Computer Vision Based Indoor Rock Climbing Analysis

SARAH EKAIREB, UCSD
MOHAMMAD ALI KHAN, UCSD
PREM PATHURI, UCSD
PRIYANKA HARESH BHATIA, UCSD
RIPUNJAY SHARMA, UCSD
NEHA MANJUNATH-MURKAL, UCSD

As personal fitness has risen in popularity, so too has the use of wearable fitness devices. There exist products for many sports, but few for Indoor Rock Climbing. Current products which provide post-climb reports utilize specialized hardware equipped with sensors. As such, they are limited in the types of information they can gather. We develop a novel tool which makes use of Computer Vision and Machine Learning to analyze video footage of the climber and produce post-climb reports that convey information which cannot be ascertained through sensor-only methods, such as the percentage of route completed and number of moves taken. To do so, we identify climb holds using Object Detection models such as YOLO, as well as the position of the climber using Keypoint Detection APIs. Our current approach produces post climb reports with an average RMSPE of 0.266 and we highlight specific areas of improvement to bolster the performance. Such a tool will provide a simple way for rock climbers to receive feedback on their climbs, with the use of a standard cell-phone camera.

1 INTRODUCTION

Rock climbing is a relatively modern sport, finding its roots in Yosemite Valley, California. Since then, rock climbing has evolved to a popular sport finding its way into the Olympics for the first time in Tokyo 2020. Since rock climbing is a solo sport, there is a need for methods of feedback and analysis of climbs. For many other solo sports, including running and cycling, there exist easy to use feedback reports that include information such as distance, speed, etc. For rock climbing, similar products require the use of wearable fitness devices to give feedback on the climbs. However, these devices are limited in their ability because they are not using any vision-based information. Hence, we have developed a tool for rock climbers that analyzes a video footage of the climber and produces a post-climb report.

Over the course of the quarter, our team has developed the Rock Climbing Coach. The aim of this project is to create software which can analyze a video of a person climbing an indoor rock-climbing wall and produce reports which summarize the climb. This tool will help both new and experienced climbers better understand their climbs. The future vision is to have a system set up in a rock climbing gym giving feedback to climbers after every climb. This would help beginners to get off the ground, and help experienced climbers reach the next level of improvement.

For this project, we have identified and broken up the project into two main components: hold detection and pose estimation. The goal of hold detection is to identify and localize the various holds on the climbing wall; whereas pose estimation's aim is to infer the position and limb configuration of the climber. The aggregation of both pieces of information allows us to produce metrics such as the percentage of the climb completed, total distance moved, number of moves, total time elapsed, move validity, and hold validity.

Authors' addresses: Sarah Ekaireb, sekaireb@ucsd.edu, UCSD; Mohammad Ali Khan, makhan@ucsd.edu, UCSD; Prem Pathuri, ppathuri@ucsd.edu, UCSD; Priyanka Haresh Bhatia, pbhatia@ucsd.edu, UCSD; Ripunjay Sharma, r4sharma@ucsd.edu, UCSD; Neha Manjunath-Murkal, nmanjunathmurkal@ucsd.edu, UCSD.

. XXXX-XXXX//6-ART
https://doi.org/

2 BACKGROUND

There are three types of indoor rock climbing: bouldering, top roping, and lead roping. For this project we focus on making our product suitable for bouldering. Bouldering is a type of rock climbing in which the routes are much closer to the ground, and no ropes or harnesses are needed [8]. We chose to focus on bouldering for this project because the short walls would allow us to have the entire route in the frame during a climb. In addition, there would be no ropes obstructing the view of the holds.

The rules are simple for bouldering [3]. The climber must start with the marked starting holds with all limbs on the wall (no parts of their body touching the ground). They must stay on the correct route color, which is determined by their starting holds and must have a controlled ending, with both hands on the marked ending holds. A single climb attempt starts once the climber is in the starting position with no limbs on the floor. A single climb attempt ends when the climber retains control of the ending holds (typically meaning they hold the position for a second), or when they fall off the wall.

There is some debate over what constitutes a rock climbing move. Some consider a shift in weight or complex foot work to be moves, while others consider a move to be any movement that bring the climbers body closer to the finish. So, for this project, we must define what positions and moves are. We define a position as a set of holds and a move as a transition from one position to another. We consider moving into the start position to be the first move. A hold is considered valid if it is of the same color as the starting hold chosen by the climber. And, we consider a move valid if the climber moves onto no new invalid holds.

3 RELATED WORK

Works which are most related to this project also address metric generation from rock climbing data. However, almost all methods currently make use of wearable devices which are equipped with sensors [9]. There exist many commercial products which address the exact problem our group aims to solve, however none of them make use of only a single camera system.

Our group is not the first to attempt a vision-based software project for Rock Climbing. There are numerous groups who have done significant work in this area. We describe a few such works below.

Many groups have attempted to create augmented-reality rock climbing games. Similar to the work of [10], these works require accurate hold and human detection. Crews uses an Xbox Kinect as their primary camera system and performs post-processing of recordings to detect holds and humans in the videos. Their work is completely computer vision based and uses OpenCV to detect objects on the rock wall as well as humans within the video.

Another work [6] aimed to display a suggested route for a climber to take on a new rock wall. This was a task our group had originally planned to address, but decided it was out of the scope of the current project. Despite this, our computer-vision based hold detection method which makes use of color information is based on Wei's work. In particular, Wei groups pixels by color in RGB colorspace, in order to separate out specific colors. Since rock holds are colored, this method allows one to easily detect rock holds. We experiment with a similar method, but group pixels in HSV colorspace rather than RGB. We find that colors are more easily separable in HSV [12].

