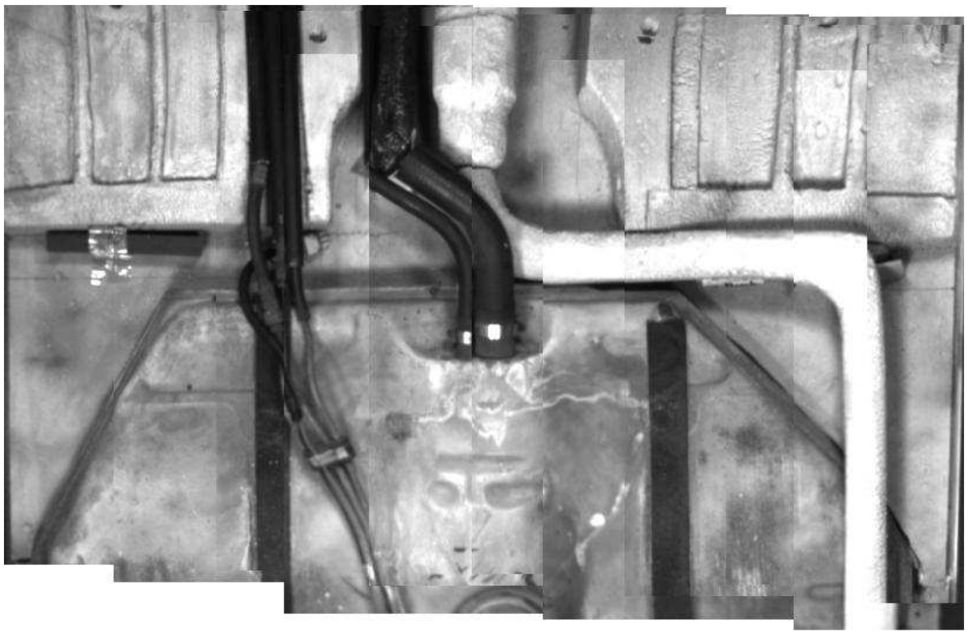


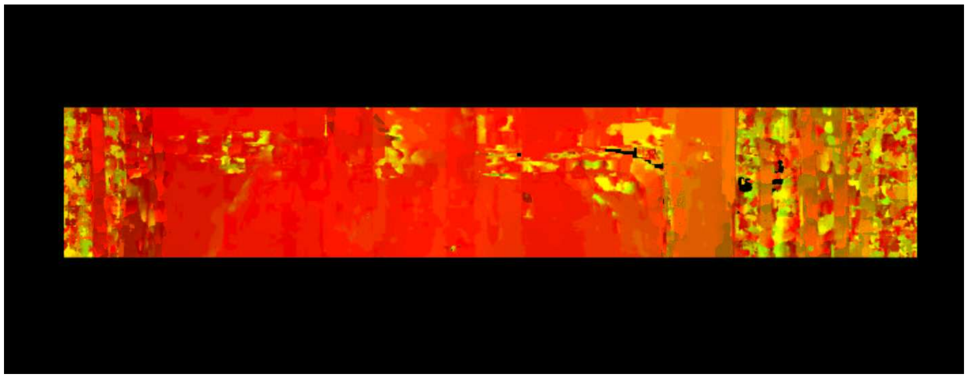


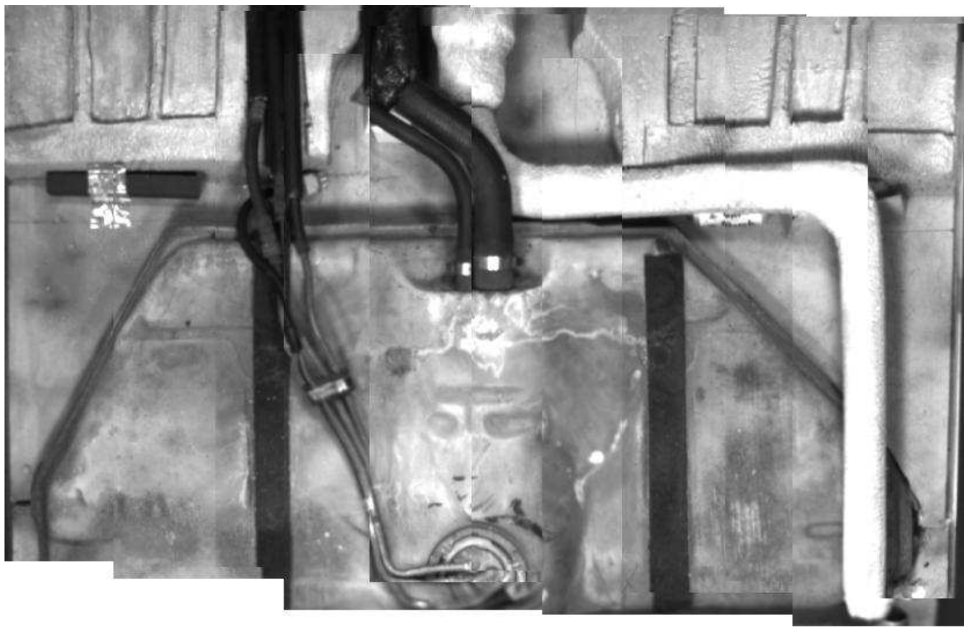


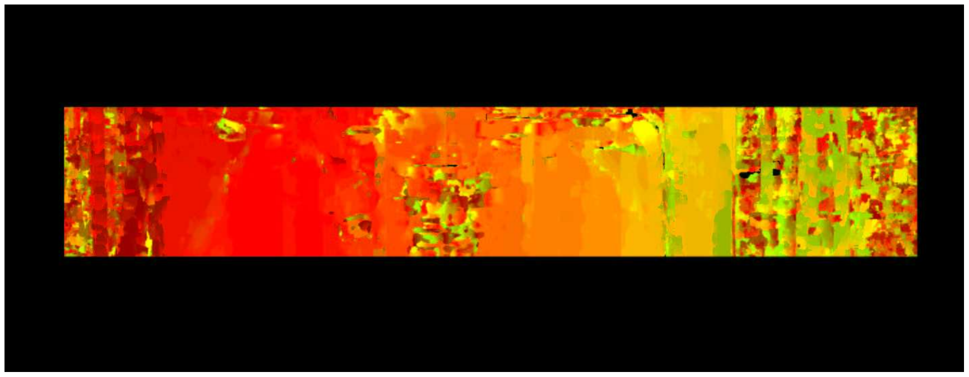


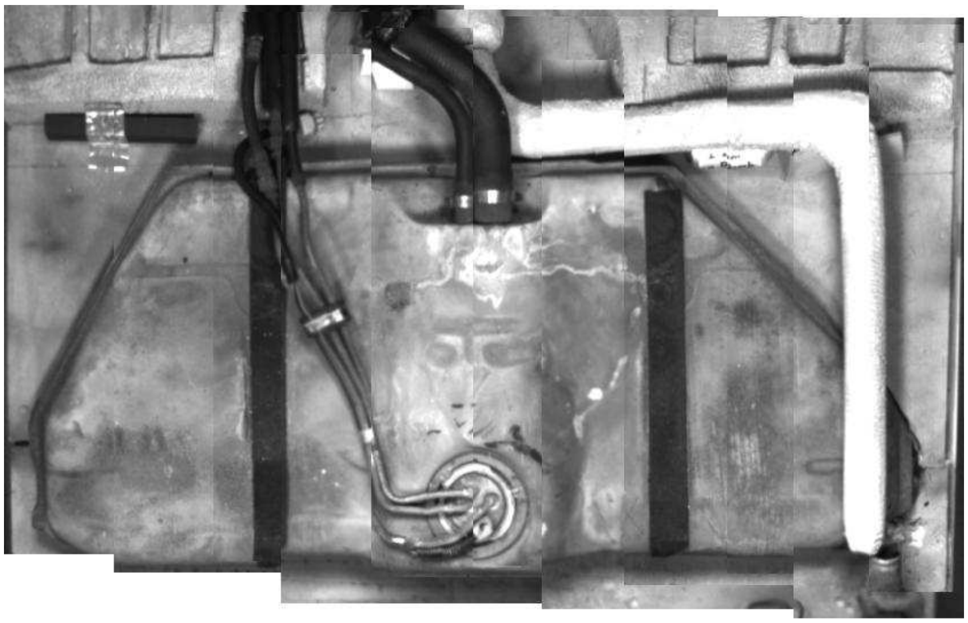



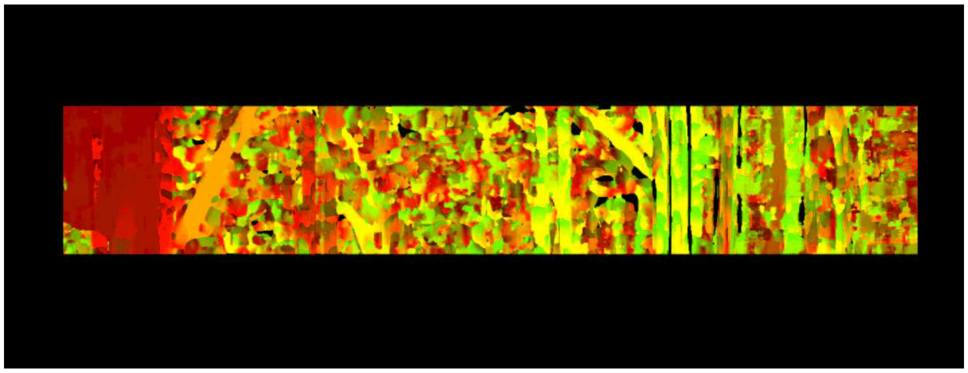


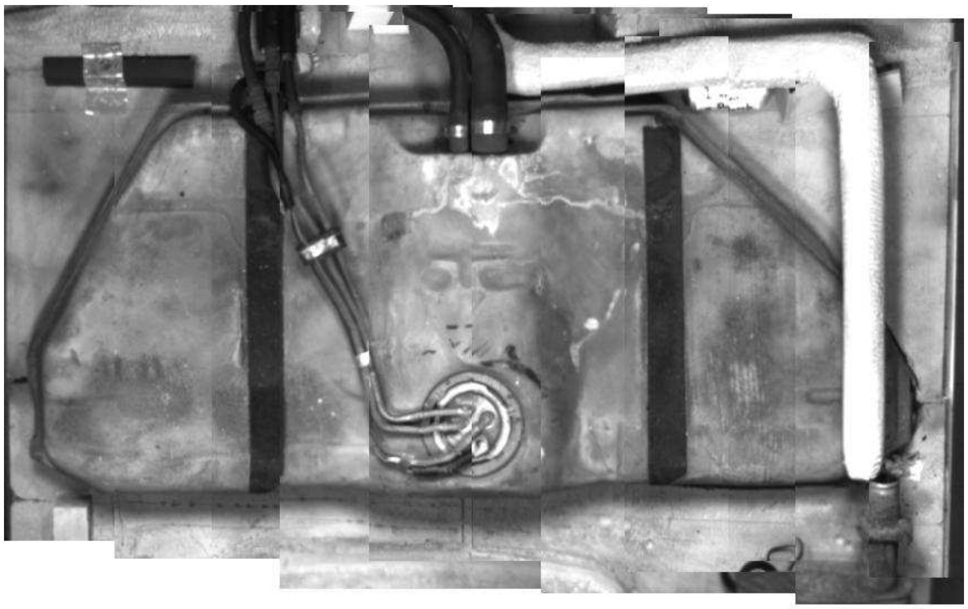


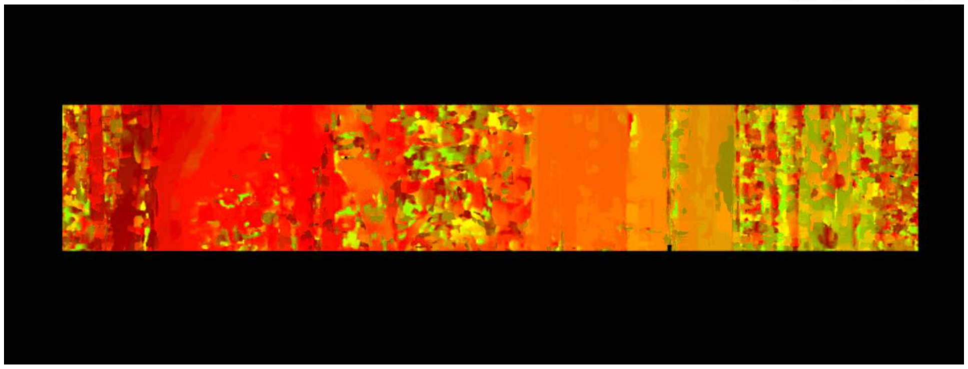


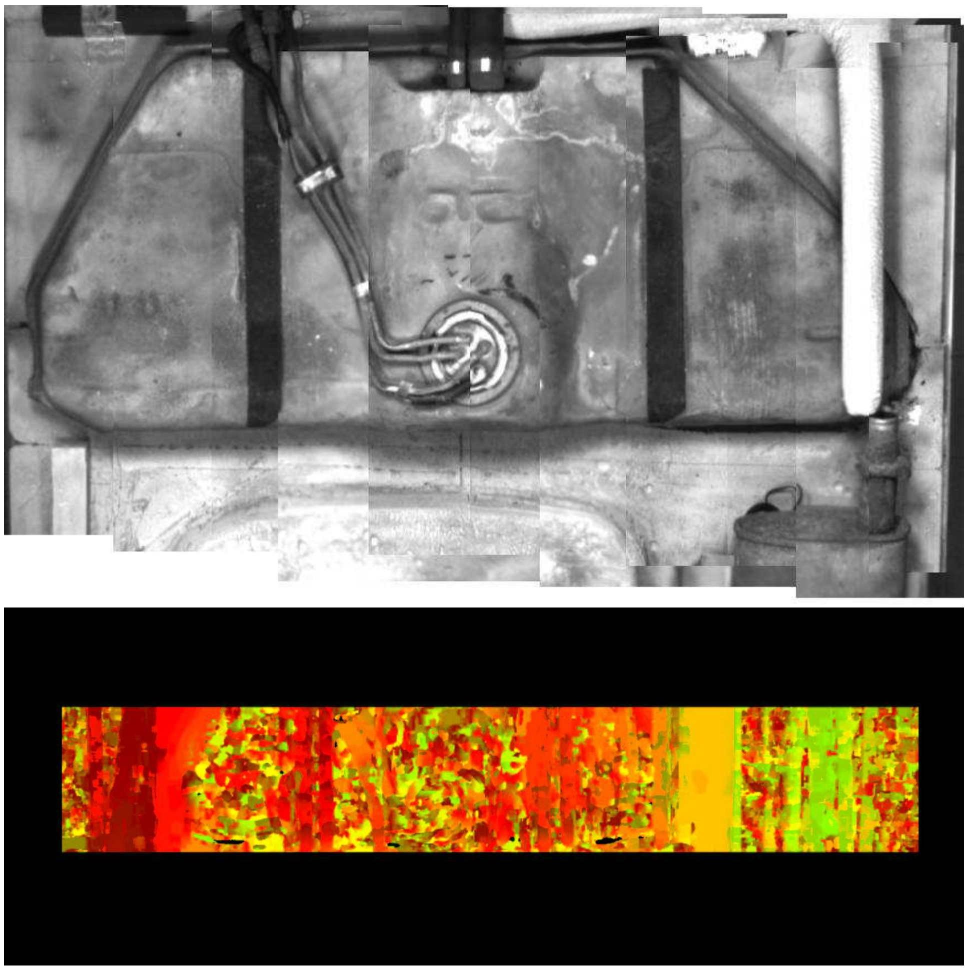


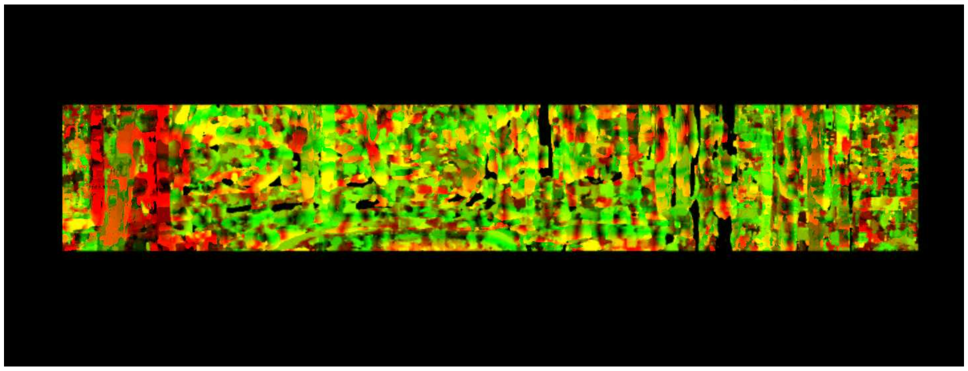


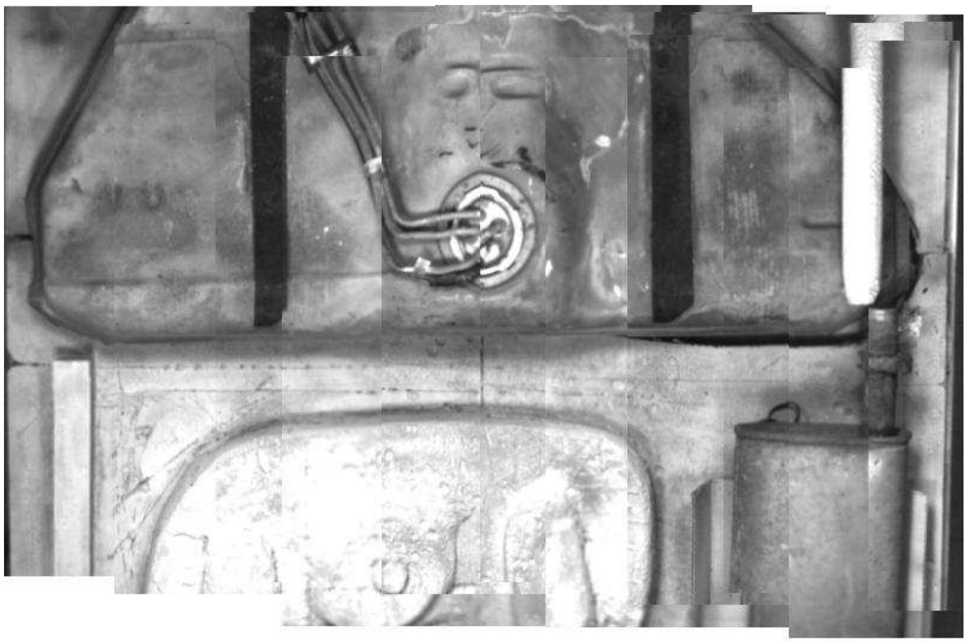












Appendix C-Paper: Paul Dickson, Mosaic Generation and System Description for an Under-Vehicle Inspection System, Synthesis Project, University of Massachusetts, Amherst, Massachusetts, 2004.

Abstract

Since September 11, 2001, the threat of terrorism has become a greater concern for the United States. As the concerns have grown, so have the needs for technological innovations tailored for tighter security. This paper proposes a system to upgrade security at airports, sporting events, federal buildings, and other locations where there is threat based on vehicles carrying contraband into the facility. The proposed under-vehicle inspection system uses an array of cameras and mosaicing techniques to generate views of a vehicle's undercarriage from different perspectives. These views provide a pseudo three-dimensional view of the undercarriage in a graphical user interface. Because the system must also run in a timely manner for it to be useful, a timing model of the system is given and discussed. This model is used to create several possible hardware configurations of the final system and to discuss the ramifications of certain aspects of the system design.

Introduction

Motivation

Inspection stations for vehicles at locations under terrorist threat have become a part of life, and one of the locations checked on vehicles is the undercarriage. Two main techniques are currently used to perform this inspection. The first technique has an inspector slide a mirror under the vehicle to view the undercarriage [1, 2]. This technique has the advantages of being inexpensive and easy to set up. The disadvantage is that the inspector cannot view all areas of the undercarriage because of the compounding problems of viewing angle, physical constraints on mirror placement, and occluding portions of the undercarriage. Typical examples of the mirrors used in these systems can be found on the Internet [1, 2]. Some work has been done to improve this technique by replacing the mirror with a camera [3]. This will solve some of the problems of viewing angles typically found with mirrors but has the same disadvantages.

The other typical method for inspecting an undercarriage uses an inspection bay. This has the distinct advantage that the inspector can see all portions to of the undercarriage and can easily focus on any suspicious portion. The disadvantage is that this both exposes the inspector to danger and requires the construction of an inspection bay.

One other method of inspection is beginning to be used but has not as yet gained the widespread acceptance of the first two methods. In this method, a vehicle drives across a line of cameras that capture images of the undercarriage. The proposed system is based on this method. The only system currently on the market simply records the input from the cameras to video tape and allows the inspector to inspect the undercarriage by moving the tape forward and backward in a viewer[4]. Like the previous methods, this technique does not allow an inspector to view the entire undercarriage simultaneously.

System overview

The goal of this paper is to describe the Under Vehicle Inspection System (UVIS), which will allow a complete inspection of the vehicle undercarriage to occur. The overall layout of the system is similar to that of the commercial system with the line of cameras that the vehicle drives over and appears in Figure 48. Images are continuously taken as the vehicle drives over the cameras. These images are then mosaiced together to generate perspective mosaics of the vehicle undercarriage from five different viewing directions. These mosaics are displayed together within a GUI to an operator who performs a visual inspection.

Prior work

The mosaic generation techniques used in UVIS were originally designed to create a compact representation of aerial video of forest canopies used for envi-

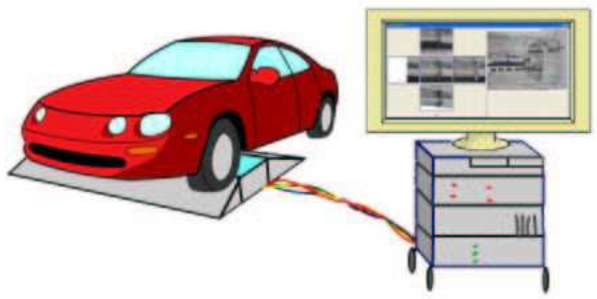


Figure 48: High-level system design

ronmental monitoring [5]. The techniques were further refined for construction of mosaics from camera movement that was primarily translational [6]. The translational shift model for a stationary forest with a plane flying over it at constant height is, for all practical purposes, the same as a stationary camera set with a vehicle driving over it in UVIS. In both cases only one part of the model is moving, and the movement is translational.

One of the main elements of previous work has been adopted by UVIS is the creation of mosaics with different perspectives, or stereo mosaics [5, 6]. These techniques will be described in detail in sections 2.2.4 and 2.3.3. The stereo mosaicing techniques were originally developed for use with rotating cameras [7, 8, 9]. The original method proposed by Huang and Hung [7] used a pair of rotating cameras to create stereo mosaics. This idea was then taken by Peleg and Ben-Ezra [8] and Shum and Szeliski [9] and refined to require only a single off-center rotating camera. As was discovered previously [5, 6], techniques based solely on rotation do not translate well to motion that is primarily translational.

