Seam Carving Through Time

Maxwell Johnson, John Zhang, Riki Singh Khorana

Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—**The project aims to shorten a video while preserving the salient features by applying 3D seam-carving to the video. The approach functions by removing a series of continuous sections of pixels to shorten the time duration of the video.

The project will implement the algorithm in software for research purposes. The team will additionally develop a system to accelerate the computation on an FPGA board. The accelerating system comprises three main system components: a software application on an external computer, a system on chip (SoC) running the Linux kernel, and the FPGA's programmable fabric to perform the acceleration.

***Index Terms*—**FPGA, Seam carving, System on Chip

I. Introduction

THE most pervasive technique for shortening the length of a video is to increase the playback rate. This method disregards the content of the video. Our project is to develop a content-aware video-shortening system with the goal of producing shorter videos that look more natural than those shortened using standard techniques. The system must be able to shorten a video to two-thirds of its original length without noticeably speeding up the salient features of the video.

In the context of our project, a seam is defined as a connected set of pixels spanning the width and height of the video. It is a surface, which may bend in the time dimension. Fig. 1 shows an example seam cutting a video, represented with time extending in the *t* dimension.

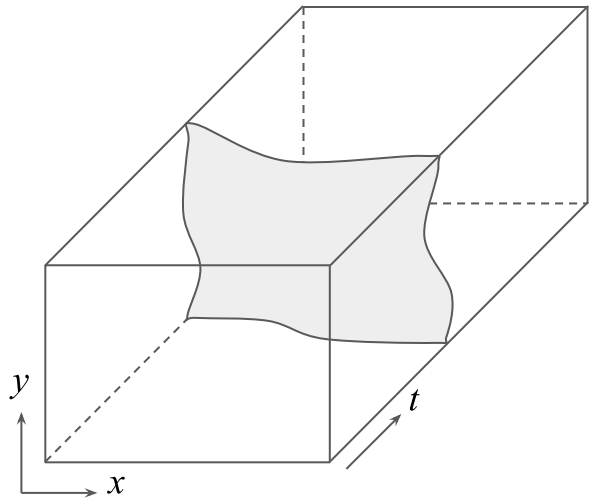

Fig. 1. An example of a seam in the three-dimensional representation of a video.

In the resulting video, different areas of a frame may contain pixels that originate from different frames. Fig. 2 shows an example of what a frame that intersects the seam may look like in the resulting video.

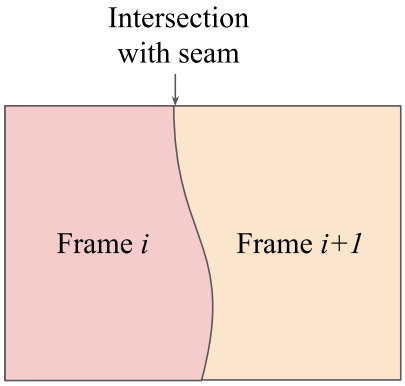

Fig. 2. An example of a frame intersecting a removed seam.

The seam carving algorithm attempts to remove the seam with the lowest cumulative energy. Energy is defined in terms of the pixel difference δ between pixels *a* and *b*, defined below.

$$\delta(a, b) = (r_a - r_b)^2 + (g_a - g_b)^2 + (b_a - b_b)^2 \quad (1)$$

Where r_p, g_p, and b_p denote the intensity of the red, green, and blue components of pixel *p* respectively. The energy function defines an energy of a pixel *p* at coordinates (*x, y, t*) denoted as $p_{x,y,t}$ is shown below.

$$E(p_{x,y,t}) = \delta(p_{x,y,t}, p_{x+1,y,t}) + \delta(p_{x,y,t}, p_{x,y+1,t}) + \delta(p_{x,y,t}, p_{x,y,t+1}) \quad (2)$$

The seam with the lowest energy corresponds with the moment of the least action. By removing the seam of the lowest energy, the seam carving through time algorithm produces a video with the slowest parts of the video cut out.

Our project comprises of two foci: research into the application of seam carving to video in the time dimension and acceleration of the seam carving algorithm. Our research will determine the families of videos on which seam carving is most effective, as well as heuristic improvements to the algorithm that improve the output quality and computation speed. To accelerate the computation, we will use a Xilinx FPGA with an embedded system on chip. The system must be able to process the video in the same time the original video takes to play. While the specific uses of this algorithm are subject to our research, condensing video to emphasize its salient features has applications in monitoring security feeds, watching tutorial videos, and viewing sports footage.

