IMAGE STITCHING WITH REAL-TIME PANORAMA CREATION FOR THE NIOSH MINE ROVER

Vaibhav Sanjay
University of Maryland - College Park, NASA Jet Propulsion Laboratory

Mentor: Erica Tevere, Co-mentor: Xavier Zapien
NASA Jet Propulsion Laboratory

ABSTRACT

The NIOSH Mine Rover will be equipped with several cameras in the front and rear of the robot, which operators will use to perceive the robot's location and react to a continuously changing environment. A video feed of the entire front and rear of the rover will assist immensely in its operation. Image stitching is the process of combining two or more images with overlapping fields of view into a singular image. This result can be achieved by detecting and matching key points between images and applying a transformation to match them. The image stitching pipeline includes several processes such as key point matching, homography estimation, warping, exposure compensation, and blending to create the final panorama. However, the process is time consuming and it is not feasible to do all these calculations for each pairing of images in a video feed in real time. We present a solution involving pre-calculation and GPU hardware acceleration to create a high quality video stitch with a consistent frame rate and latency.

***K*eywords** Computer Vision · Real-Time Image Stitching

1 Introduction

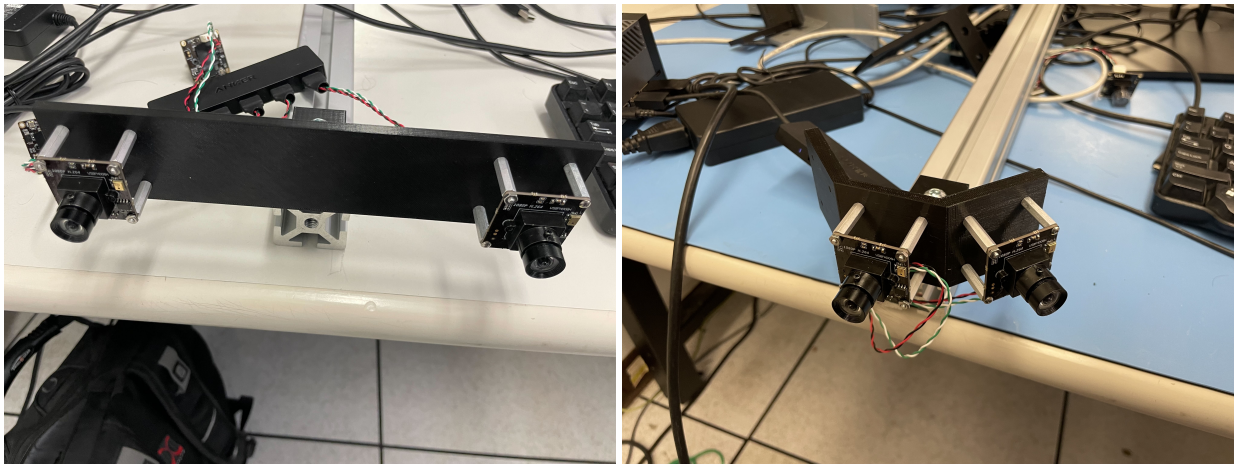
Image stitching is a well studied computer vision task in which several images taken from different fields of view are combined into one panorama with the purpose of constructing a single larger field of view. This task presents a variety of challenges including finding key points in an image, matching these points between two images, and calculating a transformation to place each image correctly on the canvas. Furthermore, seam finding and blending techniques[1] may be applied to smooth out the image and provide a coherent result. Image stitching has various applications in 360 degree panorama creation and robot planning.

1.1 Purpose

Awareness of the front and rear of the robot will be instrumental for orientation and navigation for the NIOSH mine rover, as the rover will be designed to traverse through the harsh and continuously evolving environment of a collapsing mine. To ensure that the robot operators have vision on all areas around the robot, the robot will be equipped with 8 monocular USB cameras. The cameras will process and compress images to be sent back to the operator base station. It is known from experience that operating the rover with a large number of camera views is not simple. Our purpose is to deliver the same information in a streamlined format so that the operator may control the rover with as little distraction as possible. The cameras will be paired up to form 4 pairs such that each pair has an overlapping view. From each pair, a panorama will be created, thus reducing the number of camera views required by 50%.

