

Characterization of Energy and Performance Bottlenecks in an Omni-directional  
Camera System  
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:

Robert LiKamWa, Chair  
Pavan Turaga  
Suren Jayasuriya

ARIZONA STATE UNIVERSITY  
August 2018

## ABSTRACT

Generating real-world content for VR is challenging in terms of capturing and processing at high resolution and high frame-rates. The content needs to represent a truly immersive experience, where the user can look around in 360-degree view and perceive the depth of the scene. The existing solutions only capture and offload the compute load to the server. But offloading large amounts of raw camera feeds takes longer latencies and poses difficulties for real-time applications. By capturing and computing on the edge, we can closely integrate the systems and optimize for low latency. However, moving the traditional stitching algorithms to battery constrained device needs at least three orders of magnitude reduction in power. We believe that close integration of capture and compute stages will lead to reduced overall system power.

We approach the problem by building a hardware prototype and characterize the end-to-end system bottlenecks of power and performance. The prototype has 6 IMX274 cameras and uses Nvidia Jetson TX2 development board for capture and computation. We found that capturing is bottlenecked by sensor power and data-rates across interfaces, whereas compute is limited by the total number of computations per frame. Our characterization shows that redundant capture and redundant computations lead to high power, huge memory footprint, and high latency. The existing systems lack hardware-software co-design aspects, leading to excessive data transfers across the interfaces and expensive computations within the individual subsystems. Finally, we propose mechanisms to optimize the system for low power and low latency. We emphasize the importance of co-design of different subsystems to reduce and reuse the data. For example, reusing the motion vectors of the ISP stage reduces the memory footprint of the stereo correspondence stage. Our estimates show that pipelining and parallelization on custom FPGA can achieve real time stitching.

## 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 and dad Peddakapu, brother Raghu, and late grandma Bhanumathi for their eternal love and support. I would like to thank Monish Pabrai and Dakshana, who played a crucial 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
LIST OF TABLES .....	iv
LIST OF FIGURES .....	v
CHAPTER	
1 INTRODUCTION .....	1
2 RELATED WORK AND BACKGROUND .....	3
2.1 Related Work .....	3
2.2 Background .....	3
2.2.1 Types of 360 degree videos .....	4
2.2.2 Stages in ODS Stitching .....	5
2.3 System Overview and Data Flow.....	8
3 CHARACTERIZATION .....	12
3.1 Measurement Methodology .....	12
3.2 Energy Characterization .....	13
3.3 Latency Characterization.....	17
3.4 Design Scalability .....	17
3.5 Different sensor configurations for energy efficiency and quality.....	20
4 PROPOSED OPTIMIZATIONS .....	23
4.0.1 Techniques for Low Power .....	23
4.0.2 Techniques for Low Latency .....	25
5 CONCLUSION .....	26
REFERENCES .....	27
APPENDIX	

## LIST OF TABLES

Table	Page
2.1 Prototype Specifications .....	11
3.1 Energy Characterization of Individual Stages .....	13

## LIST OF FIGURES

Figure	Page
2.1 A 360 degree capture and corresponding panorama .....	4
2.2 Left side of the picture shows the multi-view point capture and the right side shows the two ODS panorama, one for each eyes. ....	5
2.3 Equirectangular Projection of different fisheye images with offsets showing where they belong in the final ODS panorama. ....	6
2.4 From the overlapping view of adjacent views, we get stereo disparity, i.e how far each point is away from the camera. ....	8
2.5 System overview, the Camera, ISP, Processor and Storage. ....	9
2.6 Camera capture rig and the evaluation platform.....	10
3.1 X axis shows the variations of pixels across 1000 consecutive frames, Y axis shows total number of pixels with a particular variation .....	15
3.2 Chart showing the distribution of power of different components during capture and storage. ....	16
3.3 X-axis shows the pyramid level and Y-axis shows the runtime for OF ..	18
3.4 CPU execution time of different compute stages. X axis has different sub-stages in optical flow and Y axis correspond to energy per frame. .	19
3.5 Energy Efficiency of Camera ISP Stages in different configurations....	20
3.6 Image intensity histogram for low lightning conditions .....	21
3.7 Image intensity histogram for good lightning conditions .....	22

# Chapter 1

## INTRODUCTION

With the advent of AR/VR technologies there is an increasing demand to capture real world content for immersive viewing experience. The real world content needs 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 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 does 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 visually indistinguishable 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 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 uses 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 videos in real-time is limited by battery life. In-order to tackle the challenge of capturing and stitching on the 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 performance per watt 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 video 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 are due to lack of hardware algorithm co-design. In this paper we study the data flow of the stitching pipeline by building a prototype using 6 camera system. We analyze the energy and performance bottlenecks in the pipeline and analytically evaluate the proposed optimizations.

The document is organized as follows, in chapter two we discuss about the background and related work describing the general stitching pipeline for VR panorama generation. In chapter three we present the evaluation results of the prototype system design. We then discuss proposed optimizations in chapter four, and conclude in chapter five.

The contributions of our work are as follows:

- 1) Build end-to-end system for capturing 360 degree video.
- 2) Characterize individual stage power and performance 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 capture and computation.