Another method to separate pixels of an image based upon color is through color quantization. Color quantization reduces the number of colors in an image such that pixels of the same color are grouped together. Once images are quantized, groups or clusters of pixels can be found. This follows the method described in [2]. Once an image is properly quantized, it is much easier to use clustering methods such as K-Means clustering to identify clusters of pixel colors. These clusters would contain all of the pixels which belong to a similar color route on a rock wall.

Despite addressing a different task, the work of Csukas [5] closely resembles part of our approach to producing a post-climb report. Their work detects and localizes holds on a rock wall using Computer Vision techniques

such as Canny Edge Detection [4] and openCV's blob detection. Moreover, our work in color identification is inspired by their approach. In their work, Csukas identifies discrete colors by creating a histogram of the pixel values of an image of a wall. The histogram bins effectively quantize the image. We mimic this approach by defining specific ranges of colors that quantize the image more effectively. These ranges were found through extensive analysis. We describe this in Section 4.

Another group has also addressed the task of identifying route difficulties, yet they make use of Graph Neural Networks (GNNs) to do so. In particular [13] addresses a specific type of rock climbing wall called the MoonBoard. The MoonBoard is a customizable training wall which many indoor rock climbing gyms have for their members to use. Their work shows the effectiveness of GNNs in analyzing climbing routes.

While the aforementioned works address similar tasks, they do not attempt to combine information with human actions to produce post-climb reports. Human action recognition (HAR) has become a field of its own within the Machine Learning and Computer Vision communities. Moreover, HAR refers to the identification and classification of human movement to determine what type of action is being performed. There has been much work done in this area [11], and almost all works make use of pose-estimation to capture the human motion.

4 METHODS

4.1 Data Collection and Preprocessing

4.1.1 Data Collection. In order to develop our tool, we needed to collect videos of climbers bouldering. Our first step for this project was to go to rock climbing gyms and collect as much data as possible. In total we collected 30 videos from the Mesa Rim rock climbing gym, in the modern bouldering area. We chose to collect videos here because the walls were all light grey and the holds were vibrant, making hold detection a lot easier. We used 14 of the videos to develop our models, and the remaining 16 videos as our test set.

For the test set, we made sure to have a lot of variations in order to test our climb report metrics. We recorded on a total of 4 different routes, each with a slightly different angle. We started recording the videos before the climber enters the frame, and stop after the climber exits the frame. The climbs range from being 30 percent complete to 100 percent complete. And, the validity ranges from being perfect to the climber using random holds for the entire climb. We also included a few edge cases, including having the climber touch an incorrect hold before starting the climb, and having a person walk across the frame of the video.

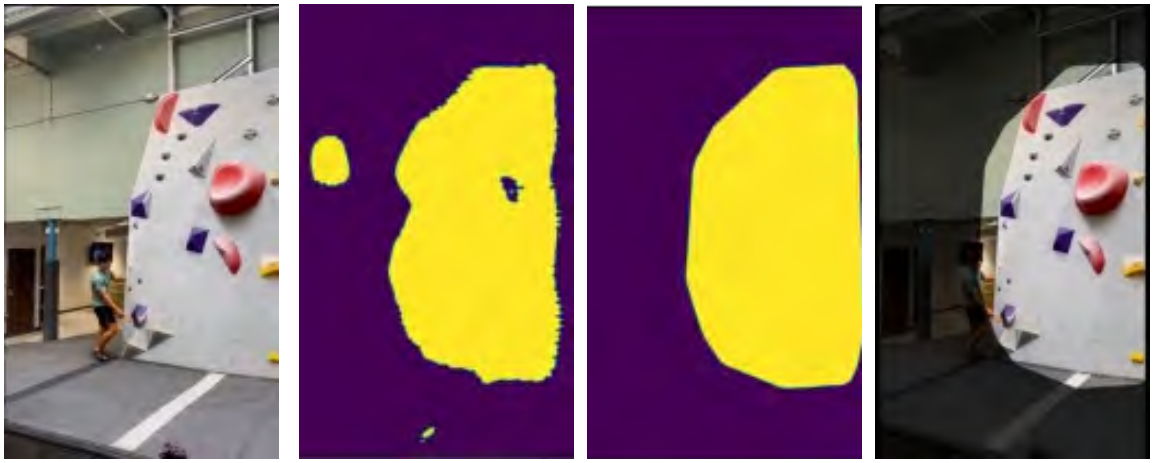
As we continued with our project, we also collected 10 videos of climbers from the UCSD climbing gym and 6 videos of climbers from YouTube. The walls in these videos are varied in colors, all different from our original dataset. We show some examples in Figure 1.

4.2 Hold Detection

4.2.1 Wall Segmentation. Our initial experiments with hold detection using both computer vision and machine learning based methods suffered from noise in the background which wasn't a part of the rock climbing wall. To get rid of this issue we thought using wall segmentation masks, which given an input frame would output a mask segmenting out the wall from the background. For the image 2a, this gives us the mask 2b which still has some noise due to inefficiency of the machine learning model. To further improve this we detect the largest contour in the ML mask and fit a convex hull on it. This gives us a much better looking mask as in 2c. Now we overlay this mask on the original image to get rid of the background as shown in 2d.