1.2 Overview

We seek to create a video stitching application that is suitable for navigation in this harsh terrain. The qualities of the stitch that we seek are



(a) Flat Configuration (b) Angled Configuration

Figure 1: Configured Positions for Camera Pairs

1. A high quality stitch that contains information from both video feeds, with little to no artifacts or seams.
2. A low latency. As each image pair comes in, the output image should be produced almost immediately. This is required so that operators can react quickly to a changing situation.
3. A high frame rate. In the best case, the frame rate should be around 15hz, so that individual frames are indistinguishable to the human eye.

While meeting individual requirements is rather simple, meeting all 3 is a challenge. To obtain a high quality stitch, one can use the full length image stitching process (see 2.2). However this process is time consuming, and will not result in a low latency and high frame rate video. On the other hand, if we sacrifice some of the steps in the pipeline to obtain a quick stitch, the resulting output is not the best quality. By applying the transformation and projecting the resulting images onto the plane, the resulting video has sufficient speed but suffers with obvious seams and distortions induced by the lenses. It seems that in either case, it is difficult to obtain all 3 points.
Our approach enables us to acquire the best of both worlds. In summary, our pipeline utilizes a precalculation scheme to obtain key points, matches, and transformations a single time. This maintains the integrity of the pipeline while significantly reducing the amount of calculations, and thus processing time, for each individual frame. Experimentation shows that this strategy indeed significantly lowers the latency and increases the frame rate for video stitching, especially with the use of the GPU.

1.3 Image Stitching vs Image Projection

While image stitching is one approach to this problem, another approach is image projection. Instead of matching key points to compute the transformation, one can use the camera calibrations and positions to construct a projection of the scene. When implemented for agreeable configurations, this system results in a quality panorama with little to no latency and frame rate drops.
We tested the camera configurations as described in 2.1 with the source code from Oehler[2]. While the resulting video has optimal frame rate and latency, the projection is likely to have clear visual seams. Observe that objects too close to the camera may be cut, and objects too far from the camera may be duplicated. The distance at which objects appear normal can be adjusted using the "focal length" parameter. However, we aim to provide a good stitch that works for all distances at least 0.3 meters away from both lenses.

2 Methods

2.1 Camera Configuration

The cameras used for experimentation are Monocular Low-Light HD USB Cameras from Blue Robotics (`https://bluerobotics.com/store/sensors-cameras/cameras/cam-usb-low-light-r1/`). The cameras have the ability to produce MJPEG, YUYV, and H264 images at rates from 15 to 60 frames per second. Each camera also