## Chapter 2

### RELATED WORK AND BACKGROUND

#### 2.1 Related Work

The methods of capturing omnidirectional stereo to capture environment maps has been around for few decades and is well established in Peleg *et al.* (2001); Kang *et al.* (2000); Ishiguro *et al.* (1990). But these techniques use rotating camera and capture multiple viewpoints of 360 degree view to generate final stereoscopic output. Such rotating setups can only be used for still captures. Others Richardt *et al.* (2013); Ho and Budagavi (2017) show VR panorama generation using a mobile camera device or with lesser number of cameras. But the first robust implementations of ODS video stitching were demonstrated by Richardt *et al.* (2017); facebook (2017). But the challenge with these and other commercial systems is that they are used only for capture and stitching ODS is offloaded to either desktop or to cloud. The cloud based stitching is not optimized for latency and power. Sending large amounts of unprocessed frames to cloud takes longer time, and the need for compression and decompression makes the real-time streaming solution difficult in offloading based solutions. Also these solutions currently use several 1000's of CPU's, if not several GPU's in order to stitch ODS at 4k resolution. Such high compute resource is not scalable to larger audience.

#### 2.2 Background

In this section, we will discuss different types 360 videos, then the system level overview and the data flow within the system. We provide some necessary background

to understand the image stitching pipeline by visually showing the intermediate outputs.

### 2.2.1 Types of 360 degree videos

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

# 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  $180^\circ$ , the pair of captured images can be processed to achieve over a spherical  $360^\circ \times 180^\circ$  area, as shown in 2.1 The equirectangular projection format is a common format for  $360^\circ \times 180^\circ$  images, allowing remapping to other projections for convenient viewing. To create equirectangular images, the paired fisheye capture data goes through multiple stages, Viz, Projection Mapping, Correspondence(optical flow), view synthesis, and compression.



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

(b) Equirectangular projected output

Figure 2.1: A 360 degree capture and corresponding panorama

## Omni-directional Stereo(ODS)

ODS output consists 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.

The inputs and outputs can be seen in 2.2.

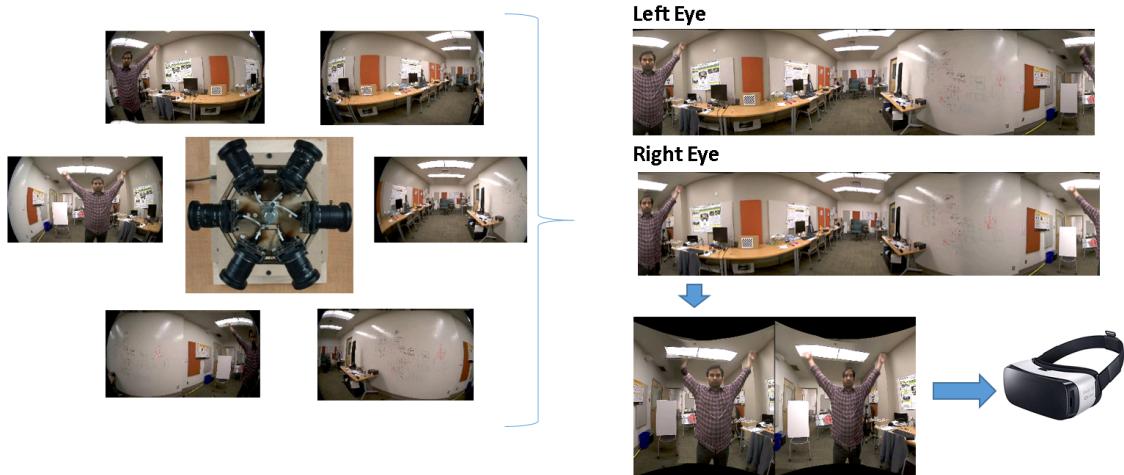


Figure 2.2: Left side of the picture shows the multi-view point capture and the right side shows the two ODS panorama, one for each eyes.

### 2.2.2 Stages in ODS Stitching

The inputs of the ODS system are fisheye images and the outputs are two stereo panoramas. We will need multiple view points so that we can capture both in 360 and in depth. But at high level both for monoscopic and stereoscopic we need to go though

the same stages for generating output viz., Projection Mapping, Correspondence, blending, compression.

## Projection Mapping

The equirectangular images shown in 2.3 are populated by sourcing image pixels from the fisheye images along a (spherical coordinate to 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.

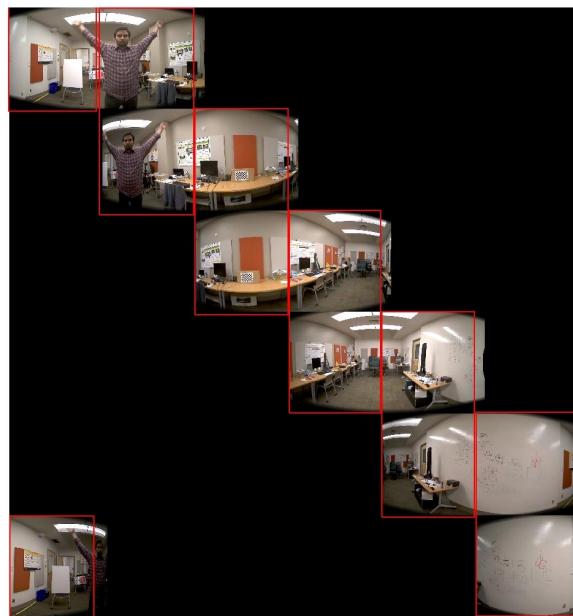


Figure 2.3: Equirectangular Projection of different fisheye images with offsets showing where they belong in the final ODS panorama.

## Stereo 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. For ODS, we need dense stereo correspondence between the adjacent views to generate novel views as shown in 2.4. We generate the stereo correspondence using spatial optical flow, i.e the optical flow between adjacent cameras. The optical flow signifies how far or how near a point is from the capture rig.

