Demystifying the Energy and Performance Bottlenecks in Omnidirectional Camera Systems

by

Sridhar Gunnam

A Dissertation 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
Your first member
Your second member
\memberThree
\memberFour

ARIZONA STATE UNIVERSITY

July 2018

ABSTRACT

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Usecases

See the commit change 2 Overview of paper

use cases - mobile 360 capture, AR, VR, MR, Autonomous Driving.

Usecase1: 360 stereo capture for experiencing in VR headsets. [xx1] Sports live stream (only few powerful ones are sufficient) For general purpose capture and streaming, we need portable and long capture time capable. Existing VR cameras are available in small and large form factors but they have limited live streaming capabilities and don't have good battery life(60 minutes max).

Usecase2: In Mixed reality headsets(military training, gaming, etc) we capture and overlay virtual objects and display. So latency critical stitching.

Usecase3: Hybrid, 360 monoscopic and stereoscopic capture and display.

Status quo: Existing 360 camera solutions.

Characterizing monoscopic, stereoscopic, bottlenecks, optimization

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. 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

Chapter 2

BACKGROUND

Existing systems

Google Jump, Facebook Surround, Mega Stereo, Samsung Gear 360

Differentiating our work

Bottlenecks in the existing systems

Data Flow

Block Diagram - With different stages - With Data Inputs and Data Outputs of each stage. (Zoom in Diagram)

Chapter 3

CHARACTERIZATION

Our work focuses on characterizing the energy and latency of end-to-end Omni-directional-stereo(ODS) Camera system. The goal is to find the bottlenecks of different components in the hardware and software pipeline and propose optimizations. As the existing ODS camera systems are built from off the shelf camera devices and the conventional stitching algorithms we see a potential research opportunity to close gaps between hardware and software. Many existing systems like Google Jump, Facebook Surround capture and compute on enormous amount of data consuming several hundreds of watts of total energy to stitch ODS in realtime 30fps. The main challenge in ODS 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.

Many have argued [Edvardo Hotmobile, Nvidia, AMD] that we need resolutions greater than 16k and frame-rate greater of 240 for true immersion in VR. At such higher framerate and resolutions there is lot of information that is redundantly captured, processed and transferred. Therefore, in our work, we study how the energy and latency of each stage get affected by the output resolution and the characterize the bottlenecks in greater detail.

We characterize both monoscopic and ODS camera systems. The difference between them is the number of novel views needed is significantly higher for ODS. Monoscopic is a special case of ODS and at core involves the same optical flow based stitching. So we will characterize generalized flow based stitching system and notify any important differences between monoscopic and ODS when necessary.

3.1 Energy Measurement Methodology

The end to end camera system, as shown in figure 3.1 can be divided into 4 major stages by energy consumption, viz., image sensor, image signal processor(ISP), processing, and Off-chip memory. For our prototype design ODS design we use six imx-274 cameras for capture and Nvidia Jetson TX2 for processing the frames. The Camera and Jetson specs are shown in figure 3.2. Jetson has power moniter IC and ways to moniter CPU, GPU, memory frequencies. We compute the energy of each individual stage by measuring the difference between the idle and active stage(when capturing/computing). (Prototype System Overview
Hardware: System 1) Dual Fisheye Camera for monoscopic 360 System 2) Six Camera Rig for OmniDrirectional Stereo) Jetson power monitering , IO Power Micron System Power Calculator for LPDDR2.

Software: Camera API, openCV, C++

1) Energy/Power
a) Individual Stage Power Characterization. X axis has different stages and Y axis correspond to energy per frame

Stage	Substage	Data Type/Domain/Format	Typical Size
	CIS	Analog Voltage/Current	Pixel
Camera	ADC	Quantized Bits	Pixel Depth
	IO	Bayer/ Raster output	Row/Coloumn of Pixels
	Raw Processing	Bayer	Set of lines
ISP	Demosaic	Bayer to RGB	Set of lines
	Color Correction	RGB	Set of lines
	Codec	RGB	Frame / Set of Frames
	Distortion Corr.	RGB	Full Frame
	Projection	RGB	Full Frame
Computation	Optical Flow	Gray	Two Adjacent Frames, Pyramids
	View Synthesis	RGB	OF + Frames
	Sharpening	RGB	Frame

b) Breakdown of Individual Stage 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. 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.