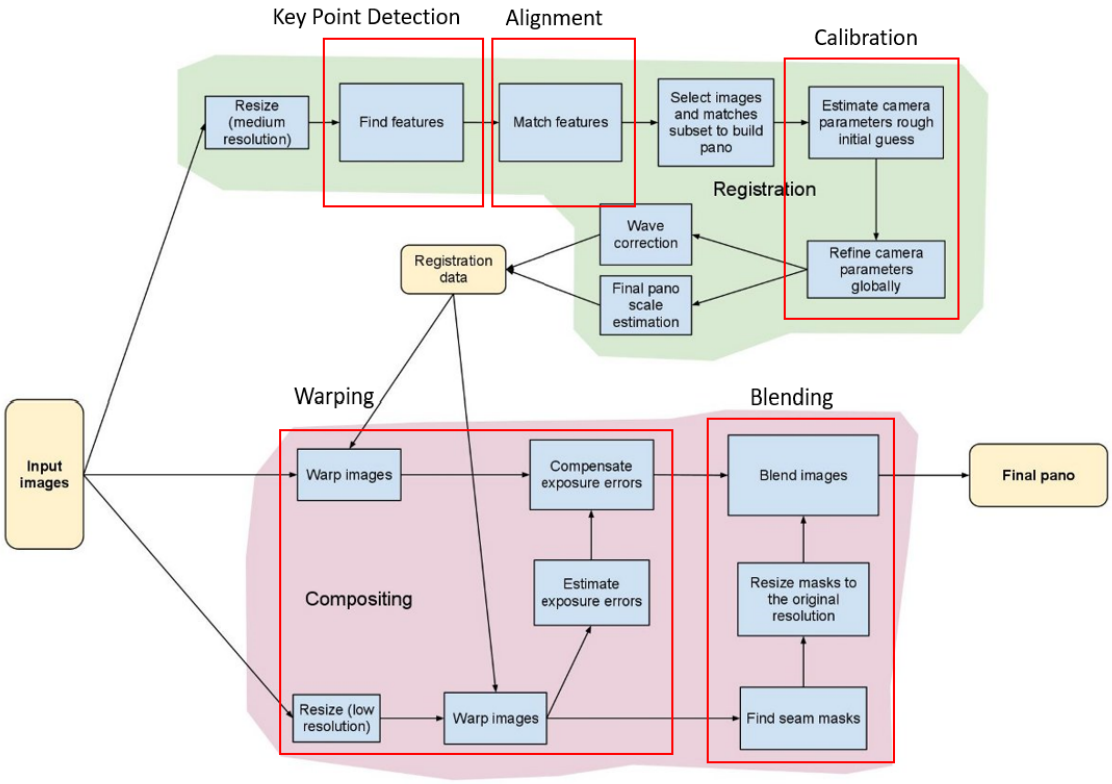


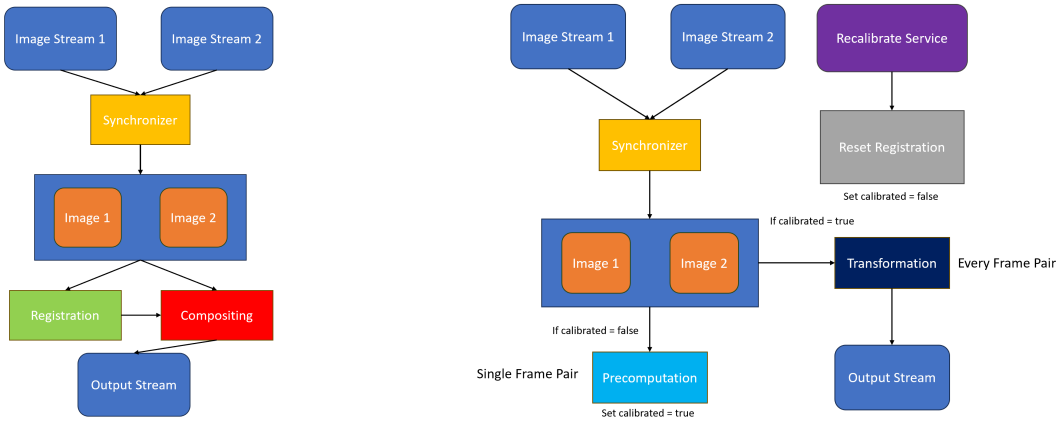
Figure 2: Illustration from OpenCV Stitcher Pipeline[3]

includes an H264 compression chip. For our purposes, the cameras were configured to produced H264 images at a rate of 60 frames per second at 640x480 resolution.
We have two cameras positioned for two different arrangements. Figure 1a shows the straight configuration. The cameras were mounted and positioned such that the optical centers were 180mm apart and parallel. Figure 1b shows the angled configuration. The cameras were mounted and positioned such that the optical centers were 70mm apart. The angle between their respective bases is 135 degrees. This configuration was created to provide a larger field of view for the output.

2.2 Original Stitching Pipeline

The original and new image stitching pipelines are influenced by the OpenCV stitching API. The various sections of the pipeline are highlighted in Figure 2. For a deeper explanation on each section see Brown[4]. Here we give a brief description of each boxed section as it relates to our source code.

