Hardware Accelerated Image Stitching Camera

ECE 445: Design Document

Cole Herrmann (colewh2)
Gautum Pakala (gpakala2)
Jake Xiong (yuangx2)

Fall 2022

1. Introduction

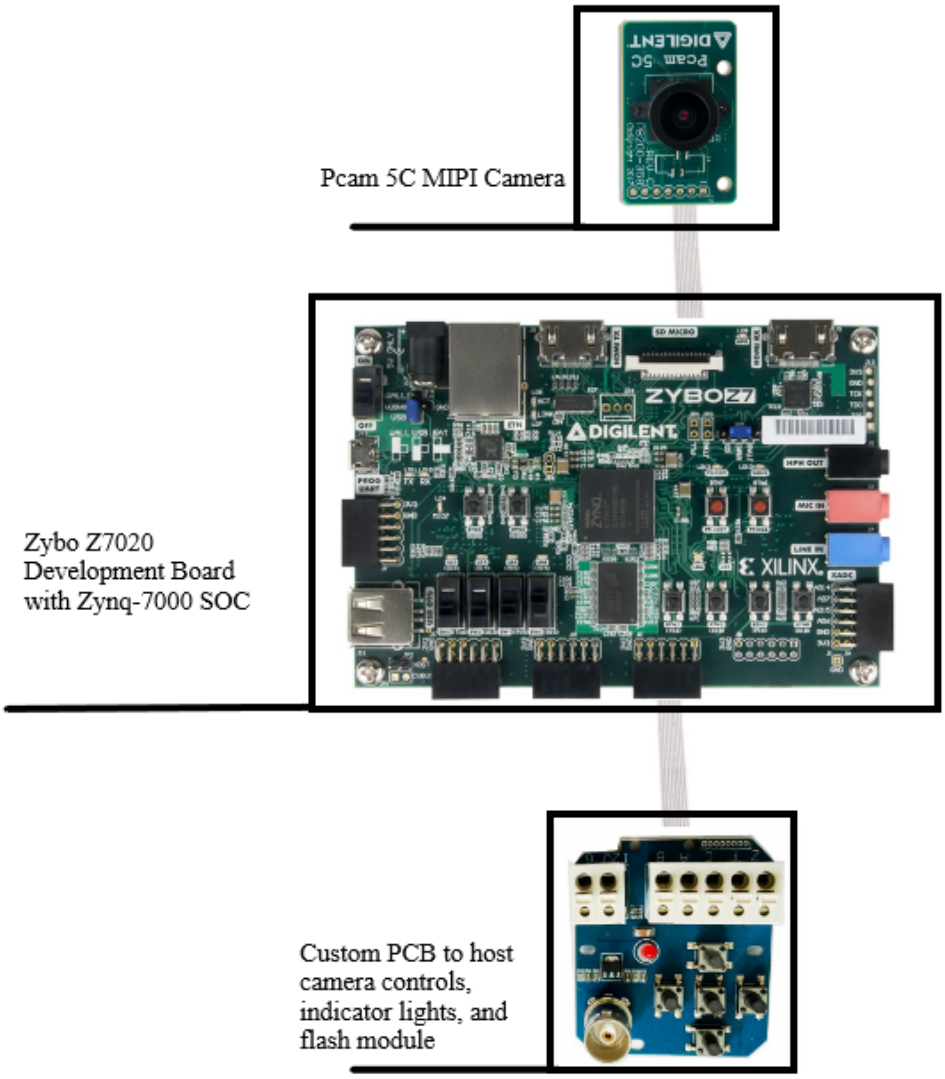
1.1 Problem

Time and energy are resources that aren't plentiful in UAVs. Traditionally when a UAV is used for aerial mapping, it will take a picture every time it flies a predetermined distance interval. Since UAVs must be kept lightweight, it's uncommon to find any with enough onboard processing hardware and energy reserves to stitch hundreds of frames into a map. That's why most mapping UAVs perform the map generation offsite on more powerful hardware than the onboard camera and flight controller. In time sensitive emergencies (open combat, search and rescue, etc), it may not be possible to land the UAV to render an aerial map, and it would be much more convenient if the drone could render the map itself, which could be viewed on a ground station through a UDP radio link.