## View Synthesis

The ODS needs novel camera views so that it can collect rays from all the directions. But as we have limited number of cameras, we can use these camera views and the dense stereo correspondences to generate the novel views which represent the images taken if there was a camera in between.

## 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°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



Figure 2.4: From the overlapping view of adjacent views, we get stereo disparity, i.e how far each point is away from the camera.

schemes, e.g., JPEG/MPEG, allow dramatic reductions in file size by discarding information that is considered to be perceptibly irrelevant.

### 2.3 System Overview and Data Flow

The end to end system consist of four main stages : image sensor, image signal processor(ISP), processor, and Off-chip memory, as shown in 2.5 The image sensor captures raw images, which are processed by ISP to generate RGB images 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.

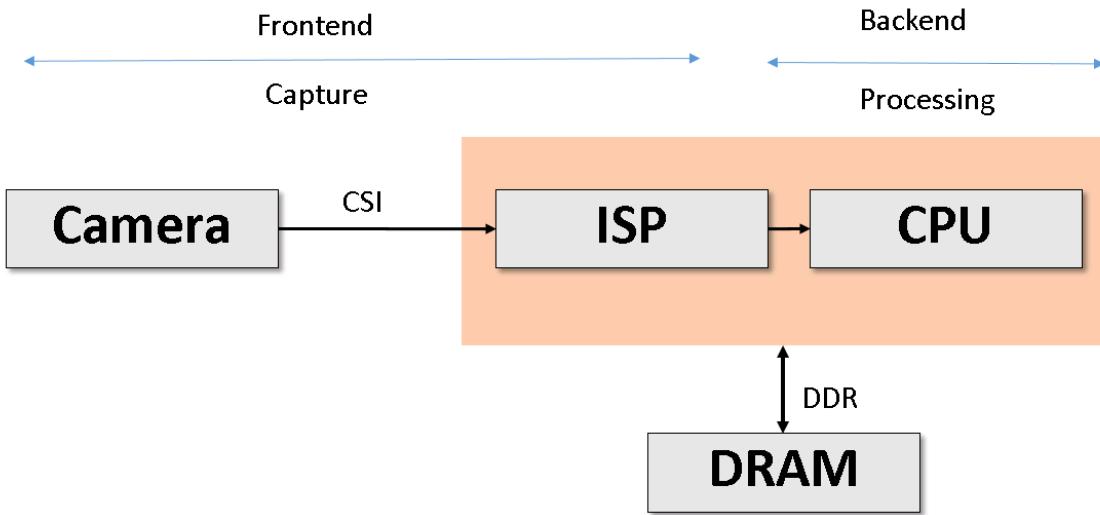


Figure 2.5: System overview, the Camera, ISP, Processor and Storage.

## Hardware

We have 3 major components, the cameras, the support rig and the evaluation platform. For capture we use six cameras with 2k resolution and 30 fps. The table 2.1 shows the specs of the cameras used for prototype design. The rig is designed with precision using laser cutting a hard cardboard. For capturing and computation we use Nvidia Jetson TX2 board.

## Software

For camera capture we use the libargus camera api provided by Nvidia. For stitching we adapt the facebook surround 360 to work on our platform.