1. **Keypoint Detection**: The identification of some feature points in each image. These keypoints can be in the form of corners, blobs, lines, or other features[5]. This task can be accomplished using a variety of feature detectors. For our application, the available detectors are SIFT[6], SURF[7], and ORB[8].
2. **Alignment**: Assuming the key points captured between each image are similar, a transformation can be determined if the features between the two images can be matched. The implementation for `BestOf2NearestMatcher` is based upon Lowe[6].
3. **Camera Calibration**: The camera intrinsics are estimated using the features and matches. In this step we also calculate the homography transformation, which describes the planar projection of the two images onto the canvas.
4. **Warping**: Once the homography has been calculated, we warp the input images so that they will be layered correctly on the canvas. Notably, this section is the most time consuming section in the newer pipeline.
5. **Blending**: This step may be omitted if the key points and matches have high confidence, however it is important in most cases. Before blending, we must estimate the location of the seam to smooth it out. This



(a) Original Pipeline (b) New Pipeline

Figure 3: Comparison of original and new pipelines

process can help alleviate ghosting effects or artifacts created by lens distortions, misalignment, and moving objects[9]. Using the seam mask and warping parameters, we can blend the image onto the canvas using feather blending or multi-band blending[10].

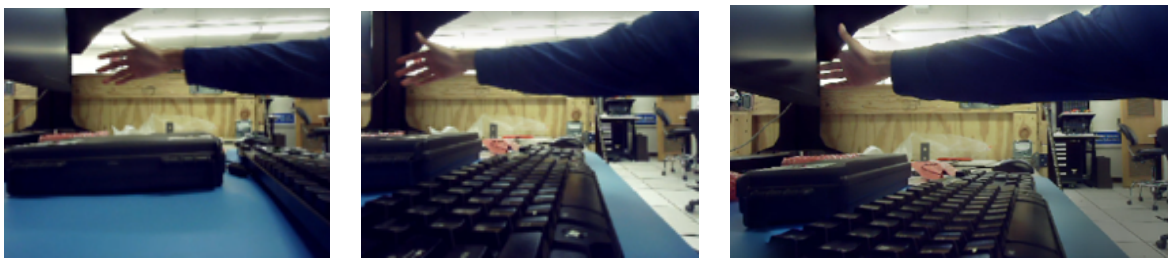
As illustrated by the Figure 2, the entire pipeline can be split into two sections: Registration and Compositing. In Figure 3a we describe the simple transformation of the OpenCV pipeline from accepting images and returning a single stitch to accepting an image stream and returning an output image stream. The only key difference is the synchronizer. As each camera is outputting its own individual stream, we obtain a pair of images by pairing up images that are outputted at similar times. The image stitching pipeline follows the same as before from this point forward. We obtain the resulting image and output in a continuous stream.
The main issue with this approach is the total time required. The largest time loss from this pipeline can be attributed to the discovery time and matching time for the key points. Since each new pair of images is different from the previous, the key points must be recalculated and matched. Even with a powerful processor this calculation cannot be completed at a reasonable rate (>15hz). Therefore, this pipeline is seemingly only suitable for image pairs or post-processing a video.

2.3 New Stitching Pipeline

The purpose of our pipeline is to have decreased latency and increased frame rate compared to the original pipeline. To obtain this efficiency we must first maintain that the camera positions remain static relative to each other. Our procedure involves separating parts of the pipeline into a precalculation section and a transformation section. When the stitching script starts up, it immediately uses the initial pair of camera images to detect the key points, matches, the homography transformation, and the stitching seam. The homography transformation is applied and the image is blended with reference to the seam for all subsequent image pairs.
Since the precalculation section is only required once, we no longer require finding key points for each image pair. Ideally, the precalculation will happen once at the start of execution and will not be repeated. This will be helpful in the case that the cameras move to featureless environments where key point matching can be difficult.
The new pipeline is illustrated in Figure 3b. The key difference is the precalculation and transformation sections. The calibration is used to inform the transformation in the following manner.

