A computer vision approach to indoor climbing analysis

Final Bachelor Dissertation

Marcel Closes i Pagan

Supervisor: **Coloma Ballester**

BSc. Computer Science Engineering

BSc. Mathematical Engineering in Data Science

2022-2023

A la Maria, a la meva família
i a amics i amigues

Acknowledgement note

I would like to express my sincerest gratitude to my supervisor Dr. Coloma Ballester for her support, guidance and expert knowledge on this dissertation. I would also like to thank UVE Solutions for allowing me to complete this dissertation while working and for the enormous knowledge provided that has helped me solve some of the problems faced while completing this project. I would specially like to acknowledge my undergrad classmates and now friends for all the support throughout these years that has led to this. Finally, I would like to thank my family for always being an unconditional pillar.

Abstract

[en] The field of sport analytics has experienced significant growth in recent years. Although traditional trackers have been widely used to record basic features, patterns and statistics for sports like distance, speed, elevation, and intervals, some sports require different analyses that are difficult to compute. The emergence of computer vision, utilizing both traditional and modern algorithms to emulate human vision and perception, has opened up a promising alternative for sports tracking and analytics. This approach has proven particularly effective for sports where traditional tracking methods have yielded imprecise results.

Indoor climbing is an example of a sport where tracking devices have not been as successful as in others, motivating this dissertation to investigate the use of computer vision capabilities to analyse indoor climbing videos. The system developed for this project employs deep learning-based object detection algorithms to identify climbers and their body parts, and to track their movements during the climbs. It also implements a new hold detection and segmentation model by fine-tuning a Mask-RCNN model (based on Faster-RCNN) pre-trained on the COCO dataset to identify climbing holds on the climbing gym wall. Thanks to a key-frame detection algorithm, the entire process is seamlessly integrated, enabling efficient computation of the final analysis inference.

The video analysis pipeline includes image preprocessing, object detection and segmentation, object matching, tracking, key-frame detection, and data extraction steps. The extracted data is then used to generate statistics related to the climbers' performance, such as the number of moves, the number of holds used, and the body position and "beta" in which these holds are used.

[ca] L'analítica esportiva ha experimentat un creixement significatiu en els darrers anys. Tot i que els aparells tradicionals s'han utilitzat àmpliament per enregistrar estadístiques bàsiques per a esports com la distància, la velocitat, l'altitud i els intervals, alguns esports requereixen anàlisis diferents que són difícils de dur a terme per aquests aparells. Gràcies a l'auge de la visió per computador en el camp, que intenta replicar la visió i la comprensió humana usant algoritmes tradicionals i moderns, ha sorgit una nova alternativa prometedora per al seguiment i l'anàlisi esportiva, especialment per als esports on els rastrejadors tradicionals no havien proporcionat resultats precisos.

L'escalada és un exemple d'esport on els dispositius de seguiment no han estat tan exitosos com en altres, motiu que ha motivat aquesta dissertació a investigar l'ús de les capacitats de la visió per computador per analitzar vídeos d'escalada. El sistema desenvolupat per a aquest projecte usa algorismes de detecció d'objectes basats en l'aprenentatge profund per identificar escaladors i les parts del cos, i per seguir els seus moviments durant l'escalada. També implementa un nou model de detecció i segmentació de preses ajustant un model Mask-RCNN (basat en Faster-RCNN) preentrenat en el conjunt de dades COCO per identificar les preses d'escalada a la paret del gimnàs d'escalada. Gràcies a un algorisme de detecció de fotogrames clau, tot el procés queda integrat, permetent una eficient computació dels càlculs finals.

El procés d'anàlisi de vídeo inclou passos de pre-processament d'imatges, detecció i segmentació d'objectes, emparellament d'objectes, seguiment, detecció de fotogrames clau i extracció de dades. Les dades extretes s'usen per generar estadístiques relacionades amb el rendiment dels escaladors, com el nombre de moviments, el nombre de preses utilitzades, i la posició del cos i el mètode ("beta") en què s'utilitzen aquestes preses.

Contents

List of Figures

List of Tables

1 Introduction

The field of sports analytics has witnessed a significant surge in growth in recent years, with the emergence of consumer gadgets (aimed at the needs of amateur runners, cyclists, etc.) or high-end equipment aimed at professionals belonging to top tier leagues and teams. However, certain sports, particularly those that involve a ball or team play, or those whose idea goes beyond physical quantities of the world (distance, speed, elevation, etc.), present unique challenges that are difficult to capture accurately using consumer-level equipment or traditional tracking gadgets based on physical sensors.

To address this issue, there has been a growing interest in the use of computer vision as an alternative approach to sports tracking and analytics. Computer vision aims to develop algorithms, systems and models that enable machines to understand and interpret visual information in a manner similar to humans. Specifically, it aims to extract meaningful information from digital images or videos, including object identification, movement detection and tracking, gesture recognition, and image semantic analysis.

One such sport that has gained popularity in recent years is indoor climbing [14], which poses significant challenges for traditional tracking devices due to the unique movement patterns and body positions involved. While distance, time, and elevation are useful general statistics for some sports, they fail to capture the nuances of indoor climbing performance [66, 12, 2, 27].

The motivation behind this research project is to explore the extent to which computer vision techniques can be leveraged to track and record more descriptive statistics for indoor climbing workouts. Specifically, the aim is to investigate how one could use computer vision-based approaches to extract statistics from a single video of a climber on a single indoor problem defined by some previous parameters (mainly the holds belonging to said problem). By doing so, this project seeks to contribute to the growing public body of knowledge on the use of computer vision in sports analytics and to pose both an interesting and fun challenge to the author, combining in one problem two of his passions.

2 Problem statement

As stated previously, the specific aim of this project is to investigate how computer vision-based approaches can be used to provide a more descriptive analysis of indoor climbing. Specifically, the project will focus on analysing a single video of a climber on a single indoor climbing problem. In order to do that, it is necessary to define what a more descriptive analysis constitutes.