The basis of stereo mosaic generation is that motion parallax is used to create the perspective images. Implicit in this technique is that the depth difference between objects in the scene are small relative to the distance between the cameras and the scene. For example, the manifold projection by Peleg and Herman [10] cannot create seamless mosaics within the UVIS domain because the depth differences between different parts of the undercarriage can approach 25% of the distance between the cameras and the vehicle. A previous version of UVIS [11] included the local alignment work of Shum and Szeliski [12] and an efficient ray interpolation technique based on the local alignment techniques in the matching phase of the mosaicing process. The new system represents a marked change in methodology, shifting from the aforementioned alignment techniques that specialize in unknown camera to object motion matching to a more rigid epipolar geometry-based matching approach that searches only specified regions for matches.

The previous version of UVIS [11] showed that it is possible to create relatively seamless mosaics within the confines of a system where the cameras are situated close to the object. It successfully demonstrated that the proposed system could effectively upgrade current inspection techniques by using a camera array to provide an inspector with a two-dimensional (2D) representation of three-dimensional (3D) information. This system relied on image acquisition through a video web server to circumvent the use of specific frame grabbers for each camera. This setup did not allow for synchronized image acquisition or for the images to be taken at necessary frequencies. In addition, this system failed to use any of the camera position information or vehicle rigid-body information implicit in the problem domain. The new system described here solves some of the problems from the previous version and uses information previously ignored. A time analysis is performed to determine the true feasibility of the system.

Design considerations

As described previously [11], a 3-inch camera separation was empirically determined to create acceptable mosaics. Applying the 3-inch camera separation in the direction of vehicle motion means a vehicle speed of approximately 5 mph over the cameras for a frame rate of 30 frames per second. The system can robustly create mosaics at higher vehicle speeds but a speed of 5 mph returns good mosaics. The ability to handle faster and varying speeds is important because it is not realistic to believe that drivers can maintain a constant speed while driving over the cameras.

Perspective distortion still has a negative effect on the matching in a mosaicing process. This effect can be reduced by increasing the distance between the cameras and the undercarriage to reduce the motion parallax. Because it is not feasible to increase this distance by placing the cameras lower, which would require the construction of a ditch for camera placement, we instead fold the optical path of the cameras (Figure 49). This allows the separation between the cameras and the undercarriage to increase without increasing the height of the camera bed, which would have the effect of creating a large speed bump (Figure 48).

The next aspect of design consideration is the camera field of view. The trade offs are that the smaller the field of view, the smaller the perspective distortion and motion parallax and the easier matching and mosaicing of the images; the larger the view, the better the information gained from the perspective mosaics. A field of view of 72^o was chosen because it created mosaics with good perspective without creating parallax sufficient to break the matching function.

Several environmental issues are associated with an under-vehicle inspection system. The major ones are lighting the undercarriage, dealing with weather as it affects the undercarriage, and eliminating grit buildup on the surface of the camera house. These issues have not been addressed as yet.

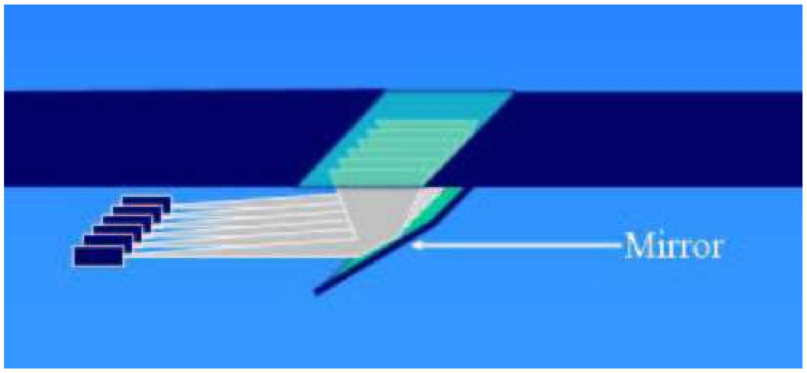


Figure 49: Camera position within the platform used to bend the optical path

System description

The high-level system description of UVIS can be seen in Figure 48. The platform that the vehicles drive appears in Figure 50. As mentioned, the camera separation is 3 inches. The transparent surface has yet to be determined, but glass or Lexan is most likely to be used. Images are taken constantly as the vehicle drives over the platform. These images are taken by a series of image acquisition machines as pictured in Figure 51 and discussed in section 2.1 and transfered to a series of machines that perform all actions associated with the crosswise mosaicing, section 2.2. These crosswise images are then transfered to the lengthwise mosaicing computers as discussed in section 2.3 before being passed to the GUI discussed in section 2.4. Section 2 also details the algorithms used and the layout of the timing model. Figure 52 shows how the steps described in section 2 take the images and first mosaic them in a crosswise manner before mosaicing them in a lengthwise manner in order to create a complete single image.

Section 3 will describe the parameters used in the model and specifics of the system. In section 4 the exact run time of the system proposed will be discussed, and in section 5 the model will be reevaluated with an emphasis on possible ways of tuning the system to make it a feasible design. Appendix D contains the details about how the timing model was determined and Appendix E contains details of the experimental model.

