Golf_SwingML

By Luis R. Alvarez Jr

Tiger Left elbow angle: 187 deg

Luis Left elbow angle: 189 deg

Golf SwingML

Abstract
Golf SwingML is a project that leverages Google's API and python libraries to analyze and detect a person in a video and then output a file with labels identifying a person and various landmarks on a person's body, such as the shoulder, elbow and arm.

The result is an excel file and graphs that contain a time-stamps with corresponding left elbow angles. Close analysis of this data frame reveals that the pro golfer keeps the left arm remarkably straight with the left elbow registering about 185 degrees for most of the swing. The armature golfer shows the left elbow in a more bent position of about 210 degrees. Also, it was noted that the pro golfer momentarily flexes the left arm into 135 degree angle. This showcases how flexibility along with power and accuracy help the pros hit the long ball.

Introduction

Off the tee, in the year 1997 the average PGA tour driving distance was 267 yards. At the same time Tiger Woods is outperforming his field by averaging 294 yards (Woods). As of August 2021, the tour average driving distance has caught up to Mr. Woods. Today a normal drive on the men's tour is a whopping 295 yds. With the number one player hitting an average distance of 320 yds (DeChambeau). If the pros can increase their driving distance, why can't you?

This project uses Python and Google's Video Intelligence Application Interface (API) to analyze golf swing video footage to better understand how the best in the world hit the long ball. Specifically, Google Cloud (GC) offers services that allow programmers to leverage cloud storage and machine learning to annotate video with labels. The GC API produces a Java Script Object Notation (JSON) file that contains a dictionary object that labels coordinates on the video and estimates what the point is, such as eyes, nose, shoulder, elbow and wrist. These annotations are then processed using python, pandas, matplotlib, and math libraries to produce a Comma Separated Values (CSV) file that contains a time series of golf swing angles.

Data Collection

Data is collected in three steps. First video is captured on a smartphone, second it is saved in cloud storage, and third the cloud storage bucket is accessed by the GC API and then outputs a file that is letter refined with python.

Driving Range and iPhone Recording: Data collection Step 1

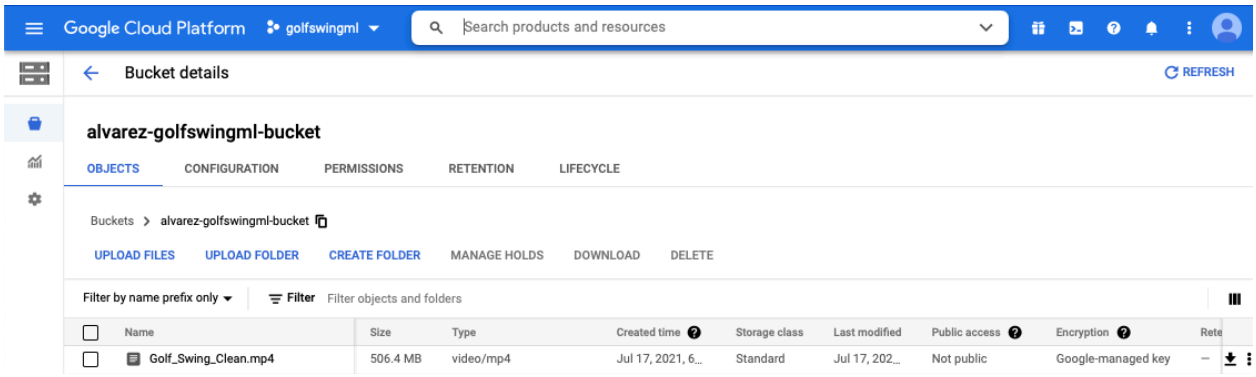
The data is captured using an iPhone and a tripod at a driving range. The video is processed with iMovie on an iMac. The video footage is cut to contain 50 golf swings at a driving range. Next, a video of a professional golfer is downloaded from YouTube to serve as a comparison at a later stage. The video is then processed in order to obtain annotations from google using two methods, namely the command line approach and programmatically. The second step in data collection is to request video annotation from the GC API.

Google Cloud Bucket Setup: Data collection Step 2

Cloud Storage is leveraged in order to further process the initial video and extrapolate the label annotations. To setup a remote server in the cloud create a cloud storage *bucket* sign into a google cloud account and browse to the Cloud Storage menu. Click on Create Bucket, type a

globally unique and permanent name, select region location, select standard storge class for data, and then select uniform access control. This last step allows for permissions to the resource via Identity Access Management (IAM). Next, browse to Create Project, click create; note the project name, organization, and location are setup here. Before the API can access the bucket and video mp4 file configure permissions by browsing to Service Accounts and selecting Create Service Account. Type a service account name, grant the role of owner and select done. Select the service account just created and click on Keys, Add Key, and Create. Check the downloads folder and there will be a unique alphanumeric JSON file that will allow remote access to the cloud resource.

