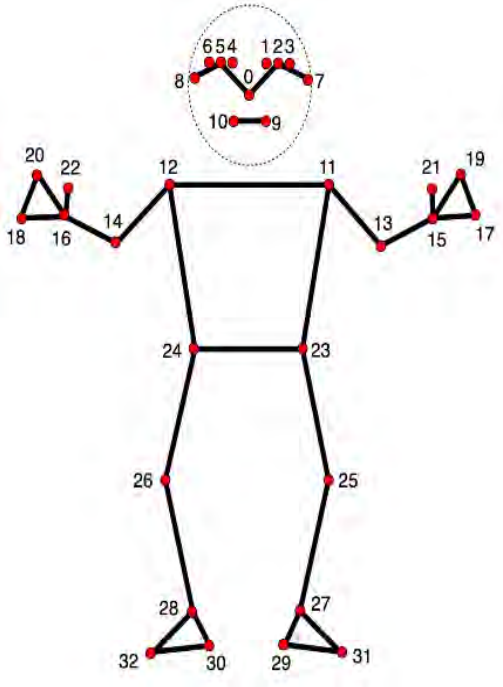
Cameras On! – Build Your Own Apps for Body Position Training Analytics

The combination of new and old tools paired with AI is making what used to be body-position and performance analytics reserved for elite levels accessible and straight forward to build and customize for yourself. I wanted to capture some of the: tools, flow of development, and considerations for how you might want to put together your own app to give you the data you need or desire on your performance. As an example, I am using a computer- vision body-position app I built for my daughter in a few days to help me better understand some of the details she is working on as a youth competitive swimmer.

Concepts and Terms to Know:

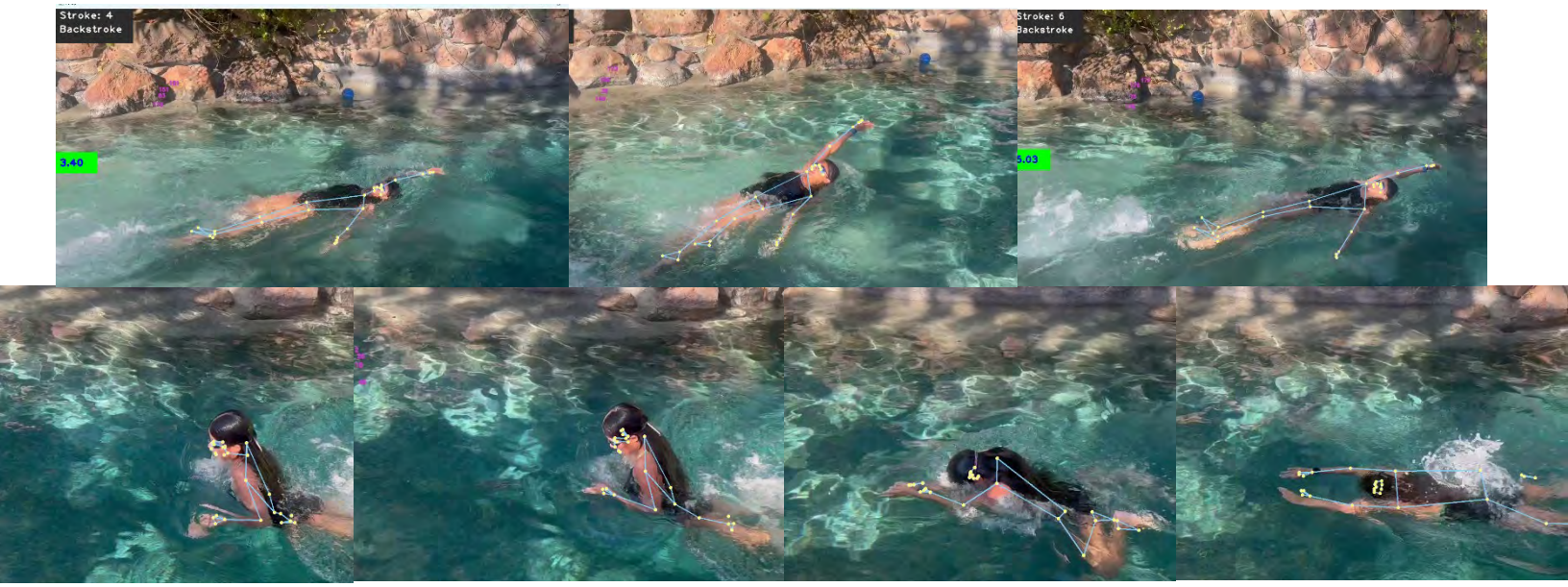
Pose-estimation – You will need to identify a pose-estimation algorithm to use in your development. This is the algorithm that will allow you to take a personal video from a mobile phone or a GoPro and identify where there is a person – or persons – in the video and ascribe **Keypoints** to specific parts of the body that can be tracked for changes in position throughout the duration of your video. Keypoints are often a set of 30-40 points that the algorithm will identify on the body and use to track movement. Keypoints should map to locations on the body that help anchor the center of the person being analyzed as they move in space and about axes of rotation and points at joints or areas of the body that have a lot of change in position as you move. At this point you are building a "Markerless Motion Capture" system. It is "markerless" because it is using computer vision computation to identify the relevant body keypoints rather than tracking stickers or sensors that are placed on the body.