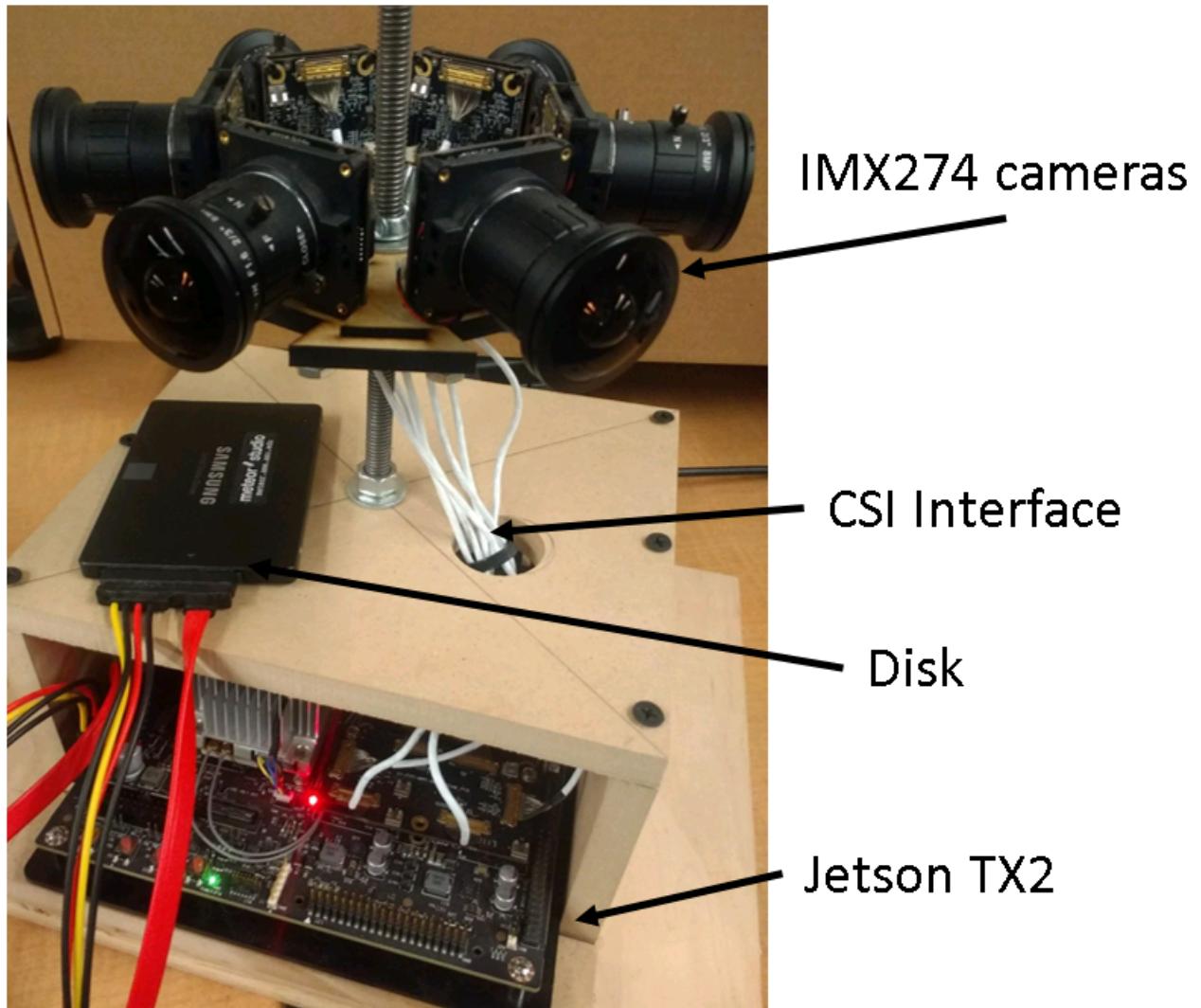


Figure 2.6: Camera capture rig and the evaluation platform

Cameras	
Type	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)
Hardware	
Board	Nvidia Jetson TX2
CPU	Quad-core ARM A57
ISP	1200 Million Pix/Sec
Software	
OS	Ubuntu 16.04
Stitching	Facebook surround 360
Implementation	c++, openCV

Table 2.1: Prototype Specifications

## Chapter 3

### CHARACTERIZATION

The goal of the work is to characterize the energy and latency of end-to-end ODS camera systems and propose optimizations. As the existing ODS camera systems are built from off the shelf camera devices and use the conventional stitching algorithms, they capture redundant data and perform redundant computations. The main challenge in ODS 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 and latency.

We classify our findings into two categories, i.e the reasons for high energy and high latency. For high energy or power consumption, we divide into following three categories

- Redundant Capture, and Computations
- High Data rates across interfaces
- Huge memory footprint

For high latency we analyze,

- Effect of sequential execution on stage and sub-stage latencies.

#### 3.1 Measurement Methodology

Jetson has INA3221 monitors and I2C capabilities to read voltage, current and power for different rails on the SOC and IO. 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.

### 3.2 Energy Characterization

We first highlight the overall system energy profile. 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. We then explain about the key findings discussed in the introduction of this chapter.

#### Individual stage energy

Subsystem	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(Cap. + Stitch*)	16.4 + 120*	19144	10.5 + 36,736
DRAM (Cap. + Stitch*)	260.4 + 105*	4792	41.6 + 1680

Table 3.1: Energy Characterization of Individual Stages

Note\*: As seen in the above table 3.1, 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. Therefore, we approximate the energy and latency for FPGA based accelerator based on Xilinx's implementation of optical flow on Zynq board, discussed in chapter 5.

## **Redundant Capture**

Natural images are redundant and but sensors are not smart to capture only the variations of the scene. Instead sensors capture and convert all the pixels for all the temporal frames. This leads to redundant conversion cycles by ADC's inside the sensor. It is therefore important to reuse the previous data to reduce the conversion cycles of the ADC to reduce the sensor power.

## **High Data rates across interfaces**

As shown in the 3.2, the interfaces consume about one third of total power during capture. The high data rates lead to higher power consumption. But having compression near sensor and decompression near ISP will increase to high latency and may increase the sub-system power. A neat technique is to use the co-design of ISP with the interface encoding schemes. The ISP has Bayer statistics for different regions of the image. As the temporal frames are redundant, the Bayer statistics can be used for encoding CSI interface data efficiently.