Acquisition

Synchronized images are continuously taken as the vehicle drives over the platform. These images are taken using a still-to-be-determined number of machines, each of which has an as yet unknown number of firewire cameras attached. The cameras used are black-and-white Point Grey DragonFlys, chosen because they require no frame grabbers, auto synchronize, and take 640x480 pixel 256 gray scale images. For a description of how the number of images

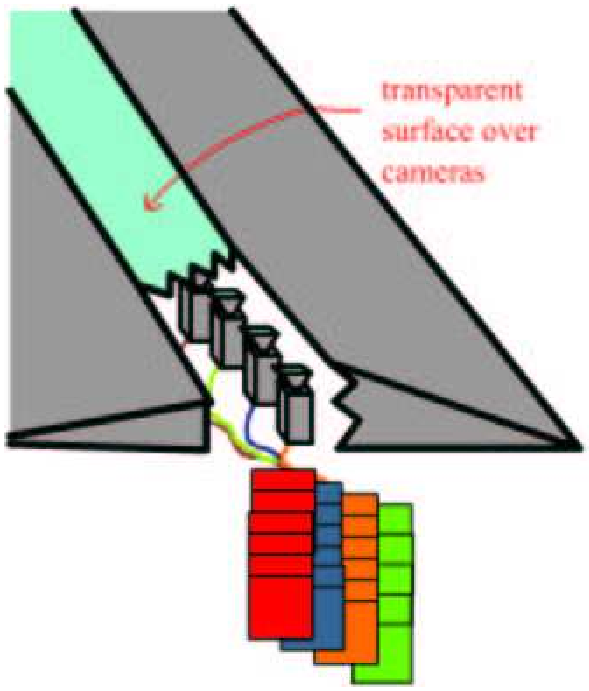


Figure 50: Camera platform; the bent optical path is not shown in order to simplify the image

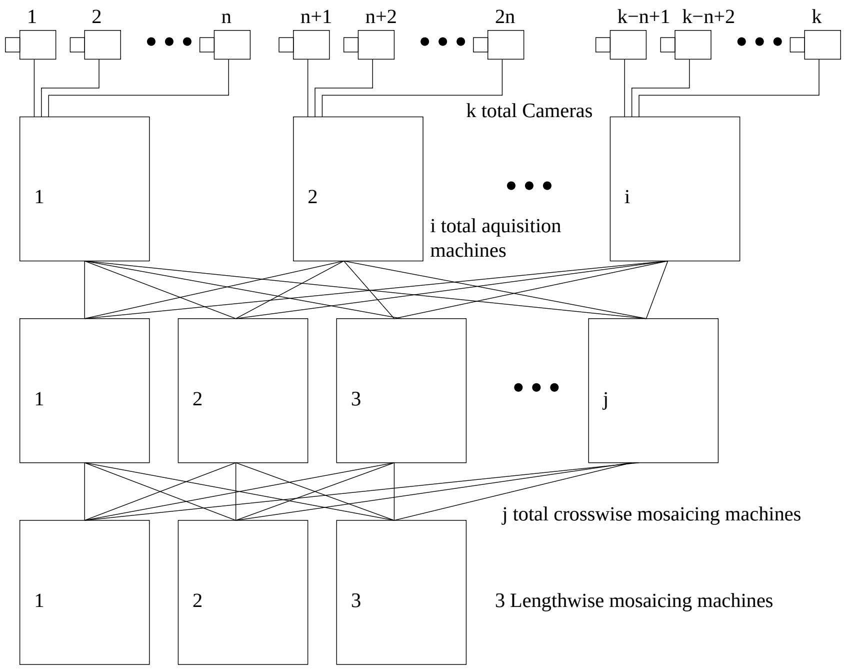

1
2
n
n+1
n+2
2n
k−n+1
k−n+2
k
k total Cameras
1
2
i
i total aquisition machines
1
2
3
j
j total crosswise mosaicing machines
1
2
3
3 Lengthwise mosaicing machines

Figure 51: Diagram of the UVIS operation

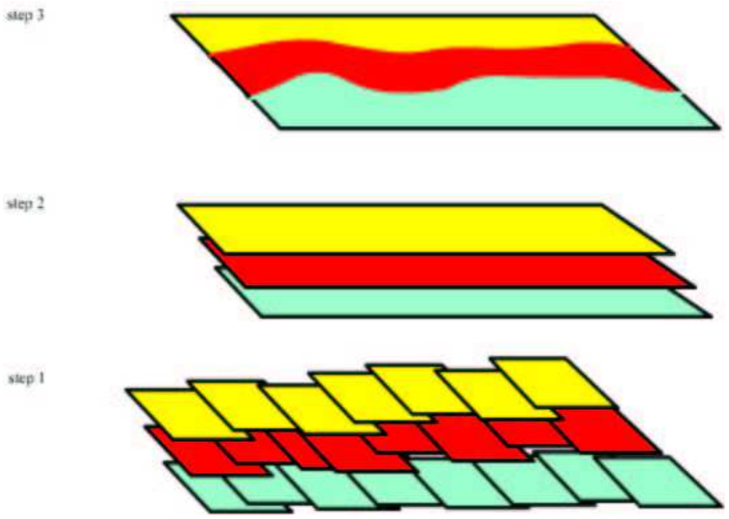


Figure 52: Mosaic building step

taken was determined, see Appendix D. The acquisition machines transfer the images as a set of simultaneous images (images from all of the cameras in the platform that are taken at the same time) to the cross-mosaicing machine.

At this time no exact image transfer time data can be listed because unknown problems are slowing down the image transfers during testing; a full explanation appears in section Appendix D.

Crosswise mosaicing

Rectification

Prior to mosaicing, a rectification step is performed on each image to remove all distortion intrinsic to the camera lenses. Each camera has been calibrated to determine both the intrinsic and extrinsic parameters. These parameters are used to modify each raw image during the rectification step. The rectification and calibration model used are those of Kovalenko [15]. The rectification process has been found to take the following time:

$$T_r = C * 0.2100 \tag{6}$$

T_r is the rectification time and C is the number of cameras in the camera platform.

Normalized cross-correlation matching

The crosswise matching process used in UVIS relies heavily on normalized cross correlation, which is the method used to match points in our algorithm. As such we will begin with a description of a normalized cross-correlation matcher (NCCM) and the algorithm used to compute the NCCM within UVIS.

NCCM is an established method within the field of computer vision for matching regions. It uses two equal-size rectangular portions of images. The first step is to determine the average pixel value for each rectangle:

$$\overline{A} = \frac{\sum_{i=1}^{N} Image_A[i]}{N} \tag{7}$$

$$\overline{B} = \frac{\sum_{i=1}^{N} Image_B[i]}{N} \tag{8}$$

$\overline{A}$ and $\overline{B}$ are average pixel values in images A and B within the correlation window and N is the size of the correlation window. N will be defined in the following sections and $Image_A[i]$ and $Image_B[i]$ refer to specific pixels within the correlation windows within images A and B. $\overline{A}$ and $\overline{B}$ are then used to determine the variance of each correlation window, Var_A and Var_B:

$$Var_A = \frac{\sum_{i=1}^{N} (Image_A[i] - \overline{A})^2}{N-1} \tag{9}$$

$$Var_B = \frac{\sum_{i=1}^{N}(Image_B[i] - \overline{B})^2}{N-1} \tag{10}$$

These lead to the determination of the match score ρ:

$$\rho = \frac{\sum_{i=1}^{N}(Image_A[i] - \overline{A}) * (Image_B[i] - \overline{B})}{(N-1) * \sqrt{Var_A * Var_B}} \tag{11}$$

As can be seen from 11, the regions with the highest variance will give the most accurate match score.

Crosswise match

The algorithm for the crosswise match appears in Figure 53. The match starts at one end of the line of cameras in the platform and continues to the other end, matching between each pair of cameras. The first step is to load in the assumed match location.

One of the chief improvements of this new algorithm over the old algorithm [11] is that it accounts for the epipolar geometry between cameras. The rectification step described above rectifies the images and at the same time creates epipolar lines between the images. The idea behind epipolar lines is that an object in space will appear on a corresponding line in both images of that space. In the context of UVIS where the camera locations are known, this means that everything that appears in one horizontal line in one image will appear on a corresponding horizontal line in the image next to it. At the time of calibration, an assumed match point can be created between every pair of cameras based on the camera locations. These match points should give a set vertical matches that do not change as the epipolar lines are set, barring physical misalignment of the image. The horizontal match can be assumed for an average distance between vehicle and cameras.

These assumed match locations are then refined by the algorithm. The match locations cannot be assumed because the horizontal match will vary with vehicle height over the cameras (i.e., the motion parallax varies with distance from cameras, so the location of a specific object in one image will vary in the other depending on height). Therefore, a first pass over the image is performed to determine suitable locations for running NCCM. This first pass is run over a subsample of the first image of the image pair. The first pass searches a window within the first image of size $(width - bufferX) * (height - bufferY)$, where $bufferX$ and $bufferY$ are given in equations 12 and 13, respectively.

$$bufferX \approx 2 * (W_s + W_c + 2) + offX \tag{12}$$

$$bufferY \approx 2 * (W_s + W_c + 2) + offY \tag{13}$$

The buffers act as cushions that keep the search within points guaranteed to be in both images. W_s is the radius of the square search window, W_c is the diameter of the square correlation window used, and $offX$ and $offY$ are the assumed X and Y match location offsets. This window is then subsampled using an input incrementor to the function to get a total number of points P_c to check, as defined in 14.

$$P_c = \frac{width - bufferX}{inc} * \frac{height - bufferY}{inc} \tag{14}$$

On each of these locations, P_c, a test is performed to see whether the variance is sufficient to match well using NCCM. The test involves determining whether the variance over the given correlation window in the first image is greater than an input threshold, signifying that it is a good location for a match to be performed. It has been empirically determined that the time to perform a given check of the variance T_c is

$$T_c = 0.0108 * 10^{-6} * {W_c}^2 + 0.0211 * 10^{-6} * W_c + 0.1707 * 10^{-6} \tag{15}$$

If a given point has a sufficiently high variance and the maximum number of points, P_i, to be checked and input to the algorithm has not been reached, then a search is performed in the second image to locate an exact match. This search is performed within a square window radius W_s around the assumed match location. For every location within this search window within the second image, the correlation from the first image is compared with the correlation window in the second image using NCCM and the best match within the search window is recorded. This match has been empirically determined to take time T_n from 16.

$$T_n = 0.0273 * 10^{-6} * {W_c}^2 + 0.0301 * 10^{-6} * W_c + 0.3023 * 10^{-6} \tag{16}$$

These recorded match scores and locations are then placed in a 2D histogram of match locations relative to the assumed match location, with a single bin for every possible location within the search window. Only matches with values above a threshold are included in the histogram. The mode of this histogram is then taken as the match between the pair of images. This total search strategy is performed between all pairs of cameras with adjacent locations on the camera platform. The time to perform the basic function overhead consisting of image loading and match determination was determined empirically and is T_k:

$$T_k = 0.0012 * C + 4.9167 * 10^{-5} \tag{17}$$

This means that the total time to perform the crosswise match T_{tc} is

$$T_{tc} = (C - 1)(P_c * T_c + P_i((2 * W_s + 1)^2 * T_n)) + T_k \tag{18}$$

Crosswise mosaicing

The mosaicing process takes the output match scores from the matching program and uses these to construct three crosswise mosaics. These mosaics form

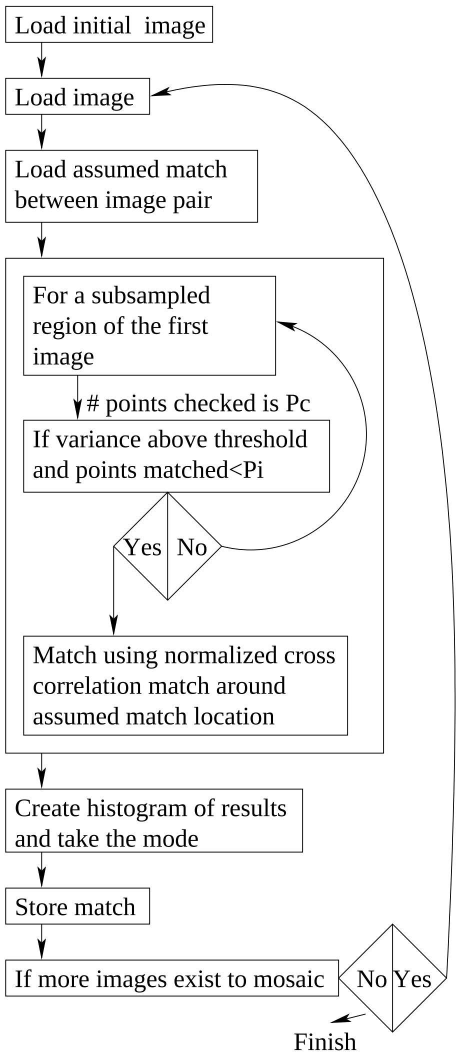

Load initial image
Load image
Load assumed match between image pair
For a subsampled region of the first image
# points checked is Pc
If variance above threshold and points matched<Pi
Yes
No
Match using normalized cross correlation match around assumed match location
Create histogram of results and take the mode
Store match
If more images exist to mosaic
No
Yes
Finish

Figure 53: Crosswise matching algorithm

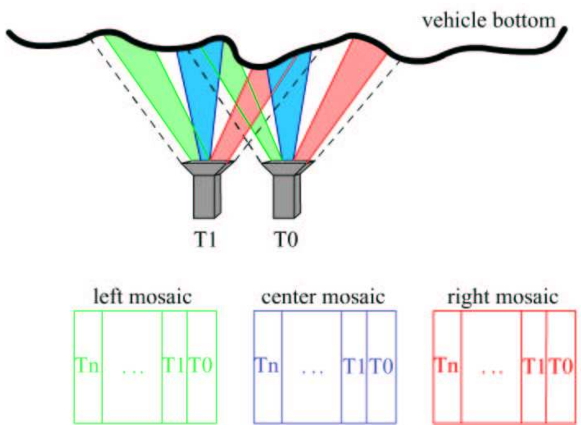


Figure 54: Example of how slices are taken from the original images to create perspective mosaics

the basis of the perspective mosaics used to give the 3D information. When a mosaic is formed, it is created from various images that are stitched together to create one continuous image. Controlling where the portions of the images stitched together come from in the original image controls the perspective view of the created mosaic. Figure 54 shows how the slices from the original images can be taken to create left, center, and right mosaics. The left and right mosaics are both created by stitching together the regions from the extreme edge of the original images. The center mosaic is created by stitching together the regions from the center portion of each image. The time to create these mosaics is T_{cm} and defined in 19.

$$T_{cm} = 0.0042 * C + 0.0236 \tag{19}$$

These crosswise mosaics are then transfered to three separate computers using the message passing interface (MPICH) on which the lengthwise mosaics are created.

Lengthwise mosaics

Crosswise mosaic transfer

The time required for transfer of the mosaics between computers is largely a function of the size of the images and hence the amount of data transferred. As such it is necessary to know how the size of the mosaics changes with the number of cameras. The *width* of these mosaics can empirically be defined as

$$width = 586 + 54 * C \tag{20}$$

The height of these mosaics can be empirically determined to be 531 pixels. The height of the crosswise mosaics varied but the maximum of all of the tests was 531, so it was chosen as a good upper bound for the timing equation. Hence the time to transfer a crosswise mosaic T_{cmt} can be written as

$$T_{cmt} = width * height * 4 * 2.101 * 10^{-8}$$

$$T_{cmt} = (586 + 54 * C) * 4.463 * 10^{-5}$$

$$T_{cmt} = 2.410 * 10^{-3} * C + 0.02615 \tag{21}$$

Lengthwise matching

The lengthwise matching function uses the same basic principles as does the crosswise mosaicing, the main difference between the two matching techniques being that there are no assumed matches in the lengthwise direction. Lengthwise assumed matches cannot be generated because they would have to rely on an assumed constant speed of the vehicle over the platform, which is not a valid assumption.