(a) Mesa Rim wall (b) UCSD Wall

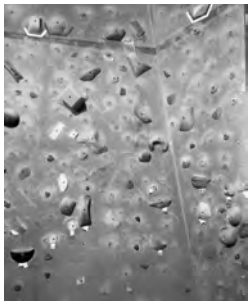
Fig. 1. Wall Types



(a) Raw Image (b) ML mask (c) Refined ML mask (d) Masked Raw image

Fig. 2. Wall segmentation

4.2.2 Computer Vision for Hold Detection. We first attempted to use classic computer vision methods for hold detection. One approach consisted of a combination of Canny Edge Detection and openCV's blob detection as seen in Figure 4. We found this method to work well on clean walls where the holds were relatively the same size, however this is not the case for most rock climbing walls. We ran into some issues with the walls being very chalky (as seen in Figure 3) which made it difficult for even a human to differentiate holds from the walls. This indicated that in order for this method to work, we would need to do a lot of preprocessing. Lastly, we also ran into some issues with having difficulty detecting blobs of all shapes.



(a) Raw Image

(b) Edges

Fig. 3. Edge Detection on chalky wall

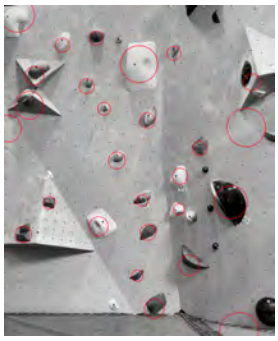


Fig. 4. Blob detection

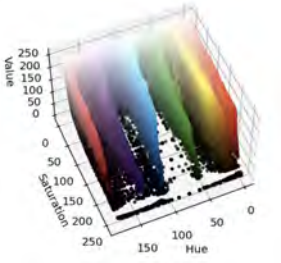
Fig. 5. Color range analysis

One drawback of using Canny Edge detection along with openCV's blob detection is that these methods work best on grayscale images. Most rock climbing walls have colored holds which indicate the color of the route. By converting images to grayscale to make use of this methods, all color information is lost in the process.

This led us to try color range analysis. For this method we create a bin for each color (red, blue, green, etc.) and for each color-bin, we find all of the pixels in the image that fall in that bin, based upon the pixels value. We then create a mask over the image containing only pixels which fell into that particular color-bin. Next, we de-noise the mask and use it to detect the holds of the specific color. This method seemed to work well on some walls, but unfortunately, could not detect holds that were a similar color as the wall. The inability to detect black or grey holds on a gray wall is illustrated in Figure 5.

We then ran some experiments with using color frequency analysis for color grouping. This would be more adaptive than having set bins. The results seem promising, but would require more research and tuning in order to implement and integrate this method. For this method, we analyze the distribution of pixel colors in the image; this can be seen in Figure 6. From the ground truth image, we are able to plot the distribution of pixel color and determine where the peaks are. As seen in the figure, there is separation between each color in the pixel graph, allowing us to easily pick out the specific color clumps.

(a) Ground Truth



(b) Pixel Graph

(c) Blue holds

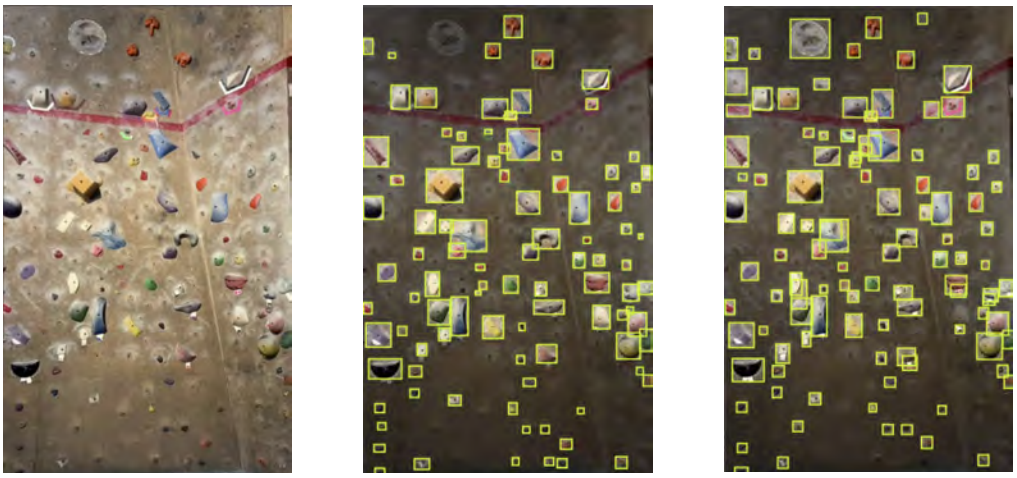
Fig. 6. Color Frequency Analysis

In the end, we decided to shift from computer vision based methods to machine learning in hopes of getting better results with minimal preprocessing and tuning. We describe this in the following section.

4.2.3 Neural Network Model for Hold Detection. We ended up using a machine learning approach for hold detection. For this, we trained a neural network model to detect holds and segment them out with bounding boxes. To do this, we made use of Roboflow, which is a service for managing Object Detection and Computer Vision datasets. It allows users to annotate images to create ground-truth labels which are used in training neural networks.

In total we annotated 53 images of rock climbing walls in order to create a dataset for the model to be trained on. We allocated 46 images for the training dataset and 7 images for the testing dataset. These images were taken from our initial data collection step. We attempted to include images of many different types of rock climbing walls in order to make our model more robust. Due to the limited size of the dataset, we used augmentations to triple our datasets size. Some of the augmentations applied include horizontal and vertical flips, hue changes, adding noise, etc.

Using a neural network proved to be much more effective than the computer vision based approaches we had tried. On our testing set, we achieved a Mean Average Precision (mAP) score of 96.3 percent. As seen in Figure 7, the model's predictions are very close to the ground truth annotations on the UCSD rock climbing wall.