An indoor climbing problem is usually compromised by a series of climbing holds affixed to a wall that are either grouped by colour and/or shape. The climber's objective is to reach the last hold (often identified by visual markers or by being positioned at the highest point) solely by employing holds that are part of the designated group or utilizing the wall itself. Additionally, climbers are usually required to start their climbing attempt with both hands and feet placed on visually indicated holds. If the climber falls or uses holds belonging to another problem, the attempt is considered unsuccessful. Figure 1 illustrates a discrete sequence of a climber start, some intermediate moves and finish an indoor climbing problem. The distinct movements, holds and body parts used to complete the climb is considered the descriptive solution to the climbing problem and is colloquially referred to as the "beta". It is worth noting that a single problem may have multiple solutions.

Figure 1: Sequence showing some time steps of a climber begin, solve and finish an indoor climbing problem defined by the pink holds

2.0.1 Research problem and objective

We define, therefore, a more descriptive analysis of a problem attempt to be the one that answers the following questions:

1. Which holds constitute the problem?
2. Did the climber finish the climb?
3. Which "beta" (movements) did the climber take to complete the problem?

By consequence, the objective of the study is to find computer vision-based solutions to answer the previously posed questions.

2.0.2 Scope and limitations

The project is bound by certain scope and limitations due to the complexity of the ambitious project, the time constraints and the interdependence of certain implementation steps. As a result, choices have been made to prioritise ready-to-use frameworks over state-of-the-art (SOTA) approaches for certain techniques and models associated to certain steps of the pipeline of the project.

In addition, to prevent the introduction of exponential complexity in some of the steps, a decision has been made to exclusively work with videos captured by stationary cameras. This eliminates the need for complex motion-based computer vision techniques, which can significantly increase the computational complexity of the system. Further constriction has been added by requiring manual input of initial frame and final frame of the climbing action, eliminating the need to create a solution for detecting the starting and finishing points of the activity.

2.0.3 Evaluation

To evaluate each individual specific step solution as well as the overall solution to answer the different problem statement questions a set of data samples has been collected . 10 validation images, 28 test images and over 2000 hold segmented masks have been defined to evaluate specifically 4.1. To obtain assessment metrics for 4.2, 4.3, 4.4 and 4.5, 4 static climbing videos have been collected. The need of setting up a static viewpoint for recording in generally crowded indoor gyms and the time constraints prevented from collecting more data points. Acknowledging the introduced bias by the limited data points, the 4 videos present enough variation to obtain an intuition on the performance of each implemented solution.

3 Literature review

The objective of this literature review section is to provide a brief overview of "industry standard" and "state-of-the-art" techniques and research on the various sub-fields of computer vision explored in the project. The latter segment of this section investigates the different approaches that have been employed to implement these techniques in the area of sports analytics and in the context of climbing.

3.0.1 Object segmentation

Object detection and segmentation are two essential tasks in computer vision. The goal of object detection is to locate and classify objects within an image, while segmentation involves partitioning an image into different regions based on their visual properties. Prior to the appearance of large-scale datasets and deep learning techniques, object detection was typically performed by objective- or energy-based optimisation methods (e.g., [47]) using traditional machine-learning algorithms such as Support Vector Machines (SVM) [31] or K-Nereast-Neighbours (KNN) [18] on top of hand-crafted feature extractors such as Scale-invariant feature transform (SIFT) [37] or Histogram of Oriented Gradients (HOG) [19]. The intrinsic feature learning capabilities of deep learning allowed problems such as object detection to be more easily implemented in specific domains, as it significantly reduces the need to manually define and select relevant features for the task in hand [45]. Two currently established methods based on deep-learning to perform object detection are RCNN [26] and YOLO [49, 50, 10, 68, 33].

RCNN (Region-based Convolutional Neural Network) is an object detection model that first proposes regions of interest in an image before applying a CNN to each region for classification and bounding box regression. Fast-RCNN [25] addresses the computational bottleneck of the two-stage approach by sharing the CNN computation across all proposed regions. Faster-RCNN [51] builds on top of Fast-RCNN by introducing a Region Proposal Network (RPN) that learns to propose regions of interest directly from the input image (instead of the search algorithm used in RCNN). Mask-RCNN [30], at the center of interest of this project, is a further evolution based on Faster-RCNN that adds a parallel branch (to the current bounding-box recognition one) that performs object mask segmentation.

YOLO (You Only Look Once) is an alternate approach to object detection that moves away from traditional classification two-stage network architec-

tures and proposes a single-stage network that is able to predict bounding boxes and class probabilities in one evaluation. Multiple versions of the architecture and model have been built over the originally proposed one, such as YOLOv3. YOLACT and YOLACT++ are two models that implement object mask segmentation based on YOLOv3. By the reasons stated in 2.0.2, this project explores the instance segmentation capabilities provided by the open-source project YOLOv8, despite the lack of a research paper and the fact that it is not a direct implementation by the original authors.

Although it is not explored, at the time of writing this dissertation, a new state-of-the-art foundation model named SAM (Segment Anything Model)[35] has been introduced, combining a new 11M image dataset (≈1B masks) with a new prompt-able segmentation model that would allow zero-shot transfer to specific tasks by prompt engineering.

3.0.2 Human pose estimation

A computer vision task that is more recent than object detection and segmentation is human pose estimation. Human pose estimation refers to the process of automatically detecting and understanding the body poses of humans from images or videos. It involves the identification and localisation of body joints and parts, such as hands, elbows, hips, knees, etc. By analyzing the spatial relationships and positions between these body parts, pose estimation algorithms can infer the configuration of the body of humans in a presented setting. Early attempts at pose estimation through *model-based methods* would work with a pre-defined parametric body model and would reverse kinematics on known body parts positions on the image. Other early approaches combining *model-* and *learning-based methods*, particularly *example-based methods*, used global and local features extracted from images or silhouette images (either created by filtering or provided along) to search for similar images in a training dataset with labeled poses [55]. Input image pose estimation would then be calculated by interpolating poses from retrieved training image examples. Agarwal et al., 2004 [3] was one of the first attempts in learning-based methods to move away from explicitly stored examples and to build a single estimation model that would generalise to unseen data. In this case, they trained a sparse Bayesian non-linear regressor from a database of training images of silhouettes by generating relevant vectors from input images to encode features. The regressor could then be used to estimate poses in silhouettes masks created from input images [4]. Other early learning-based approaches used pictorial structures as well as probabilistic graphical models to infer pose estimation, combining part detectors (local classifiers) with body-models (as in model-based methods) to

generalise pose estimation [5].

