Characterization of Energy and Performance Bottlenecks in Omnidirectional Camera Systems

by

Sridhar Gunnam

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2018 by the
Graduate Supervisory Committee:

Prof. Robert LiKamWa, Chair
Prof. Pavan Turaga
Prof. Suren Jayasuriya

ARIZONA STATE UNIVERSITY

July 2018

ABSTRACT

Generating real world content for VR is challenging in terms of capturing and processing at high resolution and frame-rates in real-time. The content needs to be captured for 360 and stereo view so that the experience is truly immersive. But capturing such content needs multiple cameras and involves huge amount of computation. The existing solutions only capture at the user and offloading the compute to the server. But offloading large amounts of raw camera feed takes longer latencies and poses difficulties for real-time applications. On the other-hand moving the traditional stitching algorithms to battery constrained device needs at-least three orders of reduction in power. we approach the problem by building a hardware prototype and characterize the end-to-end system bottlenecks like energy and latency. We later propose techniques for optimization.

We found that capturing is bottlenecked by data-rates across interfaces, where as compute is bottlenecked by both data rates and computations. The existing systems lacks hardware, software and algorithm co-design aspects leading to excessive data transfers across the interfaces and expensive computations within the individual sub-systems.The co-design aspects are especially pronounced at Image Signal Processor(ISP) stage. ISP is intermediate stage between the capture and compute, providing interesting data abstractions like motion vectors, bayer statistics to optimize the compute and sensor data traffic respectively.

We emphasize the principles of reuse to optimize the sensor and computation stage power. Reusing the previously captured data can reduce the sensor ADC power, and reusing the previous results can save significant amount of re-computations.

ACKNOWLEDGMENTS

I would like to thank my advisor Prof. Robert Likamwa for his continuous support and guidance. He helped shape my ideas and made me confidant to work in ambiguities. I would like to thank my mom Sujatha (Kaasu) and dad Peddakapu(Bucchiya Chowdary), late grandma Bhanumathi(bapanamma) for their eternal love.

I would like to thank Monish Pabrai and Dakshana, who played a crutial role when I was looking for opportunities in school. I thank my friends Kotta, Palanki, Gadda, Raks, Pathre and Penju for the moral support whenever needed in US. I would like to thank my flatmates KV, TR, Vivek Rai, and Deshpande, and my friends Druthi, and lab buddies Jinhan, Venky, Siddhant, Vraj, Alireja, Paul, and Saad. I especially thank Sahab for his willingness to help others.Most of all I would like to thank Abbu Tataya for his belief in me and supporting me financially. I thank Shanthi pinni and Hari Babi for being supportive to my family during tough times.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

With the advent of AR/VR technologies there is increasing demand to create content specific for these technologies which can either be virtual or real. Virtual content is created using game engines whereas the real world content need to be captured from cameras and then processed to the format in which the content can be viewed in AR/VR. The main characteristics of this content is to provide immersive seamless experience where user can view in any direction as fell as if they were teleported to that location. In order to have such immersive experiences, we need to bridge the gaps in several domains including optics, graphics, audio and video, etc. But during this project we focus on capture systems for 360 degree video.

What it means to have capture visually immersive real world scenes? Researchers Cuervo *et al.* (2018) predict that we need very high resolution(16k) and framerates(120+) to make the experience indistinguishable visually from reality. Current 360 stereo video systems are used mainly designed for professional videography. They consist of several camera(18 in google's jump VR Richardt *et al.* (2017)) which are bulky, and capture lot of data which will later be offloaded and used to generate AR/VR videos thereby limiting their usability. But in order to easily capture and share such experiences we need also need to focus on usability and portability of the devices to make 360 video mainstream in AR/VR.

We increase the usability and portability of the 360 devices if we can capture and stitch the panorama on the same devices. Most of the software use traditional algorithms fit for offloading based approaches and doesn't consider power budget for

implementing 360 capture using a low power portable device. 360 video is essential for VR, but capturing and stitching them in real-time is limited by battery life. Even if battery technologies improve, capturing and stitching 360 video will have heating issues, thereby increasing skin temperature. In-order to tackle the challenge of capturing and stitching on same device, we study the system level bottlenecks in energy and performance by building a prototype. We characterize the system level bottlenecks in terms of energy consumption and latency. Our findings suggest that the main reason for the inefficiency is caused by building the system from off the shelf camera and traditional stitching algorithms. Conventional 360 degree is captured using a multi-camera rig and the expensive stitching is offloaded to powerful machines. Although some systems exist where stitching is done online, they are limited by output resolution, framerate and battery life. We show that the inefficiencies in the pipeline due to lack of hardware algorithm co-design. In this paper we study the data flow of the stitching pipeline by building a protype using 6 camera system. We analyse the energy and performance bottlenecks in the pipeline and analytically evaluate the mechanisms like using motion vectors to reduce temporal data access and computation, use raster buffers instead of full image to optimize on memory, and chip area.

Although commercial 360 degree solutions exist, they are mostly used for capture and stitching is offloaded to powerful machines. This limits the usability of 360 in VR and also portability and for heating. Our goal is therefore to build a 360 camera system that optimizes the entire pipeline both in hardware and software. We characterize the traditional pipeline and propose

The document is organized as follows, in chapter two we discuss about the background and related work, in chapter three describe about the general stitching pipeline for VR panorama generation, and the prototype system design. In chapter four we

present the evaluation results of the prototype system design. We then discuss proposed optimizations in chapter five and our discussions and conclusions in the in chapter six.

The contributions of our work are as follows:

1) Build end-to-end system for capturing 360 degree video.

2) Characterize Individual Stage energy and latency and highlight the bottlenecks in the system.

3) Propose architectures to optimize end-to-end data flow and data abstractions needed at sub-system level. i.e optimizing the spatio-temporal redundancies in the data and computation.

Chapter 2

BACKGROUND

In this chapter, we will discuss about different forms 360 videos, then about the systems used for capturing each of these different forms. We then provide some necessary background to understand the image stitching pipeline.

2.1 Different types of 360 Videos

There are mainly ways 4 types of 360 capturing viz, monoscopic, Omni-directional Stereo(ODS), and Light Field cameras. In this work we will focus on monoscopic and stereoscopic cameras.

