Hardware Accelerated Image Stitching Camera

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

Fall 2022

1. Introduction

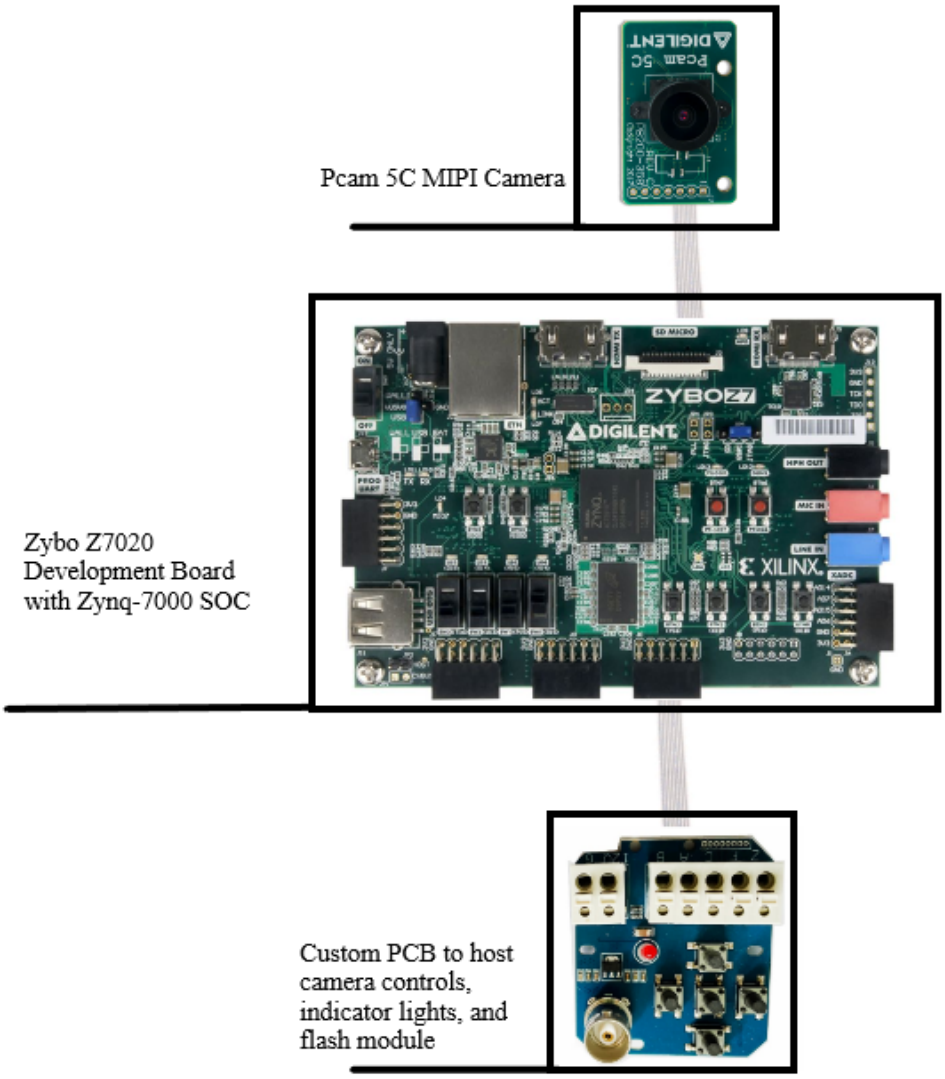
1.1 Problem

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, I 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, I plan to integrate the camera into one of my 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 Camera needs to store all pictures as YUV files on the Ext4 filesystem, accessible by the PetaLinux OS. A picture is taken by pressing a button on the external camera control PCB.