Create bucket, Configure Service Account, and Project



Async API Request: Data collection Step 3

Goolge's VideoIntelligence_v1 Application Interface (API) parses video and applies a machine learning model to detect a person in the video. Specifically, this project analyzes a video of me at the driving range hitting 50 golf balls with a driver club on a sunny day in Iowa. The API **VideoIntelligenceServiceAsyncClient (**google cloud client libraries**)** uses the annotation video method to process a preconfigured request file that contains all the arguments necessary to asynchronously parse the video to detect a person and their movements.

To setup the request, first it is necessary to configure the virtual environment with an access token. Run the command export GOOGLE_APPLICATION_CREDENTIALS="alpha-numeric_key.json" with the credentials pointing to the key file downloaded earlier and now saved in the virtual environment. Once the credentials have been setup create file named request.json and copy the code below as it contains the parameters that are used to trigger the annotation by the google service. The script can be found in the Google API documentation. The two most important parameters in this script are the *inputUri* and *features*. The first parameter gs://alvarez-golfswingml-bucket/Golf_Swing_Clean.mp4 is the cloud storage bucket and video file to be processed by the annotation algorithm. The second parameter denotes the type of annotation which in this case is PERSON_DETECTION. Note, the google API also has methods that can run facial recognition, track objects, transcribe videos, and much more.

Contents of request.json file: source: Cloud_Video_Intelligence_API.

```
{
    "inputUri": "gs://alvarez-golfswingml-bucket/Golf_Swing_Clean.mp4",
    "features": ["PERSON_DETECTION"],
    "videoContext": {
      "personDetectionConfig": {
        "includeBoundingBoxes": true,
        "includePoseLandmarks": true,
        "includeAttributes": true
       }
    }
  }
```

Using the terminal proves to be the fastest way of processing the video and has the least number of dependencies. When compared to setting up the google cloud Software Development Kit (SDK) a lot less work is needed to process a video. However, since this project is only running batch processing it is not necessary to setup a full data pipe-line with the SDK. With more time and effort, the docs clearly explain how this can be accomplished programmatically using languages like python, java and node.js. From the terminal run the following curl commands to send a POST and GET request to the annotation video method in the GC API.

Send video annotation request

```
curl -X POST \
-H "Authorization: Bearer "$(gcloud auth application-default print-access-token) \
-H "Content-Type: application/json; charset=utf-8" \
-d @request.json \
https://videointelligence.googleapis.com/v1/videos:annotate
```

Response name-string

```
{
 "name": "projects/PROJECT_NUMBER/locations/LOCATION_ID/operations/OPERATION_ID"
}
```

Configure Response command:

- PROJECT_NUMBER: found in google cloud project
- LOCATION_ID: data center location selected during bucket creation e.g., us-west1
- OPERATION_ID: unique 24-digit code to identify the long running operation

GET response command

```
curl -X GET \
-H "Authorization: Bearer "$(gcloud auth application-default print-access-token) \
https://videointelligence.googleapis.com/v1/projects/82932726978/locations/us-west1/operations/17480310946357676047 > ML_out.json
```

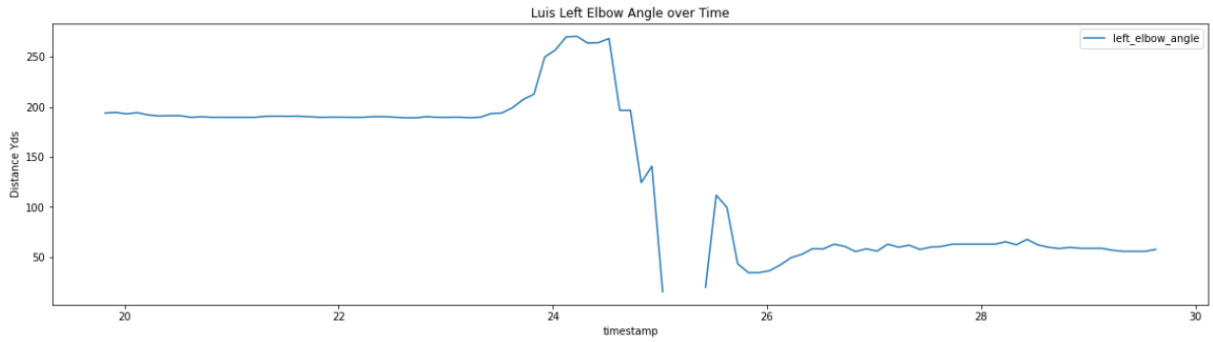
Summary access token, POST, and GET commands