(a) Pair of spherical fisheye images (b) Equirectangular projected output

Figure 2.1: A 360 degree capture and corresponding panorama

2.1.1 Monoscopic 360 Degree

Fisheye lenses allow image sensors to capture images within an ultra-wide hemispheric field of view. With two fisheye-lensed sensors that capture complementary fields of view each of over 180°, the pair of captured images can be processed to

achieve over a spherical 360°x 180°area, as shown in Figure x. The equirectangular projection format is a common format for 360°x 180°images, allowing remapping to other projections for convenient viewing.

2.1.2 Omni-directional Stereo(ODS)

ODS output consist of two panoramas one for each eye, and provide the binocular stereo needed for perceiving depth information of the scene with respect to the view point of capturing device. In order to generate such output, we need to capture stereo information from all the viewing directions. Instead of capturing from all the viewing directions, we capture in certain directions, equally distributed over the 360 degree viewing angle and later process them to get the virtual camera viewpoints. We finally get the two panoramas one for each eye, which helps see the 360 view with depth.

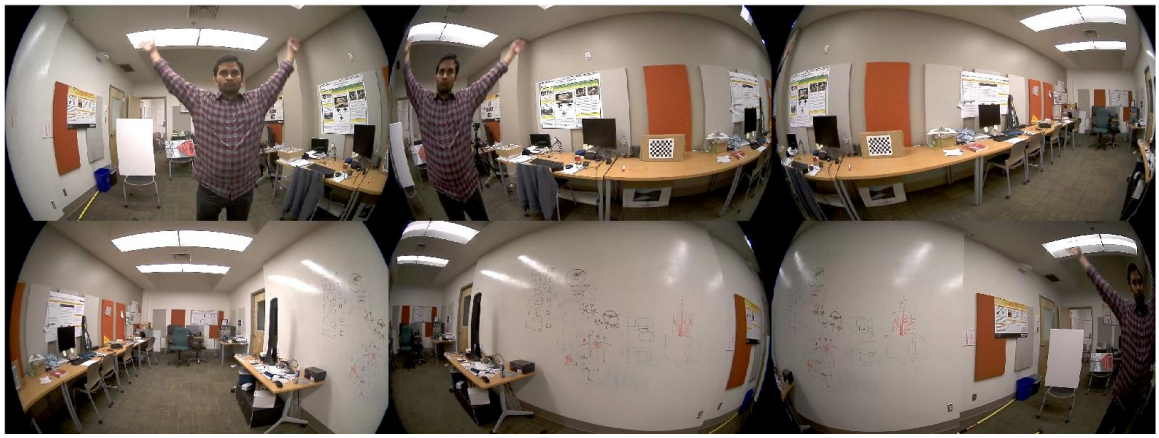


Figure 2.2: Six camera views covering full 360 Hfov

Image Representations: RGB Vs HSV Vs YCbCr(YUV) RGB is mainly for display's. HSV is easier as we have intensity channel is available. For vision applications intensity channel is very useful, for graphics people it's easier as they can change the Hue - color, Saturation - colorfulness(intensity of color, shades of color), value - overall intensity intensity of image. Y'CbCr is used to mainly for storage and trans-

Figure 2.3: ODS panorama, one for each eyes.

mission efficiency. Y' - Luma, which is intensity of gamma corrected RGB image. We use gamma correction, which is non linear encoding of luminance/brightness to adjust for the way humans perceive light and color, i.e we perceive the variations in darkness more than the variations in brightness. And also we perceive light more than color.The more perceivable luma (Y') is used with high precision, whereas color(Cb Cr) can be of low precision, thereby reducing the size of the image.

Chapter 3

DATA FLOW

The end to end system consist of four main stages viz., image sensor, image signal processor(ISP), processor, and Off-chip memory. The image sensor captures raw images, which are processed by ISP to generate RGB images, which are used by processor/accelerator, and the DRAM supports for storage of images and data for all the above stages. The ISP is typically integrated with the processor SoC, and Camera and DRAM are implemented in separate chips.

Fisheye lenses allow image sensors to capture images within an ultra-wide hemispheric field of view. With two fisheye-lensed sensors that capture complementary fields of view each of over 180 °, the pair of captured images can be processed to achieve over a spherical 360 °x 180 °area, as shown in Figure 2. The equirectangular projection format is a common format for 360 °x180 °images, allowing remapping to other projections for convenient viewing. To create equirectangular images, the paired fisheye capture data undergoes multiple stages:

Projection Mapping

The equirectangular image is populated by sourcing image pixels from the fisheye images along a (spherical coordinate âĘŠ polar coordinate) projection map. As projected pixel coordinates typically fall between integer pixel coordinates, the algorithm typically either pulls a nearest-neighbor pixel or a bilinear combination of a neighborhood of pixels.

Correspondence

As the two fisheye cameras do not precisely occupy the same point in space, objects at the edges of fisheye images appear in different positions in the images, dependent on their distances from the camera. This phenomenon is called the parallax effect. To ensure that objects appear properly, a correspondence algorithm identifies matching visual features across image pairs, warping the projection to reduce object seams in the image.

Blending

Even after projection and correspondence suggest image overlay coordinates, intensity variations from misalignments still occur between the two projected images at the stitching boundary. The blending stage combines the images through a weighted sum of pixel values to generate a seamless 360 degree image with a smooth transition.