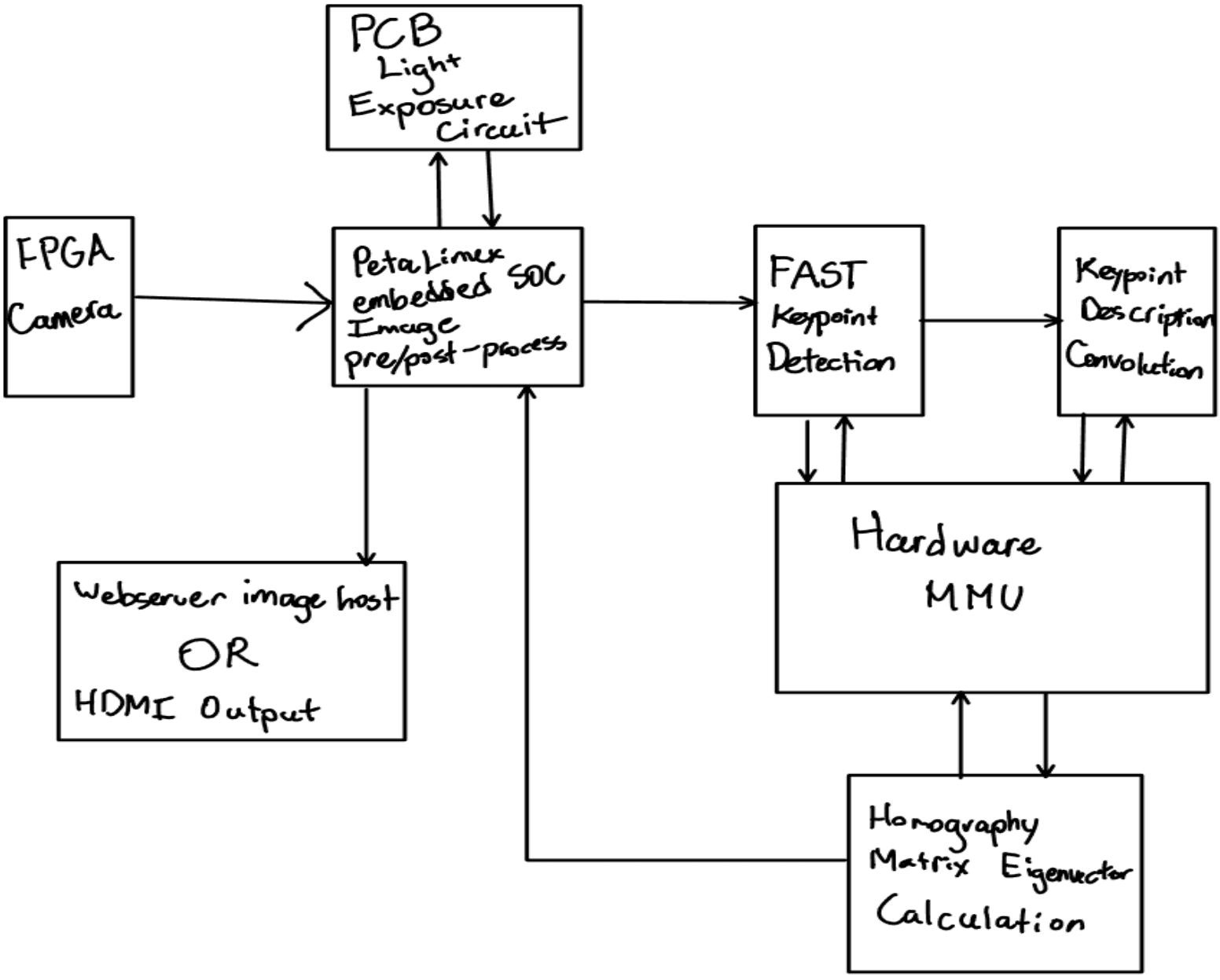
1.4.2 After two images have been saved to the filesystem, the camera needs to execute the stitching process by porting the image data to the FPGA for keypoint

matching (FAST) and homography transformation (with possible addition of RANSAC algorithm for more robust image processing).

1.4.3 After images have been processed, the final image is stored on the filesystem as a RAW image file and displayed to a monitor connected to the HDMI output port.

2. Design

2.1 Block Diagram



2.2 Subsystems

2.2.1 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.

2.2.2 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. This transformation is then applied. 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.

2.2.3 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.

3. Ethics and Safety

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.

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]

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.[3] For example, not working alone in the laboratory, not touching wires with two hands, no eating, drinking, or applying cosmetics.

Reference

[1] 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**.**

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

[3] Laboratory Safety Audit Checklist. Division of research safety. University of Illinois. https://drs.illinois.edu/Page/Programs/PlanOverview#LaboratorySafetyAudits