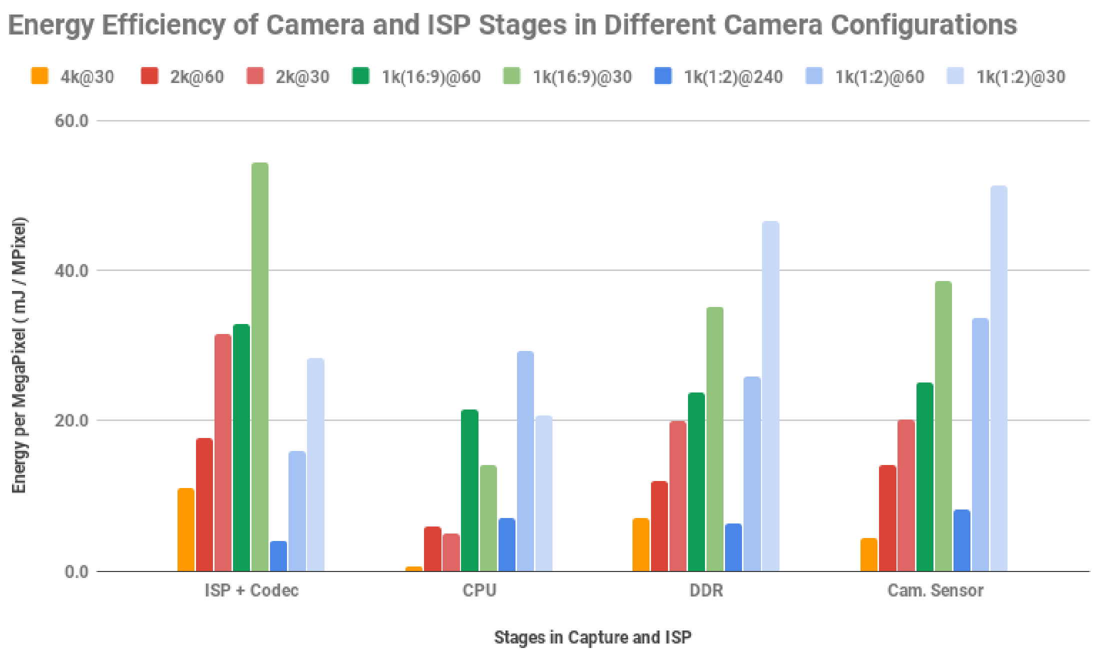

Figure 3.1: Power Efficiency of Camera ISP Stages in different configurations

Power Rail	Diff. Current(mA)	Voltage(mV)	Power(mW)
$ISP + CODEC$	102.7	19152	1966.9
CPU	16.4	19144	314.0
DDR	260.4	4792	1247.8
CAM_SNSR	375.4	3336	1252.3
Total	-	-	4781.0

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.

c) Increased frame-rate

What is the percentage of new data ffmpeg I-frame size to the P-frame ratio. Typically