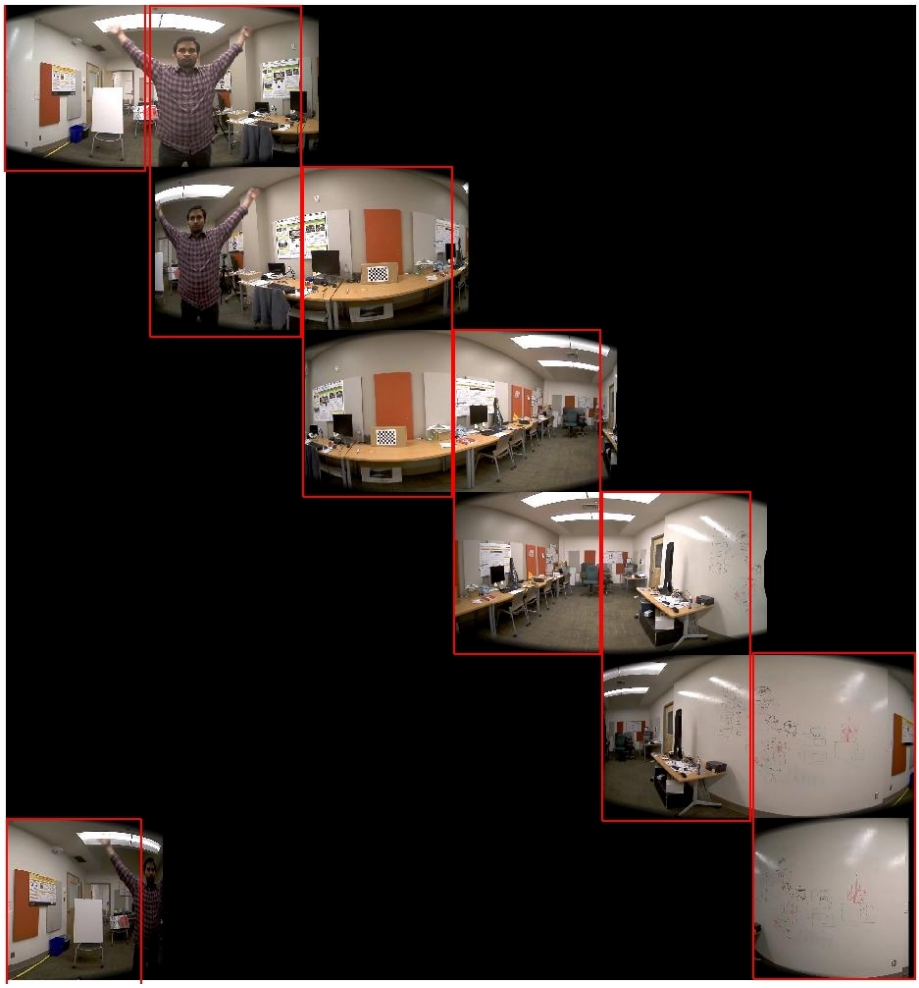
Compression

To reduce the bandwidth at the capture, networking, or storage interface, images can be compressed into representations that use smaller file sizes. Lossy compression schemes, e.g., JPEG/MPEG, allow dramatic reductions in file size by discarding information that is considered to be perceptibly irrelevant.

Hardware

The overlapping left and right images.

Hardware:Six Camera Rig. For our prototype camera design we use six IMX-274 cameras for capture and Nvidia Jetson TX2 for stitching. The Camera and Jetson specs are shown in figure 3.2.



v3.jpg

Figure 3.1: Equirectangular Projection of first and second camera frames

Camera Name	Sony IMx274
Output Image Size	Diagonal 7.20 mm (Type 1 / 2.5) aspect ratio 16:9
Number of Effective Pixels	3864 (H) x 2202 (V) approx. 8.51M pixels
Unit cell size	1.62 um (H) x 1.62 um (V)

Software: Nvidia libargus Camera API, openCV, C++

Data Flow Diagram - With different stages. The main goal of this work is to reduce bandwidth requirement and remove redundant computation during different

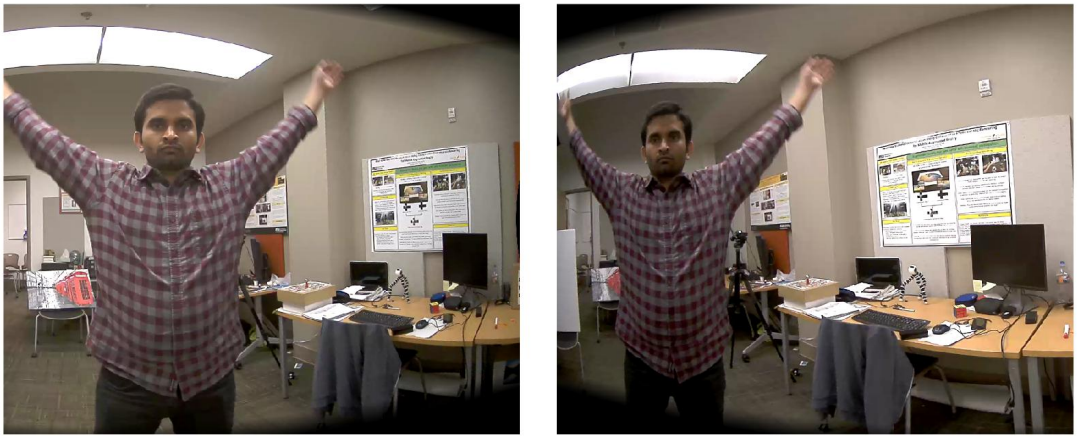


Figure 3.2: Optical flow inputs: Overlapping regions of adjacent camera images. Equirectangular Projection of first and second camera frames

stages.

Chapter 4

CHARACTERIZATION

The goal of the work is to characterize the energy and latency of end-to-end Omni-directional(OD) Camera systems both in the hardware and software pipeline and propose optimizations. As the existing OD camera systems are built from off the shelf camera devices and uses the conventional stitching algorithms, they capture, transfer redundant data and perform redundant computations. The redundancy can be attributed to the sptio-temporal correlation between the frames. The main challenge in OD panorama generation is to understand the data flow across the system and to make decisions on data abstractions needed at different subcomponents to reduce the total system power.

We characterize both monoscopic and ODS camera systems. The difference between them is the number of novel views needed is significantly higher for ODS compared to monoscopic. Both the systems at core use optical flow based stitching, so we will characterize generalized flow based stitching system and notify any important differences between monoscopic and ODS when necessary.

4.1 Measurement Methodology

Jetson has INA3221 monitors and I2C capabilities to read voltage, current and power for different rails on the SOC and IO. The can also moniter the CPU, GPU, memory clock frequencies. For evaluation we measure the absolute energy of the system and the difference between the idle and active state for individual stages of the pipeline. We also use nvidia tegra stats command to check the clock frequencies

of different components like CPU, GPU, Memory Controller for validation.

The latency of the camera capture and ISP is defined by the framerate(i.e throughput), where for computation stages we measure the latencies in terms of CPU runtime of individual software components in the stitching pipeline. One of the critical components of stitching pipeline is optical flow which can take several seconds to compute each output frame on low power embedded CPU. Therefore for realistic estimation of optical flow for accelerator based design, we measure the power and latency of optical flow implementation on zynq FPGA board.