1.2 Solution

We would like to design a camera that has onboard hardware acceleration capability to stitch images together. When stitching images together into a panorama or map, several repetitive operations are required to "prep" the images for stitching. Operations to grayscale, blur, and convolute images can be performed on a traditional CPU, but the processing time and power consumption can be improved when such repetitive operations are pipelined through an FPGA. With Cole's ECE 397 funding from last semester, he acquired a Diligent Embedded Vision bundle (https://digilent.com/shop/embedded-vision-bundle/), which we plan on using the Zybo Z7020 and PCAM 5C as the basis for the camera. The Zybo board comes with two A9 processor cores which can run Xilinx's Embedded Linux distro called PetaLinux. By running PetaLinux on the camera, we have easier access to the I/O and filesystem on the Zybo board rather than trying to create a bare metal design. After completing this project, Cole plans to integrate the camera into one of his drones, including adding serial communication between the flight controller and the Zybo board (another pro of building on PetaLinux), which would give access to a plethora of sensors such as GPS, airspeed, etc that could bring a live rendering aerial mapping drone into reality!

1.3 Physical Design



1.4 High-Level Requirements

1.4.1 A picture is taken by pressing a button on the external camera control PCB. The Camera will store all pictures as YUV files on the Ext4 filesystem (with 32 GB of space), accessible by the PetaLinux OS.

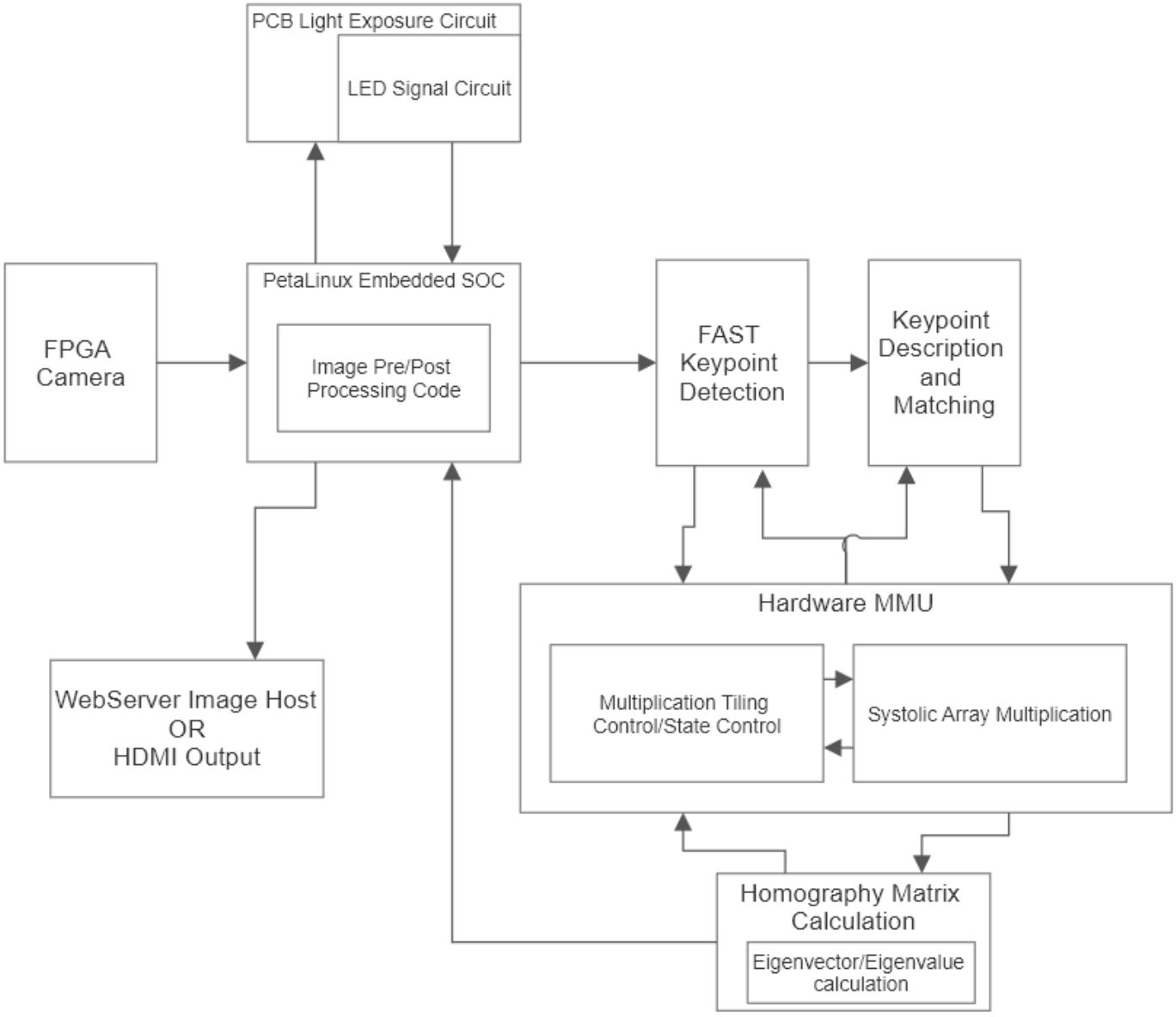
1.4.2 After at least two images have been saved to the filesystem, the user can press a button on the external camera control PCB, which stitches together the images taken into a panorama. By accelerating the stitching process with hardware, the