(a) Raw Image (b) Ground Truth Annotations (c) Model Predictions

Fig. 7. Model Predictions vs. Ground Truth on UCSD rock climbing wall (test set image)

4.2.4 Color Detection. We also used a machine learning approach for color detection. In order to detect the color of a hold, we trained a neural network to perform multi-class classification of holds. We trained the classifier to recognize 11 classes: black, blue, gray, green, orange, pink, purple, red, white, yellow, and unknown. In total we annotated 2759 holds; we used 2041 hold images in the training dataset and 718 hold images in the testing dataset. Our neural network achieved a 99.3 percent accuracy on the testing dataset.

Although this method performed very well, there are some issues with using a classifier to group the color holds. For instance, when a path color is between two classes, the model would predict some of the holds one color, and some another color. In some cases, the model would classify two different paths as the same color, if the colors were close enough or between colors. This can be seen in Figure 8c where both the green and turquoise holds are classified as green, even though there should be two separate routes detected.

This limitation can be solved by using the color frequency analysis method (mentioned above) to group the holds, rather than using a system of classification. Previously, we were attempting to use this method on the

(a) Clean Wall (b) Dirty Wall (c) Turquoise holds

Fig. 8. Color Detection

entire image, however, now that we have a working hold detection model, we could just use the color frequency analysis within the areas of the image that are contained in a bounding box. Essentially, this would pick out the clusters of colors within the bounding boxes.

4.3 Pose Estimation

4.3.1 MediaPipe Framework for outputting Climber's co-ordinates: To achieve our minimum viable product, which is to output the coordinates for hands, legs and hips at specific instances of time, we researched with various pose estimation techniques [7] and finalized Mediapipe [1] to compute coordinates for different body parts of the climber. While processing each frame, we use the framework to compute the coordinates and we tried two algorithms (1)Cosine similarity and (2)Angular velocity (for deciding on the time instances) to determine if the current frame is a significant frame.

Mediapipe Pose [1] is a ML framework for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation mask on the whole body from RGB video frames as shown in **Figure 9**. Each landmark consists of x, y, z, and visibility values, where x and y landmark coordinates are normalized to [0.0, 1.0] by the image width and height respectively, z represents the landmark depth with the depth at the midpoint of hips being the origin, and visibility is a value in [0.0, 1.0] indicating the likelihood of the landmark being visible (not occluded) in the image.

Out of these 33 landmark coordinates, we store the coordinates for: left hand, right hand, left leg, right leg, left hip and right hip as desired by the different features of the climb report aspects. We store these coordinates in a dictionary data-set as shown below:

dict_coordinates = { 'left_hand': $[(x_i, y_i)]$, 'right_hand': $[(x_i, y_i)]$, 'left_leg': $[(x_i, y_i)]$, 'right_leg': $[(x_i, y_i)]$, 'left_hip': $[(x_i, y_i)]$, 'right_hip': $[(x_i, y_i)]$ }

where, (x_i, y_i) represents x, y landmark coordinates at the i^{th} time instance.

(a) Plotting the coordinates on climber

```
----------------------
Processing a new frame
Landmarks: landmark {
  x: 0.5105810165405273
  y: 0.4993339776992798
  z: 0.05962387099862099
  visibility: 0.9999697208404541
}
```

(b) MediaPipe pose landmarks

Fig. 9. Get coordinates for hands, legs, and hips positions

4.3.2 Cosine Similarity for Significant Frames Detection: We propose to use the pose estimation information to provide preliminary feedback to climber. The primary step in the pose estimation task is to determine the accurate time instances to compute coordinates of the climber. Suppose we choose to output the coordinates according to a certain threshold (say every 5 secs in the input video) we could miss recording the coordinates corresponding to multiple moves that the climber potentially makes within those 5 seconds. Alternatively, if we output the coordinates for every frame, the eventual calculations of the climb report aspects would become memory and compute intensive. Thus, to define the time instances, we decide to output the coordinates corresponding to each move made by the climber.

To properly compute the time instances of significant frames (i.e., the frames in the video at which the climber makes a definitive move), we implement an algorithm that compares the Cosine similarity between successive frames. For the first effective frame (i.e., the frame at which the climber starts a climb), we compute the coordinates for limbs of the climber using Mediapipe Pose framework [1]. For the subsequent frames we first compute the Cosine similarity between the coordinates of the current frame and the previous significant frame frame, and store the coordinates of the current frame (i.e., define a time instance at the current frame) only if its coordinates are dissimilar to those of the previous frame, by a fine-tuned threshold. The dissimilarity of the current frame from the previous indicates that the climber has made a move some significant motion. Using this algorithm, we iterate over all the frames and store the coordinates only for the significant frames which make up our time instances.

To evaluate the efficiency of the proposed Cosine similarity algorithm, we compute the following metrics: accuracy score, precision, recall and f1-score. As defined in the context of Machine Learning, accuracy is the ratio of correct predictions out of all predictions made by the algorithm, Precision is the ratio of true positives over the sum of false positives and true negatives, Recall is the ratio of correctly predicted outcomes to all predictions, and F1-score is the harmonic mean of Precision and Recall. It is a metric (ranging from 0 to 1) that takes a cumulative account of both Precision and Recall, and thus higher f1-scores indicate a better performing algorithm. For computing the raw ground truth labels, we first record the time instances manually (in seconds) at which the climber makes a move in a video. We then record the time instances corresponding to the significant frames (i.e., the predictions) computed by the Cosine similarity algorithm. Finally, we use both the lists: raw_ ground_truth _labels and raw_predictions, comparing them to compute the accuracy score, precision, recall and f1-score. It is

observed that all the 16 test video clips have an f1-score between 0.65 to 0.85 (as shown in Figure 10a), with some outliers, which proves that the proposed algorithm exhibits low false positives and false negatives. These outliers (for instance, in Figure 10b corresponding to the reported metrics for clip 3) exist due to incorrect landmarks produced by the Mediapipe pose framework, usually observed when the body of the climber overshadows the limbs.