The lengthwise match algorithm has the same basic methodology as the crosswise match, as can be seen by comparing Figure 53 with Figure 55. Both algorithms function similarly to find the exact match location; they differ in that the lengthwise match must find an initial match instead of loading it from a file as the crosswise match does with the assumed match.

The initial matches are found in the lengthwise match process by following the same two phases as for the final match: finding high-variance regions and correlating those regions. A buffered window within the first image is again determined, this time with $buffer$ defined as

$$buffer \approx 2 * (W_{s2} + W_{cl} + 2) \tag{22}$$

W_{s2} is the radius of the second match search window and W_{cl} is the diameter of the normalized correlation window. The window to be subsampled is now of size $(width_l - buffer) * (height_l - buffer)$. This means that the number of points that have their variance checked, P_{c1}, is defined as

$$P_{c1} = \frac{width - buffer}{inc_1} * \frac{height - buffer}{inc_1} \tag{23}$$

inc_1 is an input variable to the function used for subsampling. The time to check the variance of a given point is the same as it was for T_c in the crosswise match, so T_{cl}, the time to check a given variance for the lengthwise match, can be written in terms of W_{cl}, the lengthwise correlation window diameter, as

$$T_{cl} = 0.0108 * 10^{-6} * {W_{cl}}^2 + 0.0211 * 10^{-6} * W_{cl} + 0.1707 * 10^{-6} \tag{24}$$

Again, for a given number of points defined by the input P_{i1} that have a sufficiently high variance, NCCM is run to find an exact match. Because there

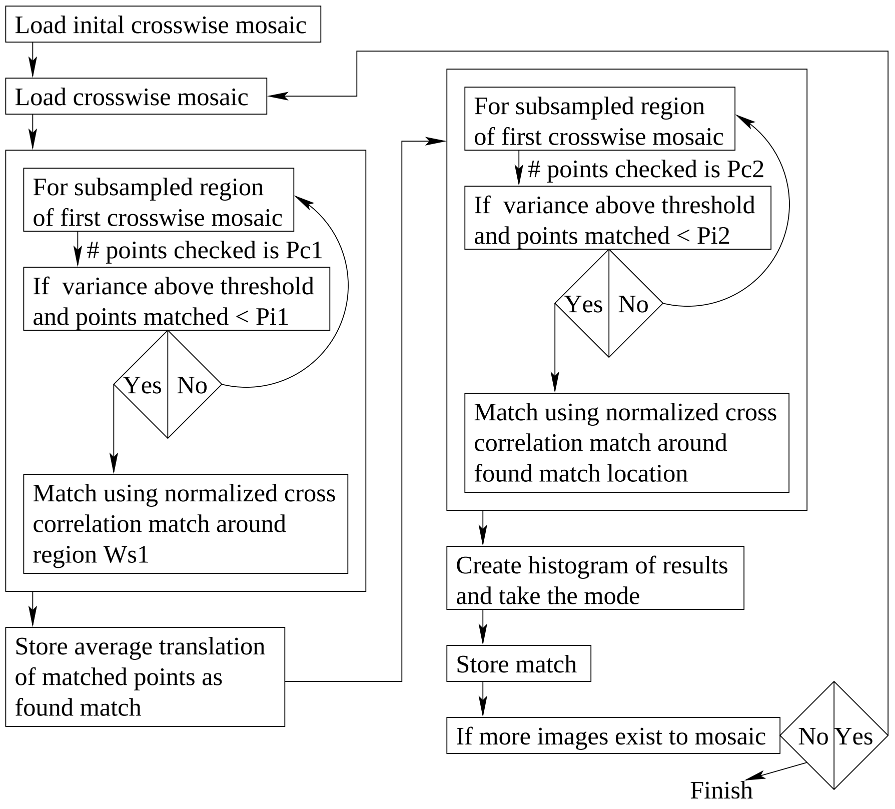

Load inital crosswise mosaic
Load crosswise mosaic
For subsampled region of first crosswise mosaic
# points checked is Pc1
If variance above threshold and points matched < Pi1
Yes
No
Match using normalized cross correlation match around region Ws1
Store average translation of matched points as found match
For subsampled region of first crosswise mosaic
# points checked is Pc2
If variance above threshold and points matched < Pi2
Yes
No
Match using normalized cross correlation match around found match location
Create histogram of results and take the mode
Store match
If more images exist to mosaic
No
Yes
Finish

Figure 55: Lengthwise matching algorithm

is no assumed speed of the vehicle to indicate where in the second image the point will match, an exhaustive search of the second image must be performed. This search of the second window is for the height of the image, minus the buffer to keep the search in the matchable region, from the given point minus W_{s2} to plus W_{s2}. This search window W_{s1}, defined in 25, is assumed to contain the point because the vehicle motion will be primarily in the lengthwise direction and hence will not shift very far off a given point in the first image.

$$W_{s1} = (height - buffer) * (2 * W_{s2} + 1) \tag{25}$$

The time to check a given pair of correlations windows again functions as it did for the crosswise matching and hence T_{nl}, the time to correlate windows in the lengthwise direction, defined in terms of W_{cl} is

$$T_{nl} = 0.0273 * 10^{-6} * {W_{cl}}^2 + 0.0301 * 10^{-6} * W_{cl} + 0.3023 * 10^{-6} \tag{26}$$

When all P_{c1} points have been matched, the results are averaged to create first-match data equivalent to the assumed matches from the crosswise match. The second phase of the lengthwise match now begins by searching a new subsampled portion of the initial image. The window subsampled is the same as that defined above. A new incrementor input to the function, inc_2, is now used to create the subsampling of the window, so the new number of points to have their variance check, P_{c2}, is defined as

$$P_{c2} = \frac{width - buffer}{inc_2} * \frac{height - buffer}{inc_2} \tag{27}$$

P_{i2} points are then matched using NCCM to determine the exact match offsets, where P_{i2} is an input to the function. The time to perform each of these matches is T_{nl}, defined in 26. The search window is defined as a square around the found match with radius W_{s2}. The best results of this search are again recorded for each point up to P_{i2}.

As with the crosswise match, the results are stored in a histogram of match locations and the mode of these locations is stored as the overall match. Again, the only results put into the histogram are those that have match scores greater then a given threshold. The time to perform all of the basic algorithm steps, T_{kl}, was determined empirically and is defined as

$$T_{kl} = (7.4478 * 10^{-4} * C + 0.0088) * I + (-7.7525 * 10^{-4} * C - 0.0022) \tag{28}$$

C is the number of cameras in the camera platform and I is the number of synchronized image sets taken as the vehicle drives over the platform. This means that the total lengthwise match time, T_{tl}, can be defined as

$$T_{tl} = (I-1)(P_{c1} * T_{cl} + P_{i1}(W_{s1} * T_{nl}) + P_{c2} * T_{cl} + P_{i2}((2 * W_{s2} + 1)^2 * T_{nl})) + T_{kl} \tag{29}$$

Separate matching functions are performed on each of the three crosswise perspective mosaics created. This separate matching will ensure that each of the final perspective mosaics generated has good continuity within itself because the images in each will have been matched specifically to those stitched around them.

Lengthwise mosaicing

The lengthwise mosaicing process functions the same as the crosswise mosaicing function. For the mosaics of the left and right crosswise mosaics, only center mosaics are generated, which creates complete left and right perspective views of the undercarriage. Three separate lengthwise mosaics are generated from the center crosswise mosaics. The perspective mosaics are from the top, center, and bottom to create the three remaining perspective views. The same mosaicing function used to generate center mosaics is used to generate center and perspective mosaics together with just a parameter change. The time to run the lengthwise mosaicer T_{lm} is empirically defined as

$$T_{lm} = (0.0017 * C + 0.0189) * I + 0.0011 * C + 0.0253 \tag{30}$$

T_{lm} is the time to generate the three mosaics from the central image. Times are not given to generate the single mosaics for the left and right perspective views because they take less time to run than do the three mosaic generations; they are run in parallel with the central mosaicing and thus do not affect the overall run time.

GUI

Length mosaic transfer

The lengthwise mosaic transfer is based on the same transfer rate ($2.101 * 10^{-8}$second per bit) as the crosswise mosaic transfer. The width is still defined by equation 20, but now the height is empirically defined by the equation

$$height = 497 + 66 * I \tag{31}$$

This height will vary in the real system as a result of the inconsistencies in vehicle speed but gives the necessary ball park figure for the model. This gives us the following lengthwise mosaic transfer equation:

$$T_{lmt} = width * height * 4 * 2.101 * 10^{-8}$$

$$T_{lmt} = (586 + 54 * C)(497 + 66 * I) * 8.404 * 10^{-8}$$

$$T_{lmt} = 2.995 * 10^{-4} * I * C + 2.255 * 10^{-3} * C + 3.250 * 10^{-3} * I + 2.448 * 10^{-2} \tag{32}$$

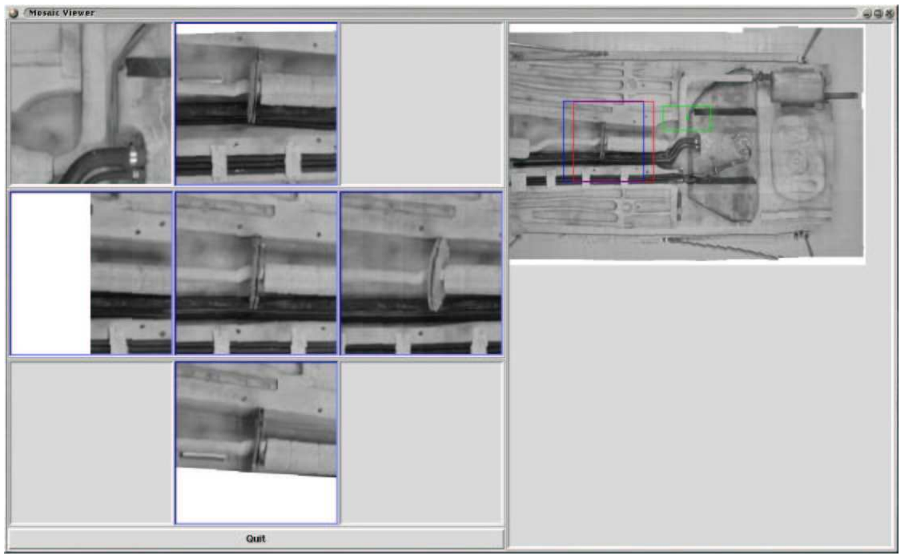


Figure 56: The GUI used to display the generated mosaics

GUI design

The GUI is the heart of UVIS. It allows the inspector to view the five perspective views in concert to present the 3D information. An image of the GUI being used appears in Figure 56. Six small images are on the left and the view of the entire undercarriage is on the right, which enables the operator to inspect the complete undercarriage. As the operator moves a box around the right-side image with the cursor, five images appear as a cross on the left side. These five images are windows into the five perspective mosaics. By moving this box, the operator is able to look around occluding objects within the undercarriage and get 3D information from the 2D representation. Because this inspection does not always yield satisfactory results, it is also possible for the operator view the original images from which the mosaics were generated. The operator can do this by selecting a region on the right image for which the original image is wanted. A sample of the GUI can be downloaded from the project web page [14].

Vision and Timing Models

This section is in two parts. Aspects of the vision model are discussed in section 3.1: known problems with the algorithms are explained and the best settings found for the parameters of the timing model are given. In section 3.2 the timing model is discussed in relation to the parameter settings. With this as a basis, models of actual system configurations and their attendant run times are proposed.

Vision model

Let us discuss the vision model by first considering each of its input parameters and then how each affects the algorithm function. The first input is the incrementor used to determine the subsampling of the crosswise images in equation 14. Increasing the size of *inc* decreases the number of points to have their variance checked, P_c, thus increasing the speed of the overall algorithm. If P_c is decreased too much, then there will not be P_i points with a variance over the given threshold and the match will take place with fewer than P_i points being matched. If *inc* is too small and $P_c$is too large, then the run time T_{tc} from equation 18 will increase, but the points matched, P_i, will all be from the top of the pair of images being matched and the overall match may falter as a result.

About 1 in 20 of the points that have their variance checked have a variance above the threshold used for these tests. With this in mind, an *inc* value of 5 has been suggested because it creates a P_c of 26 times P_i. The percentage of points that have a sufficiently high variance is completely dependent on the threshold value used. Currently, this threshold is a value that has been empirically found to return good results for matches. The amount of variance within a certain area is determined in part by the size of this area. In the future this threshold should be created as a function of the size of the correlation window W_c^2 instead of having a set value of 300 regardless of W_c as it does now. The setting of this threshold will affect the percentage of points P_c that have a variance that is high enough to use for P_i. As such, setting *inc* to a specific value and expecting that the points matched are a thorough sampling of the image space is not sound logic. In the future a rewrite of the subsampling should be performed that guarantees that all areas of the image have their variance tested equally in order to create a spatially unbiased match of the image pair.

The next input to be discussed is W_c, the diameter of the correlation window used in NCCM. Common experience has always held that the larger the correlation window used, the better the match found. This would suggest that the larger the W_c, the better the match used to create the crosswise mosaic. This has not been found to be the case within the context of UVIS. Our tests have shown that changing the size of $W_c$has little effect on the quality of the match found. Figures 57 and 58 show crosswise mosaics generated with different correlation window sizes. The mosaic in Figure 57 has fewer broken areas than does the mosaic in Figure 58. This suggests that within the context of UVIS a smaller correlation window returns better results. There are two possible explanations for this. The first is that the distortion intrinsic to images used in large motion parallax mosaicing makes it more likely for a larger window to experience more distortion than a smaller window, which negatively affects the ability of the larger window to match successfully. The second possible explanation is that because the threshold used to determine good match points is not a function of W_c, points with lower variance are getting chosen as good match locations and these low-variance regions are returning poor overall match scores as well as match scores with a spacial bias to the upper portions of the match region. This second possibility can be eliminated in the future by creating a threshold

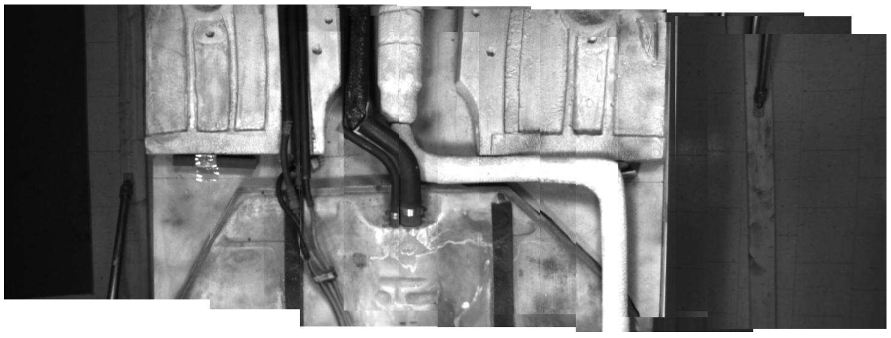


Figure 57: Crosswise mosaics with $W_c = 2$

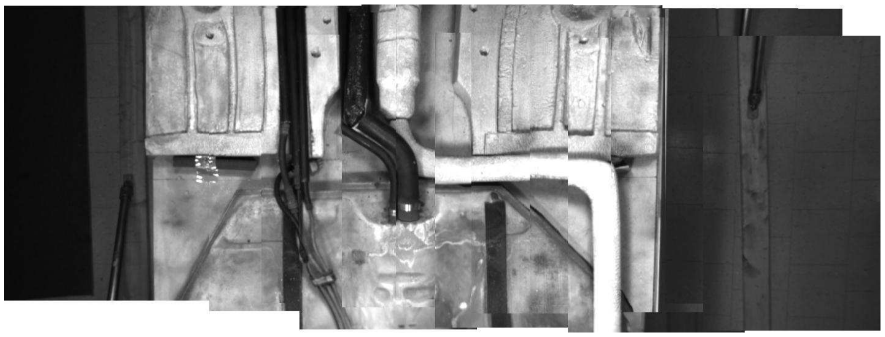


Figure 58: Crosswise mosaic with $W_c = 15$

for variance that is a function of W_c. Regardless, for the purposes of our model in this project, a value of 2 will be given to W_c within the crosswise matching model.

