CSCI 576 Multimedia Class Project
Assigned – Oct 31st 2022
Due date of demonstration – Dec 7th, 8th , 9th 2022

The course project is meant to give you an in depth understanding of some of the areas in multimedia technology. Since this is a broad field, there can be a variety interesting projects that can be done depending on your interests which can also extend to related and complementary topics that are taught in class.

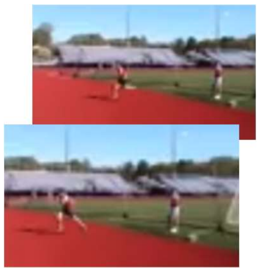
I have often found that a large project can be successfully accomplished via collaboration. Additionally, working together to design and integrate code can be a rewarding exercise and you will frequently need to work in teams when you set out to work in the industry after graduation. Accordingly, please form groups of **three (or utmost four)** students. We have started a discussion board to help you make groups, where you may post your preferred language of implementation, availability etc. to choose your partner.

This time I want to suggest a topic that is an interesting application and extension of concepts that you have learned through the lectures and assignments, specifically – block-based motion compensation, background/foreground detection, graphics transformations, creating panoramas by image stitching etc. At the heart of it is to detect foreground/background regions and objects per frame and create intermediary data structures (like a panorama) which can then be used to create novel video applications.

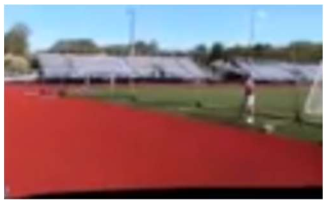
Also – for this project, you are welcome to use any 3rd party libraries and can choose your own programming environment(s) / language(s)

Generating novel videos assisted by Background-Foreground Panorama Analysis

Given an input video, where the camera is panning in a local area, eg a sports event, the project involves an analysis of the video frames to separate all the objects in motion from the background. With this separation, the foreground elements can be extracted on a frame-by-frame basis. After the extraction a composite of all the background areas of all the frames can be registered together to create a large panorama. An example inputs and outputs with panoramas is shown below being built as frames start getting processed.

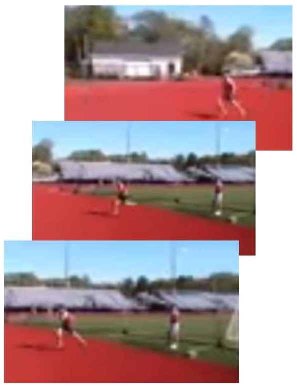


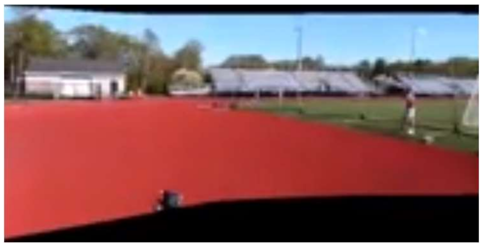
Input video frames

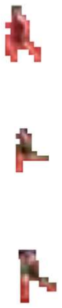


Generated background panorama

Foreground objects
for each frame







You might have correctly guessed that this is a hard problem to solve in general for all kinds of videos. But my hope here is: given well constrained datasets, both artificial and real, you should be able to experiment with this problem space and gain a good understanding of numerous subproblems that routinely occur in many multimedia projects and problems. Each of these subproblems can have different flavors of solutions. The project description and details give a direction, but you are free to read, research, experiment with different ideas and approach the problem in varying ways. 可以不按照文档来，自己商量

Ultimately, as outputs, you will assemble novel videos which were not seen before.

Inputs, Intermediary Outputs and Final Outputs of applications

Intermediary outputs are mentioned here along with final outputs because credit will be reserved for both. Intermediary outputs will give you a fair indication of how well your process is working. You will need to process, store and display both intermediary and final ouputs.

Input:

The input to your process will be a video (mp4 or rgb sequence) having the following naming convention *videoname_width_height_numofframes*. Parsing the name should give you everything about the video if you choose to use the rgb sequence. You may not need this parsing step if you are reading directly from an mp4 reader. Here are a few instances.