Similar to most other computer vision tasks, the appearance of deep learning techniques and large scale datsets revolutionised the field. Notable approaches to pose estimation using Convolutional Neural Networks were Convolutional Pose Machines (CPM) [65] and Stacked Hourglass Networks [44]. Both of these methods used a general idea of concatenating or stacking multiple stage predictions to refine a final estimation. They approached this by introducing intermediate supervision between stages during training to prevent gradient vanishing. Most recent human pose estimation techniques that leverage deep learning can be classified into two approaches: *bottom-up* and *top-down*. *Bottom-up* approaches use a first detection stage to identify and localise all body parts and joints in the image and then use a combination step to join these parts into full pose estimations for one or more persons. *Top-down* ones first use a detection step to identify persons in the scene and then apply body estimation to each individual bounding box. OpenPose [17] is a prime example of a *bottom-up* approach that achieved state-of-the-art results when presented. Developed and open-sourced by Carnegie Mellon University, it introduced a novel concept called *Part Affinity Fields*: while confidence maps predict body part location, a parallel layer is in charge of predicting 2D body part orientations to the associated body parts. This allows the model to greedily join body parts based on the orientation into single pose estimations in real time by reducing the computational cost of the *bottom-up* approach. DensePose [28] is an alternate approach to *top-down* models that uses a a custom labeled dataset based on the COCO dataset to create a model that maps every pixel of a person in a scene to a 3D location of the surface of a human body. The approach allows for new uses given the 3D body predictions instead of the traditional key-point estimations. BlazePose [7] is a pose estimation model developed by Google Research that focuses on real-time inference on light computing devices and is trained using MediaPipe [38].

Three currently state-of-the-art models in human pose estimation are AlphaPose [23], ViTPose [67] and YOLO-Pose [39]. AlphaPose introduces several new techniques focused on improving real-time multi-person pose estimation and tracking. ViTPose is an approach to pose estimation that uses the concept of attention [60] after the success of vision transformers [20, 34] in other computer vision tasks. YOLO-Pose is another iteration of the YOLO family of models based on YOLOv7 [63] that applies the one-stage detection architecture on multi-person joint and part detection by eliminating the two-stage approach based on heat-map part localisation and improves accuracy results thanks to optimizing directly the evaluation metric as a result of the single

pass predictions.

3.0.3 Computer vision in sports

The presented technological advancements in the field of computer vision have made it a very suitable solution for the world of sport analytics. Some tasks such as tracking distance or movement were made possible by wearable sensors, but these approaches were only at reach of professional teams or athletes. Moreover, certain aspects of team or ball sports are very hard to track as they require high level context understanding of the specific sport and visual analysis [42]. Despite this, open research in the application of computer vision in sports is scarce due to the private nature of professional companies and the competitive advantage it might offer. Notable studies focus on tasks associated with popular sports: ball-tracking [15, 64], player-tracking [40] and event detection [48]. Other interesting approaches to more niche sports, similar to indoor climbing, apply Mask-RCNN for waterline detection in canoe sprint videos [62] or evaluate diving performance using extreme pose estimation [43].

3.0.4 Approaches to climbing analysis

Some studies have previously used computer vision techniques or other types of analysis in solving and answering various questions for indoor and outdoor climbing. Dovgakecs et al,. 2014 [21] performed a study on climbing motion by means of 3D accelerometes and gyroscopes and proposed a probabilistic approach to classifying climbing actions into four states of behaviour based on collected data. Tai et al., 2020 [58] and Stapel, 2020 [57] attempt to solve two different problems in the same constrained space: the MoonBoard, a set indoor climbing wall with the exact 140 holds placed identically everywhere. The first analysis uses Graphical Neural Networks to classify new problems into a difficulty level based on the thousands of defined problems on the board. The second analysis uses heuristic greedy algorithms to create new proposed problems based on the dataset of defined problems. Panduveric et al., 2022 [46] uses hold detection and pose estimation algorithms to compute deep analysis of speed climbing (a modality of indoor climbing centered in completing as fast as possible a pre-defined wall problem that never changes). Their novel approach maps wall landmarks, features and key-point estimations for body joints into world coordinates and normalises climbers action to suppress viewpoint perspective and panning thanks to the static nature of a speed climbing problem. This results in highly comparable speed and movement statistics between climbers. Ekaireb et al. [22] study the same problem as this dissertation by using a bounding box hold detection model and Mediapipe pose estimation

algorithm to output metrics such as total elapsed time for problem resolution, number of moves and move validity in indoor climbing problems.

4 Methodology

This section presents a detailed overview of the steps taken to conduct the project. Each subsection provides context of which problem is trying to be solved and relates it to the problem and questions presented in 2.

4.1 Hold detection and segmentation

As previously mentioned, an indoor climbing problem is compromised by holds drilled into a wall. The first task to successfully analyze a climbing video is to be able to identify holds in a wall and discern them from the background.

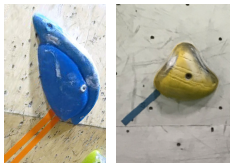

Figure 2: Example of indoor climbing holds

4.1.1 Edge and contour detection

On a 2-D image or frame, holds appear as coloured closed shapes as seen in Figure 2. A first approach to detecting holds was to use traditional edge and contour detectors, such as Canny-Edge detection [16] or colour "thresholding" on different colour spaces after applying different smoothing filters (e.g. Median filter of Mathematical Morphology).

While providing a promising start, this basic approach did not generalise well across examples, as seen in Figure 3 and 4. Each example image involved fine tuning for its specific case (filter parametrisation, colour space, etc.) and did not translate well to other example images. Furthermore, it included a fair bit of false positives and false negatives, and the masks produced were not accurate enough for further steps in the analysis pipeline.