4.2 Energy Characterization

Individual stage energy

Power Rail	Diff. Current(mA)	Voltage(mV)	Energy (mJ/frame)
Sony IMX-274 (Camera)	375.4	3336	41.7
ISP+CODEC (TX2)	102.7	19152	65.6
ARM-A57(Capture + Stitching)	16.4 + 120 (16s)	19144	10.5 + 2296*16
DRAM (Capture + Stitching)	260.4 + 105 (16s)	4792	41.6 + 105*16

Note: For the calculating the energy for frame in the above table, the camera capture is configured to 1920x1080 resolution at 30 fps, and the output resolution is 3k. [make absolute energies instead of diff. i.e active -idle]

As seen in the above table, the stitching in software is highly expensive for CPU,and DRAM blocks, i.e the computation stage energy dominates the capture stage. CPU runtime to render each output frame of 3k is 16 sec, which increases the both the energy and end-to-end latency. The other options include using GPU's,

FPGA, ASIC's. From Nvi Therefore, we approximate the energy and latency for FPGA based accelerator based on Xilinx's implementation of optical flow on Zynq board, discussed in chapter5.

4.3 Latency Characterization

4.3.1 Individual Stage latency

We define stage latency and end-to-end latency for clarity. End-to-end latency is the cumulative latency of all the individual stages. The stage latency is latency of individual stage to process a single frame.

For camera system and ISP stages the stage latency is derived from through-put, i.e inverse of fps. The end to end latency is as given in NVIDIA camera API documentation, which is one frame latency for camera stage, and one for ISP stage.

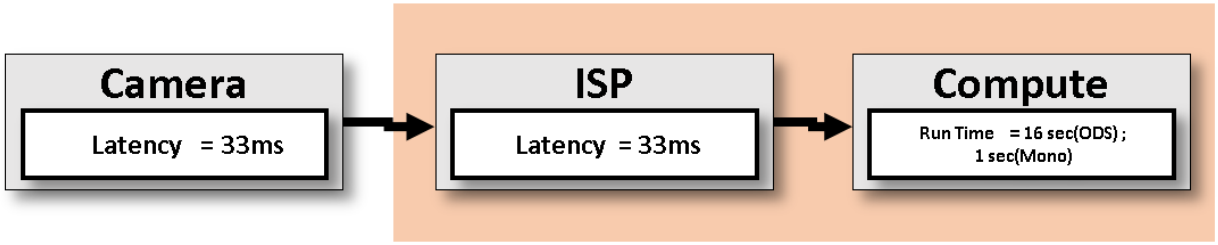


Figure 4.1: Latency of Individual Stage

We measure the latency of computation in terms of CPU runtime. The optical flow, view synthesis and image sharpening stages take 98% of total CPU runtime. Of this the major component is optical flow generation which consumes [] percent of runtime. The sharpening stage doesn't seem to benefit the image quality, so we discard the sharpening stage entirely in the current pipeline and leave it for future

work. We focus on the optical flow stage which dominates both in terms of memory usage, and computation.

4.3.2 Optical flow runtime breakdown

Time for each Pyramid search(98%) as shown in fig 3.1.

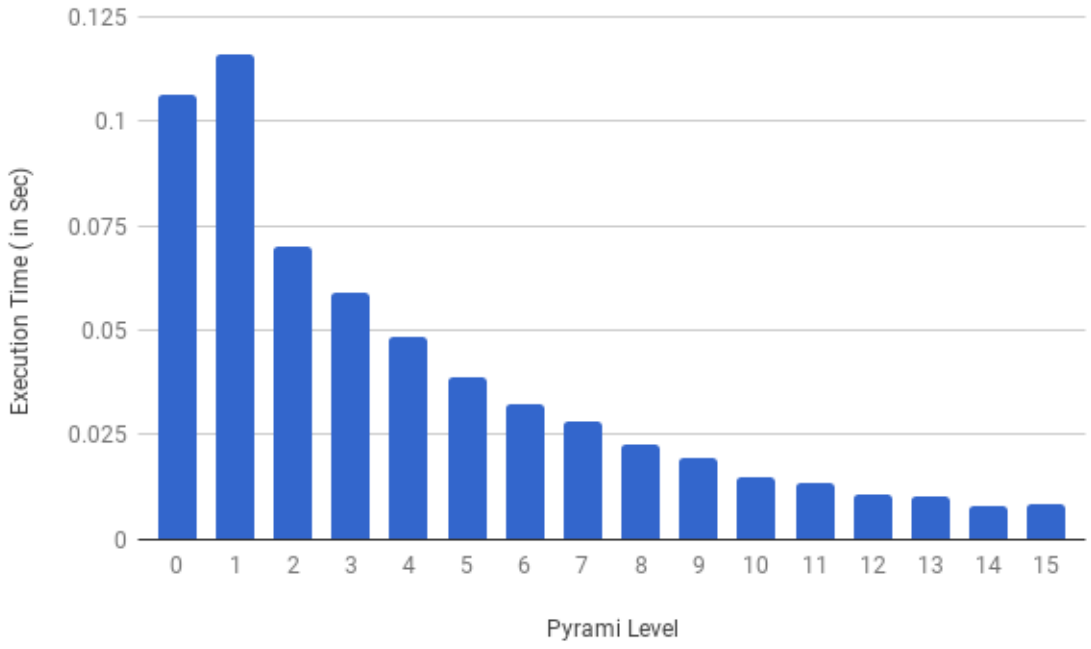


Figure 4.2: X-axis shows the pyramid level and Y-axis shows the runtime for OF

4.4 Camera Sensor Configuration

Sensor Response Time

As the ODS consist of several cameras, it makes sense to reconfigure or turn cameras or off based on the application needs and the scene dynamics. If the camera is not moving and certain portions of the camera views are static, the cameras can be reconfigured dynamically to reduce framerate, resolution, or even turn them on and

off as per the needs. We observed that the reconfiguration latency is one frame delay if there are no outstanding requests, and if there are pending camera requests, they will be served first before requesting the frame with new configuration.

The optical flow works well when the image is has high dynamic range. It is possible that some of the regions in the 360 degree view can be in low lightning, while other are in good lightning conditions. In such cases the stitching fails and can have severe artifacts. We can improve the dynamic range of the particular cameras in low lightning by reconfiguring the camera exposure time dynamically. But such approaches doesn't consider the end to end latency of the cameras and camera movements. To account for camera movements, IMU sensor data can be used to make the camera configuration decisions independent of CPU to accelerate the reconfiguration tasks.