II. Design Requirements

A. High-level Requirements

Below are requirements that are set on the overall system. They are divided into two categories: Spec and content. The former specifies the technical constraints on the spec of the video and the performance of the system itself. The latter aims to quantify the quality of the processed video in terms of our project goal.

Requirement 0.0.0: The system must process a video with time length T in 3T time.

We will measure this requirement by timing how long the system takes to process an arbitrary video. The system is timed from the beginning of the pre-processing step to the end of the post-processing step.

Holistically, we specify 3T as our benchmark because that is generally the time it takes to manually inspect a video of time T and crop out it's salient features. Technically in production, we would be able to use multiple (>3) FPGA boards to run the algorithm in parallel, where the net effect would be processing the video in time T.

Requirement 0.0.1: The system must process a 360p and 24fps video.

We will measure this requirement by simply inputting a video of 360p with 24fps into the system and verify that the system outputs some form of end product.

We have derived the quality requirements of the input video based on the most widespread standards. In particular, we have studied YouTube videos and its distribution system as an example of a common standard. From this, we set 360p as a video quality requirement, because that is the minimum resolution for a regular YouTube video, and 24fps because that was a common standard for common video formats.

Requirement 0.0.2: The system must increase playback speed of a video by 1.5 times.

We will measure this requirement by inputting a video of time T and verifying that the resulting video has time 2T/3.

Like requirement 0.0.1, we have devised this requirement via observations made on the YouTube platform. A modified playback speed of 1.5 times was commonly amongst fellow students when watching a video of a familiar topic at a higher speed. We aim to provide a 1.5 times playback speed for videos of all content, by preserving the salient features of the video using the seam carving algorithm.

Requirement 0.0.3: The system must support *at least* three popular video file formats as input.

We will measure this requirement by inputting at least three videos of select file formats and verifying that the system produces a valid output. Currently, we are planning to support .MOV, .AVI, and .MP4 as the three popular video file formats.

Requirement 0.1.0: The resulting video must have smooth frame transitions.

We define videos with smooth transitions as ones where the average energy values of the frames computed by the energy function (2) does not exceed a certain threshold value. The threshold value will be set by us, after running the energy filter through at least 20 "normal" YouTube videos, and taking the average energy + 1 standard deviation.

Requirement 0.1.1: The resulting video must have no obvious distortions to its content.

We define videos with no distortions as ones that have its salient features properly preserved. For example, if a user cannot predict the original contents of the processed video after watching it, we determine that our video has somehow lost some of its salient features.

We will quantify this requirement by conducting a user survey and asking the users to watch the processed video, predict the original video, watch the original video, and rate how close they imagined the original video to be on a scale of $1 \sim 10$. We require that we obtain an average score of above 8.5 as a result of the user studies.

Requirement 0.1.2: The resulting video must have its original order of events preserved.

We aim to develop a specific test suite for this requirement, consisting of synthetic videos with well defined "events" as inputs. We will quantify the requirement by counting how many events were misplaced in the output video compared to the input video.

B. Pre/Post Processing Requirements

The pre/post processing applications are wrappers to the entire system, which determines how the input/output is presented both internally and externally. On top of fulfilling all video spec requirements from Requirement 0.0.x, the application has several requirements set to ensure efficient communication with the hardware.

Requirement 1.0: The application must convert videos to an FPGA readable format.

The video will be represented as a three-dimensional array of pixels. We denote the width (long dimension) of the video as the x dimension, the height (short dimension) of the video as the y dimension, and the time dimension of the video as the t dimension. Each pixel is represented as a three-byte vector representing the intensity of red, green, and blue intensities. The pre- and post-processing applications must be able to convert between popular video file formats and this decompressed format.

Requirement 1.1: The application must transfer videos to the FPGA through Ethernet.

The FPGA board has a built-in ethernet port, which will be used to communicate with the pre/post processing application running on an external computer. Please see section IV Design Trade Studies for more details.