GeneratePanorame.exe inputvideo_640_480_290.rgb
GeneratePanorame.exe inputvideo_640_480_290.mp4

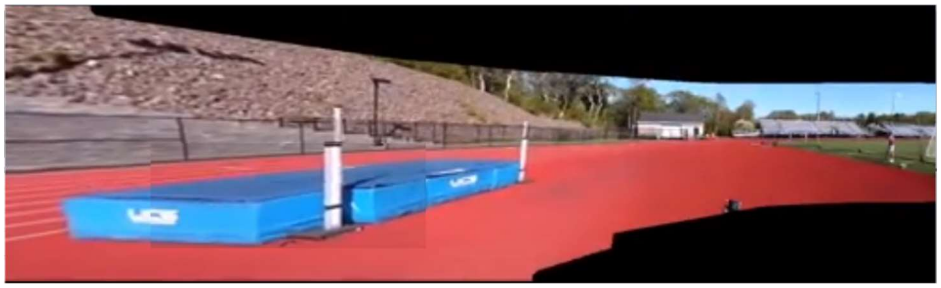
The example above indicates that the video frames have a format of width 640 and height 480. There are a total of 290 frames in the sequence. If using the .rgb files, the files will be named *inputvideo_640_480_290.001.rgb, inputvideo_640_480_290.002.rgb* …
inputvideo_640_480_290.290.rgb

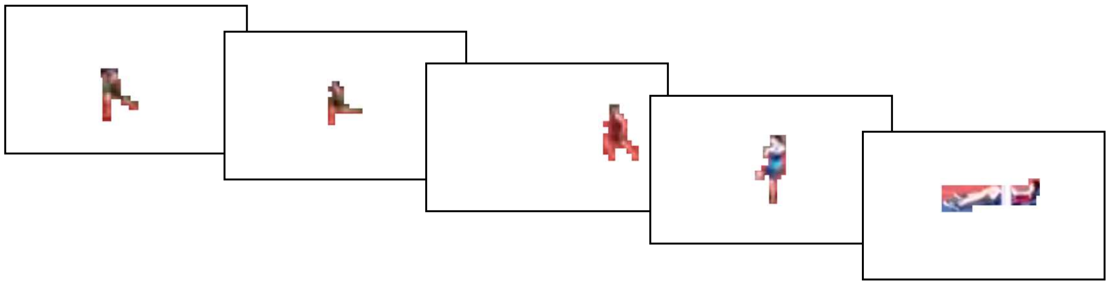
Example input video frames of an athlete running and performing a high jump is show video

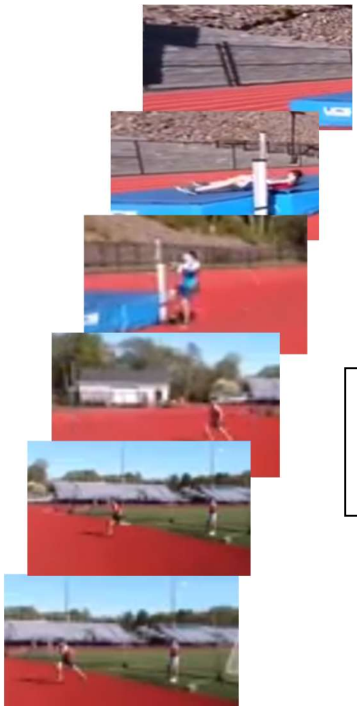
Intermediary Outputs:

Credit will be reserved for showing intermediary outputs. Please process and store your intermediary outputs as images or video as discussed below.

1. A generated panorama of the background with foreground elements removed. It is expected that after removal of macroblocks that form foreground elements, there will be gaps (holes) in the frame content. However, when the panorama is generated, these foreground holes should be appropriately filled in from matching areas appearing in past/future frames. *You are expected to display your panorama as an image. Note the image is rectangular but contents will be filled in depending on how the camera has moved in the video with many undefined (black) areas*
2. Foreground macroblock elements or objects with coordinates on a frame-by-frame basis. It is expected that the foreground elements will be define as connected macroblocks *You are expected to display your foreground objects as a video sequence against a white plain background frame by frame.*