Figure 3: Canny-Edge detection applied to an indoor climbing wall

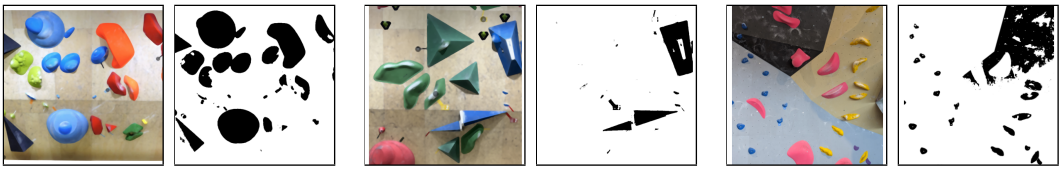

Figure 4: Binary masks created in three different examples by a value threshold on the B channel (LAB colour space) using same configuration across examples

4.1.2 Feature extractors

As stated in 3, one of the early approaches to object detection was to use feature extractors such as the aforementioned SIFT, SURF [6] or ORB [53] to extract meaningful points of interest of objects. Either by hand-crafted decisions or through traditional machine learning algorithms, these extracted features could then be used to detected and identify searched objects in the image.

Hand-crafting the decisions was historically not an easy task, and as we can observe in Figure 5, most of the key-points in our case did involve surrounding wall features rather than hold features, due to the smooth colour texture nature of climbing holds. Using machine learning algorithms, on the other hand, required having a classified dataset to train the algorithm to learn this decisions, at which point we could make use of more SOTA tools (deep learning approaches) involving self feature learning capabilities to perform the task.

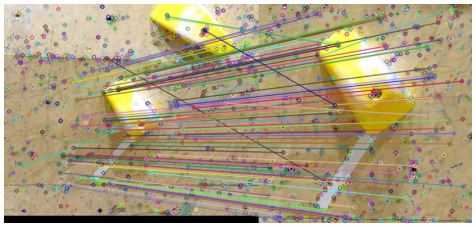

Figure 5: FLANN based SIFT key-point matches between two images involving the same climbing hold

On top of the already mentioned problems, detecting an object inside a classified bounding box still required, for further analysis, estimating a mask or contour for the hold, which would bring back up some of the challenges explored in 4.1.1.

4.1.3 Deep learning strategies for hold detection and segmentation

Having explored different solutions, given the need of high-recall detection and high-accuracy segmentation, it was fitting to try some of the more established deep learning Convolutional Neural Networks (CNNs) solutions to solve the problem.

One of the drawbacks of deep learning is the amount of data needed to converge these big networks into optimal loss minimisation to perform the task. This can be circumvented by the use of transfer learning [59]: training a neural-network on a bigger data corpus that has common or similar features and objective loss with your task and fine-tune [8, 52] the resulting network using a much smaller dataset to your specific case. In computer vision, ImageNet and COCO (Common Objects in Context) are popular datasets to pre-train foundational models [11] in classification, detection and segmentation of objects in images and videos for easy transfer learning into more niche tasks.

4.1.4 Hold data labeling

In order to fine-tune a pre-trained object detection and segmentation model a small hold dataset was created. Through scraping, downloading and friends' collaboration around **160 images** of indoor climbing walls were collected. This images were uploaded then to Labelbox through its free-to-use research and education plan. The platform offers an online editor that simplifies labeling and segmenting data. The segmented objects were tagged into two different categories that can be observed in Figure 6.

1. **Hold**: Generally smaller coloured holds that can only belong to one indoor climbing problem.
2. **Volume**: Generally bigger, with elementary shapes, coloured with neutral colours (eg: black, grey). Used to add difficulty to multiple problems by giving complexity to the wall shape.

Figure 6: On the left, examples of *holds*, on the right, in grey and black, examples of *volumes* with holds on them

This dataset was then manually segmented and labeled according to the mentioned classification criteria. This process produced a total of approximately **6000 segmentation masks** that were split into the two types of holds with the distribution shown in Table 1.

Type	**Count**	**Pct**
Hold	5738	93%
Volume	372	7%

Table 1: Manually drawn and labeled masks for each class

Figure 7 shows sample images belonging to the dataset before and after labeling.

Figure 7: Dataset image samples before and after labeling

4.1.5 Hold data augmentation

In addition to fine-tuning a pre-trained model, another technique that can be used to improve model performance, specially with a small dataset, is data augmentation [56]. Specifically, in object detection, data augmentation usually refers to transformations applied to sample images to alter them to appear as a completely new sample to the model. Data augmentation also helps model performance be more robust and better generalise to unseen data.

In our concrete case, one can see how data augmentation integrates seamlessly into the workflow. Climbing holds can be of all shapes, sizes and colours. They can be placed generally in any position of the x,y axis of the wall and in any rotation. As a consequence, any image transformation that involved cropping, rotation, scaling or color shift would theoretically result in a more robust model, on top of providing more data samples that would assist the model in better identifying and learning relevant features to detect and segment climbing holds. Given our labeled masks, the idea of applying segmentation-only

transformations was considered. This would imply performing mostly pixel based transformations into images and then merging this changes back to the original images as to only have these changes affect areas contained inside masks. Considering also the nature of our data, creating synthetic data [29] by randomly adding segmented holds from other image samples into fake background walls was also contemplated. Both data augmentations techniques were not explored further as general image transformations boosted the model performance enough to consider focusing time on other areas, as justified in 2.0.2.

The data augmentation pipeline for this project includes various pixel based and spatial transforms chained one after the other one. These transforms are applied with probability p to the image. Therefore, the probability the image sample remains unchanged is $(1-p)^n$ (where n is the number of transforms). The transforms included in the data augmentation pipeline for all the further evaluations are the following:

1. **Random crop** (crop the image to a random part)
2. **Transpose**
3. **Perspective** (random four point perspective)
4. **Horizontal flip**
5. **Hue and saturation** (random value shift)
6. **Emboss**
7. **Downscale**
8. **Color Jitter**

These transforms are concatenated in the above order and applied with probability $p = 0.2$. In 4.1.7 we evaluate the model with different rounds ($n_{da} = 0, 1, 3, ...$) of the data augmentation pipeline.