The next input to be considered is the size of the search window around the assumed match. The goal of the search window is to discover whether any small shift has occurred that would have thrown the assumed matcher off. The size of W_s, the radius of the search window, does not have to be large, and a value of 7 should be sufficient to discover shift. This search window does not take the height of the vehicle undercarriage into account. As the height of the vehicle varies, the overlap of the images will vary, which will cause massive shifts in the assumed match for any given undercarriage height change. A point that will have to be addressed in the future is that an initial match must be made and used to make appropriate changes to the assumed match before the crosswise matcher is run. This initial match would only have to find a match once and then propagate it through the crosswise mosaicing computers.

Those are all of the inputs to the crosswise matching and mosaicing functions except for the number of cameras, C, which will be left without a value, to be used later to show the scalability of the model. Let us now begin to address the inputs to the lengthwise matcher. The first input to consider is inc_1, which helps determine the size of P_{c1}. Again, the smaller the size of inc_1, the larger the size of P_{c1}; for the tests run inc_1has been set to 5. In the future, how the size of inc_1affects the distribution of P_{c1} should be investigated, and inc_1should be determined as a function of the width of the crosswise mosaics, which is determined by the number of cameras. The same theory should go into the formulation of inc_2, which determines P_{c2}. For the purposes of this model, inc_2will be set to 10 because it has been shown empirically to return good results, not because of any well-thought-out reasoning. $P_{c1}$and P_{c2}both determine the locations where matches are found and hence both should be checking locations throughout the entire images matched that can be used for the exact points $P_{i1}$and P_{i2}. The process of guaranteeing the spacial distribution has not yet been accomplished.

Again, the number of points checked, $P_{c1}$and P_{c2}, that have sufficient variance to be used to determine an exact match is determined by a threshold that is input to the algorithm. For these tests a threshold of 300 is used because it has been found empirically to work. As was described with the crosswise match, this threshold should be determined in terms of the size of the correlation window, which is set by W_{cl}. The number of points used for the exact matches, $P_{i1}$and P_{i2}, have been empirically set at 20 and 300, respectively.

Unlike the situation for the crosswise direction, the size of the correlation window seems to directly affect the quality of the mosaic generated, with the better mosaics being generated from matches performed with larger correlation windows. That said, the correlation window diameter, W_{cl}, of 15 appears to give a sufficiently good quality mosaic (Figure 59). It has also been discovered empirically that a search window with radius W_{s2} of 9 pixels is sufficient for good mosaic generation.

Timing model

With these set values for the timing model, we can rewrite equation 14 as equation 33:

$$P_c = \frac{640-82}{5} * \frac{480-22}{5} = 10222 \tag{33}$$

$offX$ is assumed to be 60 and $offY$ is assumed to be 0. By substituting in the value of W_c into equations 15 and 16, we get $T_c = 2.56 * 10^{-7}$ and $T_n = 4.72 * 10^{-7}$. With these numbers we can rewrite equation 18 as

$$T_{tc} = (C-1)(10222*2.56*10^{-7}+300((2*7+1)^2*4.72*10^{-7}))+0.0012*C+4.9167*10^{-5} \tag{34}$$

$$T_{tc} = (C-1)(0.0452) + 0.0012 * C + 4.9167 * 10^{-5}$$

$$T_{tc} = 0.0464 * C - 0.0451 \tag{35}$$

We then have the necessary information for determining the total time spent by each cross-mosaicing machine to rectify, match, and mosaic a crosswise set of images as seen in equation 36:

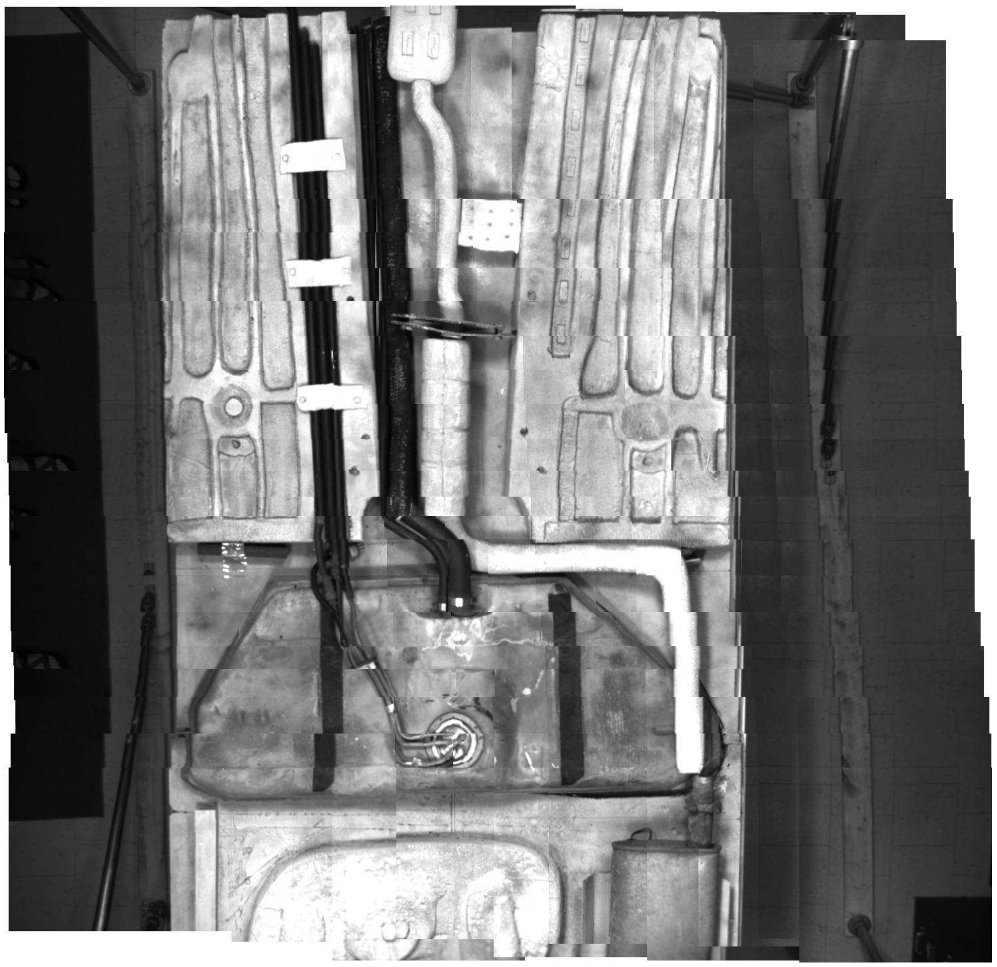


Figure 59: Lengthwise mosaic with $W_{cl} = 15$

$$T_{cwm} = T_r + T_{tc} + T_{cm} \tag{36}$$

$$T_{cwm} = 0.2100 * C + 0.0464 * C - 0.0451 + 0.0042 * C + 0.0236$$

$$T_{cwm} = 0.2606 * C - 0.0215 \tag{37}$$

The total time to complete the crosswise mosaicing T_{cwm} along with the time to transfer the three created crosswise mosaics will give us the total time taken per set of images as shown in equation 38:

$$T_{tcwm} = T_{cwm} + 3 * T_{cmt} \tag{38}$$

$$T_{tcwm} = 0.2606 * C - 0.0215 + 3 * (2.41 * 10^{-3} * C + 0.02615)$$

$$T_{tcwm} = 0.2678 * C + 0.0570 \tag{39}$$

Let us now rewrite the timing equations for the lengthwise matcher with the values from above. We can first rewrite equation 23 as follows in equation 40:

$$P_{c1} = \frac{640 + C * 54 - 56}{5} * \frac{531 - 56}{5} = 1026 * C + 11096 \tag{40}$$

With those values we can also determine the time $T_{cl} = 2.92 * 10^{-6}$, $T_{nl} = 6.70 * 10^{-6}$, and $W_{s1} = 9177$. We can then determine the number of points checked in the second phase, P_{c2}, using equation 27:

$$P_{c2} = \frac{640 + C * 54 - 56}{10} * \frac{531 - 56}{10} = 256.5 * C + 2774 \tag{41}$$

With these values we can write T_{tl} from equation 29 in terms of I and C, the number of images and cameras.

$T_{tl} = (I - 1)(3.745 * 10^{-3} * C + 1.955) + (7.4478 * 10^{-4} * C + 0.0088) * I + (-7.7525 * 10^{-4} * C - 0.0022)$

$$T_{tl} = (4.490 * 10^{-3} * C + 1.964) * I - 4.520 * 10^{-3} * C - 1.957 \tag{42}$$

With T_{tl}, the total time to run a lengthwise mosaic can now be determined:

$$T_{lwm} = T_{tl} + T_{lm} \tag{43}$$

$$T_{lwm} = (6.19 * 10^{-3} * C + 1.983) * I - 3.44 * 10^{-3} * C - 1.932 \tag{44}$$

We can now take T_{lwm} and combine that with both the time to transfer the mosaics onto the system and the time to transfer the three lengthwise mosaics from the system to get the total length time, T_{tlwm}, as seen in equation 46:

$$T_{tlwm} = T_{cmt} + T_{lwm} + 3 * T_{lmt} \tag{45}$$

$$T_{tlwm} = (7.089 * 10^{-3} * C + 1.993) * I + 5.735 * 10^{-3} * C - 1.832 \tag{46}$$

Results

Minimum time system

Now that we have a complete timing model for the system, we can determine the hardware necessary to complete the fastest possible system for generating the mosaics. We must assume that a faster transfer rate is possible from the image-grabbing machines than we have been able to simulate, because the numbers found would require the same number of input-grabbing computers as cameras. This does not take into account the fact that all of the computers are connected to the same network. The same assumption must be made for the transfer of the crosswise mosaics to the machines that will run the lengthwise mosaics, because the bottleneck present with the current transfer rates would slow the system to only as fast as the mosaics could transfer, with no perceptible increase in speed when the number of processors is increased.

Steady state run time

Although it is impossible with the current design to create a system that is guaranteed to function within a given time for any input number of images I, it is possible to create a system that will finish the crosswise mosaicing, for a given number of cameras, within a constant amount of time after the final image has been taken. This system will require enough processors so that there is always one coming free to handle a new crosswise mosaic every time a new image is taken. This number of processors can be determined by taking the time for a single processor to perform a complete crosswise mosaic T_{tcwm} and multiplying it by the speed of the cameras, 30Hz. Equation 47 gives the determination of this number of processors required in terms of the number of cameras in the platform C.

$$P_{cross} = \lceil T_{tcwm} * 30 \rceil \tag{47}$$

$$P_{cross} = \lceil (0.2678 * C + 0.0570) * 30 \rceil$$

$$P_{cross} = \lceil 8.034 * C + 1.71 \rceil \tag{48}$$

As can be seen from equation 48, this model is still not feasible because the system will take a minimum of 30 cameras, which would require 243 separate computers for the crosswise mosaicing alone, which is not an option. This would also require an additional 3 computers to handle the lengthwise mosaicing step, bringing the number of processors to 246 still not including the image acquisition computers. The total run time of this system not including the image acquisition could be defined as

$$T = \frac{I}{30} + T_{tcwm} + T_{tlwm}$$

$$T = \frac{I}{30} + 0.2678*C + 0.0570 + (7.089*10^{-3}*C + 1.993)*I + 5.735*10^{-3}*C - 1.832$$

$$T = (7.089 * 10^{-3} * C + 2.026) * I + 0.2735 * C - 1.775 \tag{49}$$

If the system were to have the 30 cameras, 90 images taken, and an estimated 3 seconds to drive over the platform, then the total runtime of the system would be just under 208 seconds.

Discussion

As can be seen from the discussion section, the system as currently designed is completely infeasible. Therefore, the results section will largely be a discussion of what changes can be made in the future to make this a feasible system.

Transfer rates

Current transfer rates are not high enough to make the system work. A small number of image-grabbing machines cannot take the images and transfer them to a line of crosswise mosaicing machines because no method exists for successfully transferring sufficient data. This can been seen as the major bottleneck in Figure 51. Without a sufficiently high transfer rate, the system will never be implementable. A close study of the data that must be transfered shows that the system pictured in Figure 51 would not be feasible with a gigabit network even without the unexplained overheads discussed in this paper. The volume of data to be transfered is more than a gigabit network can handle. The proposed system will have a minimum of 30 cameras running at 30Hz, which means that the network would have to transfer 900 images per second. If each image is 307KB, then the network would have to be transferring 269.8MB or 2.11Gb of data per second, which is more than a gigabit network can sustain.

There are three possible solutions to this problem. The first is to redesign the setup between the image capture computers and crosswise mosaicing computers as shown in Figure 60. This method will allow a gigabit network to be used but will require the crosswise mosaicing code to be rewritten to handle the new two-step mosaicing process. In addition it will increase the number of computers needed for the crosswise phase because images will have to be in essence mosaiced twice in the crosswise direction.