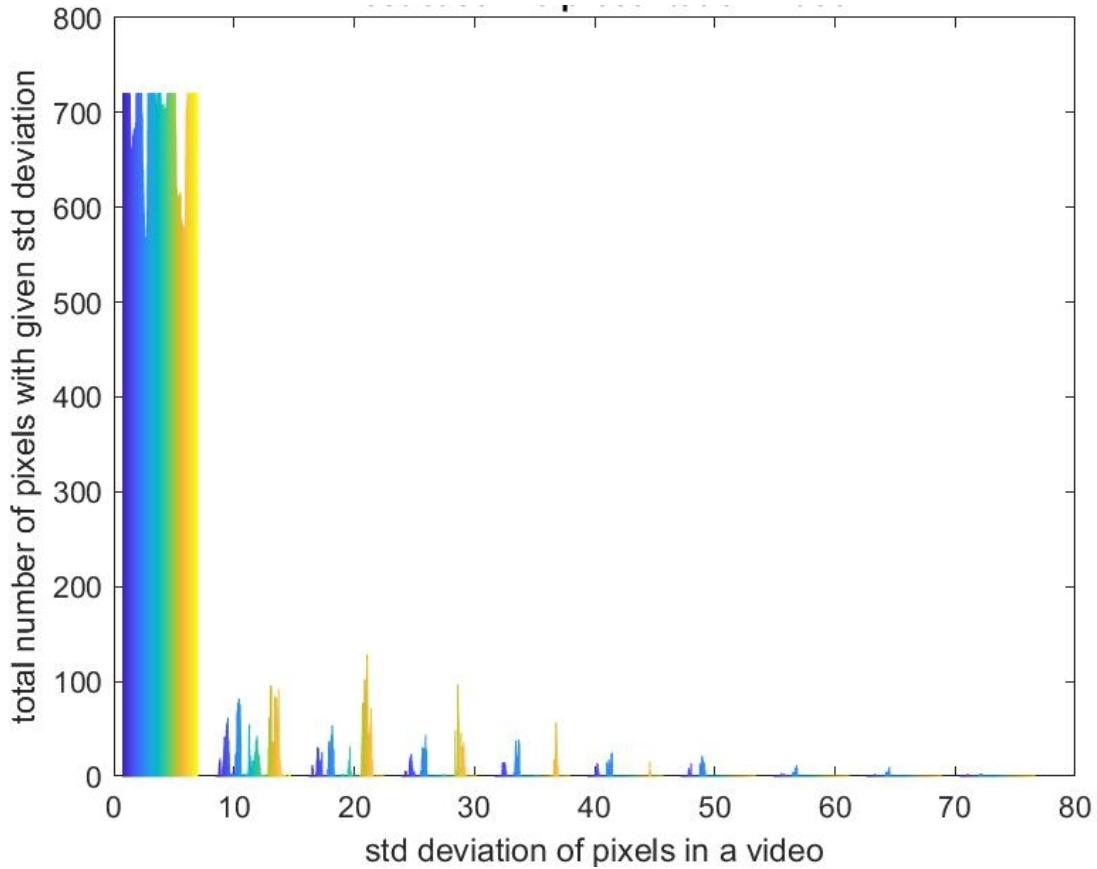


Figure 3.1: X axis shows the variations of pixels across 1000 consecutive frames, Y axis shows total number of pixels with a particular variation

### Huge Memory Footprint

The existing offloading based solutions have high memory footprint. One of the main reason being the inability to reuse the data generated by different subsystems. For example, during optical flow we load the current and previous image frames just to compute motion vector data. But this data is already available in ISP stage. Having data abstractions to reuse such data can reduce the memory requirement by up to 50 %.

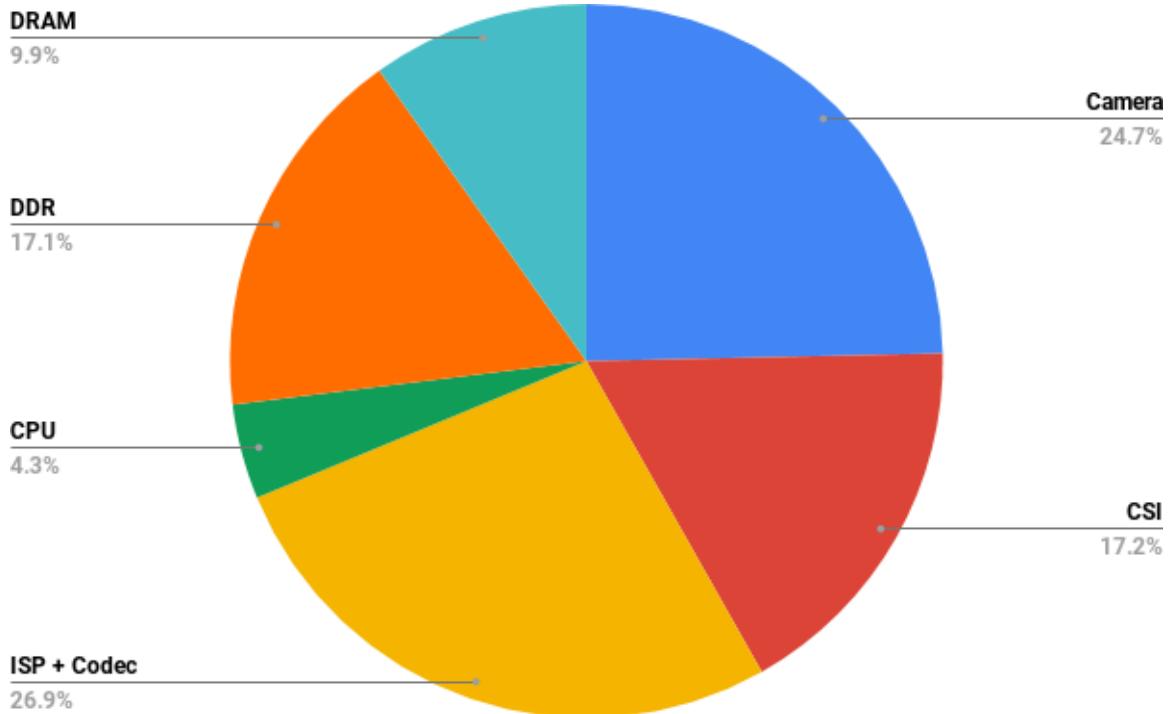


Figure 3.2: Chart showing the distribution of power of different components during capture and storage.

### Redundant Computations

Similar to the sensor capture, the stitching algorithms perform lot of redundant memory access and computations. Even though only few pixels change between frames, the existing algorithms compute over the entire image for every frame. Building custom hardware with fine power gating can be used to reduce the activity of the computation units where there is no motion across frames.

### 3.3 Latency Characterization

#### Individual Stage latency

For camera system and ISP stages the stage latency is derived from throughput, 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.