C. SoC and Algorithm Requirements

The SoC runs on the FPGA and is used to run the seam carving algorithm while delegating some of its subroutines to the programmable fabric.

Requirement 2.0: SoC runs Linux Kernel as embedded operating system.

We choose to run a Linux kernel as the embedded operating system on the ARM Cortex-A9 cores because it is well-documented, and all of our team members are familiar with it. The Linux kernel memory mapping layout has a specific area that's directly mapped with the board's DDR3 memory; this is in favor of our project as we intend to use the SoC for reading from the memory the energy map and write to the memory the video data after cutting out a seam.

Requirement 2.1: The SoC reads from and writes to the RAM.

One of the main tasks of the SoC is to read the energy map from the RAM that the programmable fabric has produced and write the resulting video data after calculating and cutting out a seam for the programmable fabric to recalculate the energy map. Therefore, it is crucial that the SoC can read from and write to the RAM directly.

We will evaluate requirement 2.1 by testing memory manipulation through the Linux kernel. The Linux kernel should be able to run scripts to read and write to the DDR3 memory on the board.

Requirement 2.2: The SoC extracts a seam made of pixels with the lowest energy from a given energy map

The SoC will take an energy map of a video calculated by the programmable fabric and find a seam to remove as described in the introduction.

We will evaluate this functionality by unit testing and comparing the calculated seams with that of a software implementation.

Requirement 2.3: The SoC reads the video data from the RAM and performs seam carving.

After calculating the seams from the energy map, the SoC cuts out seams from the video data accordingly by reading video data from the RAM and writing the resulting data back to the RAM.

We will evaluate this functionality by unit testing and comparing the calculated seams with that of a software implementation.

D. Programmable Fabric Requirements

Processing video is memory-intensive, while reading from memory is often the slowest operation. To anticipate this bottleneck, we impose the requirement on the programmable fabric that it be able to store the data of two frames in the block RAM embedded in the fabric. This allows us to compare each pixel in a frame against an adjacent pixel along each axis, as required by (2), without accessing main memory. The size of each frame is given by

320 vertical pix * 640 horizontal pix * 3 bytes/pix * 8 bits/byte = 4.7 Mb per frame

Requirement 3.0: The FPGA block RAM must have capacity of at least 9.4 Mb (1024^2 bits).

Requirement 3.0 defines a requirement on the choice of board and the layout of block rams in the accelerator hardware design. We will measure this requirement using the Vivado synthesis report.

We will be using the programmable fabric to accelerate the computation of the energy function of the video. This process must be completed for every frame before the system on chip can remove a seam. It is also used to recompute the energy of altered frames after the removal of each seam. We estimate that computing the energy function comprises roughly half of the computation in our algorithm. Using an FPGA for a very regular computation can potentially provide very speed-up; however, by Amdahl's law, there is diminishing return to accelerating only a portion of computation. To achieve a speedup of 1.98, 1% less than the optimal 2.0, the programmable fabric must process frames roughly 100 times faster than the non-accelerated system on chip portion. At our overall target speed of 24 frames per second, the programmable fabric must process the energy function at 100 * 24 fps = 2400 fps.

Requirement 3.1: The programmable fabric must process the energy function at a rate of 2400 frames per second.

We will measure requirement 3.1 first using simulation tools (VCS). On the board, we will use a hardware counter to confirm the number of cycles used per computation at a known clock rate.

Our final requirement on the programmable fabric is that it must be able to store the energy map back to RAM quickly. We must be able to write the processed frames back to memory as we generate them, so the required processing speed matches requirement 3.1.

Requirement 3.2: The programmable fabric must write the result of the energy function to RAM at a rate of 2400 frames per second.

A simulation model of the RAM and controller may be inaccurate, so we will only measure this requirement on-chip using hardware counters, in the same style as requirement 3.1.

III. Architecture and/or Principle of Operation

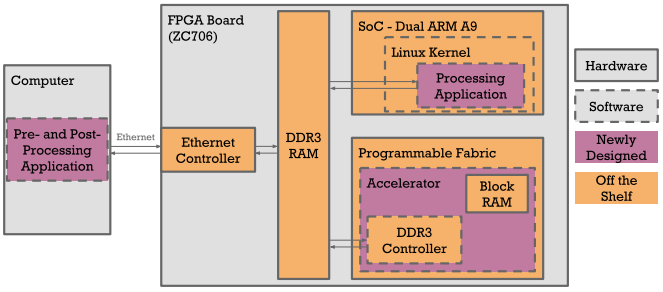

Fig. 3. High-level block diagram of system.