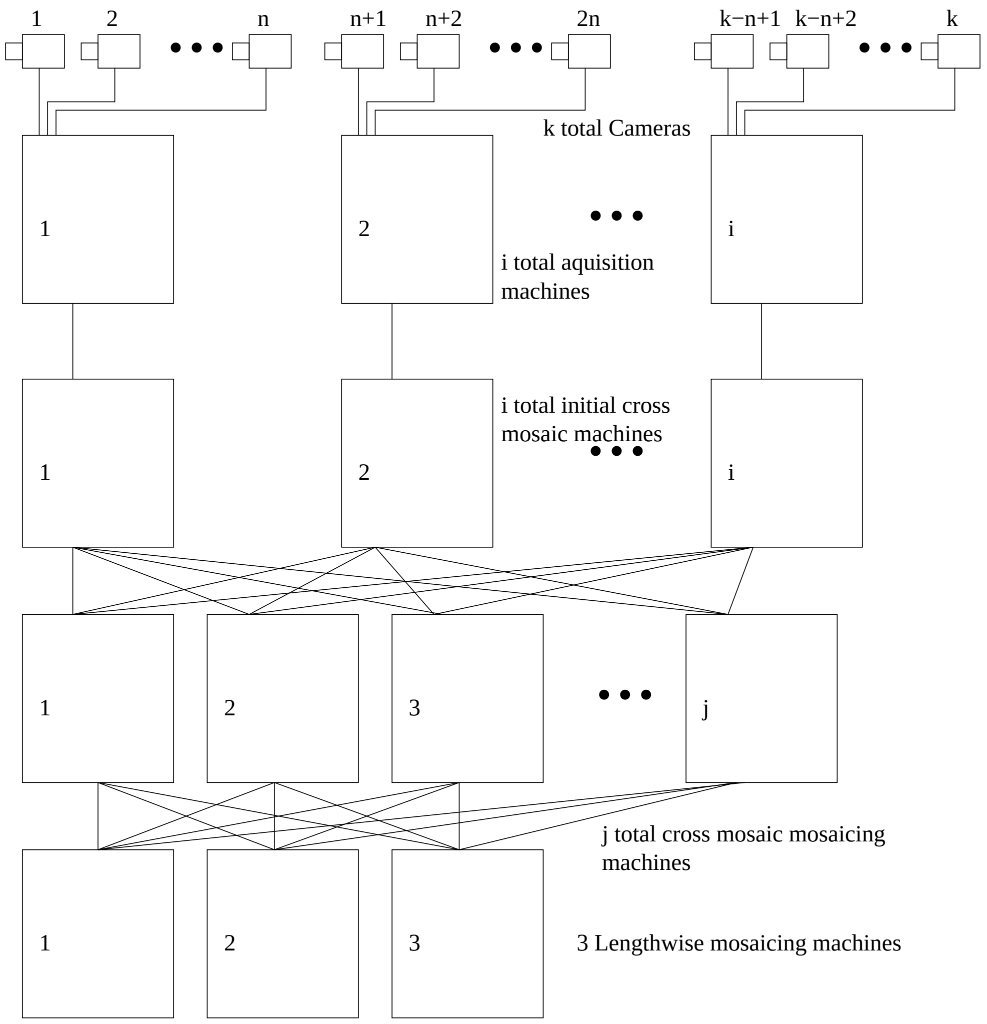

1
2
n
n+1
n+2
2n
k−n+1
k−n+2
k
k total Cameras
1
2
i
i total aquisition machines
1
2
i
i total initial cross mosaic machines
1
2
3
j
j total cross mosaic mosaicing machines
1
2
3
3 Lengthwise mosaicing machines

Figure 60: Second proposed image acquisition to crosswise mosaic network

The second solution proposed is to change from a gigabit ethernet network to a firewire optical network. The optical network will allow for greater transfer rates than from the ethernet network and will not require the algorithms to be rewritten. This solution has the disadvantage that optical networks are still expensive and so may increase the cost of the system more than additional computers would.

The third solution is to use images that are a quarter of the size of the images currently used. This reduction is reasonable because it is half the width and half the height of the original images. The data transfer rate would decrease to 540Mb per second, which is within the scope of a gigabit ethernet network. This solution will not require changes to the algorithms and will speed up the entire mosaicing process, thus decreasing the number of computers needed. The disadvantage is that this will require either the addition of a reduction stage or the use of different cameras. This solution makes the most sense because if it ever goes into production, an arrangement will surely be able to be reached with the company that builds the cameras to use a lower resolution in the CCD of the camera. The only problem with this is that the lower resolution will hurt an operator's ability to focus on an object under the vehicle. Currently more detail than is necessary exists within the mosaics, so this should not be a problem.

Crosswise mosaicing

The number of processor currently required to perform the crosswise mosaicing step is too large and will remain so, as shown in equation 48. To reduce the number of processors needed, we will have to reduce the components of P_{cross}. Most of the processor requirement of P_{cross} comes from the rectification portion T_r of T_{cwm}, which makes up T_{tcwm}. As seen in equation 6, the time required to rectify an image is a large bottleneck. The speed of the system cannot be improved without improving the speed of the rectification: it currently takes over one-fifth of a second to rectify a single image. The solution requires a hardware change. Hardware rectification devices exist that can pipeline the rectification process to the point where an image comes out of the pipeline every clock cycle. Although this will increase the cost of the system, it will increase the speed and reduce the number of processors necessary for crosswise mosaicing enough to make up for it. With this hardware added to the acquisition machines, the step could be skipped in the crosswise mosaicing and the total crosswise mosaicing time, T_{cwm}; total crosswise mosaicing machine time, T_{tcwm}; and number of processors necessary for crosswise mosaicing, P_{cross}, could be reduced from equations 37, 39, and 48 to equations 50, 51, and 52, respectively.

$$T_{cwm} = 0.0506 * C - 0.0215 \tag{50}$$

$$T_{tcwm} = 0.0578 * C + 0.05695 \tag{51}$$

$$P_{cross} = \lceil 1.735 * C + 1.709 \rceil \tag{52}$$

This alteration would reduce the number of computers necessary for the crosswise mosaicing step to 54 for 30 cameras. This number of computers is still too large to be a feasible solution. As can be seen from equation 18, the best place to achieve a speed increase is to reduce the squared term W_s, the search window. This reduction can be achieved either by reducing the size of the search window, which would make the match algorithm less robust with regard to varying depth within a given undercarriage, or by changing the shape of the window. Because the model uses epipolar geometry, there should be very little shift in the vertical direction between images. This means that the search window need go only 1 pixel above or below the assumed match location, which would change the search window size from $(2 * W_s + 1)^2$ to a far smaller $(2*W_s+1)*3$. For the W_s of 7 that is currently used, this would reduce the points searched from 225 to 45. This reduction in points searched in conjunction with the reduced images discussed for image acquisition allows us to rewrite question 35 as

$$T_{tc} = 0.009826 * C - 0.008571 \tag{53}$$

Without factoring the speedup into the mosaicing step but with factoring in transfer speedup, we can rewrite equations 50, 21, 51, and 52 as

$$T_{cwm} = 0.01402 * C + 0.01502 \tag{54}$$

$$T_{cmt} = 5.446 * 10^{-4} * C + 6.454 * 10^{-3} \tag{55}$$

$$T_{tcwm} = 0.01565 * C + 0.03438 \tag{56}$$

$$P_{cross} = \lceil 0.4696 * C + 1.031 \rceil \tag{57}$$

This proposed system would only require 16 computers for the crosswise mosaicing step and would finish the crosswise mosaicing within 1 second of the vehicle traversing the platform.

Lengthwise mosaicing

The time taken by the lengthwise mosaicing step does not need to be reduced because of the number of computers required (it runs on only three machines) but because it is excessive. The first step is to write the time in terms of the reduced image size:

$$T_{tl} = (1.666 * 10^{-3} * C + 1.370) * I - 1.697 * 10^{-3} * C - 1.363 \tag{58}$$

$$T_{lwm} = (0.003366 * C + 1.389) * I - 5.97 * 10^{-4} * C - 1.338$$

$$T_{lmt} = (7.488 * 10^{-5} * C + 8.875 * 10^{-4}) * I + 2.723 * 10^{-4} * C + 3.227 * 10^{-3}$$

$$T_{tlwm} = (3.590 * 10^{-4} * C + 1.390) * I + 6.674 * 10^{-3} * C - 1.322$$

Using Tas defined in equation 49, we can rewrite it as

$$T = (3.590 * 10^{-4} * C + 1.423) * I + 0.02232 * C - 1.288 \tag{59}$$

Using equation 59, we can see that the total run time for a 30-camera system with 90 images taken will be just under 129 seconds, which is far too long a run time for this system. Looking back at equation 58, we can see that it is the lengthwise matching that takes the long run time because the crosswise match and mosaic now take less than 1 second. By looking at equations42, 40, 41, and 28 together, we can see that the time-consuming portion of the lengthwise match is the search associated with P_i. A good method for reducing the search space could easily cut the run time of the system in half. At this time no such method has been determined. It is possible to simply cut down on the number of points for which a search and exact match are found, but this would undermine the algorithms robustness and not drastically reduce time.

Another method would be to use previous found matches to direct the first phase search in the length match. Because it is impossible for a vehicle to drastically change its speed within $\frac{1}{30}$th of a second, the match from one instant might be used to find the match of the next instant. This would drastically reduce the size of the first search window W_s. This would in turn allow for a more confined search to get an exact match, allowing W_{s2} to be reduced and drastically lessening the run time, because the second search window is $(2 * W_{s2} + 1)^2$.

Final thought

The system described in this paper shows the possibility of eventual implementation in locations where under-vehicle inspections are required. Before that can be done, though, more work is required to get the system to function in a sufficiently short time so that major delays are not created by the inspection.

Acknowledgments

This work has been partially supported by Atlantic Coast Technologies, Inc. (ACTI), by the National Science Foundation under Grant No. 0105272, and the Army Research Office under grants DAAD19-99-1-0016 and DAAD19-02-1-0133.

References

[1] "Security mirrors \ observation systems", available at http://www.insight-security.com/pf16.htm#Vehicle%20inspection%20mirrors, last checked 6/26/02.

[2] "Inspection Mirrors", available at http://www.acrilconvex.com/inspection/default.htm, last checked 6/26/02.

[3] "Search Systems Incorporated: Technical Search & Surveillance Equipment", available at http://www.searchsystems.com/contraband.html, last checked 6/26/02.

[4] "UVI - Under Vehicle Inspection System", available at http://www.uvisystems.com/ last checked 6/26/02.

[5] Z. Zhu, A. R. Hanson, H. Schultz, F. Stolle, E. M. Riseman, "Stereo Mosaics from a Moving Video Camera for Environmental Monitoring", First International Workshop on Digital and Computational Video, December 10, 1999, Tampa, Florida, USA, pp. 45-54

[6] Z. Zhu, E. M. Riseman, A. R. Hanson, "Parallel-Perspective Stereo Mosaics", The Eighth IEEE International Conference on Computer Vision, Vancouver, Canada, July 4001, vol I, 345-354.

[7] H.-C. Huang, Y.-P. Hung, "Panoramic stereo imaging system with automatic disparity warping and seaming", Graphical Models and Image Process., 60(3): 196-408, 1998.

[8] S. Peleg, M. Ben-Ezra, "Stereo panorama with a single camera", CVPR'99: 395-401

[9] H. -Y. Shum, R. Szeliski, "Stereo reconstruction from multiperspective panoramas", ICCV99, 14-41, 1999.

[10] S. Peleg, J. Herman, "Panoramic mosaics by manifold projection", CVPR'97: 338-343.

[11] P. Dickson, J. Li, Z. Zhu,A. R. Hanson, E. M. Riseman, H. Sabrin, H. Schultz, G. Whitten, "Mosaic Generation for Under Vehicle Inspection", ACV'02:251-256.

[12] H. -Y. Shum, R. Szeliski, "Construction and refinement of panoramic mosaics with global and local alignment", ICCV'98: 953-958.

[13] R. Kumar, H. Sawhney, J. Asmuth, J. Pope, S. Hsu, "Registration of video to geo-referenced imagery", ICPR98,vol.4: 1393-1400.

[14] "Vision Lab at UMass - UVIS", available at http://vis-www.cs.umass.edu/projects/uvis/index.html, last checked 6/26/02.

[15] Sergei's masters thesis

Appendix D: Timing model accuracy

This appendix covers the details of the model determination of UVIS. Each section is a mirror to a section of the paper and includes details of how the model for that section was determined.

Acquisition

The number of cameras that can be connected to the image acquisition computers is determined as follows. Two bottlenecks occur when images are taken; the first occurs when the images are acquired. Through empirical examination on a Pentium 4 1.6 GHz machine, the maximum number rate for images to be stored from the cameras into RAM was found to be 22 cameras running at 30 images per second. Because the images must also be sent out at the same rate as acquired and the RAM cannot read data out any faster then it writes data in, this sets our maximum number of 11 cameras per computer.

The maximum rate that data from cameras can be taken without hitting a bottleneck when saving them into RAM was determined as follows. A program was written that creates image files using the same protocols as those used by the camera drivers. This program times how long it takes to write a given number of images from the camera being simulated. Because it was empirically discovered that the memory could handle about 22 cameras at 30 images per second, a test run was set up to simulate 1 million images being written, thus creating a test run that would take 1461 seconds to run, which would eliminate any timing errors caused by the limitations of the C programing languages time library (which has a resolution of 1 second). These tests were performed with the images being written to a RAM-disk partitioned into memory for this test.

The second bottleneck appears in transferring the images to the next set of processors, which will handle the image rectification and the crosswise matching and mosaicing (see Figure 51). The transfer rate for the raw images over a 1 gigabit network is 38 images per second, which gives us the maximum transfer rate of one camera per input computer for the acquisition phase. The image transfer is accomplished using MPICH, the free distribution of MPI, and shows obvious overhead in the file transfer. A transfer of 38 images that are each approximately 300KB would mean a sustained transfer rate of just over 11MB per second, which would be about 90Mb per second, which is under one-tenth the bandwidth of a gigabit network. This level of overhead means that a problem is occurring with the MPICH program that is transferring the files. If this system is to be developed, this problem with have to be found and fixed.

Rectification

The rectification model was determined by timing the rectification process for simulated data of 2-16 cameras and 1-19 images. For all images rectified, the test showed a steady state average time of 0.2029 seconds per image with a standard deviation of 0.0061 seconds. Because the goal of this paper is to design a system that can run within a certain amount of time, using the upper bound of 0.2100 images per second for the rectification in equation 6 seemed advisable.

Crosswise matching

The timing model to determine the run time of the crosswise match was determined by first analyzing the structure of the algorithm and then empirically testing to determine the coefficients within each equation of the system model.

The number of points checked is determined as shown by equations 12, 13, and 14. The assumed offsets in 12 and 13 cannot be known though, so no exact model of P_c can ever be determined. The model presented proved to be accurate within 14% of the actual P_c values tested with a 11% standard deviation. These numbers were determined by timing the matching algorithm with 32000 different combinations of the input variables to the function.