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. The CPU implementation takes 24 seconds for generating output at 6k resolution, of which optical flow generation consumes nearly 70 % of runtime. We therefore focus on reducing the optical flow stage which dominates not just in terms of latency but also in power consumption.

#### Optical flow runtime breakdown

The individual stages and the sub-stages of computation are sequential, leading to higher runtimes. By pipelining the different pyramid computation stages of optical flow, we can reduce the latency to 0.1 seconds from 16 seconds as shown in 3.3.

### 3.4 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,

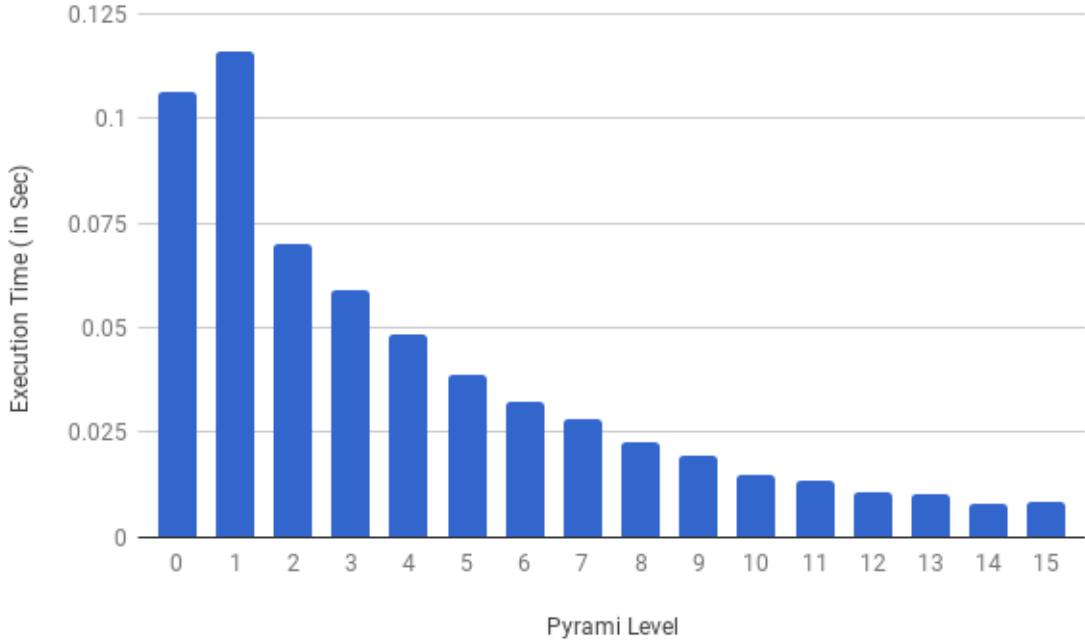


Figure 3.3: X-axis shows the pyramid level and Y-axis shows the runtime for OF

DRAM controller with increasing resolution and observe [linear] dependency of clock 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 design to decrease the chip resources, and use 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).

## CPU time of compute stages for different output resolution

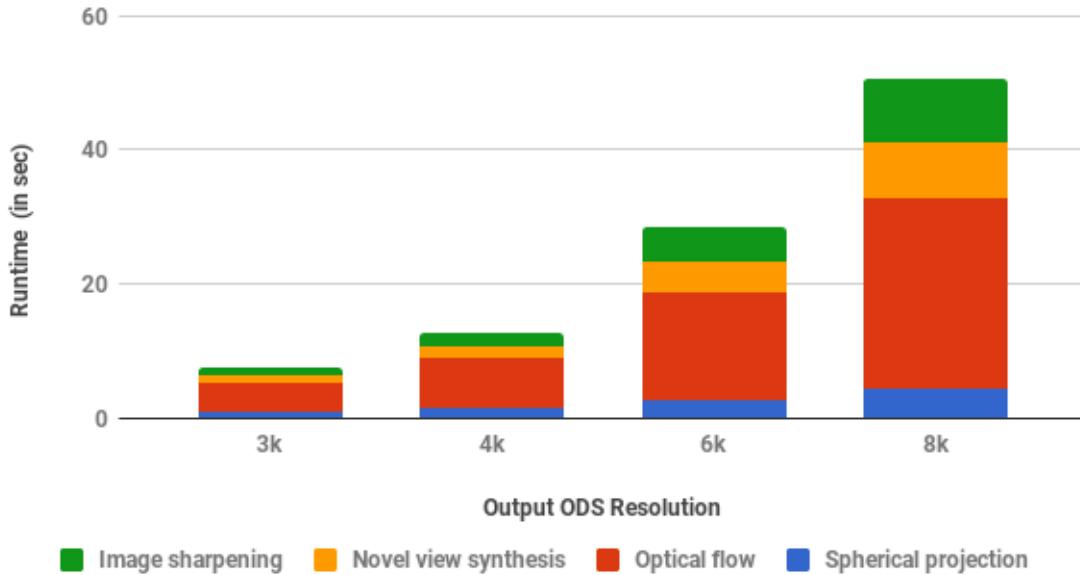


Figure 3.4: CPU execution time of different compute stages. X axis has different sub-stages in optical flow and Y axis correspond to energy per frame.

### Resource scalability with the resolution

We measure the DRAM capacity 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.