The figures show the differences in low light capture with and without brightness correction. [Integrate the images with histograms]

4.5 Design Scalability

Runtime scalability with the resolution

As discussed in chapter 2 related work, the resolution required for next generation VR is at-least 16k and frame-rates greater than 120. For our evaluation, we assume that energy scales linear with frame-rate and focus our evaluation on scalability of increasing resolution. We can see in fig that even the runtime scales almost linearly with the resolution. Notice that optical flow dominates the total runtime, followed by view synthesis and image sharpening. We also measure the frequencies of CPU, DRAM controller with increasing resolution and observe [linear] dependency of clock

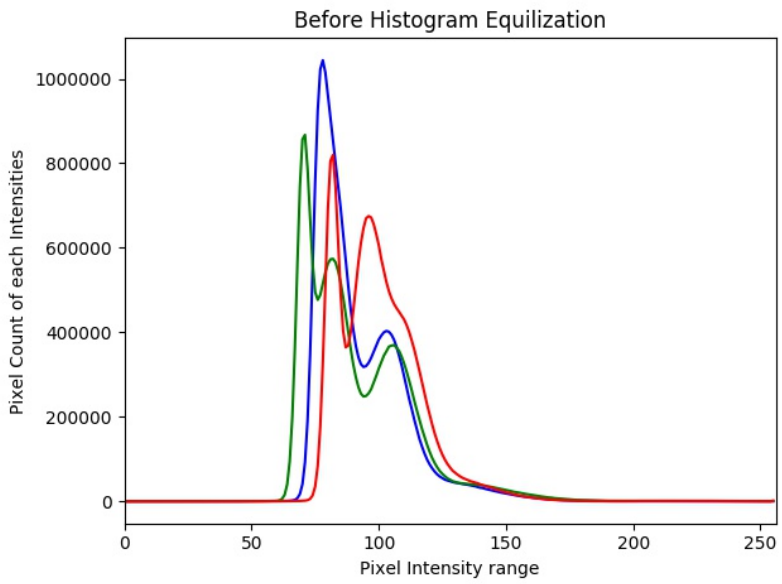


Figure 4.3: X-axis shows the pyramid level and Y-axis the runtime tile search and propagate.

frequency on the resolution.

The main takeaway is that the existing hardware and software scale linearly with the increasing resolution and framerate, which is bad considering the VR panorama requirements. We therefore propose directions to exploit the spatio-temporal redundancies within the frame and across the frames to reduce the data flow and computation. We propose rasterbuffer based designs to decrease the chip resources,and using data abstractions at different hardware IP blocks to share data to reduce temporal redundancies(eg. motion vectors from ISP can be used by optical flow stage, thereby removing the necessity to store previous frames and recomputation of motion data at a later time. The same motion vectors can be used to encode spatial frame data to reduce redundant data transfer).

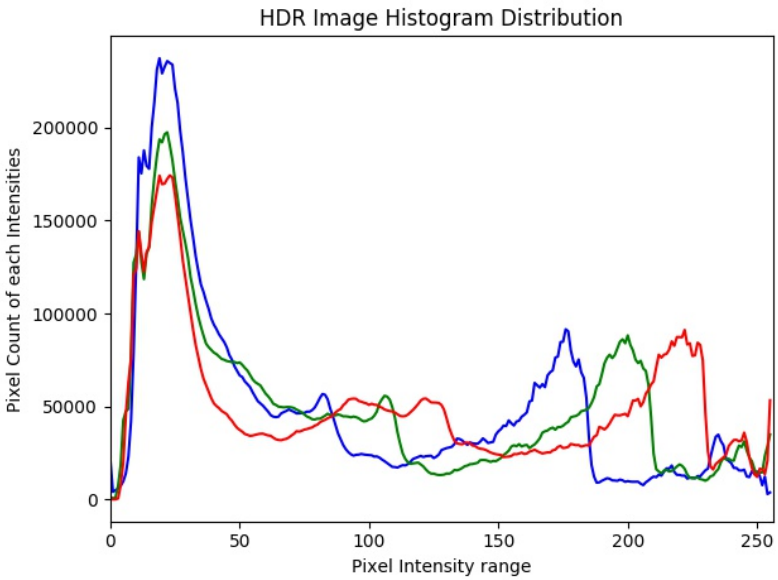


Figure 4.4: X-axis shows the pyramid level and Y-axis the runtime tile search and propagate.

Resource scalability with the resolution

We measure the DRAM capacity required and bandwidth needs as we increase the resolution as a parameter for resource scalability. Higher capacity indicates the need for better encoding schemes and high bandwidth can indicate the temporal redundancy in the data, thereby increasing the bandwidth requirement. For 3k, 4k, 6k and 8k output resolution.

Although we built a system where all the cameras are capturing at same resolution and framerate at a given time, we expect the future cameras make these decisions dynamically to save power. Therefore, we measure the efficiency of capture and ISP processing at different modes of operation and measure the efficiency of capture and processing in power consumed per pixel at different modes.