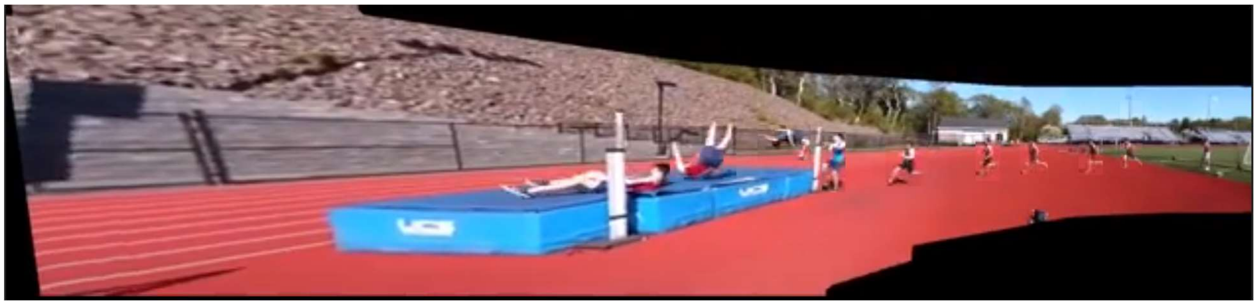

Intermediary output 1 – background panorama


Intermediary output 2 – video showing foreground elements

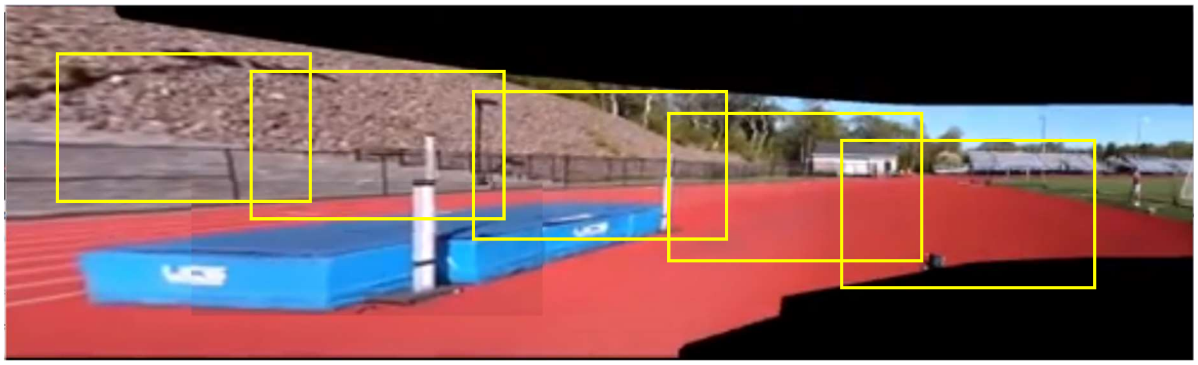

Input video frames

Application Outputs:
Suggested below are three applications that you will need to create by processing your intermediary outputs – background panorama and foreground objects.

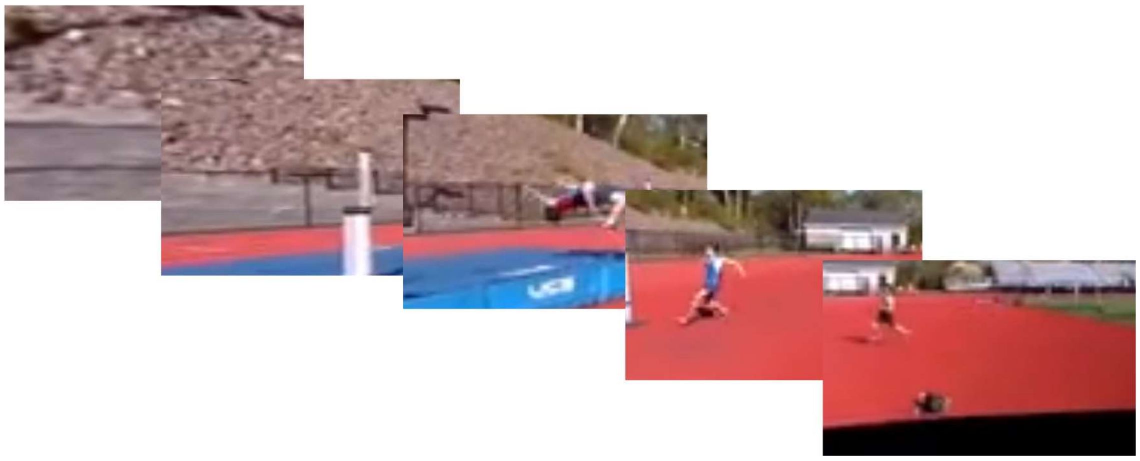
1. Display motion trails by compositing every nth foreground element on the panorama, where n is your choice of a number. The motion trail is a nice image showing a map of where the foreground objects were in the panorama.