Video	Threshold	Accuracy	Precision	Recall	F1_Score
1	0.9999	0.666666	0.666666	0.888888	0.761905
2	0.9999	0.5	0.555556	0.833333	0.666666
3	0.9999	0.375	0	0	0
4	0.9999	0.6	0.625	0.8333333	0.714286
5	0.9999	0.571428	0.8333333	0.5	0.625
6	0.9999	0.714286	0.727272	0.888889	0.8
7	0.9999	0.777778	0.857143	0.857143	0.857143
8	0.9999	0.652174	0.666667	0.857143	0.75
9	0.9999	0.5625	0.777778	0.583333	0.666667
10	0.9999	0.375	0.5	0.333333	0.4
11	0.9999	0.727272	0.714286	0.833333	0.769231
12	0.9999	0.7	0.733333	0.846154	0.785714
13	0.9999	0.6	0.647059	0.846154	0.733333
14	0.9999	0.714286	0.692308	1	0.818182
15	0.9999	0.62069	0.5625	0.692308	0.62069
16	0.9999	0.666667	0.615385	0.8	0.695652

(a) Evaluation of Cosine similarity algorithm

(b) MediaPipe landmark fault (body overshadows limbs)

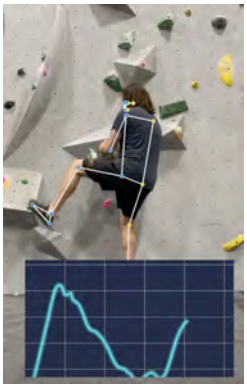
Fig. 10. Efficiency of the proposed Cosine similarity algorithm

4.3.3 Motion Graph for Significant Frames Detection:: The main limitation(s) of the cosine similarity approach was that fine-tuning threshold for each video had to be done individually and yet the algorithm ended-up generating a lot more significant frames than expected. Secondly, we mainly aimed to mark a frame as significant when the position of the climber changed from one set of holds to the next set but cosine similarity approach resulted in marking frames where the climber positions were changing drastically in any condition (eg. when climber's hands moved over some holds, when climber's body moved without changing position with respect to holds used). Due to these limitations in the cosine similarity approach, many climb report features generated were inaccurate.

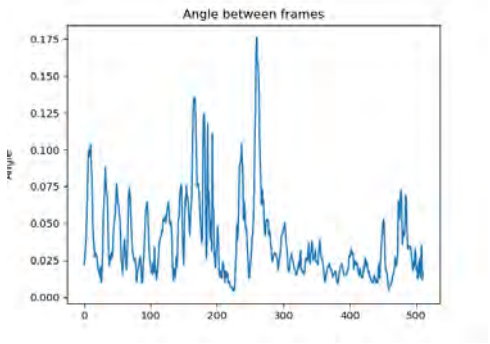
To address the issues faced above and improve robustness, we implemented an extension to the cosine similarity by considering angular velocity to detect significant frames. Methodology is described as - We consider i^{th} frame and $i - step^{th}$ frame, where step is a hyper parameter. Next, we calculate the angle between these frames using cosine similarity method described in the previous subsection. This process of angle calculation is done for i in [step, length of the video] storing each angle in a list resulting in a motion array. Then perform a smoothing operation on the motion array using a savgol filter (in built in scipy) and find the time steps for minimas (troughs) in the motion graph. These time instances would provide significant frames.

The figure below shows the entire motion graph consisting angles for all the frames(518) for the video. Out of the 518 frames we were able to get 13 significant frames for this video which is a huge improvement compared to the Cosine similarity method.

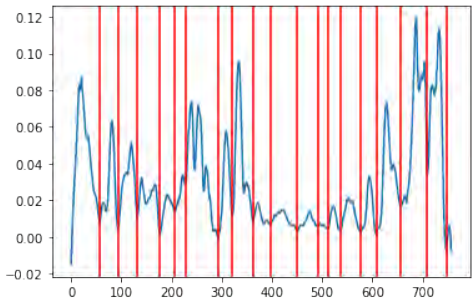
To visualize the angular velocity approach, we generated motion graphs for a video during run-time. The figure below depict the the motion graph for a climbing video which clearly shows spikes when the climber makes a move.



(a) Spike in graph for a move



(b) Angles detected for all the frames for the video



(c) Significant frames detected

Fig. 11. Motion graphs

The table below shows the comparison of significant frame detection using cosine similarity vs angular velocity.

Video	Total frames	Cosine similarity	Angular Velocity
1	459	351	14
2	282	202	9
3	495	383	13
4	350	280	9
5	506	188	15
6	403	312	10
7	305	192	9
8	728	534	20
9	507	356	16
10	849	395	23
11	289	174	8
12	518	377	13
13	618	459	18
14	387	274	9
15	939	584	25
16	553	324	17