### Energy Efficiency of Camera and ISP Stages in Different Camera Configurations

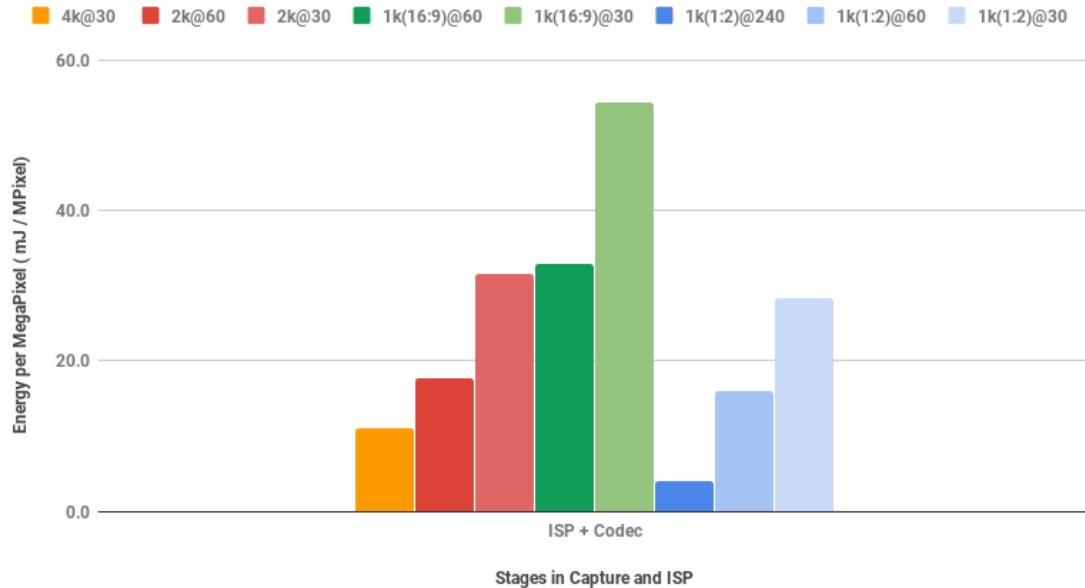


Figure 3.5: Energy Efficiency of Camera ISP Stages in different configurations

### 3.5 Different sensor configurations for energy efficiency and quality

#### Saving sensor power

As the ODS consist of several cameras, it makes sense to reconfigure or turn cameras on 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.

## Improving quality of images in low lighting

The optical flow works well when the image has high dynamic range. It is possible that some of the regions in the 360 degree view can be in low lightning, while others 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 do not 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 3.6 and 3.7 show the differences in low light capture with and without brightness correction.

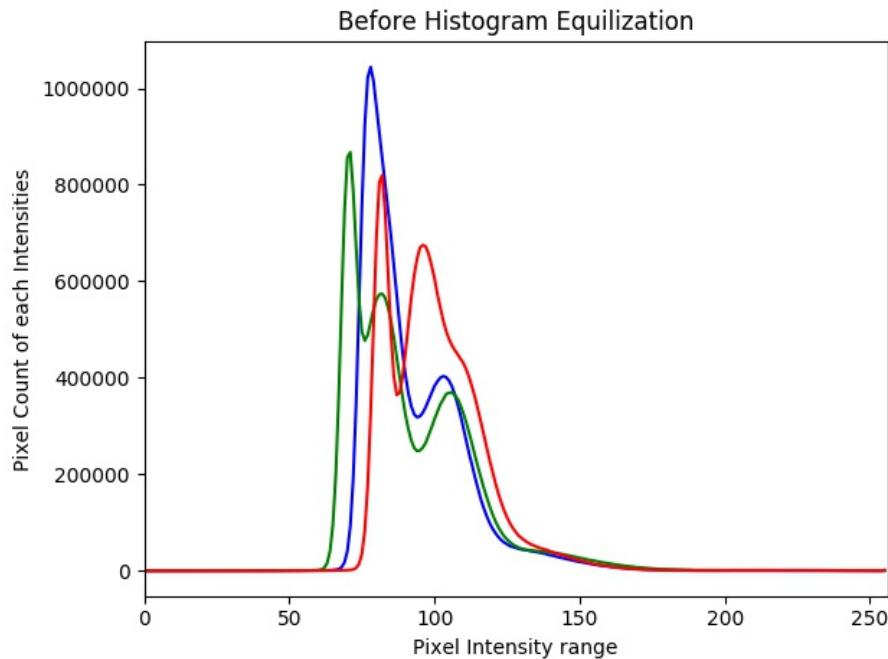


Figure 3.6: Image intensity histogram for low lightning conditions

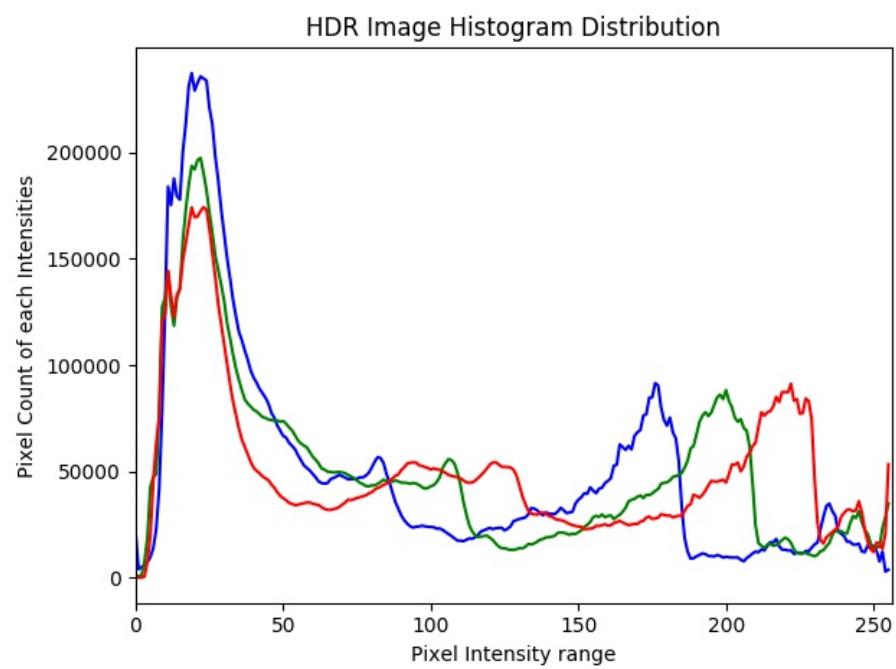


Figure 3.7: Image intensity histogram for good lightning conditions

## Chapter 4

### PROPOSED OPTIMIZATIONS

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.