image processing only takes a few milliseconds, depending on how many images are being stitched together.

1.4.3 After all images have been processed, the resultant panorama will be displayed through the HDMI display. To portray the image at full 1080p resolution, the panorama can be "panned" on the display through the forward and backward buttons on the PCB.

2. Design

2.1 Block Diagram



Many subsystems in the block diagram incorporate the hardware matrix multiply unit. This is a simple systolic array that will be speeding up most of the image processing. The hardware mmu consists of the systolic array doing the multiplication and the control logic for the array input and output.

The FAST algorithm utilizes computer vision techniques for keypoint detection and description within the images. From there, the keypoints are matched between the images using convolution.

The images are then officially stitched together in the same plane using the homography matrix to overlap the images.

The images then proceed back to the SOC for image post-processing and blending. The SOC is also responsible for initializing the image for the accelerator.

While the images are being taken, the light sensing circuit on the PCB sends a signal to the FPGA. The FPGA will communicate with the PCB telling it whether to flash the LEDs for light exposure in the images.

The final stitched together images will then either be hosted on the web or output through HDMI in order to be viewed.

2.2 Subsystems

2.2.1 Hardware MMU

Multiply two 3x3 matrices (inputs)

- Keep the final result in PE accumulators

$$\begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

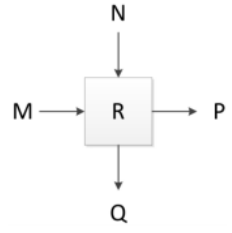
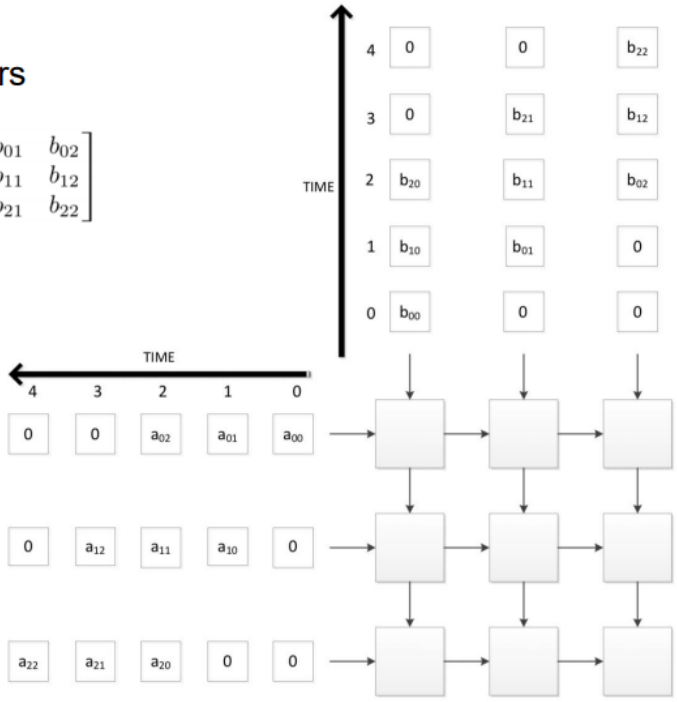