2. Create a video a *new video* by defining a path in the panorama image, but it should overlap with the foreground objects in some capacity with the expectation that the foreground objects are composited in time synchronized manner.



New video path outlined on the panorama – you may use your choice of resolution and path



New output video with time synchronized foreground objects previously detected composited in.

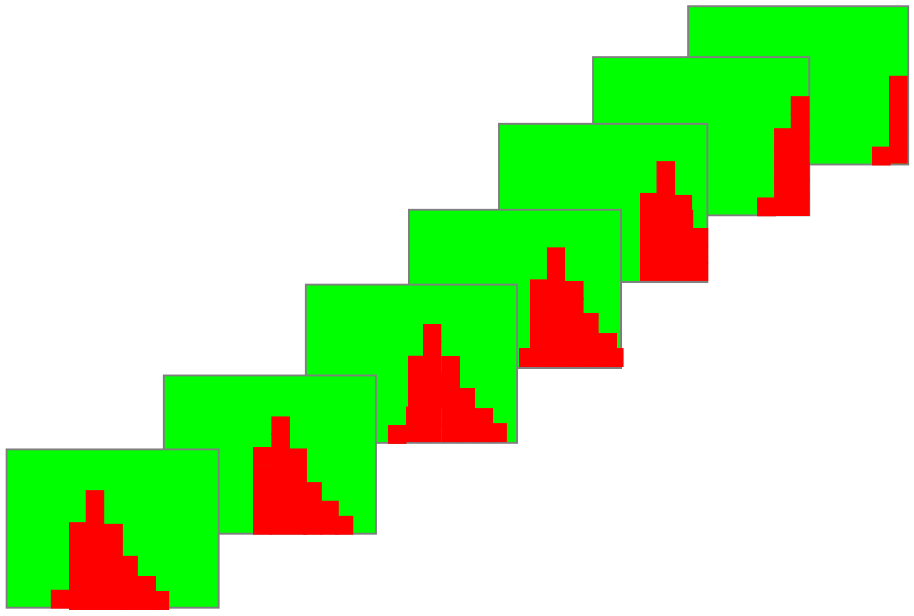
3. Remove objects from video: When your video is processed, you have a background panorama and you have one (or more) foreground objects. Can you recreate the same video by removing one of the foreground objects but keeping the rest? *(Note: if there is only one foreground object then you have a video with no foreground objects)*

Algorithm and Processing:

Step 1: Detecting foreground and background macroblocks using motion compensation
You are required to divide each frame of the video into **background** and **foreground** macroblocks objects. Note that your macroblocks may not be precise, especially at the boundaries but nevertheless you should be able to segment out contiguous blocks of foreground elements and separate the background. Enumerated below are some useful guidelines:

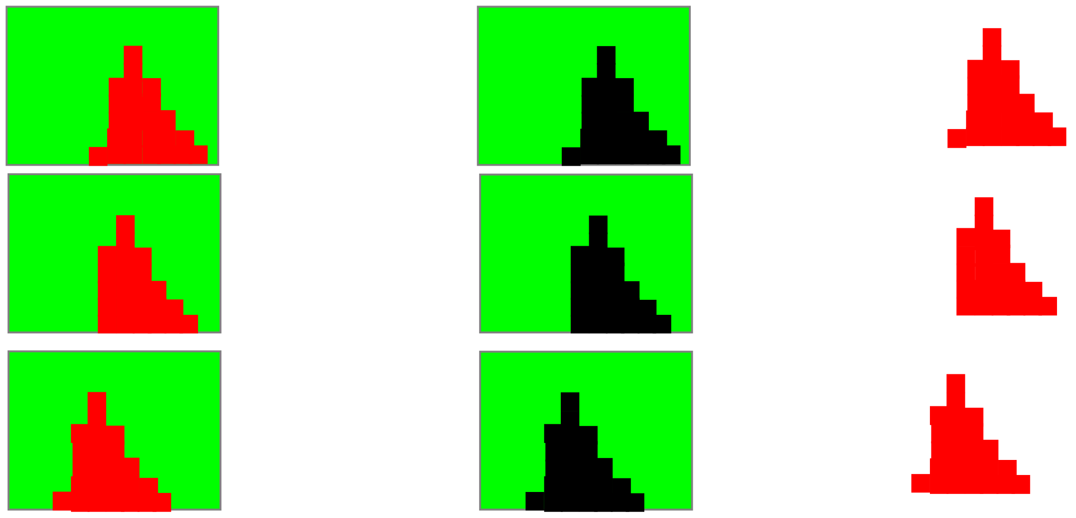
1. Divide each image frame into blocks of size 16x16 pixels
2. Compute the Motion Vectors based on the previous frame – this is the block-based MAD (mean absolute difference brute force search) that we learned in class, or you may also use any fast motion estimation (FME) technique. At the end of this step, each macroblock of the frame should have a correct motion vector.
3. Organize the blocks into background and foreground based on the similarity of the directions of their motion vectors. Background macro blocks either have a close to zero motion vectors (if camera is not moving) or a constant same motion vector (when the camera is moving). On the other hand, foreground macroblocks have similar motion vectors with macroblocks of a moving region and are directionally different from background motion vectors. Contiguous or adjacent foreground macroblocks may form the area for a foreground object.