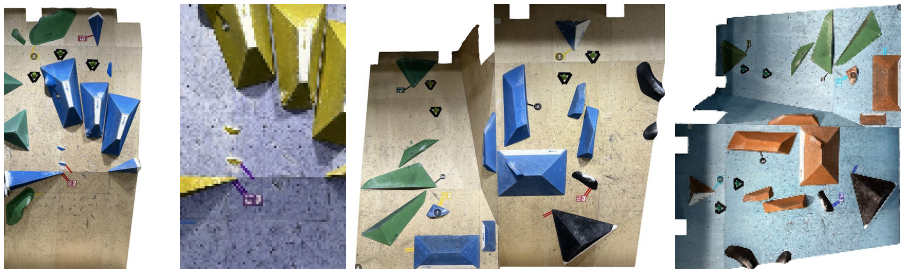

Figure 8: Original samples and new augmented samples after transforms

4.1.6 Model training

Two different model architectures were considered for the training step: Mask-RCNN and YOLO. Initial tests showed that Mask-RCNN produced more accurate masks as well as less false negatives (YOLO displayed a tendency to skip small holds in samples with high number of individual instances). Time constraint was a deciding factor in exploring deeply one architecture instead of both. As mentioned in 4.1.3, the aim was to use a pre-trained COCO instance segmentation model and fine-tune it with the custom indoor climbing hold labeled dataset presented in 4.1.4. Figure 9 shows a simplified diagram of Mask-RCNN heads over two different backbones and how the model builds on top of Faster-RCNN to incorporate instance segmentation through a parallel branch.

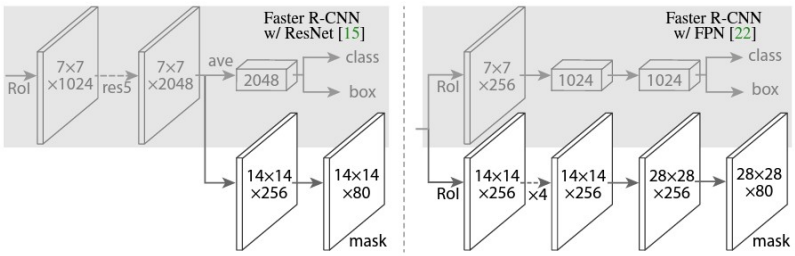

Figure 9: Diagram of two Mask-RCNN head variations over two backbones: ResNetC4 and FPN. Source: [30]

To train the hold detection and segmentation model the detectron2 framework was used. Detectron2 is an open-source library developed by Facebook AI Research (FAIR). Mask-RCNN was proposed by FAIR so the library provides ready-to-use tools to train and fine-tune multiple models, including the RCNN family. The hold model is trained by fine-tuning *mask_rcnn_R_50_FPN_3x*. The base model is defined by a ResNet backbone already pretrained on ImageNet classification tasks and incorporates a Feature Pyramid Network [36] (which adds robustness to detecting climbing holds of very different sizes). The base model is trained on the COCO dataset for ≈37 epochs. The model provides the best speed/accuracy trade-off. The fine-tuning is done on a NVIDIA GeForce 1050Ti and therefore training speed and inference was also considered. The original hold training dataset contains 109 images and ≈4600 segmentation masks. This set is used to train the model after initialisation with the base model weights throughout different configurations and data augmentation iterations.

4.1.7 Model evaluation

The model training runs use the validation set and test set introduced in 2.0.3 to prevent over-fitting and to compute model performance metrics respectively. To evaluate the different training runs the COCO standard evaluation metrics are used. These metrics build on top of Intersection over Union (IoU) for both bounding box and segmentation masks to define AP, $AP^{IoU=0.50}$, and $AP^{IoU=0.75}$. All metrics are computed for both bounding box predictions and segmentation masks estimations. Average Precision (AP) averages area under the precision-recall curve over different IoU thresholds (specifically 10 values in [0.5, 0.95] with 0.05 steps). $AP^{IoU=0.50}$ and $AP^{IoU=0.75}$ is area under the precision-recall curve at the defined set IoU thresholds. Generally COCO AP can be interpreted as mAP, for both 2-class and uni-class configurations. The different model training runs can be observed in Table 2

Config				**BBox**			**Mask**		
n_c	n_{da}	s	epochs	AP	$AP^{0.50}$	$AP^{0.75}$	AP	$AP^{0.50}$	$AP^{0.75}$
2	0	109	10	0.54	0.66	0.62	0.55	0.65	0.61
1	0	109	10	0.66	0.81	0.78	0.67	0.81	0.77
1	1	218	10	0.68	0.83	0.80	0.69	0.83	0.79
1	3	436	10	0.68	0.82	0.79	0.70	0.82	0.79
1	5	654	10	0.69	0.85	0.81	0.71	0.85	0.81
1	5	654	15	0.69	0.85	0.82	0.71	0.85	0.81
1	10	1199	10	0.70	0.85	0.81	0.71	0.85	0.81
1	**10**	**1199**	**15**	**0.71**	**0.86**	**0.82**	**0.71**	**0.86**	**0.82**

Table 2: Training runs and evaluation metrics for hold detection and segmentation model. n_c: number of classes, n_{da}: number of data augmentation runs, s: final size of training dataset

4.1.8 Discussion

One decision that helped the model performance metrics improve significantly was to transform both holds and volumes (4.1.4) into a single class. For the problem in hand, class was an irrelevant factor to answer the posed questions in 2. The initial assumption that different classes for different looking holds would help the model estimate better segmentation masks was proved wrong. On Table 2 the first two rows show the evaluation for metrics for the two-class and uni-class configurations. The uni-class change represents the biggest jump on model performance in all the table. Subsequent runs explore the improvements data augmentation adds to the model given the relatively small

initial training set size. We can observe a $\approx 5\%$ improvement between the second row run with no data augmentation added and the selected model on the last run. Data augmentation increases AP by improving bounding box and mask accuracy for small and medium holds as well as reducing the propensity of false negatives. This is specially true when averaging AP across high IoU thresholds such as $AP^{IoU=0.90}$. Performance on big holds remained mostly stable throughout the different training runs. The AP performance of the last model configuration was deemed good enough for further steps. Subsequent iterations of n_{da} and epochs were not explored given the convergence on the AP - training time trade off. Figure 10 shows instance and segmentation predictions of the highlighted model for different test images.