Figure 1: A systolic array processing element

P = M
Q = N
R = R + M*N



One of the key features in the design is the hardware matrix multiplication unit. This is a key unit in any modern hardware acceleration for a multitude of applications including machine learning. Our hardware mmu will consist of a tiled systolic array system similar to the one in the image shown above. A systolic array is much easier to implement for this hardware acceleration than a gpu and is more lightweight in terms of power consumption and fitting on the FPGA. The tiling aspect is to additionally save area on the FPGA for the separate subsystems and the SOC.

2.2.2 Keypoint Detection/Description, and Matching

As mentioned before, the development of this project will be done on the Zybo board that has the embedded Linux environment. The majority of code base and algorithms below would be written in SystemVerilog for the hardware portion. There may be some image pre-processing done in the Linux environment if that is easier to implement.

All image stitching for panoramas has 3 main processes: Keypoint Detection/Description, Keypoint matching, and Homography Transformation.

Keypoint Detection is the process of identifying key points in an image that are recognizable from different angles, lighting, and scale. Many computer vision algorithms accomplish this goal such as SIFT, SURF, and FAST to name a few. We are choosing to implement the FAST algorithm for keypoint detection, not just because it is faster than most other algorithms, but also because it is the least resource intensive for the FPGA to execute. These algorithms already take into account scale and rotational invariance for the images.

Keypoint Description gives each identified keypoint a unique descriptor that can be used to identify each keypoint on the image. Again, there are many methods of doing this, but the simplest is to compile a matrix of the gradient vectors around each keypoint that can be obtained through convolving the image with specific filters.

Keypoint matching occurs when the keypoints are detected and described in each image. If the difference between the descriptors is below a certain error threshold, the key points in each image are said to be a match. Typically, a minimum of 4 keypoint matches is needed for Homography Transformation.

2.2.3 Homography Transformation

When image stitching, the angle of the images needs to be rectified to create a clean output panorama. Homography Transformation is a common problem that

transforms the coordinate system of an image into the plane of the reference image through a 3x3 homography matrix. The homography matrix can be calculated using the keypoint matrix and solving a constrained least squares problem in order to find the eigenvector with the lowest eigenvalue. The method of calculating this matrix is shown below with x being the source image pixels and Y being the destination image pixelsThis transformation is then applied to the destination image. One issue with the homography transformation is that the result can be skewed with outliers in the keypoint matching process, where there are keypoint matches detected, but they are not really matches. A common solution to any outlier problem like this is the RANSAC algorithm. This is easily transferable to hardware and can be used to make the computation of the homography matrix more robust. After the images are warped (transformed) and overlapped, there may be some image blending required for a cleaner result which can be done in the Linux environment.

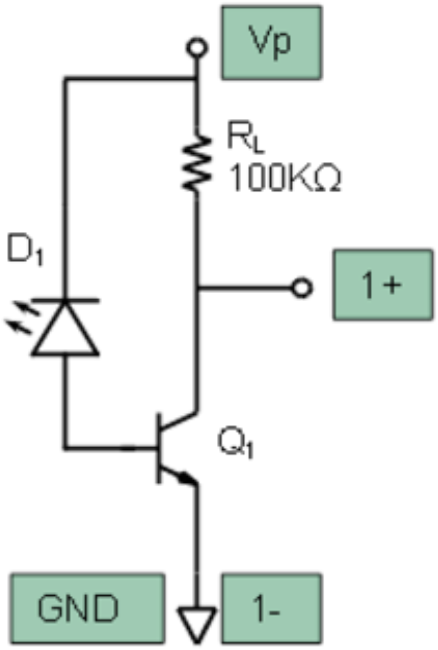
$$\begin{vmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -y_3X_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -y_4X_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1Y_1 & -y_1Y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2Y_2 & -y_2Y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3Y_3 & -y_3Y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4Y_4 & -y_4Y_4 \end{vmatrix} \bullet \begin{vmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{vmatrix} = \begin{vmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{vmatrix}$$