The equation for T_c was known to be polynomial because the number of locations that had to be checked to determine the variance grows as a function of window diameter W_c squared. A test run to time the determination of the variance of a given point for a minimum of $5 * 10^8$ times took a minimum of 113 seconds. Because the accuracy of the time library in programing language C is only accurate to within 1 second, a maximum error in time of under 1% was considered sufficiently rigorous. The error rate dropped even lower for the larger W_c values and was under 0.25% for tests of W_c equal to 7 or greater, which is the value for W_c when UVIS has typically been run. A least-squares fit was run on the data to determine the coefficients of T_c in 15. The average time per point checked is plotted with T_c in Figure 61.

The method of determining T_n was the similar to that of T_c, which is also known to take polynomial time in terms of the size of the correlation window W_c because of its structure. Again, NCCM was run a minimum of $5 * 10^8$ times and it took a minimum of 216 seconds, which leaves a possible timing error of under 0.5%, which is a timing error of 1 second as described above. The error rate was under 0.1% in the area of W_c equal to 7, where the system is usually run. Again, a least-squares fit was run on the data to determine the coefficient of 16. T_n is plotted below with the empirical data from which its coefficients were generated 62.

The baseline time for the program T_k was determined by running the algorithm repeatedly, with the algorithm never running either the variance test or NCCM. The algorithm was run repeatedly for each number of cameras until the total run time was greater then 1000 seconds to make sure the timing error was under 0.1%. The run time was assumed to be linear in terms of camera size because all portions of the algorithm were run a set number of times except for the variance test and NCCM, which were not included. The results were fitted with a line using the least-square method, and the coefficients of the equation of T_k were determined to be those seen in equation 17. The equation and the actual $T_k$data appear in Figure 17. The algorithm was only run with a simulation of up to 16 cameras, which is the maximum number of cameras that could be simulated within the lab environment.

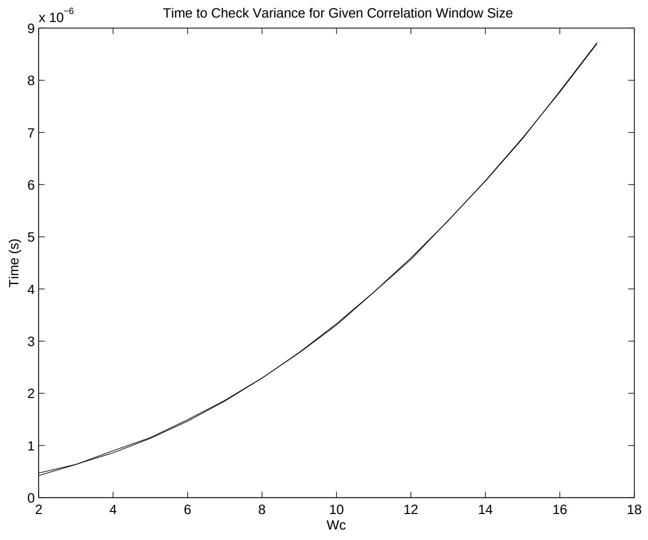


Figure 61: Time for testing the variance of a correlation window empirical data and timing model

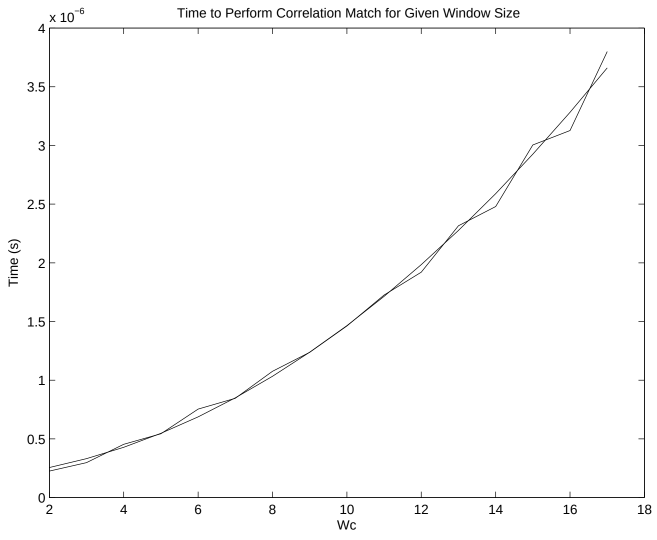


Figure 62: Time for testing NCCM of a pair of correlation windows empirical data and timing model

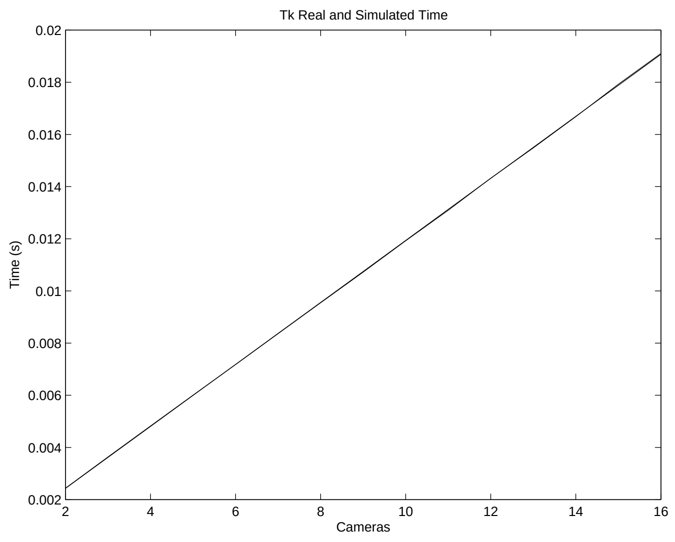


Figure 63: Time to run the baseline match algorithm and the equation of said baseline time plotted vs. the number of cameras

The overall model of run time for the crosswise match T_{tc} can be found in 18. 18 comes directly from analysis of the algorithm, with each portion of the match relating to the number of pairs of images on which it is run, where the number of pairs is the number of cameras minus 1. The $(2 * W_s + 1)^2$term comes from the fact that NCCM is run on a search window with radius W_s around the assumed match.

When the total time model is compared to actual run times of the algorithm the results are found to be within 3% of each other when averaging the run time difference of 32000 separate tests. These 32000 tests are runs of the crosswise matching algorithm with differing inputs and each specific test took more then 1 second to lower the error from the Time function under Linux that has precision of .01 seconds. The 3% difference in total run time was found with a standard deviation of 4%.

Crosswise mosaic

The time to run the crosswise mosaicing T_{cm} was determined by timing the mosaicing process for each given number of cameras 2-16 for synchronized image sets 1-19. The average time to mosaic each simulated number of cameras from 2-16 was taken, and a linear function was fit to those data using the least-square method. This determined the coefficients used in 19. Figure shows T_{cm} plotted with the empirical data it was created from.

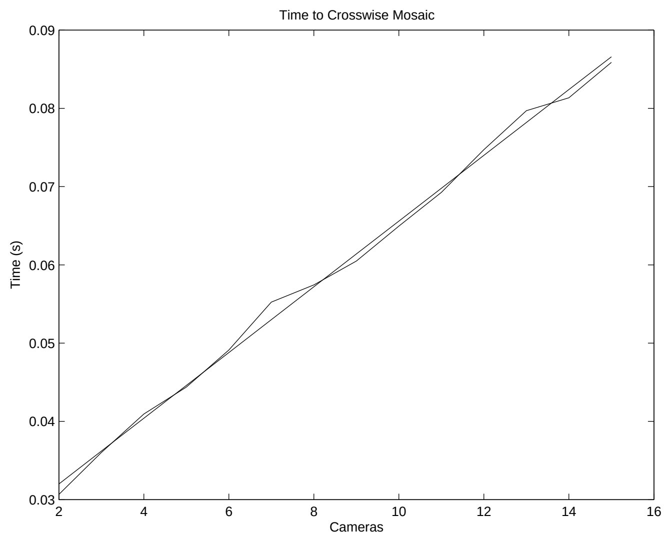


Figure 64: Time to perform the crosswise match empirical data and model

Crosswise mosaic transfer

The times to transfer the crosswise mosaics will also include whatever overhead is causing the slowdown of the single image transfer. The amount of data to be transfered per image is the number of pixels within the mosaic to be transferred (the width times height) multiplied by the size of the data type of each pixel, which is 4 for the black and white pgm images being used. The transfer rate was found to be $2.101 * 10$ seconds per bit of data to be transfered. This rate is still only approximately 45 bits per second, which is far lower then the gigabit ethernet network should be able to handle. The times used to determine the transfer rate were determined by repeatedly sending mosaic files of various sizes (corresponding to crosswise mosaics generated from 2 to 40 cameras) and timing the process. Each image was sent 8200 times to get minimum total transfer times of 235 seconds, thus eliminating error caused by the C timing library.

Lengthwise matching

The number of locations that must have their variance tested in the first pass, P_{c1}, is determined by the evaluation of the algorithm.

T_{cl} and $T_{nl}$are the same as $T_c$and T_n, respectively, and have the same equations, 24 and 26 and 15 and 16, respectively. The variables are named differently to avoid confusion when discussing the crosswise matching vs. the lengthwise matching.

P_{c2} is like P_{c1} and is determined through analysis of the algorithm.

The final piece of the lengthwise matching puzzle is the baseline match algorithm time, T_{kl}. The algorithm was analyzed and the equation for T_{kl} was

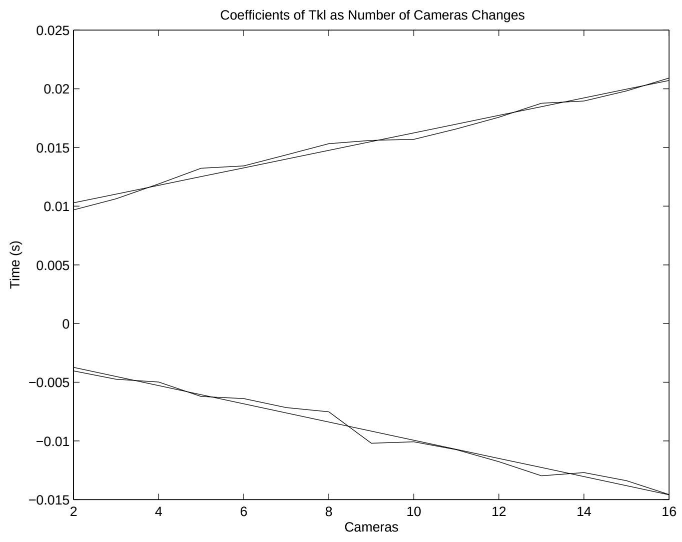


Figure 65: Baseline time equation coefficients and the lines fit to them

determined to be linear in terms of the number of images I. Because the linearity was determined by the size of the crosswise mosaics matched and the crosswise mosaics size was determined linearly by the number of cameras, T_{kl} is written in terms of both I and cameras C. Equation 28 was determined by first fitting equations to the baseline times to match for crosswise mosaics of 2-16 cameras. Equations were then fit to the coefficients of these equations to get 28. The equations to get 28 are plotted in Figure 65 with the baseline times from which they are derived.

The total time for the lengthwise matching, T_{tl}, is found from analysis of the algorithm to be 29.

Lengthwise mosaicing

The lengthwise mosaicing step is linear in terms of the number of images because it merely stitches parts of different images together. The size of these images and hence the time to mosaic depends on the size of the crosswise mosaics, so the time to mosaic must again be written in terms of both the number of images and the number of cameras. The coefficients for the lengthwise match time are determined for each set number of cameras, 2-16, and then the coefficients for the lines that fit the previous coefficients are determined, again using a least-squares fit. These coefficients are used to write 30. The coefficients of 30 and the data from which they are determined are plotted in Figure 66.

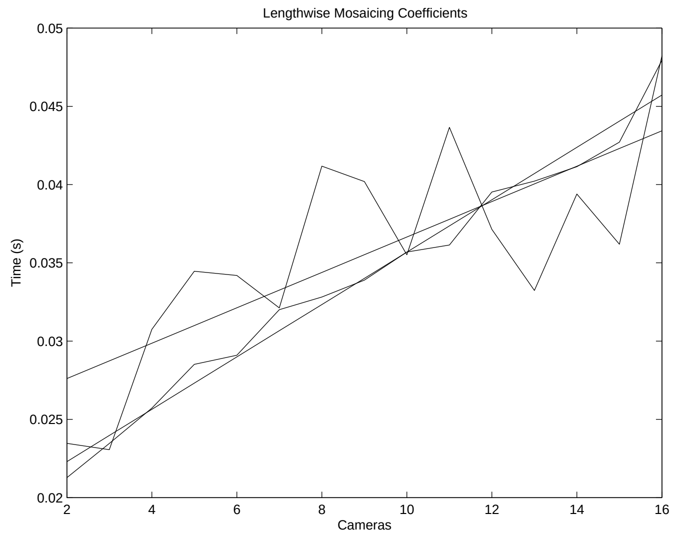


Figure 66: Coefficients of lengthwise mosaicing and empirical data

Lengthwise mosaic transfer

The lengthwise mosaics transfer was postulated from the crosswise mosaic transfer data.

Appendix E

Experimental setup

Because it is not feasible to build a scale working prototype of the system; one has had to be simulated in the lab environment. To this end a scale model of a vehicle undercarriage was created as shown in Figure 67. The model is a combination of Styrofoam that has been carved and painted and actual car parts.

The cameras are mounted to a frame around the undercarriage as shown in Figure 68. The idea behind the model is that moving the cameras above a stationary model is equivalent to having a moving undercarriage over stationary cameras. The camera-to-undercarriage movement will be the same either way. As can be seen from Figure 68, the cameras are attached to a metal frame above the model. This frame allows the cameras to be set at any location above the simulated vehicle undercarriage.