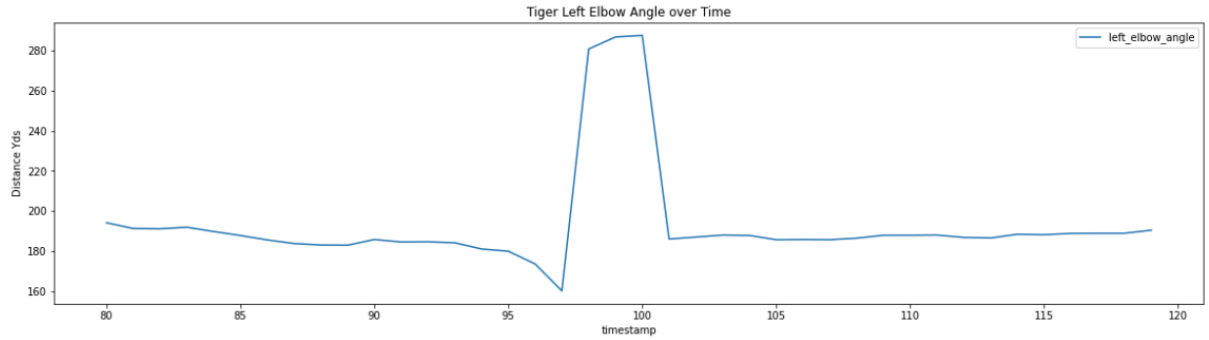
```
(my_Virtual_Env) (base) luisalvarez@Luiss-iMac my-folder-name % export GOOGLE_APPLICATION_CREDENTIALS="golfswingml-b92ada60eb03.json
"
(my_Virtual_Env) (base) luisalvarez@Luiss-iMac my-folder-name % curl -X POST \
-H "Authorization: Bearer "$(gcloud auth application-default print-access-token) \
-H "Content-Type: application/json; charset=utf-8" \
-d @request.json \
https://videointelligence.googleapis.com/v1/videos:annotate
{
  "name": "projects/82932726978/locations/us-west1/operations/12834128508130435308"
}
(my_Virtual_Env) (base) luisalvarez@Luiss-iMac my-folder-name % curl -X GET \
-H "Authorization: Bearer "$(gcloud auth application-default print-access-token) \
https://videointelligence.googleapis.com/v1/projects/82932726978/locations/us-west1/operations/12834128508130435308 > ML_out.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   427    0   427    0     0    747      0 --:--:-- --:--:-- --:--:--   746
(my_Virtual_Env) (base) luisalvarez@Luiss-iMac my-folder-name % ls
ML_out.json                   golfswingml-b92ada60eb03.json   my_Virtual_Env                request.json
(my_Virtual_Env) (base) luisalvarez@Luiss-iMac my-folder-name %
```

Example of raw data returned from GC API

```
{
            "name": "right_ankle",
            "point": {
              "x": 0.48058289,
              "y": 0.73199129
            },
```

Visualizations Post Processing with Python





Description

The graphs compare one golf swing from Tiger and one from Luis. The data collected with the iPhone is choppy and miss collecting some of the swing. This is due to the fast action of the golf swing and the relatively low frame capture rate of the iPhone. The average golf swing for Luis takes about 2-3 seconds with the down swing taking tenths of a second to occur. The second golfers graph shows a much smother and complete sample. This is due to the high-speed footage used to record Tiger. By estimation, the Tiger video captures his swing about 7 times faster than the iPhone used for Luis.

Tiger's swing shows that the golfer keeps the arms straight while winding up the swing and after releasing it. The line graphs show how the greatest elbow bend degree occurs at the top of the swing above 200 degrees. Luis' graph also shows how the sampling is best during the slow wind up, but then is unreliable after the release. The data is missing annotations regularly for the iPhone footage specifically during the rapid release action. Luis' graph shows the elbow angle less than 100 degrees after the gap in data which is not accurate. Further inspection of the video overlayed by a stick figure representing the predicted labels for the body part shows that API struggles to keep up with the lower frame rate of the iPhone.