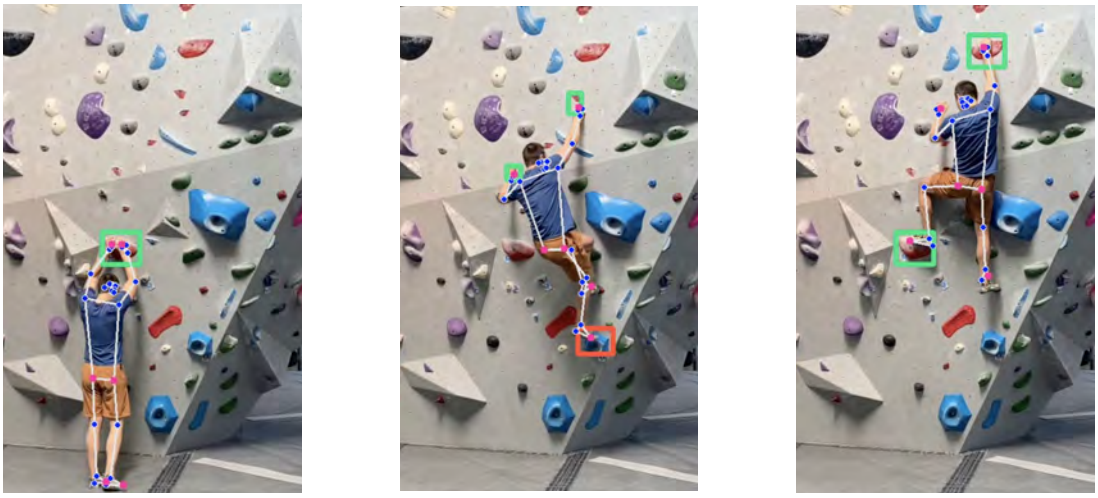
Fig. 12. Comparison of no. of significant frames detected by cosine similarity vs angular velocity

4.4 Climb Report Features

4.4.1 Percent Complete: The first feature of our climb report is the percentage of the route completed by the climber. In order to calculate this, we first identify which route the climber is taking by analyzing the most frequent color of all holds used within a climb. We then identify the lowest and highest holds of this color on the rock wall. Lastly, we find the frame of the video which contained the highest hand position, either right or left. Within this frame, we choose the position of the lower hand as the climber's final position. The calculation of the percentage complete is as simple as $\frac{p_f - p_l}{p_h - p_l}$, where p_f, p_l, p_h are the final position of the climber, the position of the route's lowest handhold and the position of the route's highest handhold.

4.4.2 Validity: There are a multiple different ways of evaluating the validity of a climb. Here we define two measurements of validity: hold validity and move validity. For hold validity, we calculate *# of unique valid holds used / # total unique holds used.* A hold is considered valid if it is the same color as the route color. For move validity, we calculate *# of valid moves / # total total moves.* We previously defined a move as a change in holds, and a valid move as a change in positions where all new holds used are considered valid.

In figure 13, we show a demonstration of when the climber is a valid and invalid position.



(a) Red route is taken (b) Foot incorrectly on blue hold (c) Using correct hold group

Fig. 13. Move validity (Red box: incorrect, green box: correct)

In order to obtain the route color used for validity, we choose the color that the climbers hands and feet are on for the longest period of time in the video. Generally, in rock climbing, the color route is determined by the first hold used by the climber. However, we found this to method to have a low accuracy due to issues with color classification and position estimation. As seen in Figure 14, the accuracy of using the most frequently hovered hold to predict the color of the route is 100 percent, whereas using the first hold used is 58 percent.

Prediction of route used color

NAME	Truth	First	Used	Hovered
clip1	purple	purple	purple	purple
clip2	purple	black	purple	purple
clip3	red	red	red	red
clip4	red	red	red	red
clip5	blue	blue	blue	blue
clip6	black	black	black	black
clip7	blue	black	blue	blue
clip8	red	red	red	red
clip9	green	orange	green	green
clip10	green	green	green	green
clip11	black	yellow	black	black
clip12	black	yellow	black	black
ACCURACY		0.58	1.0	1.0

Fig. 14. Training set accuracy of methods for determining route color

4.4.3 Distance center of climber moved overall: The aim of this aspect is to compute the overall distance moved by the climber (how much the center of mass is shifted). Thus, by being conscious of their center of mass when making a move, a climber can anticipate how large of a jump they must take to reach the nearest acceptable hold.

To compute the distance center of climber moved overall we use the hip coordinates calculated at different significant time instances in the climb video. We first calculate the difference between hip position from one point to another, i.e. we calculate all the distances for each consecutive jump and then sum-up these distances to get the overall distance center of the climber (hips) moved.

It is difficult to evaluate the computed distance center of the climber as we do not have the ground truth of the distance moved by the climber in appropriate metric. Hence, to visualize the hip movement in each consecutive jump we plot the distance vs time instance in the graph as shown in Figure 15 . This graph helps us to understand the nature of the jump taken by the climber at each consecutive move.

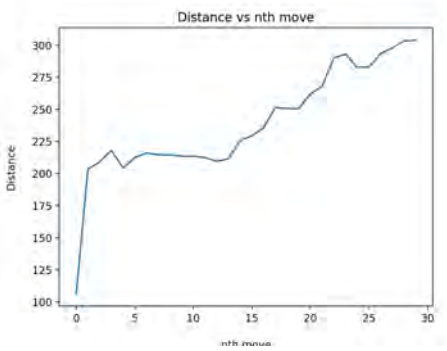


Fig. 15. Distance moved (in pixels) per move analysis

4.4.4 Number of moves taken: The aim of this aspect is to compute the total number of moves taken by the climber during the entire climb, by determining the change in coordinates of hands, legs and hips at each interval and incrementing the count.

To compute the number of moves taken we integrate the outputs of hold detection with functionality with pose estimation capabilities. This integration provide us with the ability to detect a particular move taken by the climber by knowing which specific holds are being used at any given point in time during the video.