Figure 10: Detection and segmentation hold predictions for test images

4.2 Hold matching

After implementing a good enough solution for 4.1, the first question presented in the problem statement, as described in 2, is still partially unresolved. While it is generally assumed that holds of the same color belong to the same problem, each indoor gym has its own specific rules. Consequently, an interactive problem definition process was implemented. This interactive process allows for an easy selection of holds that conform the given problem based on a static image of the indoor wall. Alongside the video to be analyzed, the process requires an image of the climbing wall. An interactive screen is prompted with the provided image that allows the user to select the holds that belong to the problem. Once the climbing problem has been defined on the reference image, the task at hand is to match the holds into the analysis video frames. Since in 2.0.2 we defined the video as static, this would implicate matching the holds on the initial frame or another non-occluded frame. This interactive process allows to define the indoor problem once in a reference image and identify the same hold problems in any provided video.

Several potential solutions were considered at this stage. Given the previously implemented hold detection and segmentation techniques, a deep learning approach was explored. This involved creating bounding boxes around each detected hold in both the reference image and the target frame. Classification algorithms utilizing ViT embeddings [20] or traditional convolutional neural networks (CNNs) for image classification could be applied to these bounding boxes pairs to estimate similarity. The drawback of this approach was it did imply building another dataset of hold-to-hold pair matches and therefore was not pursued further. Ultimately, traditional methods for object detection and matching were employed to implement the hold matching process. Template matching [13], where the bounding box of the hold in the reference image is convoluted as a sliding window over the target image at various scales to identify highly correlated regions was briefly tested. However, its inconsistent performance quickly led to a shift in focus towards more recent traditional feature extractors, as discussed in 3 and 4.1.2.

4.2.1 SIFT key-points and descriptors

SIFT, which stands for Scale-Invariant Feature Transform, extracts distinctive and robust local features from images, enabling reliable matching and recognition across different scales, rotations, and lighting conditions in other images. Given its robustness, proven performance and general availability, it was chosen as a feature extractor for hold bounding boxes in both reference image and target frame.

Defining N as the set of selected holds in the reference image conforming the problem and C as the set of detected holds in the target frame, the matching process consisted in estimating relevant key-points ($point_r$) and its corresponding descriptors in bounding boxes of N in the reference image. Key-points ($point_t$) and descriptors were then computed on the entire target frame also. After that, each bounding box bb_n (bb_n = bbox of hold $n \in N$) was matched to the target frame using key-points and descriptors with a certain distance threshold δ_d. Let's define M_n as the set of key-point matches between selected hold n bounding box and the target frame and d as the match distance. Let's define also bb_c as the bounding box for candidate hold $c \in C$. Then

$$f_c(M_n) = |M_d|,$$
$$M_d = (m_1, m_2, ..., m_n) \; \forall \; m_i \in M_n \text{ s.t. } d_i \leq \delta_d \text{ and } point_t(m_i) \in bb_c$$

$f_c(M_n)$: # of good matches between reference hold n and candidate hold c.

The algorithm finally selected the top-K ($K = |N|$) candidate holds from C in the target frame that had more good matches to the reference holds' bounding boxes. This notion of number of matches for a candidate hold could be summarised as

$$F_c = \sum_{i=1}^{N} f_c(M_i)$$

The matching results were a good start, but holds that produced big bounding boxes (either by the hold size or by its orientation) in C were favoured. As shown in 4.1.2, a good amount of key-points were generated by wall features rather than intrinsic features of the hold. This introduced bias into the matching algorithm by selecting holds that created bounding boxes containing a lot of wall space. In addition, SIFT descriptors failed to encode colour information (which can be considered one of the most important features and descriptors of holds) as it works with gray scale images.

Results on reference selected holds key-point matching to target frame can be seen in Figure 11. Results of the only-SIFT hold matching process can be seen in Figure 12.

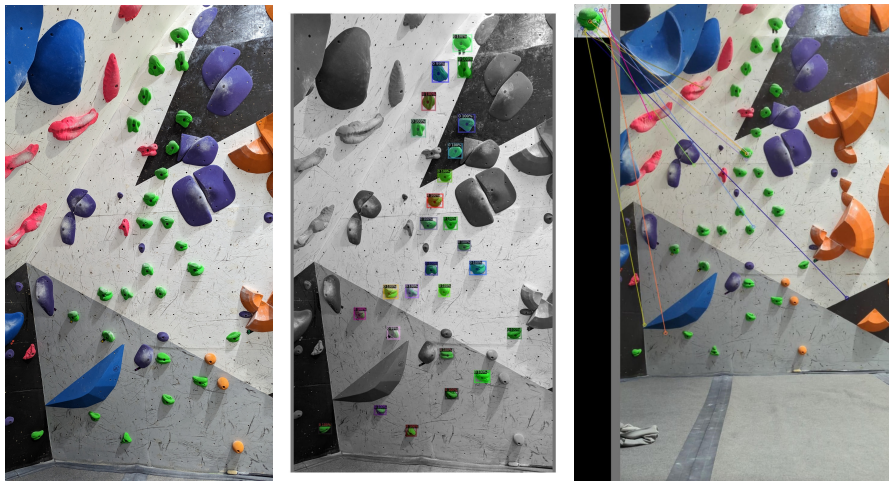

Figure 11: *Left:* Reference image example. *Middle:* Selected holds in reference image. *Right*: SIFT key-point matches example from hold bounding box in reference image to initial video frame.

4.2.2 RANSAC

RANSAC [24] (Random Sample Consensus) is an iterative algorithm commonly used in computer vision for robust model estimation (for instance, for computing the homography relating two images capturing the same planar object) in the presence of outliers. RANSAC can be used together with SIFT to

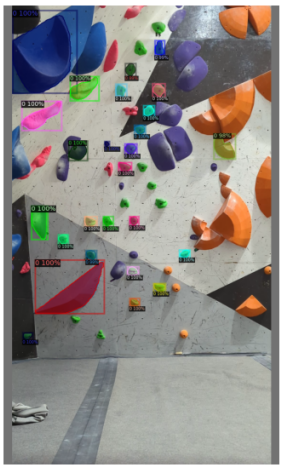

Figure 12: Candidate holds proposed by hold matching (goal: green route holds) that favours (wrongly) holds with big bounding boxes. Check Figure 11 for reference

improve the process by providing robustness to the matching algorithm in the presence of erroneous key-point pairs. RANSAC achieves this by randomly selecting a subset of key-points and estimating the transformation model, such as an affine or homography matrix, that relates the key-points between the images. This estimated transformation model is then evaluated by measuring the number of key-points matches that fit the model (inliers) within a certain threshold. The RANSAC algorithm iteratively repeats this process, selecting different subsets of key-points and estimating different models. The final model is determined by selecting the one that maximises the number of inliers.