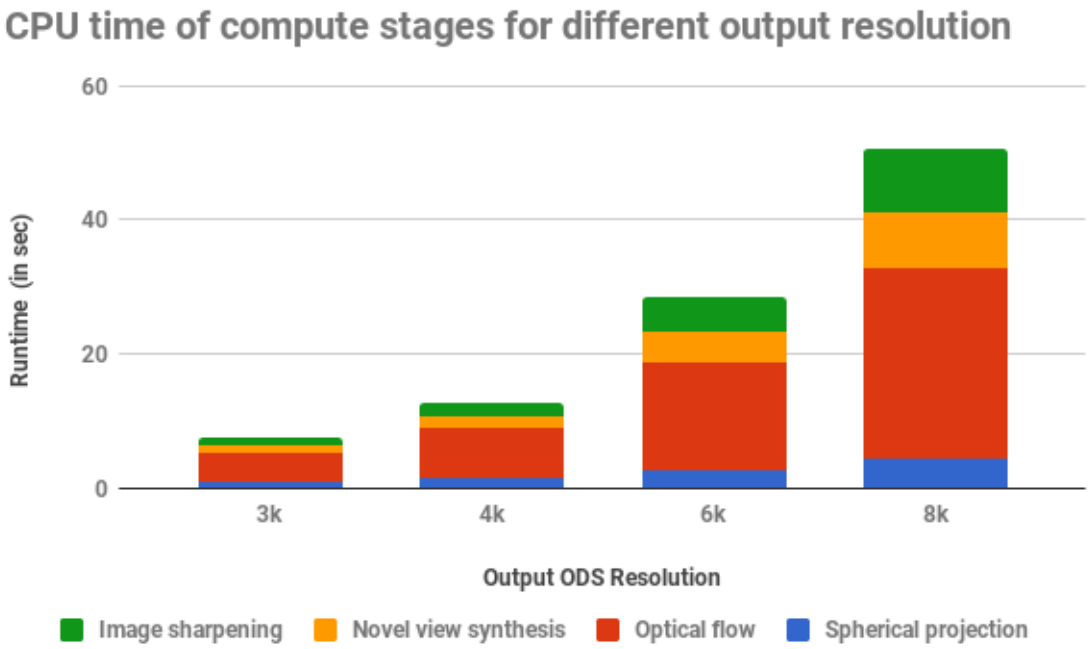


Figure 4.5: CPU execution time of different compute stages. X axis has different sub-stages in optical flow and Y axis correspond to energy per frame.

4.6 Evaluating data-flow redundancies

Evaluating redundant computations in optical flow. Accuracy Vs Energy&Perf tradeoff. Next we evaluate the optical flow for a scene where foreground has movement and background is static. The temporal flow difference is found out to see the variation of flow. As expected we see the flow change only for the regions where the objects are moving. This observation suggests that accuracy of optical flow can be trade-off with computation time to save energy and latency. It also shows that the accuracy drop is less than _ percent which can be acceptable

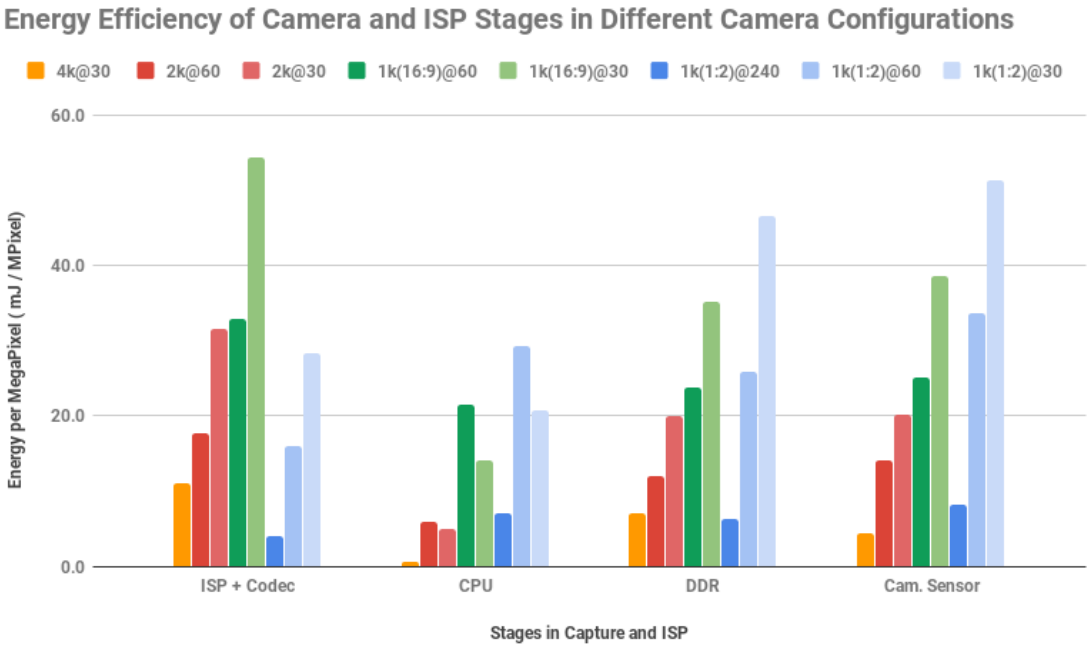


Figure 4.6: Power Efficiency of Camera ISP Stages in different configurations

4.7 Misc

Increased frame-rate

What is the percentage of new data ffmpeg I-frame size to the P-frame ratio. we usually have I,P, and B frames in a compressed video, but in case of realtime compression we will have only I and P frames and disable B frames as it adds latency to the pipeline. Typically I-frame to P-frame is about 4 times and one I-frame occur for 30 P-frames. This implies we save a lot on interface power if we can push the computation to near sensor. [Numbers - savings from IO datarate reduction]

3) Survey of Camera and ISP stage energy breakdown Camera Sensor and ISP power directly taken from literature. Computation Power split into sub stages.

For power characterization of camera sensor and ISP, we run the camera in different resolutions and framerates and see how the various sub-component power changes.

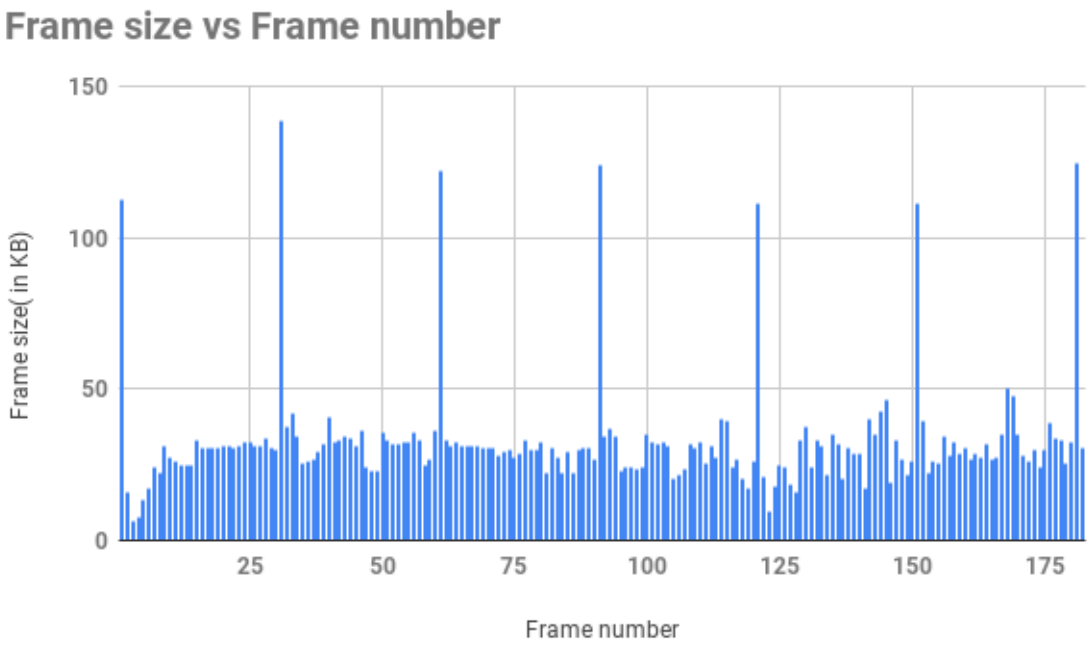


Figure 4.7: Framesize of I and P frames

The components include Camera Sensor, I/O, ISP, CODEC, DDR, and CPU. The ISP, and CODEC power are combined as they belong to same SOC voltage rail.

