Analysis of Runner Biomechanics Through Image Processing

Asa Kusuma | TJHSST Computer Systems 2007-2008 | June 8, 2008

Figure 1: The program in action

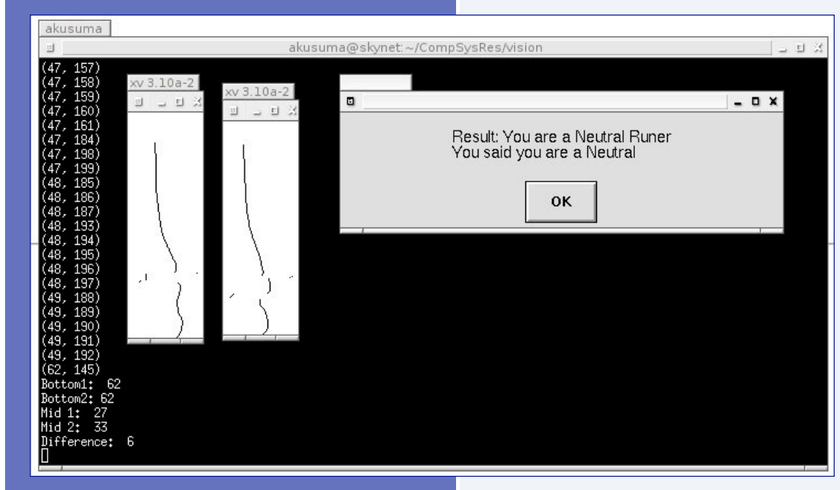

Figure 2: Program output

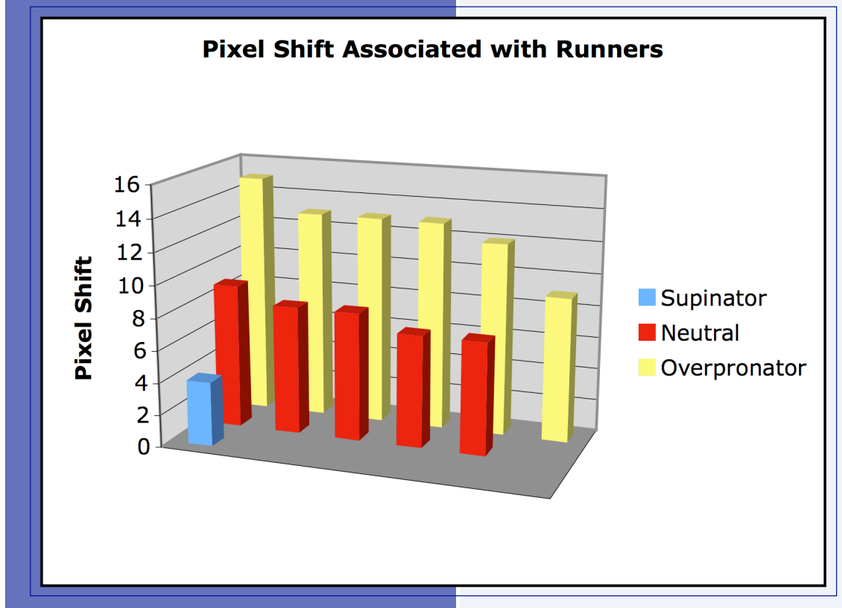

Figure 3: Data collected regarding pixel shift

Figure 4: Image collection setup

Introduction

The biomechanical features of a runner in an image can be analyzed by using certain image processing techniques, the primary method being edge detection. By constructing an accurate, two-dimensional model of a runner's lower body from a rear angle, it is possible to extrapolate the underlying qualities of that runner's biomechanics. This is done by creating an outline of a runner's lower leg and feet. An edge detection algorithm is applied on an image to create this outline. By comparing the results of the edge detection algorithm input with images of the runner before and after impact, biomechanical features can be determined. In this type of situation, algorithm speed is not a very relevant issue; accuracy is far more important, the reason being that you only need to analyze a few images to create a two dimensional model of the lower body, as well as the fact that the time it takes to analyze a runner does not directly affect his performance as a runner.

Background

The goal of this project is to analyze images of a runner and extract biomechanical information about the runner from the images. Among runners, a major cause of injury is overpronation. Pronation is the natural inward rolling of the ankle to absorb impact. All runners should pronate to a degree, but many runners pronate too much, causing misalignment, knee problems, and problems with the muscles and ligaments around the ankle. Even worse, overpronation puts abnormal stress on the inside shin bone, the Tibia. This can lead to shin splits and even stress fractures. Conversely, many runners don't pronate enough. This situation is called supination--such runners are called supinators. Supination can cause problems similar to those stemming from overpronation, but instead, the problems are usually with the outside of the leg. For instance, supination causes stress on the outside shinbone, the Fibula. Like in overpronators, stress fractures can result from supination.