2.2.4 HDMI Output

HDMI output is a system that operates with the TMDS protocol. There have been plenty of people who have created image renderers for HDMI. Our goal is to be able to transfer the image that is being processed in the accelerator through an HDMI renderer we design and output to the HDMI port for an instantaneous results viewer. If the process of creating the accelerator is too long, it would be simpler to host a webserver and display the image in the Linux environment.

2.2.5 PCB Light Exposure Circuit

The PCB circuit would contain two elements, the light level sensor and the LED circuit. The light sensing circuit would be a simple photodiode current amplification sensor shown below. This signal would be sent to the FPGA which would process whether the LED circuit on the PCB should be turned on or off. If the light levels are low enough to trigger the photodiode voltage, then the LEDS would turn on when taking the picture on the FPGA.



2.3 Tolerance Analysis

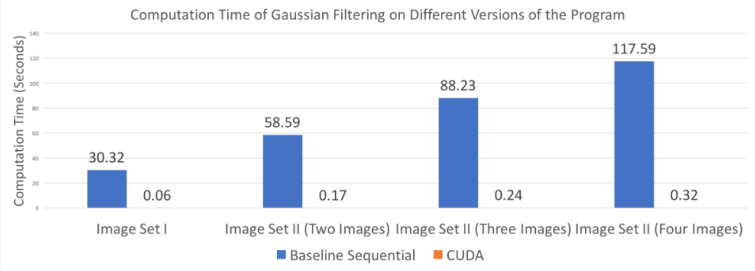
a) Floating point multiplication is extremely resource intensive, especially on an FPGA. For this, the FPGA has dedicated DSP slices, but when multiplying 640x480 pixel images, this floating point multiplication will become unfeasible to incorporate on the FPGA extremely fast. For this reason, the tiling method is implemented to chunk the image into sections. Despite this there still may not be enough resources on the FPGA to fully implement the systolic array for matrix multiplication. According to the FPGA reference manual [2], the Zybo board has 80 DSP slices, 35,200 flip-flops, and 26,400 LUTs. With synthesizable optimized RTL, 32 * 32 multiplication came to 48 DSP slices, 13,420 FFs and 23,293 LUTs [1]. This is clearly unable to be scaled anymore since the number of LUTs has almost hit the maximum. Despite this, the proposed design had a latency of 10

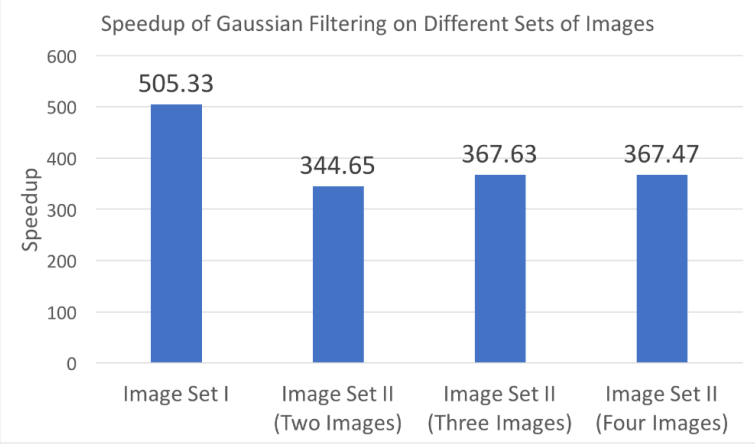
microseconds. If we scale this to our 640x480 image. The stitching process would take anywhere from 200-300 microseconds. The worst case scenario when stitching 3 images together would be around 1ms which is still extremely fast and significantly more lightweight than the large amount of code-bases doing the image stitching in modern day computers.

b) Two algorithms are usually applied for key point matching, parallel and sequential image stitching with their tradeoffs.