Fig. 3 shows the block diagram of our overall system. We have divided our system into three main hardware components: the external computer, the on-board system on chip, and the programmable fabric. We will address the purpose and dataflow of each component individually.

A. External Computer

The computer is the user interface to our system. This is where the user will select their video and the new playback rate. The pre-processing application will parse the video from a standard playable format into a three-dimensional array of pixels. Each pixel will be represented as three bytes, representing the intensity in that pixel of red, green, and blue, respectively. It is in this format that the video will be transferred to the FPGA board over ethernet connection. The resulting video is received over the same connection in the same decompressed format. The post-processing application formats the video into a standard playable format and saves the result to file.

B. SoC

The system on chip is a core embedded in our FPGA. We will use it to perform the more dynamic portions of seam carving: seam selection and seam removal. It is responsible for analyzing the energy map of a video output by the programmable fabric, determining a low-energy seam to remove, and performing the memory operations required to remove the seam. To determine a seam to remove, the system on chip will read the energy map from RAM. After selecting a seam, it will read the video data from RAM, modify it, and write it back.

C. Programmable Fabric

The programmable fabric will accelerate the computation of the energy function given in (2). The input to this component is the video in the uncompressed format stored in the RAM. It is accessed through the Xilinx DDR3 controller IP [6] instantiated in the programmable fabric. The programmable fabric computes the energy function of the video, storing temporary data in block RAM within the programmable fabric. The energy map of the video is written back to the RAM through the same DDR3 controller. When the system on chip removes a seam, the programmable fabric will recompute the energy function of the altered frames.

IV. Design Trade Studies

A. Algorithm

We have decided to utilize the OpenCV library as part of the pre/post processing applications. In order to achieve high speed while leveraging the rich functionality of OpenCV, we decided to use the C++ distribution instead of the familiar Python version. A naive implementation of video pixel processing showed a speed up from 14.9 seconds to 3.5 seconds using C++ instead of Python. We have made a tradeoff between faster implementation and learning cost of a new language.

A recurring suggestion for the seam carving algorithm was to use grayscale instead of color for edge detection, allowing us to implement a faster and less memory-intensive solution. However, we have decided to keep the pixel information in color, as we would like to aim for a high-quality solution. The added memory of the color will also likely not be the bottleneck.

B. Accelerator

As part of our design, we will use an FPGA board with an embedded system on chip. The system on chip will perform the more dynamic stages of the seam carving algorithm, while delegating tasks to the programmable fabric, which will accelerate the more regular computation. Using an embedded system on chip will reduce the communications overhead between the two processing-heavy components of our system.

The primary decision to be made in the accelerator subsystem is what FPGA board to use. The board provides all the hardware used by the accelerator, which we decompose into the following subsections:

- Programmable fabric
- System on chip
- Main memory
- Off-board communication

The programmable fabric is the portion of the board that will be directly used to accelerate computation. This is the source of our primary constraints on board choice. Design requirement 3.0 requires that the block RAM of the FPGA be able to hold the data of at least two frames to reduce memory accesses, which totals 9.4 Mb of block RAM. Table 1 shows the block RAM capacities of the available boards. Only the Xilinx ZC706, which uses a Zynq 7045 FPGA [1], has the block RAM capacity to hold the required two frames.

TABLE I. Projected Resource utilization

Board	Block RAM capacity
Xilinx ZC706	19.2 Mb [2]
DE10-Standard	5.6 Mb [3]
ZedBoard	4.9 Mb [2]
Pynq-Z1	4.9 Mb [4]

All four boards use the same system on chip, a dual-core ARM Cortex-A9 [2][3][4]. They are clocked at comparable speeds, though again the Xilinx ZC706 is the most performant at a 1 GHz clock rate.

The main memory dictates the length of video that we can process at one time on the FPGA. The pre-processing application can break the video into shorter chunks to be processed independently, so we will only require that the main memory hold roughly 10 seconds of video. 4.7 Mb/frame * 24 fps * 10 sec = 141 MB. The ZC706 has 1 GB of DDR3 RAM [1], which provides ample space for the video and leaves space for the system on chip's operating system and seam carving application.