In our hold matching algorithm we applied RANSAC to compute the homography relating both views and to further filter out wrong key-point matches. For each bounding box proposed, key-point matches were discarded if the estimated homography model considered them outliers. In other words, we applied the following extra condition on the scoring function introduced in 4.2.1

$$f'_c(M_n) = |M_d|,$$

$$M_d = (m_1, m_2, ..., m_n) \ \forall \ m_i \in M_n \text{ s.t. } d_i \leq \delta_d \text{ and } point_t(m_i) \in bb_c \textbf{ and}$$

$$||\mathbf{point_r(m_i)}, \mathbf{H}^* \cdot \mathbf{point_t(m_i)}|| \leq \varepsilon$$

$f'_c(M_n)$: # of good$'$ matches between reference hold n and candidate hold c.

$^*H$: Homography matrix estimated by RANSAC

In line with the 4.2.1, then

$$F'_c = \sum_{i=1}^{N} f'_c(M_i)$$

In Table 3 we can see the improvement RANSAC provided over the only-SIFT algorithm. Even though RANSAC did improve the accuracy of the algorithm, the hold matching process did still not use one of the most visual features of holds: colour.

4.2.3 Colour matching

Last improvement area that was added to the hold matching algorithm was to make use hold colours to improve the matching results. CSIFT [1] is a proposed alternative to leverage colour information by building key-point descriptors in a colour invariant space. Unfortunately, a ready-to-use implementation for CSIFT was not found. Given the reasons stated in 2.0.2, a more basic approach to use colour information was taken.

This basic approach makes use of the segmentation masks created for both reference holds and candidates holds. Iteratively, along the SIFT computation, the average colour inside the mask V_n for the reference hold is computed (average hue value in HSV colour space). The same is done for candidates holds in the target image and their respective masks. The algorithm assigns then θ points to each candidate hold whose average mask hue value V_c is inside a range centered around V_n (range values δ_r and θ can be parametrised).
We can define this process as

$$g_c(V_n) = \begin{cases} \theta, & \text{if } V_n + \delta_r \geq V_c \geq V_n - \delta_r \\ 0, & \text{otherwise} \end{cases}$$

$g_c(V_n)$: Points assigned to hold candidate c based on its hue colour and the hue colour of reference selected hold $n \in N$

4.2.4 Combined SIFT+RANSAC and colour scoring

As mentioned in 4.2.1, the algorithm goal is to take the top-K ($K = |N|$) candidate holds in the target frame as the problem holds in the video frames. To combine both scoring functions we define score S_c for candidate hold $c \in C$ as

$$S_c = \sum_{i=1}^{N} f'_c(M_i) + g_c(V_i)$$

Therefore, the K candidates with higher score S are selected as the matches for N in the initial frame.

4.2.5 Hold matching evaluation

To evaluate each one of the approaches and parametrisations as well as the incremental improvements and the final scoring technique, we used the samples presented in 2.0.3. Each of the 4 examples was used to define 5 theoretical indoor problems (by selecting holds in the reference image): the real one tied to the video, two other ones with colour consistency and two extra ones of random holds. Consequently, the following evaluation was done across 20 hold matching executions. To quantificate the quality of the hold matching algorithm process we defined the *precision* metric (which in this case mimics *recall*) as

$$Precision = \frac{correct\ hold\ matches}{|N| = K}$$

We did not use *accuracy* to normalise *precision* across number of possible candidates as it did not correctly reflect how well the algorithm was doing for the task in hand, although in other situations it might be considered a more fair assessment.

The evaluation results across examples for the 3 different scoring methods can be seen in Tables 3 and 4. The evaluation examples had both reference image and initial frame downscaled to size [576, 960] for better inference time both in hold detection and hold matching. All evaluations used set parameters $\varepsilon = 5.0$ and $\delta_r = 15°$.

	$\delta_d = 0.9$	$\delta_d = 0.75$	$\delta_d = 0.5$
F_c	0.62	0.82	0.83
F'_c	0.77	0.83	0.83

Table 3: *Precision* for SIFT-only and SIFT+RANSAC approaches across different match distance thresholds

4.2.6 Discussion

We can see in 4.2.5 that each addition to our scoring method improves the *precision* of the algorithm results. RANSAC provides an improvement over

	$\theta = 5$	$\theta = 10$	$\theta = 20$
S_c	0.83	0.84	0.84
S_c^{**}	0.92	0.94	0.93

Table 4: *Precision* for combined scoring across different θ values. Fixed $\delta_d = 0.75$

SIFT-only by filtering matches that do not comply with the projection estimation, specially when we relax the match distance threshold (bigger δ_d). This helps avoid some of the problems presented in 4.2.1. Though at first appearance the combined scoring method S_c does not present a significant improvement over F_c', when analyzing results this is produced by climbing problems whose hold colour is grey or black. In these cases, since we are only using the values for the hue channel in HSV space, the scoring based on colour range does not fully take advantage of the feature that $g_c(V_n)$ aims to extract.

S_c^{**} are the *precision* results for the combined scoring method when subtracting the conflicting samples. We can see in that case how incorporating a colour based scoring function into the algorithm, even if basic, does present a significant improvement over F_c' alone. We can see results of the combined scoring approach in Figure 13.

Finally, the matching algorithm seems at first to solve correctly 3 out of the 4 sample examples presented in 2.0.3 (4th one being the one were holds are gray) when the image and the initial frame are left at their original resolutions: [2268, 4032] and [1080, 1920] respectively. Despite this, the algorithm has not been tested extensively with high-quality images to provide any real conclusion.