The most used configuration for our project when all the six cameras are capturing 1920x1080 @ 30 fps. At this configuration below is the split of different component power.

e) Quality Tradeoff's with input resolution

Sharpening Reduction in fidelity of unwarped image, as interpolation is not being done. Reducing number of pyramid's f) High motion Vs low motion differences. Size of motion vector to that of size of full frame. g) File IO power h) Breakdown in terms of type of Memory used i) Breakdown in terns of type of Computation j) Breakdown in terms of IO bandwidth bottlenecks

Chapter 5

DISCUSSION AND CONCLUSION

Based on the characterization of end-to-end system pipeline we can group the proposed optimizations into two categories, i.e to optimize for low power and another for low latency. For low power we talk about techniques to reduce sensor power and then about spatio-temporal data reuse during computation. For reducing the compute latency we talk about use of pipelining and parallelism and the expected benefits.

5.0.1 Techniques for Low Power

Reducing the ADC Power

In a multi-camera system, the combined power of sensors becomes a critical bottlenecks. Most typical image sensors are not intelligent to compute over the only the changing pixels. The temporal frames have high amount of redundancy, when there is less motion of camera and/or the objects in the scene. Therefore it is possible to have different modes of camera operation through which we can save power. The different modes could be reducing the resolution and frame-rates, which are explored in previous works[jinhan]. In addition to such optimizations, we propose the adaptability of camera's ADC circuitry to save power even further.

A camera capturing 4k, 30fps typically consumes 500mW of power, and the ADC contributes to 60% of total sensor power. We observed the dynamic range is smaller for local regions which in frame, i.e the pixels will have same most significant bits(msb) and only vary for the least significant bits(lsb). This information can be used to reduce the number of ADC conversion cycles in a SAR like ADC's. The conversions can only

start for the last few bits thereby reducing ADC conversion cycles by more than 50%. It should also be noted that the voltage swings for resolving msb bits is higher than the lsb bits. Therefore there is extra savings in terms of reducing the voltage swing. But when the predicted starting point of ADC conversion goes wrong, the conversion should start for msb bit.

Reducing Camera Serial Interface Power

The sensor interface power consumes about 17% of total system power during capture. Currently the interfaces stream the raw camera data to ISP. But this power can also be reduced by using light weight compression techniques and encoding using Huffman encoding. But in order to be able to use Huffman encoding, we need the entropy data. As recent ISP's have a bayer statistics accelerator embedded, we can easily generate the approximate entropy of frame for various regions. Such co-design techniques should drastically reduce the interface power and even enhance the effective number of pixels transmitted for a given channel bandwidth.

Real world content generation for VR is an emerging research problem. VR content needs capturing for 360 degree view and 3D immersive experience. Although commercial solutions exist, the dataflow is broken resulting to long latencies. The workflow consist of camera that captures the multiple views and offload the computing to desktop or cloud for stitching the video. These solutions needs several 1000's of CPU's if not multiple GPU's consuming consuming upto 1 kilo Watt power to stitch a 4k video in realtime. Cloud based solutions will have privacy concerns considering the user may share personal content in VR in real-time We envision that capturing and rendering near camera helps improving the end-to-end latency and reduce the power by close integration of the capture and rendering system. The existing 360 camera devices have portable camera rigs that capture and offload the expensive computa-

tion to cloud, or powerful desktops. This limits the scalability of stitching operation and increases the end-to-end latency. But performing capture and generating the VR panorama on the same device is computationally expensive. Our work focuses on characterizing the energy and latency of end-to-end Omni-directional(OD) Camera systems. The goal is to find the bottlenecks of different components in the hardware and software pipeline and propose optimizations. Capturing consumes 24fps/Watt at 4k resolution whereas computing has efficiency of 0.01fps/Watt on a quad core ARM-A57. DRAM-less Stacked Image Sensors ADC readout power, Rolling vs global shutter Abstractions for hardware software co-design a) How to find out which data to sense, send without the intervention of computationally intensive vision algorithms. b) How to detect them early in the vision pipeline c) Data Driven

Instead of approximating 3D object movements as 2D motion block translation, we need to model the 3D model rotations and translations in compression algorithms.

Optical flow in spherical coordinates.

3) Develop methods to perform computation on compressed data.

4) On compression: Evaluate different compression algorithms and compression rates and how they affect data transfer and performing computation on compressed data. We may need to invent/improve existing compression algorithms according to requirements of our application. 5) Dynamically perform region based compression. If region based compression is performed dynamically, we need architect the system to enable communication between stages to provide necessary compression information. (This might be use case in continuous vision sensing for surveillance application where the region of interest could be a moving object. We may want different compression rates based on region of interest.) 6) In region based compression, the following stages should be aware of which region is compressed and should accordinglyâĂŃ perform computation in it. We should check for feasibility to multiple compression ratios on

the same image frame for different regions but still perform computations appropriately. Some of the research challenges include: Making use of as less cameras as possible. Computation Vs Capture tradeoffs Need for novel computation methods to reduce design size and data transfer. Can we generate High resolution Stereoscopic capture using Low resolution monoscopic 360 video capture + low resolution Stereoscopic capture. Image representation for high compression, decompression and view transformation. Can we generate one equirectangular representation from another with simple transformations. Where should this views be generated? Can we do some sort of encoding between the two views for highly correlated regions? (Assign ID to common regions, transmit only one of them)