Note that you can have different foreground regions moving in your video sequence. By the end of this first part, you should know for each frame for each macroblock – whether they represent a foreground or a background macroblock. The general-purpose solution of this problem is not easy, but as long as your program comes up with a algorithmically valid demarcation of foreground and background based on motion, that is fine. An example is demarcation is shown below where the green macroblocks represent the background, and the red macroblocks represent the foreground object of macroblocks in motion.



Step 2: Creating background only plates and foreground objects:
For all the frames, zero out all the foreground objects, but keep track of them for future processing. The background plates with holes need to be composited to form the background panorama. Content is the

holes in frame n will be filled in by the correct and continuous background content using the next step. The figure below shows a pictorial understanding of the process.



Original video frames

Foreground objects removed from frames. The background plates can be combined into a panorma

Foreground objects in frames

Step 2: Creating a panorama for the background –
Next we need to choose an "anchor" frame to initialize your panorama. The anchor frame is typically first frame, last frame or middle frame. The background panorama is created by warping the neighborhood frames around the anchor frame and compositing the warped "missing" content at each step. This amounts to computing a *transform* from every frame to see how it fits into the panorama image.

a. If the camera has moved horizontally or vertically, this transform may be approximated as a translation matrix. This may be true for some of the given datasets.
b. If the camera has rotated about is pivot, this transform may be approximated as a rotation matrix. This may be true for some of the given datasets.
c. *In general, given the various degrees of freedom (T, R, S and perspective changes), this transform is best approximated as a 3x3 perspective transform (homography).*

$$\mathbf{p' = Hp}$$

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Please refer to upcoming class lectures on how to compute the H matrix using corresponding points between images.

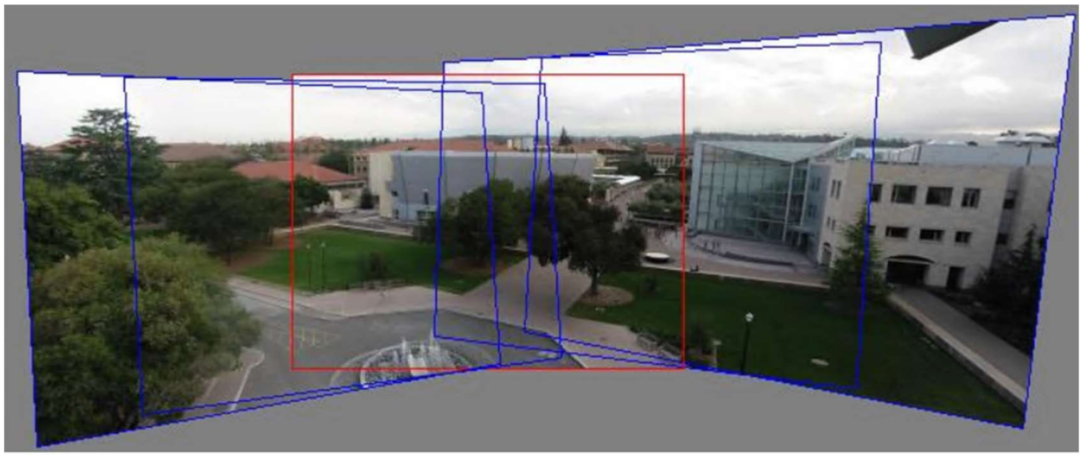


Image example taken from https://inst.eecs.berkeley.edu/~ee290t/fa19/lectures/lecture8-homography.pdf