Here is the Keypoint location mapping used by Google Gemini in their latest Pose-estimation algorithm that is part of their Media Pipe Open-Source code and tools for building computer vision computation into useful tools. I highly recommend reading and using tools you can find at: Google's Media Pipe Development Tools

This is the AI tool that was just used by Google in their partnership with Steph Curry for the awesome AI Basketball Coach – Check it out: Steph Curry / Google Partnership - AI Basketball Coach

I have integrated this keypoint set by using the MediaPipe tools with some added code and analytics into a swim-stroke analytics app I've built for my daughter. You can see the keypoints and their movement as she swims. In this instance, I am starting with this code-base in Github. Swim analysis – at least from outside the water – can be a tough problem for a pose-estimation algorithm because some part of the body is often underwater and occluded during every moment of the video.



You can see it is doing well given this comes with the territory, but there are limitations to what you can compute accurately or considerations for which model you might want to choose. For example, some pose-estimation models do better and optimize for handling occluded moving objects whereas others out-perform for handling of multiple persons in the scene (eg a soccer pitch). In my example, I am using Media Pipe which uses Google's pose-estimation algorithm Blazepose.

Once you have stable tracking of the keypoints you can run and process any analytics you wish. For example, you can easily capture from the data above metrics of: stroke rate, body rotation, arm acceleration, line consistency, angles of entry, and durations between different key movement positions.

What to consider in choosing a tool:

Are you tracking a single person or many? Some algorithms are better than others for solving this problem.

Will the entirety of the body be visible for the duration of the video and your intended tracking or do you anticipate occlusions?

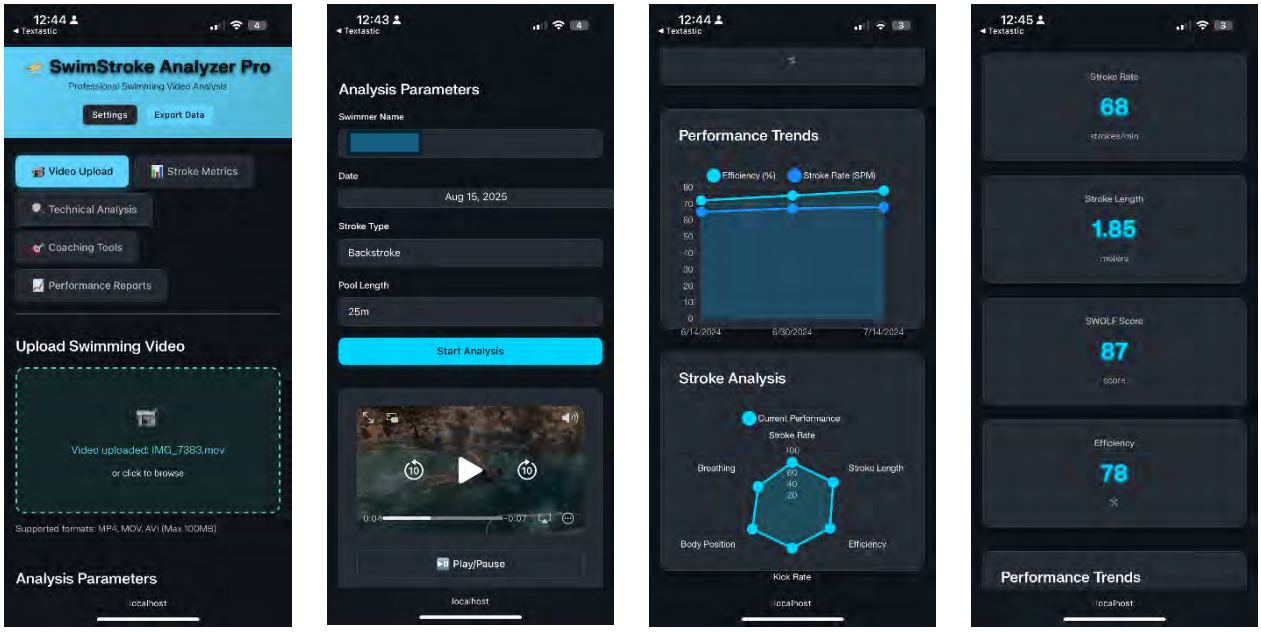
Do you have control over the angle of capture that you will measure? For example, the swimming pose-estimation I show above would be easier if I could capture the video from over the swimmer. However, that is not a practical use case. In contrast, if I were trying to capture yoga poses or body-position during any sort of strength training there is much more flexibility and stability in the position of the camera and video that I can capture during the duration of my pose estimation.

