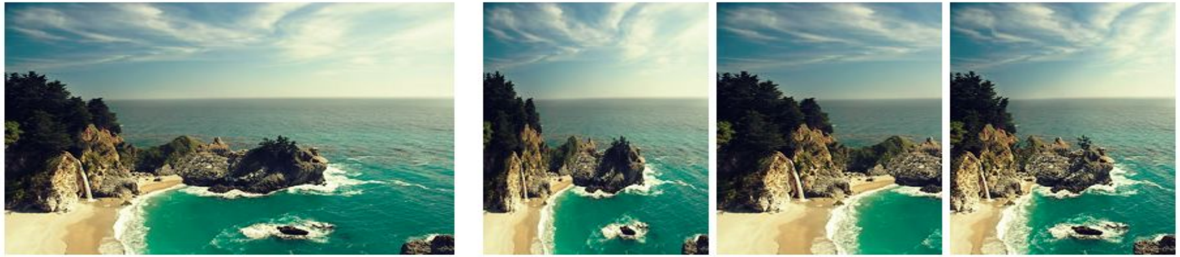
FPGA Accelerated Seam Carving for Video

B2: Kimberly Lim, Eshani Mishra, Shruti Narayan

Use Case



Scale Crop Seam

Content-aware re-scaling takes into account features within the frame, and intelligently targets parts of the frame to remove or interpolate during the re-scaling process.

Computational complexity becomes the *bottleneck* of the implementation of the algorithm. A hardware-oriented seam carving algorithm using FPGA is proposed to improve performance.

ECE Areas: Hardware (FPGA), Software (Seam Carving Algorithm)

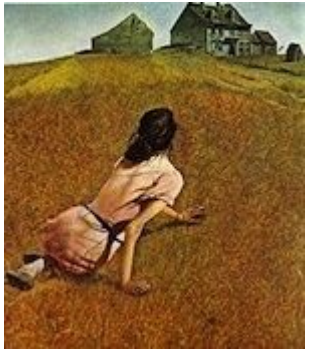
Requirements

- Resize video for different display sizes, while maintaining important content integrity
- Video processing on FPGA with a 10x speedup over using only software processing
- Static seams: most useful when the camera is stationary, and the foreground and background are separated
- Access and process user uploaded videos with a maximum resolution of 480p x 360p with a max length of 5 seconds at 30fps through microSD card
- Convert processed video into viewable format on the monitor by converting from binary to .bmp file

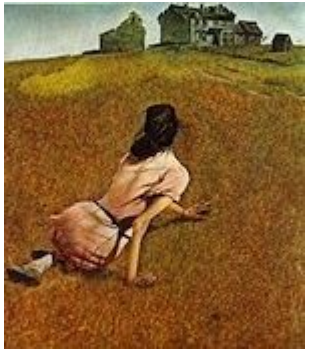
Potential Extensions

- Forward Energy
- Adding Seams (make videos larger)
- Process videos taken with FPGA camera
- Object/face detection
- Graph cut: useful for videos with moving subjects or a moving camera (out of scope due to memory constraints)

Input



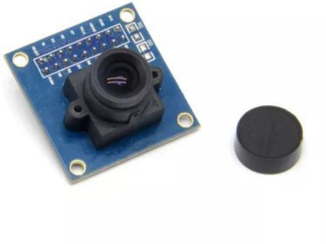
Forward energy

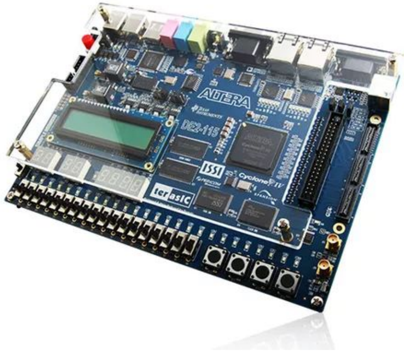


No forward energy

Technology

- DE2-115 FPGA board (for computation)
- OV7670 CMOS camera (to interface with board and provide live video)
- Monitor (for projection)
- HDMI cable (FPGA connection to monitor)
- Micro SD card (storing video files)
- Raspberry Pi (conversion of .bmp to .hex file; potentially for post-processing of output into viewable format)





