SmartFat: A Fully Automatic Timing System with Hip Number Recognition for Track Races

Trevor Houchens, Eliot Laidlaw
Brown University
10 May 2019

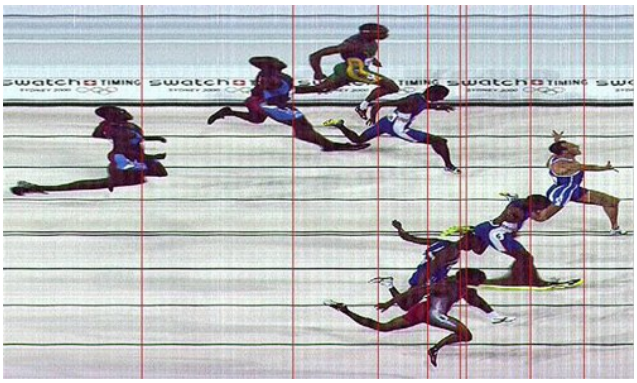

Figure 1. Cross-sectional finish line image

1. Introduction

The goal of this project was to create a fully automatic timing (FAT) system for timing and judging the finished of track and field races. We attempted to build a system that would:

1. Output a cross-sectional image like the one shown in Figure 1 that shows where in time each runner finished. In the image below, every column of pixels is the finish line at a certain time. The horizontal axis represents time, so the runner on the right finished first and the one on the left finished last.

2. Output a list of the hip numbers (in track races each runner has a sticker on their hip with a large number on it for timing/identification purposes) in the race in the order that they finished

This project is inspired by our own running and track careers. Both of us are leaders of the Brown Running Club and we have both competed in track meets and organized a track meet. The precision and accuracy gained from an FAT system is necessary for high level competition, but existing systems are extremely expensive, so we set out to create our own.

2. Related Work

Unfortunately, the vast majority of existing FAT systems are proprietary and we were unable to get insight into the inner workings for our own project. Instead, we used the techniques learned in class to build our system from scratch based on our own intuition and with the guidance of Haoze, our TA.

Our system is written in Python and relies on a number of libraries: Numpy, FFmpeg, PyAV, OpenCV2, SciPy, the SciKit Image Library, Pillow, and the SciKit Learning Library.

We collected all the data for our project ourselves. At a track meet, we used a slow motion phone camera to take several videos of the finish line as runners were passing by.

3. Method

The first thing we tackled was finding the time that each runner finished and using that information to construct the cross-sectional finish line image (Figure 1).

To accomplish this, we first loop over the video, frame by frame. At each frame, we take the absolute difference between the image and image of the background (the first frame in the video usually). This gives us an image with near-zero values everywhere except where there is a runner, as shown in Figure 2.

We then convolve this image with a vertical h x 1 filter of 1's that tapers to 0.33 at the bottom and is then divided by h. This tapering is because runners in lane 1 (the lane closest to the camera) appear larger than runners in the other lanes, and therefore their extremities are similar size to the entire torso of a runner in a farther lane. The runners in lane 1, however, are lower in the image, so the tapering reduces this issue. The result of this convolution is a w x 1 image that is essentially the difference image compressed down to a single row. We then set everything in the row that is above a certain threshold to 1 and everything below to 0.

By taking the row from every frame and stacking them up, we get a very useful image, shown in the first panel of Figure 4. In this image, time is represented in the vertical direction,

Figure 2. Video frame; background image; resulting subtraction

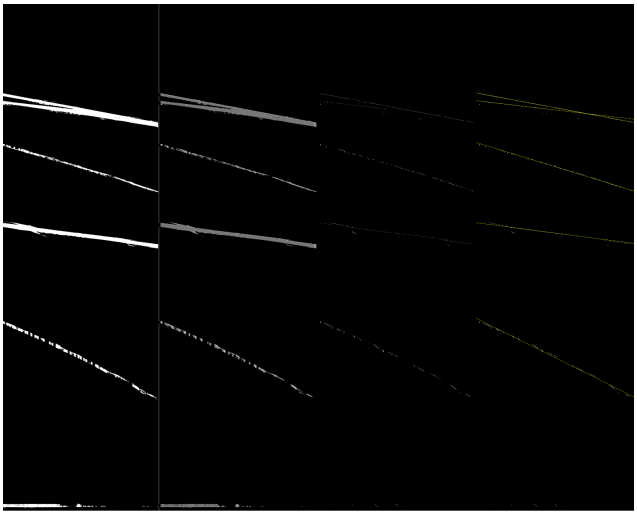

Figure 3. Stacked rows; edge detection; non-max suppression; line fitting

and the white stripes represent the runners passing through the frame. Steeper lines represent faster runners (or runners closer to the camera), and the y-coordinates of the endpoints of each line represent the frame at which the runner entered and exited the frame.

The most useful piece of information to be extracted from this image, however, is the intersection of the top edge of each line with the finish line, which is a vertical line approximately halfway through the image horizontally. The top edge of the runner's line represents the front of the runner (because the chest is what matters in track races) and the y-coordinate of this intersection point is the frame at which the runner crosses the finish line.