Here are a few resources to help decide which algorithm and model to work with: Trade-Offs in Pose-Estimation Models

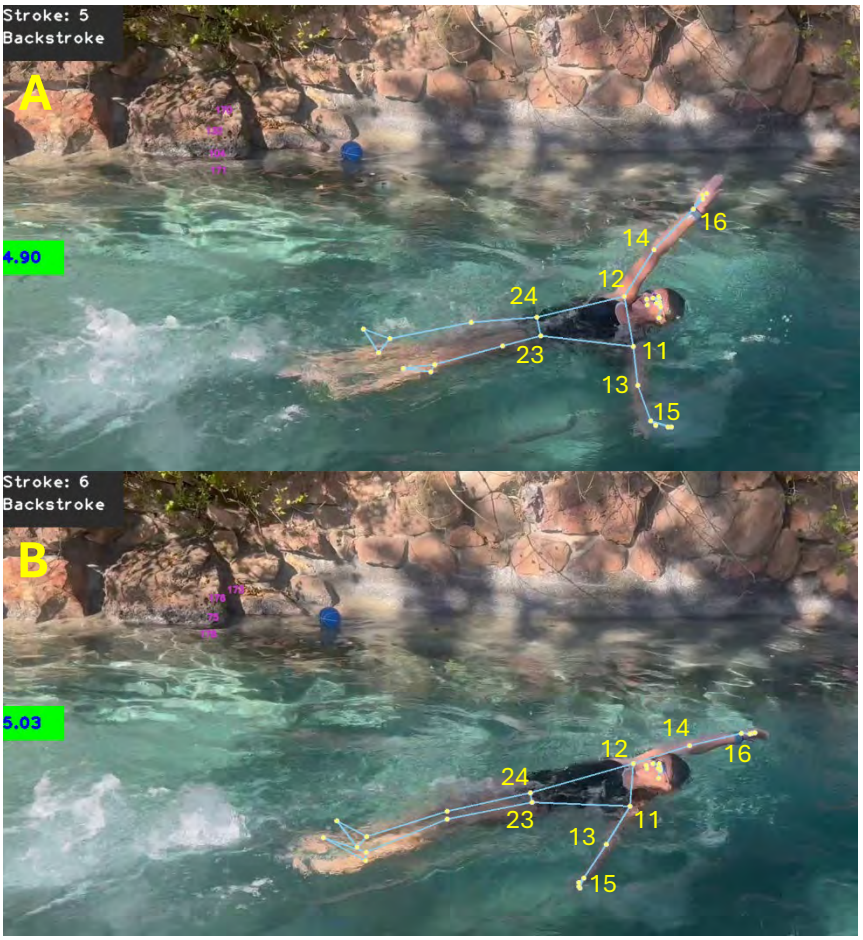
Useful Tools and Resources:

Perplexity Labs

Perplexity Labs requires the paid "Pro Version" and is different than the base Perplexity in that it produces assets and code you can use immediately. In the example I will walk through, I leverage Perplexity Labs for quick development of a basic mobile application and user interface. It's not refined, but entirely usable. Note – Perplexity does not currently have its' own computer vision pose-estimation model. This means if you decide to use Perplexity for any app development, you will need to use a computer-vision model and code outside of Perplexity for the basic pose-estimation algorithm and data generation. The Media Pipe tool kit linked is a great alternative. You can pull this data into perplexity for further processing in the application you have built. I will give an example of how you might do this. There are a few open-source options. Here is an example of the functional mobile application I could quickly mock-up through prompts (it isn't elegant but is functional for my needs) from Perplexity Labs to upload a video to my pose-estimation model and run analytics on the keypoint output. There are some early metrics here, but they could be customized to whatever you wish, and you can definitely improve the UI and UX with improved prompts and direct code modifications.