Solution Approach

- User can upload videos to a library which our system will be able to access (to SD card)
- Read pictures to the FPGA from stored files on SD-card using pre-processed images that’s been converted to proper format (binary)
- Preprocess video data for use in the algorithm
- Based on traditional seam carving, the algorithm can be divided into 3 steps (sequential)
    - We will modularize this part of the project based on those 3 steps and remove static seams
- Video carving with a speedup over using only software processing
    - Speedup would be achieved through parallel processing capabilities of the FPGA, as well as eliminating latency from using interpreted languages
- Convert processed video into viewable format on the monitor

FPGA Memory Utilization

- Rough time estimate for algorithm: 30 cycles per pixel = 6*10^-7 s ~ 1 μs per pixel
- 5 second Youtube 360p video has 480 by 360 pixel resolution, 30fps
- FPGA has 3,080,000 bits for memory
- Spatial map is the most time/space consuming operation in the algorithm
    - Without parallelization:
        - 480 * 360 * 6* 10^-7 s/pixel * 30 fps * 5s = 15.552 seconds per seam
    - Parallelize the work by row
        - make groupings of each cell in the row and surrounding 8 and copy into RAM
        - 360 pixels/row * 9 pixels* 8 bits/pixel = 25,920 bits (*much more space left on FPGA - can parallelize more rows*)
        - 480 rows * 6* 10^-7 s/pixel * 30 fps * 5s = 0.0432 seconds per seam
    - Other proposed blocking methods to explore
        - Parallelize more rows at a time