The algorithm uses the bounding boxes of the holds and the pixel coordinates of a joint to determine if a joint is "on" a hold. It does so by calculating whether the pixel coordinate of the joint is within the boundaries imposed by the hold's bounding box. In order to avoid false detections of hold usage, we set a threshold of the number of consecutive frames that a joint must be within the bounding box to truly be considered "used". This is necessary as climbers often wave a limb near one hold, only to not use it as they were truly reaching for a different hold.

At each specific instance of time, the position of the climber and the holds used is highlighted by a bounding box as shown in the figure. We compute number of moves by tracking the climber's position changes when a hold is used.

(a) Raw image

(b) Detected move image

Fig. 16. Move detection

We computed number moves based on both the significant frame detection algorithms discussed above. The table below shows the comparison of number of moves detection using cosine similarity vs angular velocity for frame significance respectively. As we can see, angular velocity approach using motion graph beats the cosine similarity approach for every video.

We found one major issue with our current method of identifying if a hold is in use. More specifically, the Mediapipe module we use for pose-estimation sometimes infers the joint position of a climber when their limb is occluded from view. This leads to our method detecting certain holds as in use when the climber is not actually using them. This is seen more clearly in Figure 18. As shown, the climber's right leg is hidden from view, but the Mediapipe solution *infers* the leg position, thus leading a new hold being detected as in use. Future work may remedy this by leveraging Mediapipe's `occluded` output which indicates that the limb is out of view.

Video	Ground truth	Cosine similarity	Angular Velocity
1	10	52	11
2	7	27	7
3	14	87	10
4	8	89	8
5	19	94	15
6	10	67	9
7	6	72	6
8	19	157	18
9	20	93	14
10	23	125	16
11	5	29	17
12	10	69	8
13	14	83	18
14	11	76	6
15	14	177	20
16	12	91	14

Fig. 17. Comparison of no. of moves output by cosine similarity vs angular velocity

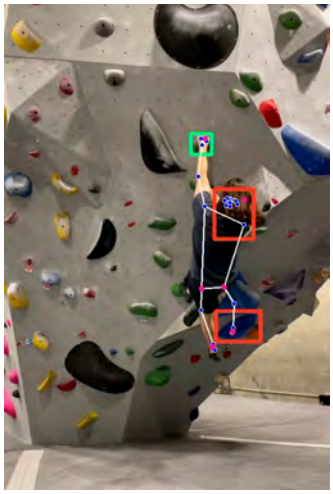


Fig. 18. Occluded Pose

4.4.5 Total time elapsed: The aim of this aspect is to to provide a statistic for determining the overall time taken by the climber for a climb. While a more sophisticated metric like time elapsed per move would help the climber better, overall time elapsed is a good starting point, aiding the climber in analyzing the gap between individual and ideal performance and optimizing for time. Many indoor rock climbing enthusiasts compete for the fastest time to the top of the climbing wall. Thus, motivated by the essence of the time aspect in the sport (especially if we consider speed climbing too), this feature aims to compute the total time taken by the climber for a climb (in seconds). The feature is not only a climb report aspect, but it also aids creation of a mask for computing the accuracy, precision, recall, and f1-score from raw_ground_truth_labels and raw_predictions for the Cosine

similarity pose estimation algorithm (discussed in section 4.3.1). Since the lists: raw_ground_truth_labels and raw_predictions (in the context of Cosine similarity algorithm) are of varying lengths, the evaluation metrics cannot be computed directly, and hence a mask (essentially a constant length array of size = total time elapsed + 1) is prepared such that each index i in the array corresponds to a time instance in the video and a value 1 at a time instance i represents that the coordinates have been stored for this time instance by the Cosine similarity algorithm.

Therefore, for preparing the mask to compute the evaluation metrics for the Cosine similarity algorithm (section 4.3.1) and for determining the overall time elapsed for a climb, we first record the ground truth labels, i.e., the time instances at which the climber starts and ends a climb. The feature then finds the first frame in the video such that the limbs of the climber are on some hold (and the body is off ground) and also finds the last frame in the video (for simplicity, we define the last frame to be the frame in which the position of hips of the climber is the highest). Using the computed first frame and the last frame, the feature then predicts the total time elapsed to be = $(end\_frame - start\_frame + 1)/fps$, where fps is the 'frames per seconds' recorded for the video using the OpenCV Computer Vision library. Finally, using the prediction and the ground truth label for the total time elapsed, we evaluate the error rate for the feature by calculating: $error\_rate = abs(total\_elapsed\_time\_ground\_truth - total\_time\_elapsed\_predicted)/total\_elapsed\_time\_ground\_truth$ and the accuracy for the feature by calculating: $accuracy = 1 - error\_rate$. We observe that all the 16 test video clips have an accuracy between 70% to 95% (as shown in Figure 19a), with a few outliers, which proves that the feature is able to efficiently identify the start and end frames in the videos. These outliers (for instance, Figure 19b) exist due to incorrect hand landmarks produced by the Mediapipe pose framework, due to which the hand coordinates do not fall over a hold, rendering the feature unable to compute the start frame in the video.

Video	Ground Truth	Prediction	Accuracy
1	11	14.2	0.70909
2	8	9.133333	0.858333
3	16	7.666667	0.479167
4	10	9.8	0.98
5	Mediapipe	Landmark	Error
6	13	13.366667	0.971795
7	9	8.533333	0.948148
8	19	22.666667	0.807018
9	Mediapipe	Landmark	Error
10	26	23.133333	0.889744
11	8	10.133333	0.733333
12	16	19.766667	0.764583
13	Mediapipe	Landmark	Error
14	12	13.8	0.85
15	21	28.066667	0.663492
16	14	20.1	0.564286

(a) Evaluation of total time elapsed

(b) MediaPipe landmark fault (inconsistency between hand landmark and hold)

Fig. 19. Efficiency of the predicted total time elapsed

5 RESULTS

In order to quantitatively evaluate our climb-report features, we had all of our team members annotate each video with their perceived ground-truth label. We communicated a common evaluation scheme to use such that the

annotations were consistent among the whole team. With these labels, we take the average for each climb-report feature and let that be the actual ground-truth label.