Related Work

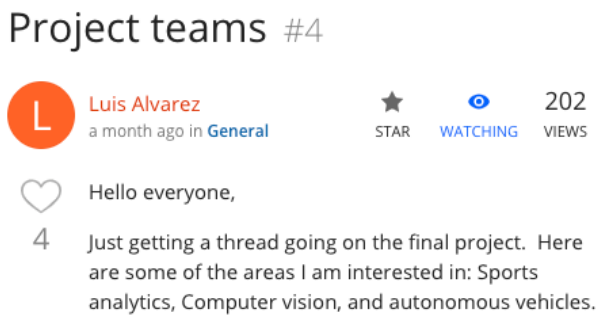
In order to stich GC API annotations with the video footage I used open-source software Video Intelligence Player that parses the json file using regular expressions. The software is fairly simple to use and only takes three commands to setup and use. However, I discovered a bug, or use case, that prevents the software from reading the json file. I believe that the application cannot successfully parse data that is generated by the backend for the REST/CL data pipeline. I tested the application with two sets of data one that was generated with the GC REST/CL method and one that was generated by the GC Python backend method. The data from the python method works every time while the other does not. I filed a ticket on the GitHub repository called *Stick Figure Not Visible* and closed it since I later discovered that the issue was maybe an unplanned use case scenario.

In the early stages of the project, I researched iPhones native approach to the computer vision task of labeling video data. With this early exploration and preliminary work, I found a way to use Apples application of a machine learning model in the Swift Language. I watched a video from a World-Wide Developer Conference and worked my way through tutorials on how to implement a computer vision model using Apples Software Development Kit. I was able to obtain a live stick figure overlay on live video footage on my iPhone. However, I pivoted away from this approach since I am not very familiar with the Swift language.

While trying to complete this project programmatically using python the documentation explained that a variable called *name* is needed. However, this detail lacked an explanation and details. Much of the API reference materials reference a name-string that can be used to retrieve the parsed data via a python coroutine. After finding out more about coroutines and stumbling on not being able to find the string-name for quite some time I regrouped by researching how to perform this operation using the Command Line. From here I was able to find that the project name-string is returned by the videos:annotate method and in my case it was configured with information I knew except for the unique 24 digit operation id.

Team Organization

This project was completed by one person in the span of about 3 weeks. Originally, I was in a group the fell apart when the other member expressed concerns about being able to contribute to the team. Thanks to an open and honest conversation the previous group dissolved and I formed my own group. However, it is noted that I started the search for a team member early in the course. Thanks to this early start the dissolution of the first team allowed for ample time to regroup.



Software & Development Tools

The project mainly uses the Google Cloud Software Development Kit to access its Video Intelligence API. Also, this project leverages the Python libraries Pandas and Math to process raw JSON file into a palatable format. The google documentation has four Video Intelligent APIs. Version one is the application interfaced used for this project, but version beta version 2 and version 3 are also explored during troubleshooting and the first iterations of the project.

The sensor used is a camera from an iPhone 11 running software version 14.6.
Computer: iMac 2017 running macOS Big Sur version 11.4
iMovie version 2.3.2
Video Intelligence Player
Python 3
Pandas 1.3.0
Numpy 1.21.1
Jupyter Notebook
Google Cloud's VideoIntelligence_v1 API
REST/Command Line Interface
Mac Terminal Z-shell

List of Mile Stones

July 16-18: Collect Golf Swing Data
July 19-22 Setup environment to use GC API and Parse raw data
July 23-26 Build posture skeleton, build data frames, build visualizations
July 27-31 Build video presentation, GitHub repo, build readme.md, submit
Aug 4 Presentation

Results and Discussion

After collecting the annotations results the data is processed with Jupyter Notebook. Specifically, the data is analyzed by mapping two vectors originating from the left elbow. One vector points to the left shoulder and the other to the left wrist. The angle of the elbow is calculated using high school geometry. The notebook contains code that pulls out the 2D coordinates for each point on the arm during a swing and along with time series data to a CSV file for further analysis. The code can be refactored to calculate other interesting angles, such as the position of legs, hips, eyes, and nose. However, this analysis focuses on the left arm and elbow during the takeoff and down swing. The analysis compares a pro golfer and me a weekend warrior.

The file Luis_best_swing.csv contains the left_elbow_angle for a 200 yard drive that can be considered the best drive out of 50. The golf swing can be broken down into two phases. The first phase is the takeaway and is the wind up of the golf club. At address my left arm is at its straightest and is about 187 degrees at the elbow and is basically straight. Midway through the takeaway the left arm starts to bend at the elbow significantly with an angle of about 201 degrees. At the top of the swing just before the downswing phase my left arm is about 263 degrees. The downswing is the second phase and is fastest part of the swing taking place in tenths of a second. A good high-speed camera captures more detail. Right before contact the left elbow is at a respectable 207 degrees and ends in 197 degrees just past the point of contact.