Direct memory access (DMA) provides a way for off-board communication to write directly to memory, which provides the highest-throughput communication. On the ZC706, there are three connections that provide DMA: USB 2.0, Ethernet, and SDIO. SDIO is not designed for inter-processor communication. The maximum speed of USB 2.0 is specified as 57 MB/s [5]. The maximum speed that our ethernet controller supports is

1000 Mbps = 125 MB/s [2]. Both devices can transfer our video faster than it can be played and are likely sufficient. Xilinx provides IP cores allowing the programmable fabric to interface with either controller. We decided to use ethernet because of its ease-of-use on the software side as well as the higher data rate.

V. System Description

A. Pre/Post Processing

The role of this unit is to disassemble and reassemble original and processed videos into user-readable formats, and to apply additional filters to the video as needed. It also serves as the master to the hardware application, and therefore requires ethernet network programming to optimize the communication.

The pre/post processing application runs on the external computer and is implemented using C++. Most of the dissection of the video file into FPGA readable bitstreams and the assembly of it into a video output uses the OpenCV library. Additional processing on the video itself also uses OpenCV, but primitive pixel-level manipulations may be directly implemented without using the library. The network interface to the FPGA board also lives in this application.

B. SoC

The SoC takes up the tasks of calculating lowest energy seams from the energy map calculated by the programmable fabric. It will also read video data from the DDR3 memory and use the cut out the calculated seams of pixels from the video.

The SoC will run a Linux kernel as an embedded OS, enabling direct mapping to the DDR3 memory and therefore facilitated memory reading and writing.

C. Programmable Fabric

To compute the energy function defined in (2), our design must square nine differences. The Zynq 7045 has 900 discrete signal processing (DSP) units [2], each of which contains a multiplier with a pre-adder that can be used to compute the square of a difference. The required horizontal resolution is 640 (see Requirement 0.0.1), so our design seeks to parallelize the energy function computation across a row of pixels in a frame. The design consists of 640 processing units, one of which is pictured in Fig. 4.