Below I've tried to provide a sequential walkthrough of one approach you can take to create a custom performance tracking app. Follow the bullets and links. In this case, I have built an html application and used cloud services to run the pose-estimation processing. Alternatively, you can use Blazepose or some of the Google tools that will run code directly on a mobile device. There are trade-offs in development tools. In my case, I was already running in the cloud. I will walk through how to get this up and running:



1. Identify an appropriate pose-estimation algorithm. Let's assume you are using Google's Media Pipe and Blazepose. Modify the pose estimation video processing code to calculate the desired pose metrics. This is really where your opportunity for customization comes into play. You are taking the keypoint values that your pose-estimation algorithm provides and computing some metric of change or performance on this data. For example, in the swim videos above and the one right here, stroke count can be computed in multiple ways from the keypoints by triggering a count every time the arms keypoints (14,13, 16, and 15) move in and out of identified value ranges. I've indicated these on the images above. Similarly, we can track torso rotation by looking at the values of keypoints: 11, 12, 23, and 24 and establishing a metric between the values of the four points that is computed continuously. You can clearly see how keypoints 23 and 24, and 11 and 12, are closer together in **Image B** in comparison to **Image A**, which is an indication that the swimmer has rotated to the right (looking down) on their longitudinal axis (head-to-toe side-to-side rotation). Accelerations, velocities of specific keypoint speed and movements, can offer immense insight.
Once you are happy with the metrics you are computing on the keypoints you will need to:

2. Create a Docker image of the pose processing code. A Docker image is a way of containerizing and assuring all the code, supporting packages, and libraries are contained and located in a common place that will allow your algorithm to run. To know what this list is you can use Perplexity Labs to generate the "Requirements" list and create a first-pass Docker image.

3. Upload your Docker image to Amazon AWS. Again – For this instance, I am running the code in the cloud and built an HTML App to be system agnostic. If you run the processing directly on the mobile device you can bypass using AWS, however the there are limits to some of the pose estimation algorithms, differential support per pose-estimation algorithm

for different operating systems, and differential outcomes if you want to make use of an AI asset generation tool.

4. Create AWS S3 buckets for raw (input) and processed (output) video and results

5. Create AWS Lamdas to trigger the Docker image to run on the input video uploaded to AWS S3 and place the processed video and results into the associated AWS S3 bucket. This is likely straight forward for anyone who works in algorithm/software development, but if not, you can make use of AWS Code Whisperer, Perplexity Labs or an alternative such as Replit to generate the code needed to make these calls to AWS from your app as well as identify where and what is needed in your specific scripts.

6. Setup an Gateway HTTP APIs to interface the S3 buckets

7. Now you can build an application to run the processing and report the analytics, or you can use an AI asset creator like Perplexity Labs to build a first-pass app and interface to your algorithm and metrics. In Perplexity Labs, generate prompts that specify you want to build a real-time analytics app that targets some sort of human performance. Be specific and iterate with the output to further specify the details of your UI and user behaviors. In the example above, I wanted to build a swim-stroke analytics app that would work on both my mobile device and laptop. In your prompts you can specify metric support, behavior, layout, color design, etc. . . .

8. Use Perplexity or whichever AI tool you are using to identify the proper position in the code and calls in your code to upload your video and push from the App into the S3 bucket. Do the same for the pull of the processed video with the pose-estimation keypoint processing back into the App for viewing in the App from the processed S3 bucket. Include the initial metrics and data from the algorithm that you want to plot and represent in the application.

9. If you chose to build an HTML app like I show above, you can quickly run a functioning app on your laptop. To run the same code on your mobile device you can make use of an app like Textastic or another service to quickly run the index.html file of the app from your mobile device.

Ok! You should now be set up to start using your app! Be creative and dive into the data and insights you can gain when you have AI computer-vision algorithms paired with your personal knowledge plus the knowledge of your favorite coaches and experts working for YOU and only you.

Cheers,

Poppy