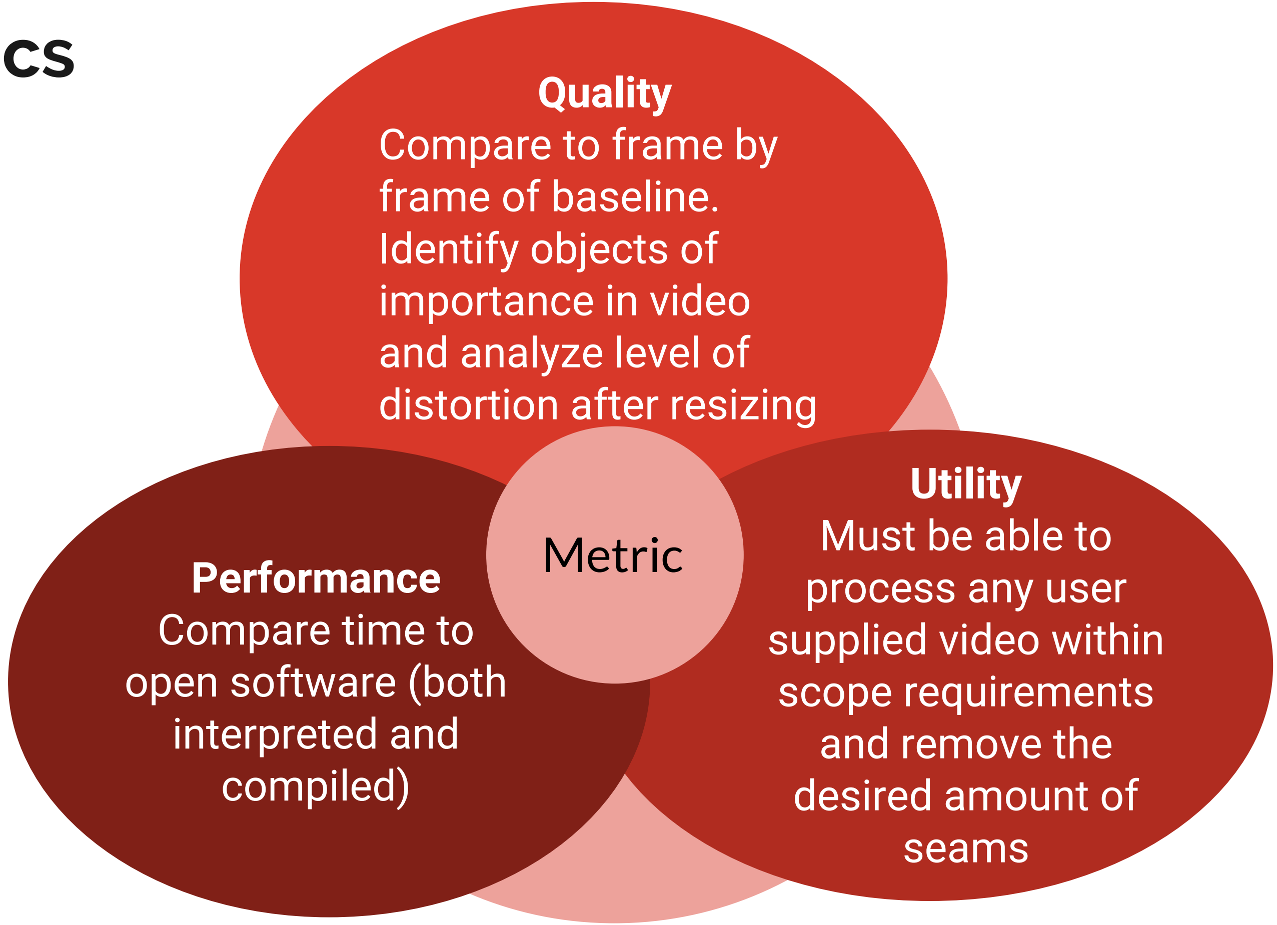
Testing + Meeting Requirements

Video resizing of different display sizes	Compare our implementation to videos to those presented in research paper of the algorithm
Speedup of processing on FPGA	Benchmark energy-map computation implemented C and compare to our FPGA using cycle counts
Static Seam removal	Identify all important features in test videos and print viewable seam lines on frames to ensure most important features are being preserved
Convert processed video into human-viewable format	Third party independent tester for verifying video output matches resolution

Verification

- Meets use cases and customer needs if:
    - Our final video preserves content better than alternative cropped, scaled, or low resolution versions
    - Performs at least 10x faster than software solution
    - Resizes video to correct resolution without distorting important features in the video (improvement over running the algorithm separately over every frame)
    - Resized video is viewable on monitor

Metrics



Quality
Compare to frame by frame of baseline. Identify objects of importance in video and analyze level of distortion after resizing

Metric

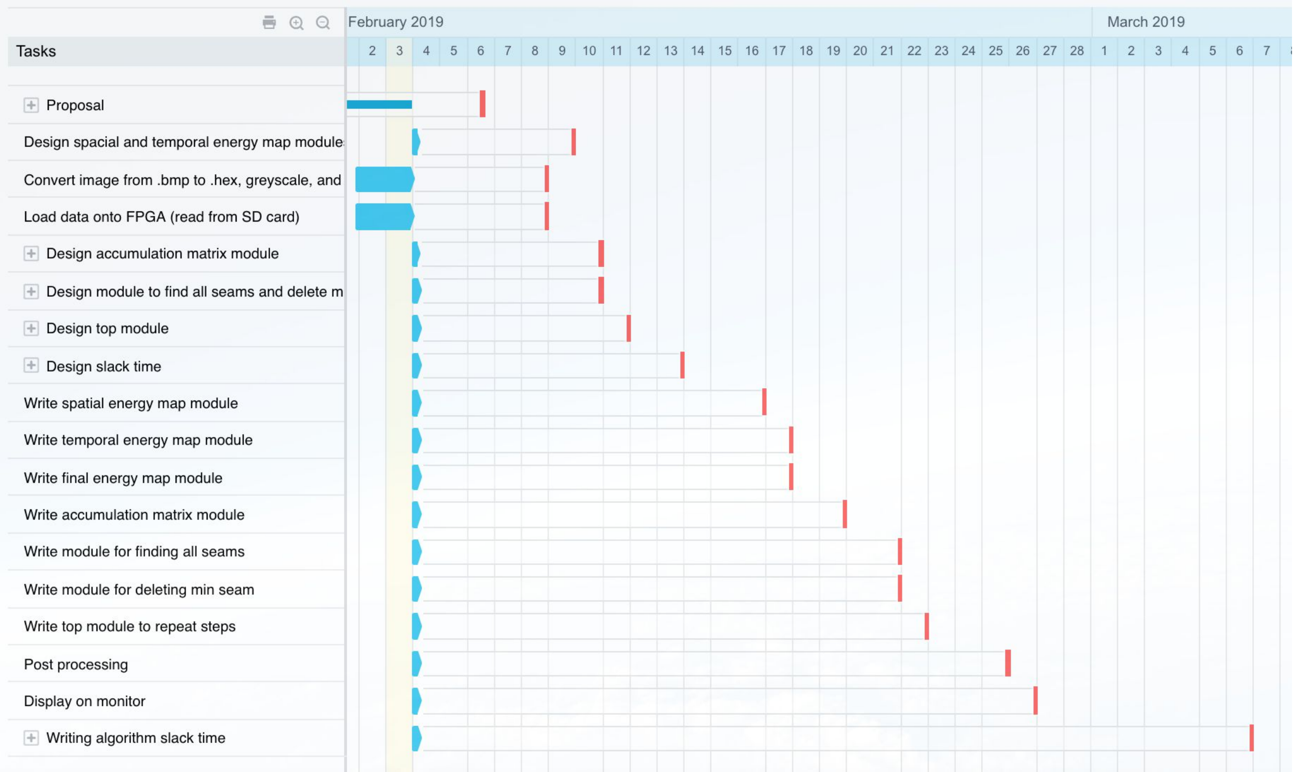
Performance
Compare time to open software (both interpreted and compiled)