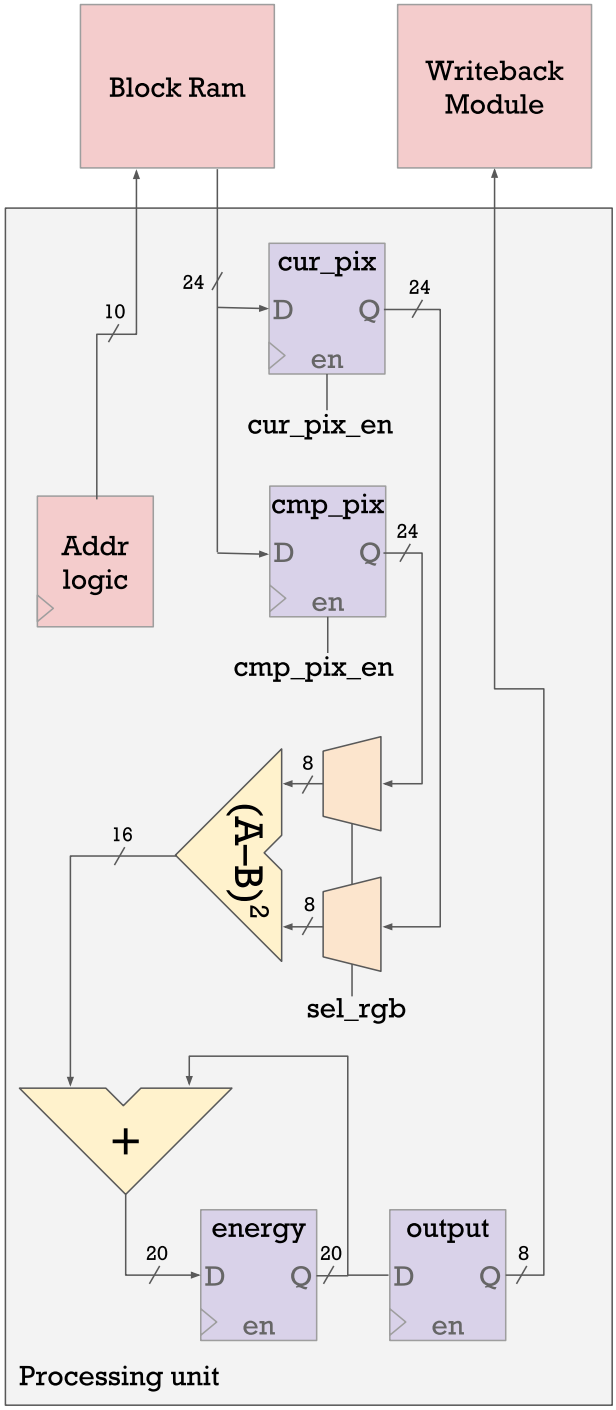

Fig. 4. Hardware design diagram

Each unit receives pixel values from the shared block RAM. In the first clock cycle of the process, cur_pix_en is asserted to load the value of the reference pixel. In subsequent cycles, cmp_pix is loaded with adjacent pixels and δ(cur_pix, cmp_pix) is computed (see (2) for function definition). The final energy is held in the output register so the writeback module can write the resulting data to RAM. A final component of the system is the module to load the block RAM with video data from main RAM, which will use the Xilinx DDR3 RAM IP block [6].

Computing each δ takes three multiplications, each of which require one cycle. Three such computations are required, for a total of nine cycles of computation. One cycle is required to load cur_pix, which cannot be overlapped with the first load of cmp_pix, so we can expect the computation of each energy to use ten cycles. At a 50 MHz clock, we will be able to process 50 MHz / (10 cycles/row * 320 rows/frame) = 15.6 kfps, which satisfies requirement 3.1.

We have designed and synthesized one processing unit, from which we can extrapolate the overall usage of board resources. These numbers are reported in table 1.

TABLE II. PROJECTED RESOURCE UTILIZATION

Resource	640 Units	FPGA capacity [2]	Utilization
Block RAM (Kb)	9600	19620	48.9%
Look Up Tables	41600	218000	19.1%
Flip-Flops	51200	437000	11.7%
DSP slices	640	900	71.1%

Our projected utilization is closest to 100% in DSP slices, as designed. We could introduce more processing units at the cost of more complexity in the division of labor; however, it is advantageous to leave some DSP slices unused to ease the place-and-route process and to keep the clock frequency high.

VI. PROJECT MANAGEMENT

A. Team Member Responsibilities

The design and implementation work are divided as follows. Maxwell is in charge of the FPGA fabric design and implementation, aiming to accelerate parts of the seam carving algorithm. John is in charge of the SoC on the FPGA, aiming to establish a seamless communication channel between the FPGA and the external computer. Riki is in charge of the pre-processing and post-processing applications on the external computer, aiming to process the video in a way that would optimize the seam carving results. All members are involved in design reviews, and plan to collaborate during the integration of the various components to make up the whole system.

B. Budget

We have no external orders planned at this time, as all of our work can be implemented locally on our laptops and the Xilinx FPGA provided to us by the school.

C. Risk Management

The team has identified several risk factors that could potentially hinder the progress or outcome of the project.

I. Uncertainty of outcome

Given the research nature of this project, the outcome of seam carving through time is unknown. We know that it can be applied to video as in [9], but this project did not apply seam carving to the time dimension. To mitigate this important risk, we've planned out the schedule such that the first task is to implement a prototype of seam carving through time. This will allow our team the time to perform the necessary refinement and evaluation to apply the seam carving algorithm in the time dimension.

II. Unfamiliarity with SoC

None of the team members have previously worked with the system on chip, a key component in running the dynamic portions of the seam carving algorithm. To minimize the impact of this, we have prioritized the task of interfacing to the SoC and we have assigned two of our team members to contribute to the development of the SoC application and interface.

III. Unknown processing time to find a seam

We will be accelerating the computation of the energy function, but we will be running the seam-finding portion of the algorithm on the embedded core. Without our prototype seam carving application, our estimates of processing time are still very approximate. Again, we are mitigating this risk by prioritizing the development of the software prototype. In the worst case, we can move the seam-finding off of the SoC and to the external computer. This comes at the cost of increased communication to the external computer, but if the embedded SoC does not have enough processing power, this may be a worthwhile trade.