Like most biomechanical features in the human body, pronation is a visible phenomenon, but hard to recognize to the un-trained eye. Pronation happens very quickly, and the movement is miniscule. This type of movement is hard for humans to see, but much easier for a computer, armed with a 20 frame-per-second camera. Using only images from a camera, the project will determine the degree of pronation of a runner. Such an ability could be instrumental in determining the proper shoe type and diagnosing injuries. The project will strictly be involved in analyzing images from a controlled environment and determining biomechanical features from analysis of images. This means that the project will not be concerned with selecting images from a video feed or trying to analyze images taken in random and widely varying situations. The images used in the project will be taken from the back of a runner running on a treadmill, not from a runner running in stormy weather in an urban environment, taken at an awkward camera angle. There is very little purpose in trying to determine the biomechanics of random people walking in the street, so focusing on controlled environments makes the project much more feasible at almost no cost to applicability in the real word.

Development

For the actual program, a programming language had to be decided on. Originally, C++ was going to be used, but after further though, Python was chosen. Because the nature of the project concerned developing, testing, and trial-by-error coding of algorithms, Python, a very easy to code and simple language, was chosen. The downside is that Python is slightly slower than C++. The program was written as a python script that can be run on any computer that has python installed.

The first step in the process is to acquire the right images, namely, images of a runner's leg in motion, before and after foot impact with the ground. In order to increase the accuracy of the algorithm, it is important to develop a proper and uniform setup for capturing images. There are two variables that need to be constant when devising the system: image resolution, distance between the runner and the camera. Runner speed needs to be faster than jogging speed and slower than sprinting speed. In other words, the runner must be lifting up his knees, but he shouldn't be up on the balls of his feet.

With these parameters in mind, a concrete system can be devised. To capture the images, the camera is placed behind the treadmill, with the lens placed just above the treadmill running surface. See Fig 4. for a picture of the physical setup. Once the runner is moving on the treadmill, the camera begins capturing video. The video is loaded onto a computer and the frames are extracted and converted to raw images. An image taken before impact and an image taken after impact are manually selected and input into the

Within the program, the images are prepared for edge detection using Gaussian blurring, noise removing techniques, and outlier removal algorithms. In order to reduce noise I developed an algorithm for targeting continuous lines. I found that the top of images tended to be more accurate because there are no irregular lines around the middle and upper portion of the lower leg. Thus, the algorithm finds the edge near the top of the image and works down the edge, only including edge pixels that are near the pixels above it. If a gap in the line forms, as vertical size of the gap increases, the algorithm allows for pixels to have a larger difference in x-coordinates from the nearest pixel above.

After preparation, an edge detection program creates an outline of the inner leg. Once the two edges from the two images are derived, the edges need to be aligned properly so that they can be properly compared. Often, one edge is larger than the other edge. This usually happens when one image is blurrier or has slightly different lighting, so the computer can't accurately find as many edge pixels for the edge. Because of this discrepancy, it wouldn't be accurate to compare both images, so the sizes and positions of the edges must be equalized. This is done by reducing the size of the larger edge, to match the size of the smaller edge. Then an algorithm is applied to both edges, in order to find the average x values of the outlines. These two values are compared, producing a pixel gap, or the difference in pixels between the two edges. The larger the pixel gap, the higher the degree of pronation. However, the output of this method, the pixel difference between the two averages, is relative to the camera resolution. The same level of pronation recorded with a camera with a larger resolution will look like more severe pronation. After testing the program on several runners, each running at a different distance from the camera, I found that as long as the camera was placed in the same position, at the end of the treadmill, there was no significant effect on the accuracy of the program.

In order to produce a practical program that can be used by others, it is imperative to develop some kind of graphic user interface that the average person can use. While I didn't have the time to develop a completely stand-alone graphical user interface, I did implement a graphical user interface within the python terminal program. In other words, the user must open a terminal and run the program using the python command, but once that step has been completed, the rest of the user experience has a graphical user interface.

Results

After testing the program on multiple runners with all types of biomechanics, I found that if the program found a pixel shift of zero to five pixels, then the runner was a supinator. If the program found a pixel shift of six to ten pixels, then the runner was a neutral runner. If the program found a pixel shift of more than ten pixels, than the runner is an overpronator. My program has been tested on twelve runners; see Fig. 3 for the data results. Only one of the tested runners was incorrectly assessed by the program.

Conclusion

While the scope of the testing of the program is not quite large enough to cement the total accuracy of the system, it is clear that the project has successfully created a way to analyze runner biomechanics. The data in Fig. 3 shows a definite trend with an increase in pixel shift associated with an increase in pronation. Unfortunately, the program must still be run on a Python-equipped machine, but there is an implemented graphical user interface.