Key point matching utilizes the difference of gaussian approach in which we blur the image and subtract the images to find the difference with different levels of gaussian blur. The key points are the pixels that are locally distinct, and we utilize the gaussian pyramid to find multiple key points with the approach. The next step is computing the descriptor in which we compute the gradient of the area and collect the gradient for histogram and find similar local key points.

Computing the gaussian pyramid is usually time consuming and different approaches can be used in the steps. The process of parallelizing gaussian filtering benefits from the parallelism, which reduces the process within seconds[3].





Another challenge of gaussian filtering is the memory constraint. We find out that although the actual computation on a CUDA device is only 0.43s, transferring the images to a CUDA device takes 0.73s, which is almost twice as much as the actual computation time.[4]

The sequential image stitching approach stitch images with optimal seam finding and transition smoothing processes. The sequential panorama stitching procedure enables us to process large source images and create high resolution panoramic images on limited-resource devices such as mobile phones.[5] The steps of the procedure are optimal seam finding and transition smoothing.

During panorama stitching, the approach only needs to keep the panoramic image and the current source image other than all source images in memory, which is good for implementation on mobile devices.[5]

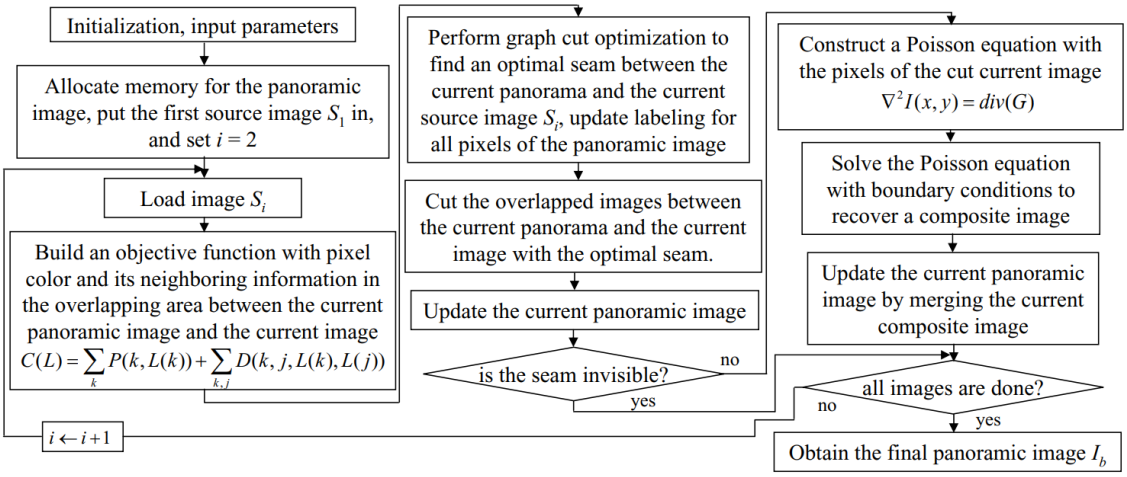


Fig. 1. Work flow of the sequential image stitching approach for creating mobile panoramic images.

2.4 Requirements and Verification Table

Requirements:	Verification:
The homography matrix should be able to be computed within 500 microseconds for the 640x480 images.	The equipment for verification would be the Xilinx simulation environment. 1) The RTL would be loaded into the simulation environment. 2) The testbench would simulate images being inputted to the homography calculation module.