Utility
Must be able to process any user supplied video within scope requirements and remove the desired amount of seams

Tasks + Division of Labor

Eshani	**Preprocessing of data**	● BMP to HEX file ● Load Image into FPGA
Shruti Kim Eshani	**System & Algorithm Design**	● Spatial & temporal energy map ● Accumulation matrix with weights ● Path-sum calculation and min ● Min-seam removal
Shruti Kim	**System Implementation**	● Files for energymap, pathsum, *main* ● restitch of video for next seam ● Top module to repeat steps
Eshani	**Post processing of result**	● Conversion from HEX to BMP ● Write result to SD ● Display result to monitor

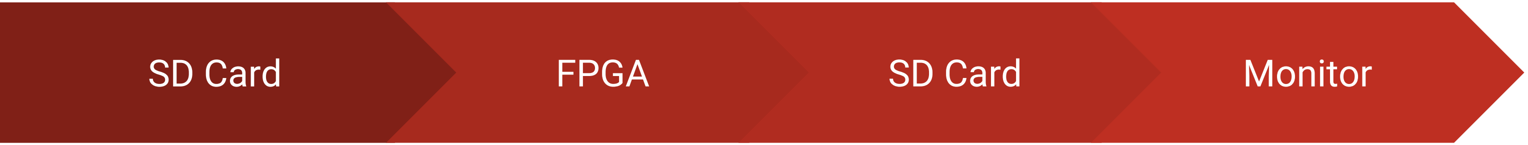
Schedule



Tasks

- Proposal
- Design spacial and temporal energy map module
- Convert image from .bmp to .hex, greyscale, and
- Load data onto FPGA (read from SD card)
- Design accumulation matrix module
- Design module to find all seams and delete m
- Design top module
- Design slack time
- Write spatial energy map module
- Write temporal energy map module
- Write final energy map module
- Write accumulation matrix module
- Write module for finding all seams
- Write module for deleting min seam
- Write top module to repeat steps
- Post processing
- Display on monitor
- Writing algorithm slack time

System Diagram



SD Card

FPGA

SD Card

Monitor

1 5-second-video of 30 fps is converted to 150 images of the same resolution. (1 .mp4 file -> 150 .bmp file)

Convert each .bmp to .hex file and store into SD card

Read binary files from SD card and process video frames according to algorithm

Store post-processed frames in SD card due to memory limit of FPGA. At end of program, all result frames are on SD card. 150 .hex files post-processed

Use HDMI or VGA on FPGA to output processed-video.