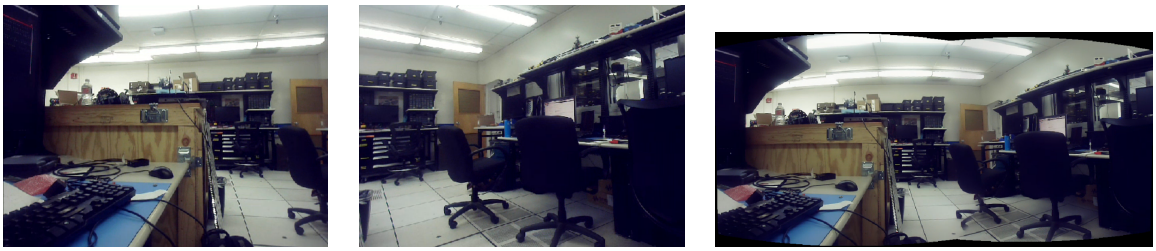
1. **Precalculation**: The precalculation is similar to registration but requires more depth. We calculate the homography as before, but continue on to calculate any parameters needed for the final warp. This includes the camera parameters, seam mask, and warping corners/sizes.
2. **Transformation**: The transformation step uses the saved parameters from the previous step to quickly obtain and transform the resulting stitch from the latest pair of images. The seam mask is used along with a blender to smooth out the final image, which is subsequently published to the output stream.

Upon completing the calibration, the parameters calculated will indefinitely be used in the transformation unless the recalibration service is called. When the recalibration service is called, the precalculation is completed again to recalculate the parameters. An example scenario for this use case is in the case that the cameras must be moved, or if



Left Image Right Image Stitched Result

Figure 4: Input pair and corresponding stitched result for flat configuration



Left Image Right Image Stitched Result

Figure 5: Input pair and corresponding stitched result for angled configuration

the original calibration was not ideal.
We can see a frame of the result for the flat configuration in Figure 4. Qualitatively the stitch represents what we desire because it includes both camera images in a seamless image. The quality of the image does not change as the cameras move around, since the transformation is kept constant throughout the run of the program. The angled configuration provides a much wide result as expected. In addition, the quality of the stitch does not waver very much when the seam is affected by close and far objects. This is due to the smaller distance between the focal centers of the cameras.

3 Results

3.1 Setup

To compare the pipelines we employ a simple experiment. The 3 pipelines we test are original pipeline, new pipeline (no GPU), and new pipeline (GPU). For each pipeline, we test the total timing in seconds of all calculations. We do not count the time for sending and receiving the image, only the time for all the calculation. We also test the frame rate in hertz that images are published for each pipeline. Each stitching pipeline was executed 3 times to obtain a reasonably accurate result. The numbers for latency and frame rate that are displayed are taken from the average of the 3 attempts.

3.2 Outcome

The results can be found in Figure 6. Between the old and new pipeline without GPU, observe that there is a significant drop in latency and a significant increase in frame rate. The latency fell from 0.23 to 0.08 (65% decrease) and the frame rate increased from 3.2 to 10.8 (237% increase). Visually, this makes a drastic improvement for the operator. The latency is now within the reaction time for the average human, which will give the operator enough information to react to any situation in time.
The impact of the GPU is considerable as well. Between no GPU and GPU, the latency fell from 0.08 to 0.06 (25% decrease) and the frame rate increased from 10.8 to 17.7 (64% increase). The frame rate increase puts it above the ideal value of 15 frames per second.
As for the stitch