Communication and Power Regulators Humans communication is adaptive to situations. We speak fastly/slowly, change languages, our protocols are adaptive i.e we interrupt others sometimes/ and don't few other times. But the system interfaces are static and are not adaptive, they have fixed protocols and data rates. Can we get inspired from nature and make these interfaces more programmable and dynamically adapt to the requirements of the system?? What is stopping us?? Where do such adaptive interfaces comes into picture?? We do have some sort of dynamic nature at interface hubs. But are they programmable? Optimizing number of cameras using software modelling in Unity/xyz using multiple virtual cameras. Create virtual camera rig, set camera parameters of virtual camera and capture images [Stanford Work] Stitch images captured by virtual camera Compare with the ground truth, i.e the actual 3D environment https://blogs.unity3d.com/2018/01/26/stereo-360-image-and-video-capture/

5.0.2 future work

Future Work: Divide it into: Hardware/Software and New Technology

Sub categories of different domains and fields. Eg: Vision, Graphics, ML, Systems, Networking etc Graphics: Model generation Vision: Systems: Light Field Cameras

Challenges in immersive 360 degree capture for natural environments: Reflective challenges in 360 camera capture. Different cameras see different reflections, will the math still be the same. It won't be, because we have not considered illumination.

Filling the holes using AI

Image representations

Viewpoint aware static and dynamic scene recognition Integration of codecs Near sensor ADC Motivation for SAR or hybrid SAR to single slope ADC(state of the ART) Reducing computation

5.0.3 Research Questions

1) Data Flow: Reducing redundant computation and transfer.

Cause ⇒ Effect

Cause:

Temporal frame re-reference(reuse) for motion vector calculation.

Previous optical flow re-reference for temporal regularization of current optical flow.

Effect:

Increased DRAM memory consumption

Increased Memory Bandwidth

Increased end-to-end pipeline latency

Cause:

Re-computation of entire flow pyramids for each frame. (Pyramids of previous flow, estimated flow on current frame, current gray image, and alpha channel, all in floating point)

Effect:

Increasing the number of computations needed for each frame.
(Differentiating between the background and foreground and motion vector inputs to reduce the computations. background information is available in the form of optical flow correspondence, motion vectors are available from the ISP/motion estimation block)

2) Data abstractions and formats? When and where should we find motion vectors? How much is size of local buffers at each stage? Make analytical producer consumer model? How much energy saving at ADC readout? How much savings because of DRAM organization? and bit quantization?

Data Abstraction and Data Representations

Equirectangular format stretches the region near poles thereby storing redundant data. It is also difficult to compress as the existing motion estimation for video compression[facebook] doesn't work on equirectangular format. Cubemap on the other hand reduces the storage of redundant information at the poles. It also helps in compression as we can use existing compression techniques. There is also a trend for equi-angular cubemaps which reduce the file size even further by mapping the spherical images onto smaller cubes.

Which is better format to represent 360 videos monoscopic videos? Equirectangular or cubemap or new formats which can represent spherical images that reduce both data storage and rendering effort? As these formats keep changing, we need design systems that are independent of the type of projection format.

5.1 Conclusion

Characterization summary Combination of feature based and dense correspondence based optical flow.

REFERENCES

Cuervo, E., K. Chintalapudi and M. Kotaru, "Creating the perfect illusion: What will it take to create life-like virtual reality headsets?", in "Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications", pp. 7–12 (ACM, 2018).

Richardt, C., J. Tompkin, J. Halsey, A. Hertzmann, J. Starck and O. Wang, "Video for virtual reality", in "ACM SIGGRAPH 2017 Courses", p. 16 (ACM, 2017).

APPENDIX A

MONOSCOPIC STITCHING IMAGES

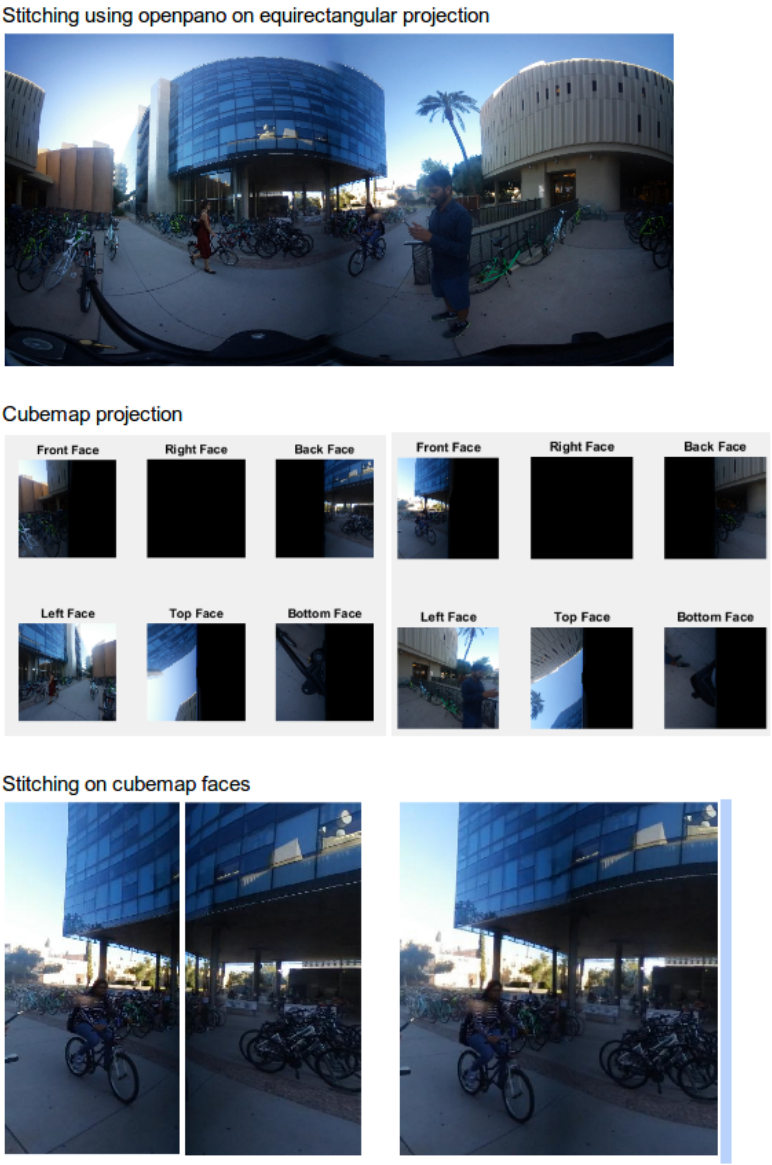


Figure A.1: Cubemap based Stitching