At most driving ranges there are markers that allow for the estimation of the distance the golf ball travels. In this particular day, my average swing was about 140 yards and my best driving being about 200 yards. In order to understand how to hit the ball further I analyzed some high-speed footage of professional golfer Tiger Woods. At address both Tiger and I have our arms in similar positions with our left elbow angle close to 190 degrees. However, that is where the similarities end. Once Tiger starts his takeaway it is very evident that his left elbow angle is constantly close to 190 degrees. In contrast, my left arm bends at angles greater than 200 degrees. One startling surprise was observed midway through Tiger's takeaway.

Tiger Vs Luis

Tiger Left elbow angle: 135 deg

Luis Left elbow angle: 201 deg

At this point the pro golfer's arms exhibit excellent flexibility. The left wrist drops below the left elbow and shoulder almost as if the golfer was double jointed and registers a 135 degree left elbow! At this stage my arms are at about a 201 degree angle. My left arm forms a magnitude vector that opens up while Tiger's opens down. This flexibility along with control and power are likely why the pro can hit the ball on average 290 yards.

Conclusions

Based on the annotated video, Tiger keeps his left elbow almost straight if not locked through his swing. Right before contact, Tiger's arms stay at a steady 185 degrees. The main takeaway from Tiger's swing is that he is able to generate energy by keeping his left elbow as straight as possible and even has a period where he forms an acute angle with this joint before he unwinds. This swing can aptly be described as a pendulum that starts in his shoulder and is extended by the head of the graphite club. Furthermore, in an attempt to emulate this golfer, I try to keep my left elbow as straight as possible while slowly progressing through the swing. Needless to say, the task is very challenging and gives the sensation of almost falling over. I think I will leave the long ball to the pros.

This approach of processing data can be successful with better quality video input. The swing data after the downswing action is not as reliable from the iPhone as the high-speed YouTube video. As a result of this limitation the data parsed for Luis' has gaps and erratic elbow angles during the downswing. However, this project shows that it is still possible to get good data from a swing using an off the shelf machine learning model.

Future Work

Given more time, I believe it is possible to write a python program to automate video annotation using the Google Cloud's VideoIntelligence_v1 API instead of using the batch approach with the POST and GET commands. Also, an interesting evolution of this project is to use the beta version of this API to annotate video in real time to get instant feedback on golf swing. In order to build upon the proof of concept in this project the GC API will need to be mastered.

A secondary goal of the project is to use the video footage to track the club head in order to calculate the swing speed. This setup is possible by setting up the camera on a wide angle instead of portrait mode in order to keep the club head in frame. Another interesting iteration of this project can include footage from two cameras. One from the side and another from the back point of view. All while analyzing multiple angles of the body at once and playing them back to a coach on in real time.

References

"Authenticating to the API." Google, https://cloud.google.com/storage/docs/authentication, Accessed 23 July 2021.

"Build an Action Classifier with Create ML." Apple WWDC2020, https://developer.apple.com/videos/play/wwdc2020/10043, Accessed 7 July 2021.

"Coroutines and Tasks." Python Org, https://docs.python.org/3/library/asyncio-task.html. Accessed 22 July 2021.

"Creating Storage Buckets." Google, https://cloud.google.com/storage/docs/creating-buckets, Accessed 24 July 2021.

"Detect People." Google, https://cloud.google.com/video-intelligence/docs/people-detection, Accessed 20 July 2021.

O' Beirne, William. "Video-Intelligence-Player." GitHub, https://github.com/wbobeirne/video-intelligence-player. Accessed 17 July 2021.

"Explore the Action & Vision app." Apple WWDC2020, https://developer.apple.com/videos/play/wwdc2020/10099/, Accessed 5 July 2021.

Markowitz, Dale. "Can AI make me a better athlete? | Using machine learning to analyze penalty kicks." YouTube, https://www.youtube.com/watch?v=yLrOy2Xedgk, Accessed 14 July 2021.

Markowitz, Dale. "Sports_AI_Analysis.ipynb." GitHub https://github.com/google/making_with_ml/tree/master/sports_ai, Accessed 17 July 2021.

Murugavel, Manivannan. "Find the Angle Between Three Points From 2D Using Python." Medium, https://manivannan-ai.medium.com/find-the-angle-between-three-points-from-2d-using-python-348c513e2cd, Accessed 22 July

"Service Accounts." Google, https://cloud.google.com/iam/docs/service-accounts, Accessed 21 July 2021

"Services for Google Cloud Videointelligence v1 API." Google, https://googleapis.dev/python/videointelligence/latest/videointelligence_v1/video_intelligence_service.html. Accessed 22 July 20221.

"Set up the Vision API." Google, https://cloud.google.com/vision/docs/setup. Accessed 18 July 2021.