Next, we compare the climb-reports outputs for all videos to these ground-truths and compute RMSE across all videos for 5 of the 6 climb report metrics. We leave out total distance traveled from the evaluation as it was we could not come up with a labeling scheme for this metric. We show the results in Table 1.

Feature	RMSE	GT σ
% Complete	9.61	5.38
Time Elapsed (s)	5.06	1.34
Num Moves	2.71	0.667
Hold Validity %	23.7	2.61
Move Validity %	33.6	4.22

Table 1. Climb Report Performance

In the first column of the table we detail which climb-report feature is being evaluated. The second column shows the RMSE of outputs of this feature across all 16 videos in our dataset, when compared to the ground-truth derived from averaging all annotations. The third column denotes the standard deviation between the annotations from each team member (ie. GT σ).

Since the features do not share the same units, these RMSE metrics cannot be aggregated to compute a single performance metric for the entire climb-report. In order to account for this we use RMSPE, which is similar to RMSE, but calculates the percentage of the label which is achieved. RMSPE values closer to 0 are better, and they represent a form of “correctness". Our report’s **overall RMSPE is 0.266**.

6 MILESTONES

When we started on this project, our MVP was to create a piece of software that can take as input a video of someone climbing, and output the percentage climbed and the coordinates of the climber at each specific instance in time. We have not only successfully completed this task, but have also included outputting the number of moves, distance climbed, move validity, hold validity, and time elapsed. We have completed all of the milestones outlined in the milestone report.

7 CONCLUSION

We have created a computer vision based rock climbing tool to provide insightful post-climb analysis to climbers. Our software takes in a video of a person climbing, and outputs a climb report containing the following metrics: percentage of the climb completed, total distance moved, number of moves, total time elapsed, move validity, and hold validity. This tool allow both new and experienced rock climbers to better understand their climbs. We are able to produce these metrics with an RMSPE score of 0.266. For hold detection, we are able to obtain an mAP score of 96.3 percent, and an average color detection accuracy of 99.3 percent. For tracking the climbers moves, we achieve an RMSE of 2.71.

8 FUTURE PLAN

In the future, we plan to improve our existing implementation and make it more robust. This would involve obtaining a larger dataset of climbing videos, and doing some more research into methods of hold detection and significant frame estimation. We have currently identified a few areas of improvement. As mentioned previously,

we need to transition from our color classification method to a color frequency analysis method of color grouping. In addition, we need to fine-tune our logic for detecting when the climber is making a move.

We would also like to create an application with a nice user interface for our project. This would ideally be a phone app that a climber could use to analyze videos post-climb. In order to achieve this goal, we would need to work on improving the speed of our algorithms to be able to run it more efficiently on mobile devices.

Once we are able to improve our existing implementation and create an user application, we could explore adding additional features to the climb report. One possible additional features is move difficulty analysis. This would involve a better understanding of the climbers movements, and of the difficulty of the holds.

REFERENCES

[1] 2020. *MediaPipe Pose Estimation.* https://google.github.io/mediapipe/solutions/pose

[2] Rosebrock Andrew. 2014. Color Quantization with OpenCV using K-Means Clustering. (2014).

[3] Bouldering Boss. 2022. *Bouldering Rules: using the Starting and Finishing Holds.* https://boulderingboss.com/bouldering-rules-using-the-starting-and-finishing-holds/

[4] John Canny. 1986. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8, 6 (1986), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

[5] Sean Csukas. 2016. Identification and Classification of Holds for a Rock Climbing Wall. (2016).

[6] Wei Eric. n.d.. *Indoor Rock Climbing Wall Route Displayer.* https://stacks.stanford.edu/file/druid:bf950qp8995/Wei.pdf

[7] Tomas Simon Shih-En Wei Yaadhav Raaj Hanbyul Joo Ginés Hidalgo, Zhe Cao and Yaser Sheikh. 2020. *OpenPose Estimation.* https://github.com/CMU-Perceptual-Computing-Lab/openpose

[8] indoorclimbing.com. 1998. *Climbing Competition Types and Formats.* https://www.indoorclimbing.com/comp_types.html

[9] Eleonora Mencarini, Chiara Leonardi, Alessandro Cappelletti, Davide Giovanelli, Antonella De Angeli, and Massimo Zancanaro. 2019. Co-designing wearable devices for sports: The case study of sport climbing. *International Journal of Human-Computer Studies* 124 (2019), 26–43. https://doi.org/10.1016/j.ijhcs.2018.10.005

[10] Crews Nick. 2016. *Kinect Climbing.* https://github.com/NickCrews/kinect-climbing

[11] Tansel Özyer, Duygu Selin Ak, and Reda Alhajj. 2021. Human action recognition approaches with video datasets—A survey. *Knowledge-Based Systems* 222 (2021), 106995.

[12] Gang Qian Shamik Sural and Sakti Pramanik. 2002. *Segmentation and histogram generation using the HSV color space for image retrieval.* https://www.cse.msu.edu/~pramanik/research/papers/2002Papers/icip.hsv.pdf

[13] Cheng-Hao Tai, Aaron Wu, and Rafael Hinojosa. 2020. Graph Neural Networks in Classifying Rock Climbing Difficulties. (2020).