D. Schedule

In Fig. 5. (following page), Maxwell's tasks are in red, John's are in yellow, Riki's are in blue, and joint work uses the secondary colors made by combining the relevant individuals' colors.

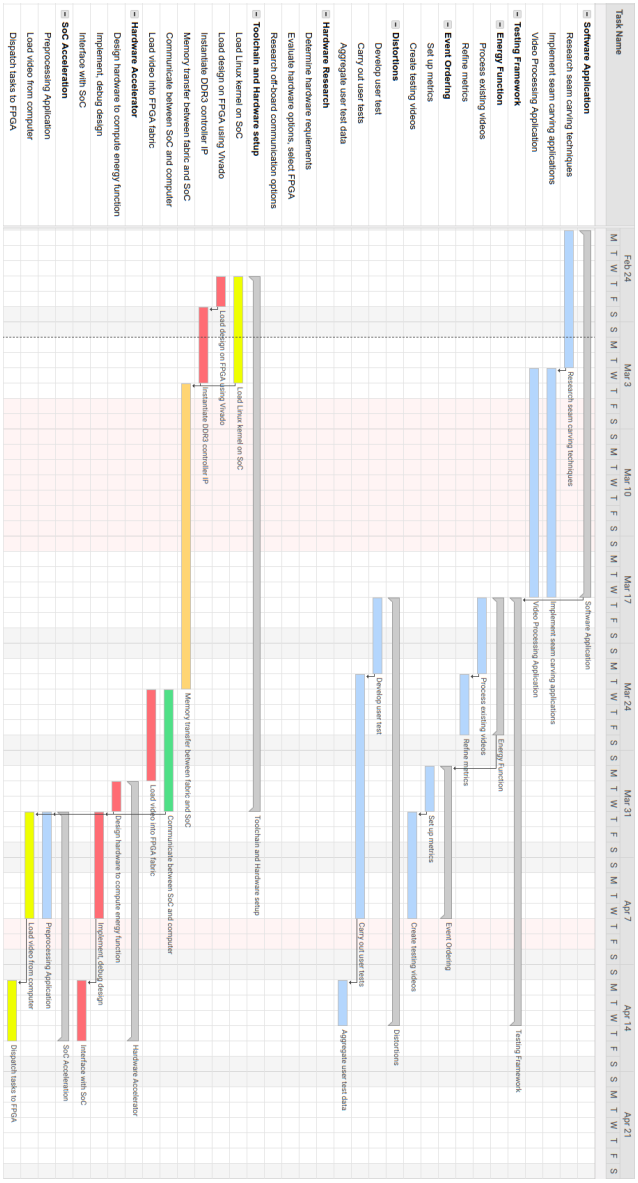

Fig. 5. Team Gantt Chart

VII. REFERENCES

[1] Xilinx ZC706 Evaluation Kit. https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html#hardware

[2] Xilinx Zynq-7000 SoC Family Guide. https://www.xilinx.com/support/documentation/selection-guides/zynq-7000-product-selection-guide.pdf

[3] Terasic DE10-Standard Board. https://www.terasic.com.tw/cgi-bin/page/archive.pl?CategoryNo=205&No=1081&PartNo=2

[4] Digilent Inc, Xilinx PYNQ-Z1 Board. https://store.digilentinc.com/pynq-z1-python-productivity-for-zynq-7000-arm-fpga-soc/

[5] Universal Serial Bus Specification, Table 5-10. http://sdphca.ucsd.edu/lab_equip_manuals/usb_20.pdf

[6] Xilinx DDR3 Controller. https://www.xilinx.com/products/intellectual-property/ddr3.html

[7] Kernel Memory Layout on ARM Linux. https://www.arm.linux.org.uk/developer/memory.txt

[8] ZC706 Evaluation Board for the Zynq-7000 XC7Z045 SoC User Guide. https://www.xilinx.com/support/documentation/boards_and_kits/zc706/ug954-zc706-eval-board-xc7z045-ap-soc.pdf

[9] Improved Seam Carving for Video Retargeting, Rubinstein, Shamir, Avidan. http://www.faculty.idc.ac.il/arik/SCWeb/vidret/index.html