	3) The total runtime from the simulation should be recorded.
The software should be able to blend the images to where end-users could only notice the difference if they really looked.	The webserver hosted by the SOC would be used for this test. 1) The image would be inputted into the Linux environment from the accelerator. 2) The image would be sent to the web host and viewed on the monitor. 3) The results of the image can be qualitatively observed and recorded by the users.
The FAST algorithm should be hardware accelerated to output identifiable keypoints.	The equipment for verification would be the Xilinx simulation environment and a python environment. 1) The RTL would be loaded into the simulation environment. 2) The testbench would simulate image arrays being inputted to the FAST keypoint algorithm. 3) The output of the algorithm would be outputted to a python environment which can display the images with identifiable keypoints. 4) The keypoint accuracy and latency of the simulation should be recorded.

3. Cost and Schedule

The cost of our project is shown in the table below. We do not require much hardwares as our project is mainly based on FPGA. We also take the labor cost into consideration. We expect the team members to work at least 1 hour per day with a salary 45 dollars per

hour. Therefore the labor cost is 45 dollars/hour × 3 members × 1 hour/day × 90 days = 12150 dollars.

Description	Manufacture	Cost
Embedded Vision Bundle	Digilent	400$
LED signal circuit	PCB supplier	50$
HDMI cables	KabelDirekt	10$

The total cost = 12150 + 460 = 12610

Schedule:

Week of 10/3:

- Fully done with research for design, remaining research includes HDMI output and more checks on FAST algorithm

Week of 10/10:

- PCB Light Exposure Circuit done and PCB design done and ready to be ordered.

Week of 10/17:

- FAST algorithm and Hardware MMU fully implemented and synthesizable.

Week of 10/24:

- Homography Matrix module synthesizable and Web Server Host OR HDMI Output finished

Week of 10/31:

- Integration should be almost fully complete with image blending done in the Linux environment. Should have an almost clean image viewable on screen.

Week of 11/7:

- Final touches on integration, including image blending and any algorithmic issues regarding homography or FAST.

4. Ethics

Our project, accelerated panorama image stitching camera upholds the code of ethics I as it has societal implications and great potential applications in dangerous situations.[1]

The image stitching camera can be used for traffic control and cartography. As it provides fast and high quality image output. The image stitching camera also has commercial prospects because it reduces the burden of communication systems as only one panorama is sent per frame contrary to sending several images through the wireless system.

Besides the societal implications, drones equipped with our camera can be deployed in various urgent and perilous natural hazards such as forest fire and earthquake when the wireless stations are shut down or disabled. The information can be quickly processed and the danger is responded to faster compared to traditional methods of communication.

5. Safety

We are also aware of the safety and ethical problems that will come with the project. As there are few high buildings in the Champaign urbana area, it would be safe to control the drone in an open area with few crowds. We should be aware of the intrusion of private properties and right of portrait when experimenting with the camera as it potentially violated the IEEE code 2 “hold paramount the safety, health, and welfare of the public”.[2]

The PCB only requires low and safe voltage. But we should pay attention to the testing of the board, which might cause a short circuit and damage the board. We would follow the fire procedure when the burning gets out of control.

Personal safety is also important when we are working in the lab. We will not engage in experiments with hazardous materials or high voltage electronics, therefore, we will abide by the laboratory safety guidance provided by the University of Illinois.[2] For example, not working alone in the laboratory, not touching wires with two hands, no eating, drinking, or applying cosmetics.

6. References

[1] A Zynq Accelerator for Floating Point Matrix Multiplication Designed with Vivado HLS. https://docs.xilinx.com/v/u/en-US/xapp1170-zynq-hls

[2] Zybo Reference Manual. https://digilent.com/refrence/programmable-logic/zybo/reference-manual

[3] Yingen Xiong and Kari Pulli, *Sequential image stitching for mobile panoramas,* Information, Communications and Signal Processing conference 2010. https://www.researchgate.net/publication/224107399_Sequential_image_stitching_for_mobile_panoramas

[4] Xin Xu and Zhuoqun Chen, *Parallelizing Image Stitching,* https://github.com/JamesOnEarth/Parallel-Image-Stitching

[5] IEEE code of ethics I. IEEE Policies, Section 7 - Professional Activities (Part A - IEEE Policies). https://www.ieee.org/about/corporate/governance/p7-8.html**.**