#### 4.0.1 Techniques for Low Power

##### Reducing the ADC Power

In a multi-camera system, the combined power of sensors becomes a critical bottleneck. 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 Hu *et al.* (2018). 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 of the data. As the recent ISPs have a Bayer statistics accelerator embedded inside them, 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.

### **Improving the data abstractions across sub-systems**

The existing system pipeline consist of multiple subsystems that are not optimized for data sharing to optimize the system globally. For example, the motion vectors are computed during ISP stage, but as these results are abstracted away from other blocks like compute stage, it increases the overall memory footprint and extra computation cycle in compute stage. Similarly, the image statistics generated by ISP stage can be used for optimizing the sensor and sensor interface power. Sensor power can be optimized by providing the region of pixel values for the ADC, and entropy information of frame statistics at ISP can be used to encode the interface power without extra cost for generating the entropy information at the sensor end.

#### 4.0.2 Techniques for Low Latency

The camera and ISP stages are optimized in hardware generating realtime inputs needed for ODS stitching. So the main contributor for latency is the computation of dense stereo correspondence and the view synthesis stages. We estimate the latency reduction of applying the techniques like pipelining and parallelization in the following sub-sections.

#### Pipelined and Parallelized Execution

The main source for high latency is the sequential execution of different stages and sub-stages of the stitching pipeline. The stitching pipeline consist of stages like projection, optical flow and view synthesis which are sequentially executed. Pipelining the stages by would bring the latency to max of the latencies of the individual stages, which is 16 sec of the optical flow. The optical flow again consist of computing results for multiple pyramids, which maximum latency of higher level of pyramid comes around 0.1 sec. By pipelining we can bring down the latency from 16 seconds to 0.1 second. We estimate these numbers from SDSoc (2017) implementation of dense stereo optical flow reference design. In the next section we discuss how to go from 0.1 sec to real time in the range of 30 fps.

All the tasks in existing software pipeline can be accelerated using multiple cores. The dense optical flow for 4k resolution on FPGA takes only 17 milliseconds compared to 16 seconds by CPU implementation. We use Zynq (2017) to approximate the compute power for ODS generation at 6k resolution and 30 fps, which is 5.4 Watt. An ASIC implementation of the same design should further bring down the power to the order of 1 Watt. Therefore, the total system power including capture will be in the range of 3-5 Watts for 6k output ODS resolution.

## Chapter 5

### CONCLUSION

Real world content generation for VR is an emerging research problem. VR content needs to capture 360 degree views and 3D immersive experiences. The existing 360 camera devices have portable camera rigs that capture and offload the expensive computation 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. Through the rigorous process of building prototype and evaluating the power and latency of the system, we found that by reusing and reducing the data across different stages of pipeline, we can optimize the system for power. By pipelining and parallelizing the compute in hardware, we can reduce the latency. With proposed optimizations on a ASIC implementation, total system power will be in the order of 3-5 Watts and end-to-end system latency in the order of 100ms, making low power realtime stitching possible.

## 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).
- facebook, “Facebook surround 360”, URL <https://github.com/facebook/Surround360> (2017).
- Ho, T. and M. Budagavi, “Dual-fisheye lens stitching for 360-degree imaging”, in “Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on”, pp. 2172–2176 (IEEE, 2017).
- Hu, J., J. Yang, V. Delhivala and R. LiKamWa, “Characterizing the reconfiguration latency of image sensor resolution on android devices”, in “Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications”, pp. 81–86 (ACM, 2018).
- Ishiguro, H., M. Yamamoto and S. Tsuji, “Omni-directional stereo for making global map”, in “Computer Vision, 1990. Proceedings, Third International Conference on”, pp. 540–547 (IEEE, 1990).
- Kang, S. B., R. Szeliski and P. Anandan, “The geometry-image representation tradeoff for rendering”, in “ICIP”, pp. 13–16 (2000).
- Peleg, S., M. Ben-Ezra and Y. Pritch, “Omnistereo: Panoramic stereo imaging”, IEEE Transactions on Pattern Analysis and Machine Intelligence **23**, 3, 279–290 (2001).
- Richardt, C., Y. Pritch, H. Zimmer and A. Sorkine-Hornung, “Megastereo: Constructing high-resolution stereo panoramas”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 1256–1263 (2013).
- 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).
- SDSoC, X., “Dense pyramidal lk optical flow”, URL [https://www.xilinx.com/html\\_docs/xilinx2018\\_1/sdsoc\\_doc/oqm1517269835518.html](https://www.xilinx.com/html_docs/xilinx2018_1/sdsoc_doc/oqm1517269835518.html) (2017).
- Zynq, X., “4k60fps dense optical flow on zynq”, URL <https://www.slideshare.net/embeddedvision> (2017).