To get these intersection points, first we convolve the image with an edge detecting filter that is 15 x 1 with the first 11 entries being 1's and the last 4 being -1's (divided by 15). This results in the second image in Figure 4. By using non-maximum suppression row by row, we get the third image. We then use RANSAC to fit the lines seen in the fourth panel. The RANSAC process works as follows:

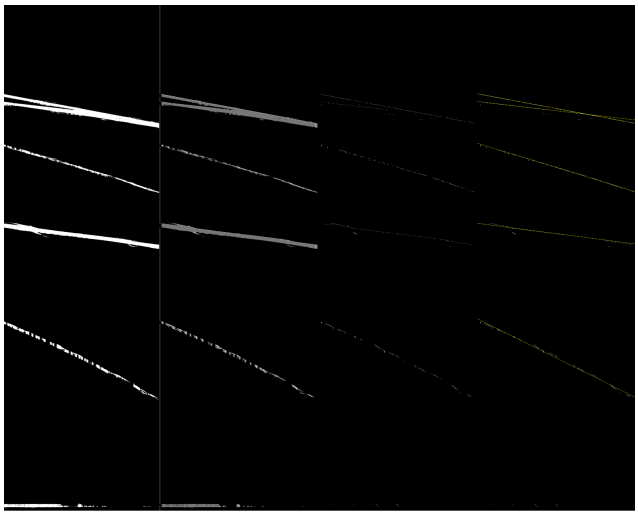

Figure 4. A two that has been modified to be classified by an MNIST trained CNN

1. Choose n points from the output of the non-maximum suppression.
2. Fit a line to those points
3. Count the number of inliers
4. If that number is greater than some threshold, accept that line as a runner, and remove all inliers from the array of remaining points.
5. If there are still several points left to be fit and we've looped less than some number of times, go to step 1 and repeat.

We can now easily calculate the frame at which each runner crossed the finish line and use that to label a cross-sectional image. The cross-sectional image is created by taking the vertical h x 1 strip of pixels at the finish line of every frame and horizontally concatenating them.

The second part of the project was to detect the hip numbers of each runner. We begin this process by clipping the image of each runner's finish to a region around their hip (shown in Figure 5). To clip the image horizontally (extract a vertical stripe that contains the runner) we use the same w x 1 rows of pixels that were stacked to make the images in Figure 4. We use the section of this row that is 1's near the finish line to clip the image. We then use the same process on the other axis, convolving our vertical stripe (minus the background) with a horizontal filter of 1's and using where that column is high to extract a horizontal stripe from our vertical stripe where the runner is. The result is an image clipped to the runner's body. We then take only the middle 40% of this (vertically) because the hip will always be around the middle of the runner.

Figure 5. Examples of clipped images to be searched for numbers

Next we scan over the small image with a hip-number-sized window. The scan is influenced by two variables, step, which determines the number of pixels to shift for each sampling, as well as a ratio which specifies how wide the window should be as a fraction of the whole image's width. During the scan, each region of the image is assigned a probability from 0 to 1 that indicates how likely it is to contain the hip number. The region with the highest probability is taken as the hip number. The probability is produced using a CNN that was trained on over 1000 data points of hip numbers that we personally collected. The CNN was created using tensorflow and a stencil from the tensorflow website. It consists of two convolutional layers and one max pooling layer. It does not need to be particularly deep since the input images are only 28x28.

Once the hip number is located, it needs to be classified as an integer. This is accomplished using another CNN. This CNN was taken from tensorflow and trained on the MNIST dataset which was also provided by tensorflow. Before a digit prediction is made by the CNN, the hip number is adjusted. In the hip number image, a function is applied to each pixel. The function keeps the current pixel value if at least one white pixel occurs above the current pixel, below it, to its left, and to its right. If this doesn't occur, or if the current pixel is white, its value is set to 0.95. This serves to make the background of the image white, while the actual digit remains black. This is more similar to the MNIST data so it makes the MNIST trained CNN more accurate in its predictions.

4. Results

Results of our program can be seen in Figure 7.

The outputted hip numbers for these images are as follows:

Figure 6: 4, 3, 2, 5, 2, 2

Figure 7: 2, 3, 2, 2

These numbers are about 30-40% accurate from our analysis. For finish detection, the program is more accurate, placing a line at about 80% of actual finishes. It also, however, includes at least one false positive per video, from our analysis.

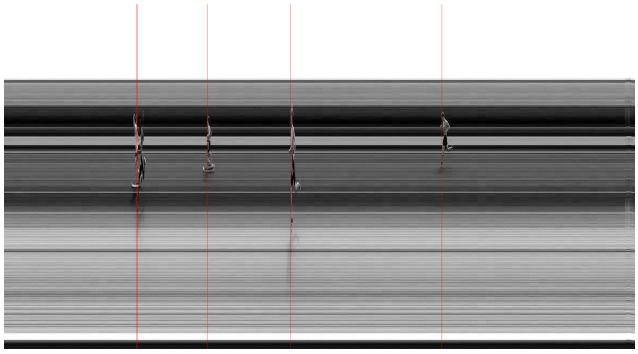

Figure 6. An example cross-sectional finish image

Figure 7. Another cross-sectional finish image

4.1. Discussion

Something that we noticed was that our CNN that we trained to find hip numbers was quite accurate (it located the

hip number within the image about 75 percent of the time), but our MNIST classification often returned incorrect results. This is likely due to the discrepancies between the training data for the MNIST trained CNN which is comprised of hand written digits, and the noisy hip number digits that we were attempting to classify. One way that we could potentially improve our classification would be to add more augmentors when training our MNIST CNN. The hip numbers are often very rotated and quite noisy, so it would be good to train it on digits that were rotated or had some background noise.

5. Conclusion

Overall we are relatively happy with our results. We (kind of) successfully processed a video of a track race to determine when each runner finished and what their hip number was. This system, if refined a little, could be very useful for our Running Club endeavors and provide a cheap alternative to expensive FAT systems.

Team contributions

Eliot Laidlaw Eliot helped to collect data and handled the part of the project that pertained to finding when each runner finished, outputting the cross-sectional image, and clipping the finish images to the runner's body.

Trevor Houchens Trevor helped to collect data, built data annotation software, annotated data and dealt with number detection and recognition.