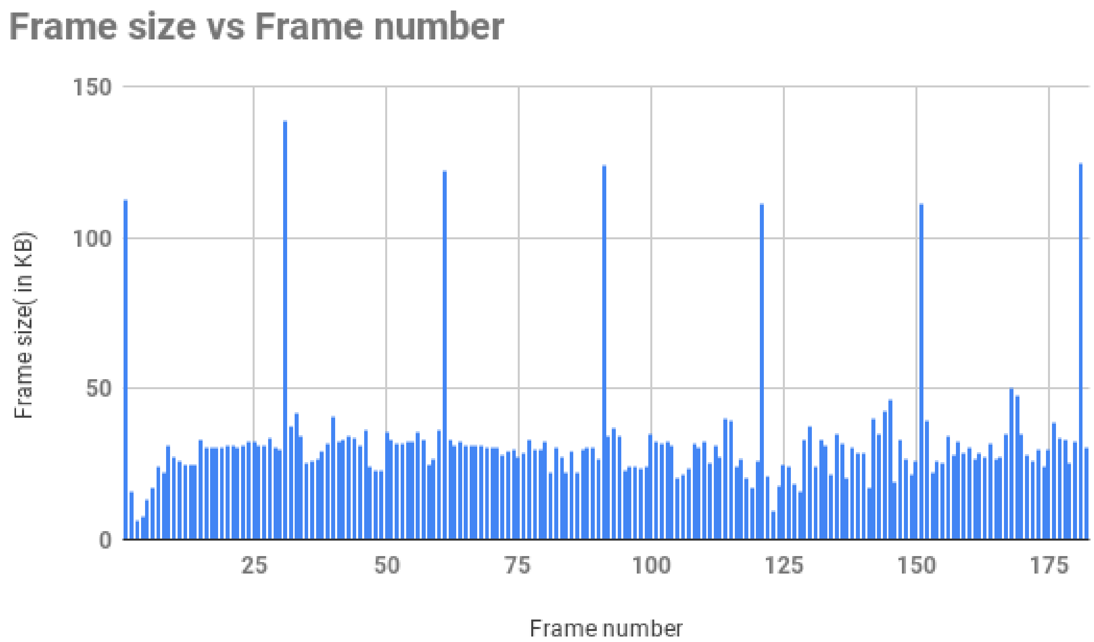

Figure 3.2: Framesize of I and P frames

I frame to P frame is about 4 times and we have one I for about 30 P frames. This implies we save a lot on interface power if we can push the computation to near sensor.

d) Scaling with the resolution

Outputs: 4k, 6k, 8k, 12k Discuss scalability of resolution for different stages. Especially the scalability of cache, DRAM, CPU power.

e) Quality Tradeoff's with input resolution

Sharpening Reducing number of pyramid's f) High motion Vs low motion differences 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

2) Performance

a) Individual Stage Latency Characterization b) Breakdown of Optical Flow Individual Stage Time for each Pyramid Generation c) Breakdown of sharpening

Chapter 4

PROPOSED MECHANISMS FOR ENERGY AND PERFORMANCE SAVINGS

Technique ⇒ Benefits

1) Hardware software co-design: Re-using motion vectors generated by ISP stage to reduce the re-reference of previous image frames to calculate motion vectors for optical flow generation. (How to check if we need to compute everything with intensive vision algorithms or just use the precious results, what granularity to compute. Eg. Pyramids updating. Reactive to reconfiguration and powering on and off.) a) Reduction of DRAM capacity requirement b) Reduction of DRAM bandwidth requirement c) Reducing end-to-end latency in generating dense optical flow
2) Data driven execution: Use of motion vectors and previous optical flow to update the pyramids to make use intrinsic properties of foreground, background and motion in the scene. a) Reduces the number of computations i) For building pyramids. ii) For calculating SAD(Sum of Absolute differences) during pyramid block matching. b) Reduces off-chip accesses c) Reduce end-to-end latency

3) Streaming Architecture: Using raster buffers across the entire optical flow pipeline.(Number of rows is constrained by maximum motion in the scene) a) Helpful for scalability to higher resolution b) Reduces size of local SRAM and off-chip memory access.

Case for low power 360 capture. Real time Stitching 30fps ¿4k resolution with low power. Latency of GPU, CPU makes them unusable for vision tasks in AR, VR.

Case for algorithm software Co-Design for a line buffer based streaming architecture.

1) Energy Characterization of end-to-end pipeline

Camera, ISP, Computation

Split of energy in computation

2) Runtime Characterization

a) End-to-end pipeline

b) Split in computation execution

3) Performing motion estimation prior to computation stage

a) Savings in DRAM capacity, bandwidth(Normalized)

b) Savings in DRAM Bandwidth

c) Savings in overall energy

d) End-to-end latency reduction

4) Optimizing of computation in pyramids

a) The execution time split for creation of pyramid, finding optical flow of pyramid, refining/updating the pyramids, upscaling the pyramid.

98 percent is to generate optical flow(dense pixel correspondence). But only 20-30 percent actually needs to be recomputed.

Main optical flow method time is 0.560256 Total time for entire optical flow is 0.584954

5) Sense the environment in gray scale and perform color mapping later? How much are you saving?

Chapter 5

DISCUSSION

Summarizing the proposed optimizations and future directions. 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

Chapter 6

CONCLUSION AND FUTURE WORK

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

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

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

REFERENCES

APPENDIX A

RAW DATA