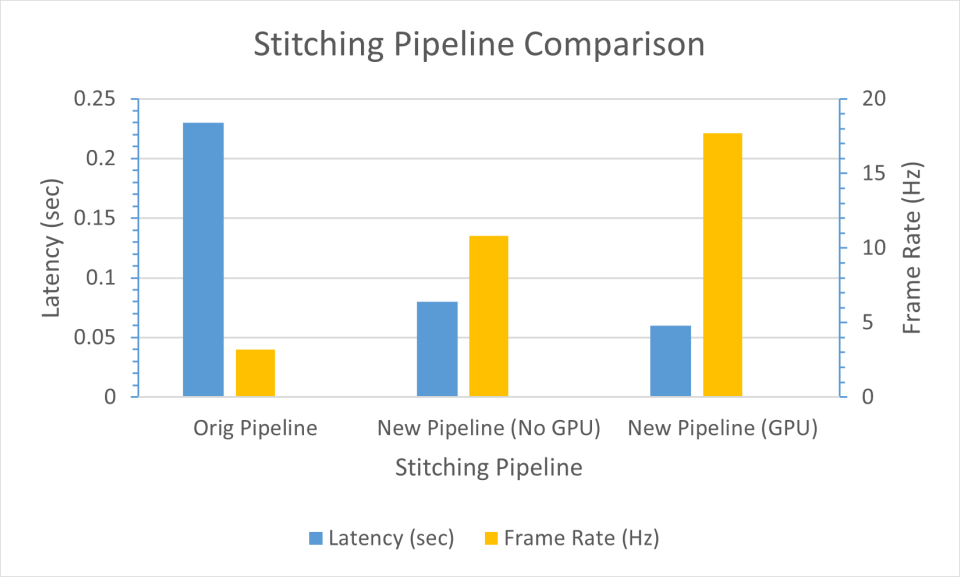


Figure 6: Comparison of original pipeline, new pipeline without GPU, and new pipeline with GPU

4 Conclusion

Video stitching is a variation of the well-studied problem of image stitching. The approach to real-time video stitching requires more nuance than image stitching in terms of facilitating an acceptable frame rate and latency for the output feed. In this paper, we discussed a new pipeline for video stitching based on that of Brown[4]. Our solution addresses the long time required for image stitching by precomputing the key features, matches, transformation, and seam for the outputted frame to produce the same quality of stitch with suitable frame rate and latency. We add support for GPU Hardware Acceleration to further increase the speed for all parts of the pipeline. Our new pipeline performs much faster than the original stitching pipeline for video footage while still maintaining the same image quality. There are some improvements that could possibly improve the stitch quality and speed for our application. Here we list a few possibilities.

Stitch Correction

Our solution relies on the fact that the cameras do not move in relation to each other. They may move, but the distance and angle between them must remain static. In the case that relative movement does occur, the stitch quality could be lowered. The work done by Yang[11] shows a possible solution for adaptive seam finding that could be implemented in our code.

Transformation Correction

Especially when the cameras are far apart, the transformation can appear fine at one distance, but look incorrect at another. Another possible improvement could be implementing a recalibration step when the camera is operating at a different range than before. This feature would require a system to understand when a stitch is off. A second program could be started to check the camera feed to make sure that the stitch visually seems fine, and a recalibration requested could be issued in the case that the range changes.

Darkness Testing

The stitching tests could be performed in darkness. Although ideally the best precomputation would occur in the light, it is possible that a recalibration would need to occur in darkness (with some LEDs). Testing the quality of the stitch after this change would be important to make sure that this function operates as expected.

5 Acknowledgements

Special thanks for my mentor Erica Tevere for providing support and guidance throughout the project. I would also like to thank the entire NIOSH team led by Dr. Rudranarayan Mukherjee with members Erica Tevere, Justin Koch, Xavier Zapien, Katherine Tighe, Eric Ambrose, David Zhu, and Bradley Kern.
This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Summer Internship Program and the National Aeronautics and Space Administration (80NM0018D0004).

References

[1] Ashley M. Eden, Matthew Uyttendaele, and Richard Szeliski. Seamless image stitching of scenes with large motions and exposure differences. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2:2498–2505, 2006.

[2] Martin Oehler and Oskar von Stryk. A flexible framework for virtual omnidirectional vision to improve operator situation awareness. In *2021 European Conference on Mobile Robots (ECMR)*, 2021.

[3] OpenCV. Images stitching. `https://docs.opencv.org/4.x/d1/d46/group__stitching.html`.

[4] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision*, 74(1):59–73, aug 2007.

[5] Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–10, 2018.

[6] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. Similarity Matching in Computer Vision and Multimedia.

[8] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[9] Hadi Hejazifar and Hassan Khotanlou. Fast and robust seam estimation to seamless image stitching. *Signal, Image and Video Processing*, 12, 07 2018.

[10] Peter J. Burt and Edward H. Adelson. A multiresolution spline with application to image mosaics. *ACM Trans. Graph.*, 2(4):217–236, oct 1983.

[11] Tao Yang, Fenlin Jin, and Jianxin Luo. A fast and robust real-time surveillance video stitching method. *Journal of Physics: Conference Series*, 1651:012170, 11 2020.