Figure 69 shows the actual cameras as they are attached to the frame pictured in Figure 68. The cameras are attached to circuit board material with wires. This setup allows for easy camera movement to test different separations of the cameras. It does not lend itself to a stable platform or careful camera

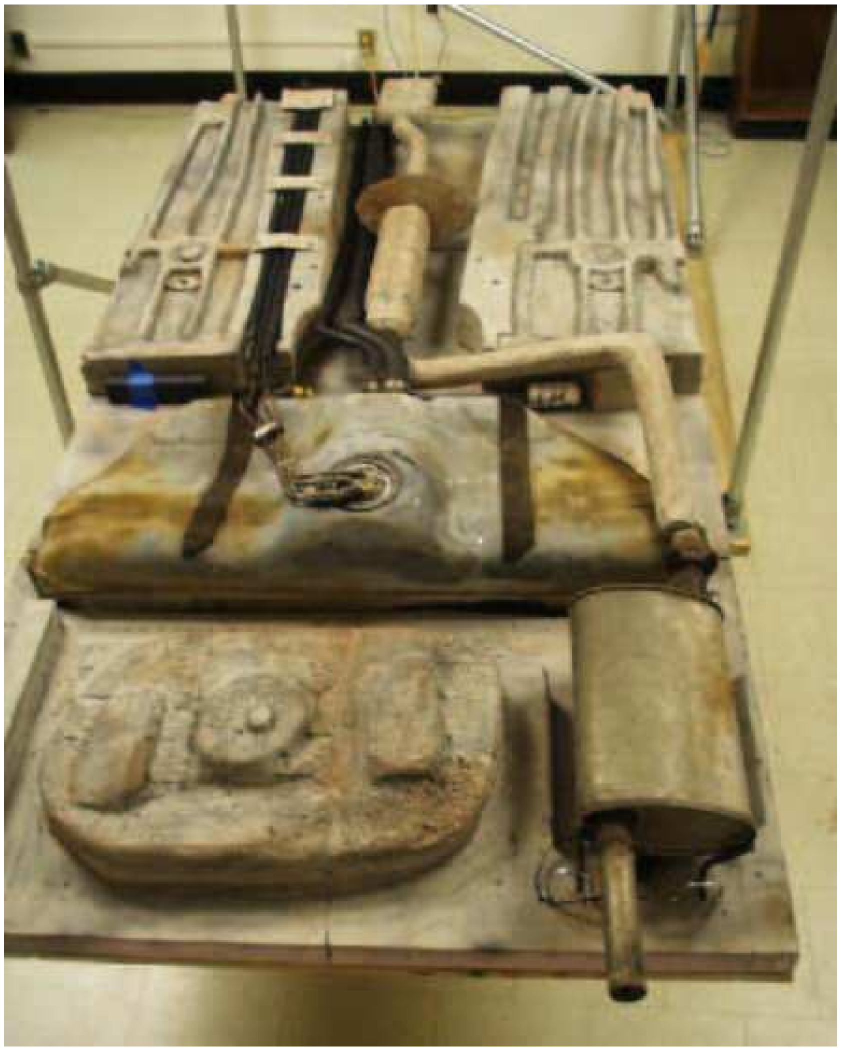


Figure 67: Image of scale vehicle undercarriage model

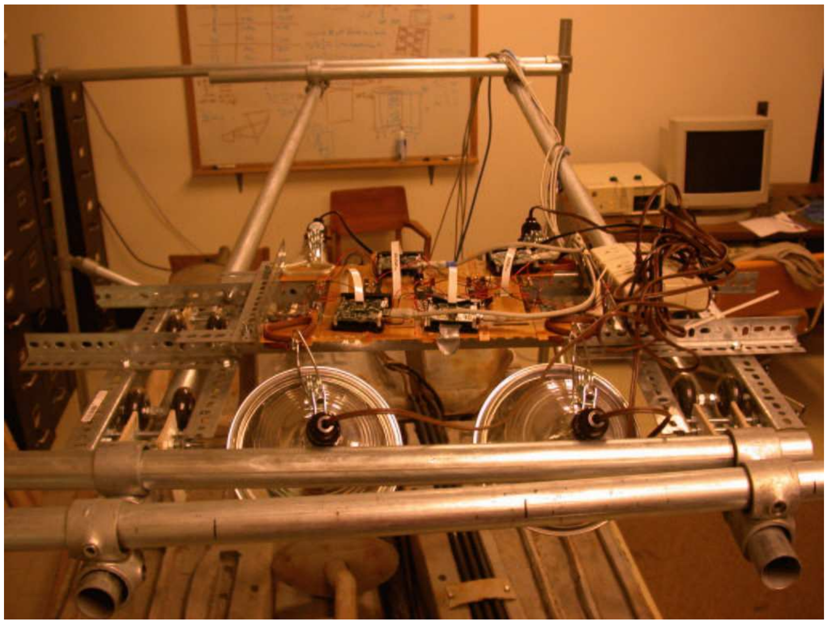


Figure 68: The frame on which the cameras are mounted above the undercarriage model

alignment. In the eventual system the cameras will have to be rigidly mounted in place and their fields of view controlled for a consistent overlap. In the model setup the camera mounting does not allow for this careful placement and the fields of view do not have a consistent overlap. Because this would affect the assumed ability that is to be mathematically derived, the assumed matches were instead created by hand matching points between the four camera images.

The cameras themselves sit on a sled (Figure 68) that rides down the frame on rail runners designed for this project (Figure 70). These runners allow the camera sled to be pulled down the frame at varying speeds to simulate the actions of a vehicle traversing the platform with an uneven speed.

Camera setup

The Point Grey DragonFly cameras used were calibrated and rectified using techniques and software provided by Kovalenko [15]. Figures 71 and 72 show, respectively, a raw input image and a rectified image. The radial distortion shown in Figure 71 demonstrates the need for image rectification. Without removal of the distortion, no possibility exists for matching the images.

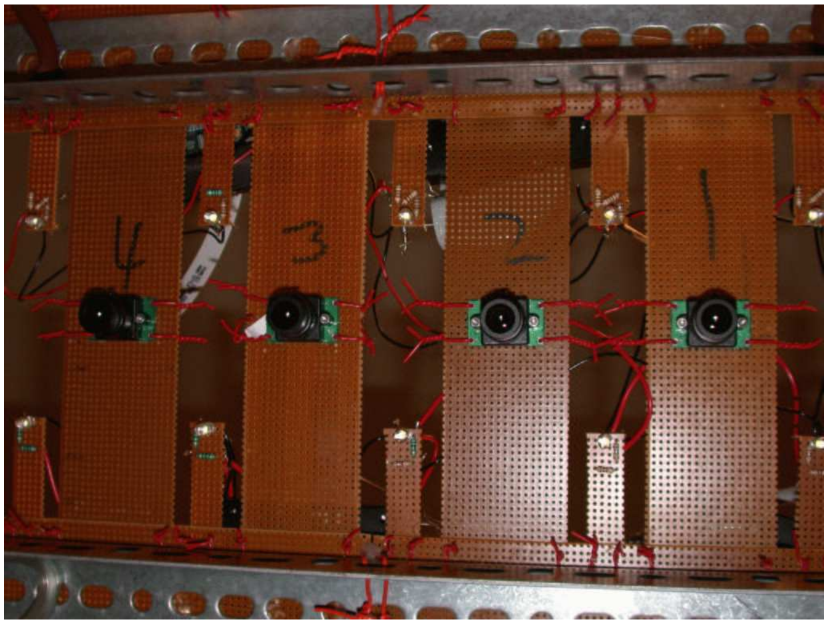


Figure 69: View from under cameras

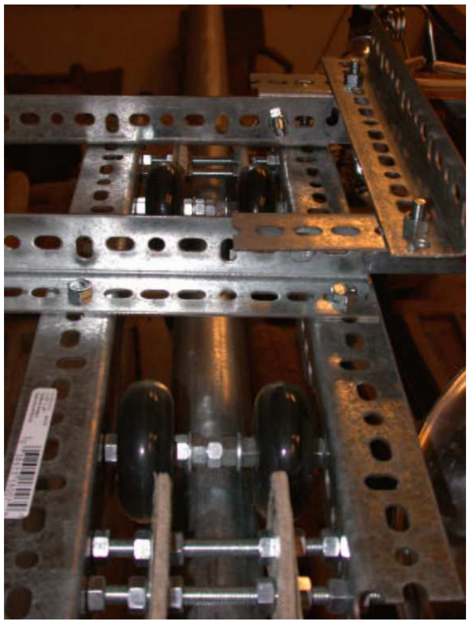


Figure 70: Camera platform runners

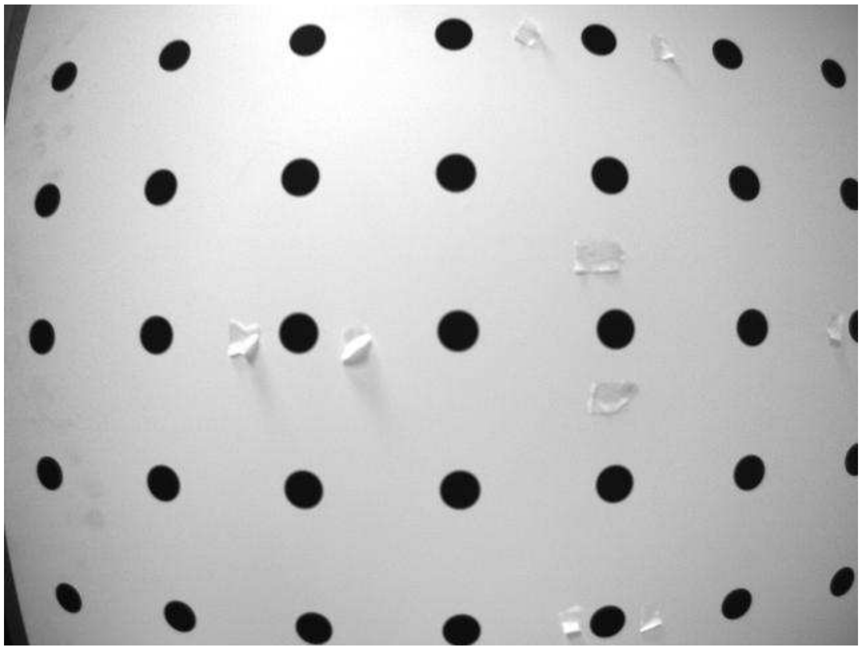


Figure 71: A raw input image from one of the cameras

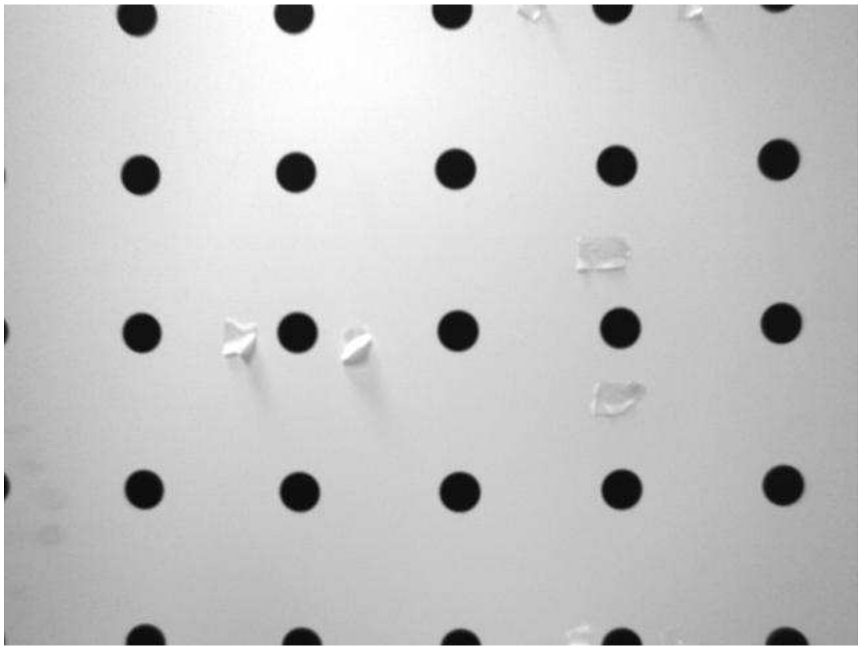


Figure 72: The same image as in Figure 71 rectified

Experimental model

Only 4 cameras are available, so it is not possible to test the system with the 30+ cameras the final system will have. Because four points of data are not enough to reasonably extrapolate to a 30+ camera, systems behavior from more cameras had to be simulated. A 16-camera system was simulated by taking the 4 cameras and taking images from them at four different locations across the vehicle undercarriage. This was done by shifting the longitudinal pipes to which the cameras are attached (Figure 68). These 16 cameras had to have the same simulated movement in the direction of simulated car movement; therefore, the cameras were drawn down the length of the model, taking images every 3 inches for each horizontal location of the cameras.

Because the 4 cameras needed to simulate the 16 cameras with known assumed offsets, the 16 simulated cameras were hand matched for each of the 19 locations down the model length. This 16-by-19 grid of images is the set of images from which all timing model information is determined. The images appear in Figure 73.

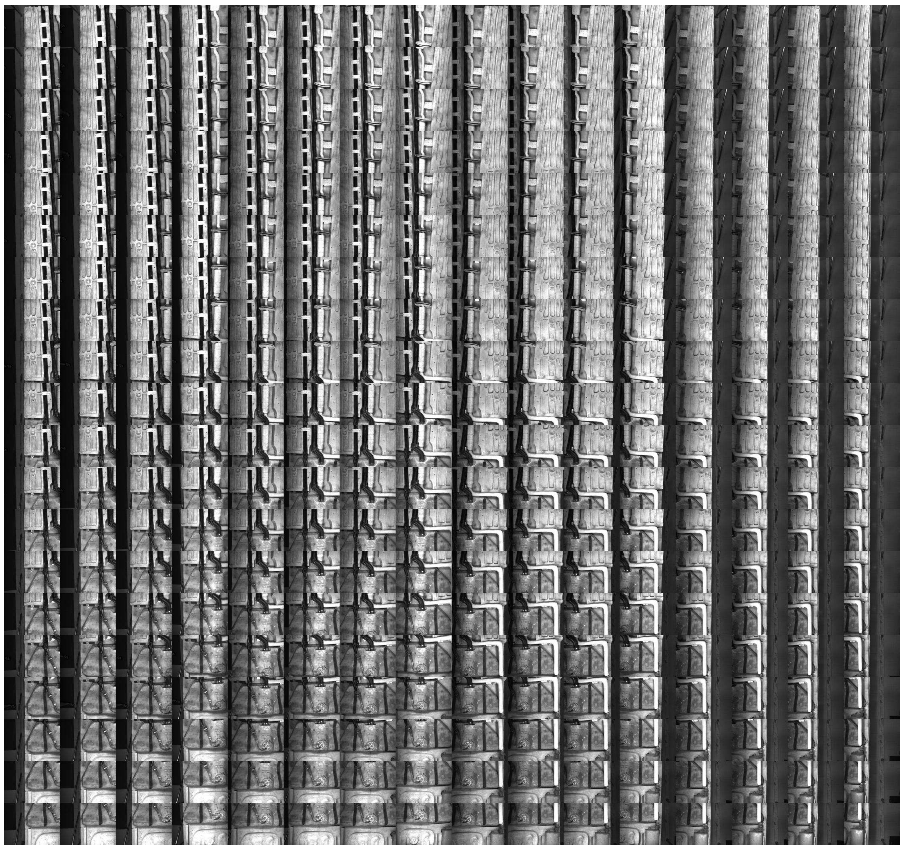


Figure 73: Complete set of images used for timing model