4.3 Pose estimation

An indoor climbing set up is composed of two clear concepts: the holds itself and the climber. Having found an appropriate solution for identifying holds on the video subject to analysis and matching the predefined problem into the holds detected, the task in hand is to identify the climber and its movements. To do this, we attempted to use pretrained human pose estimation (HPE) models to track the climber movement and joints.

Figure 13: Selected holds by matching algorithm using combined scoring. *Left:* Match results for problem defined by grey holds. *Right:* Match results for problem defined by green holds.

4.3.1 Approaches

The decision to use an out-of-the-box pretrained model for HPE was taken given the limitations exposed in 2.0.2, the lack of a clear framework to fine-tune HPE models and the complexity of creating a contextualised climbing labeled pose dataset. The pretrained models used were BlazePose (through MediaPipe's framework) and YOLOv7.

Figure 14: BlazePose estimated poses for climber in sample video. *Left:* Low accuracy on joint location. *Right:* False negatives on joint detection.

Despite the apparent success of Ekaireb et al. on their use of MediaPipe's framework for pose estimation, pretrained BlazePose on our test videos did not show the same results. Overall, the results displayed multiple false negatives

on important body limbs, accuracy for part location was low and consistency between frames for tracking was lacking. This problems can be observed in Figure 14. Since the initial test, at the time of writing this dissertation, it seems MediaPipe's framework has been updated and reworked. A quick test on their demo website shows better performance for still images of climbing frames but still displays errors, as seen in Figure 15.

Figure 15: BlazePose estimated poses for climber in new web demo. *Left:* Better pose estimation accuracy. *Right:* False negative on climber detection.

Given the unreliable results produced by BlazePose, a more state-of-the-art model was used. The model needed to be robust against extreme poses, be reliable estimating back-facing poses and provide an interactive framework to work with body joint location estimations for further analysis. Out of the three SOTA approaches presented in 3.0.2, YOLOv7 trained on the MS COCO dataset was chosen for the stated reasons. The single-pass architecture of YOLO also presented improvements on inference time compared to BlazePose. Pose estimation results on sample videos using YOLOv7, despite some small inconsistencies, showed good enough results to be used for further stages of the analysis pipelines. Some examples of pose estimation results in video frames can be seen in Figure 16.

4.3.2 Pose estimation evaluation

Given the lack of manually tagged pose labels for the sample videos, there was no direct way to compute an evaluation metric for HPE results. Because of this, evaluation was delayed after finding appropriate solutions for 4.4 and 4.5. Therefore, it was decided a fair assessment for applying YOLOv7 pose estimation in climbing videos would be computed by evaluating errors either in key-frame detection or final inference analysis that were caused directly by

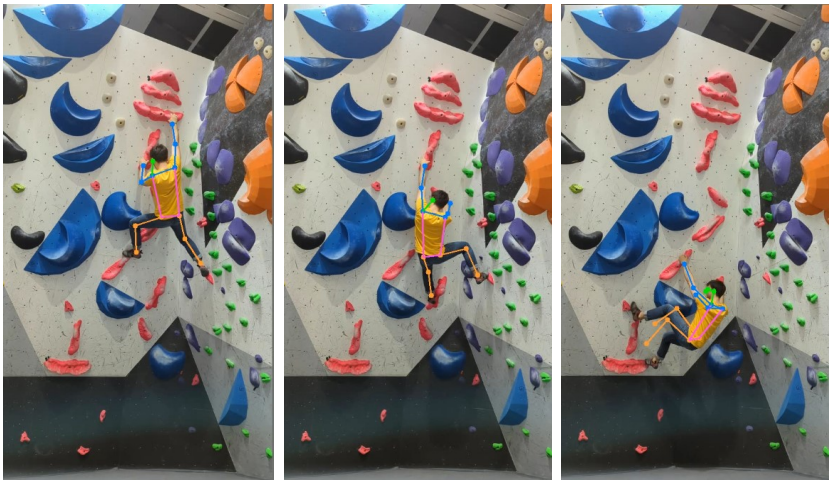

Figure 16: *Left, Middle:* YOLOv7 estimation for similar poses as in Figure 14. *Right:* Small inconsistency in YOLOv7 pose estimation.

wrong pose estimations. This would help also contextualise the evaluation in the problem, given the lack of any major adjustments to the model other than small changes in parametrisation.

4.4 Key Frame detection

The idea of key frame detection is inherited from [22, 21]. Both studies try to classify climbers actions into different activities based on movement data. In our case, to tackle the third question in 2, we need to reduce the search scope for 4.5 both for optimisation reasons and to avoid false positives. We define key frame in our video analysis problem to be a frame in the video that has interesting information to answer the posed questions. In this case, to find a suitable solution to describe the "beta": how the climber solved the climbing problem composed by the selected holds. To simplify that question, "beta" is defined to be hand and foot placements on the different holds. This definition will be further refined in 4.5. For key frame detection then, we're interested in detecting "moves". But not movement frames specifically, but rather frames when a move has been completed (the climber's starting position of the problem being the only exception of an interesting frame without a previous move). A visual description of a move can be seen in Figure 17. In that sequence, we would be interested in analysing the climber's left hand position on the first frame and the last frame, and avoid any computation while the left hand is moving.

Figure 17: Frame sequence describing a left hand move

4.4.1 Key frame detection algorithm

Having described what constitutes a key frame in our video analysis problem, the goal was to detect frames where the analysed joint remains static after completing a move. To do that, euclidean distance traveled between frames by tracked joint j could be computed using pose estimation output (x^j, y^j, t). Angular velocity ω could too be computed from traveled Euclidean distance.

$$\Delta^j_{t,t-1} = \sqrt{(x^j_t - x^j_{t-1})^2 + (y^j_t - y^j_{t-1})^2}$$

Distance traveled by joint j from frame $t-1$ to frame t

Overall, despite success by Ekaireb et al. and Dovgalecs et al. on the use of angular velocity, tracking distance traveled by joints on the viewpoint plane showed more robustness to the small inconsistencies produced by YOLOv7 on estimation of climber's joint locations. Given all data points for distance traveled between frames by joint j in the video, the objective was to find 1 frame for all intervals where joint j remains static between intervals where joint j displays movement.

To do that, curve $c^j(t)$ was defined by chaining all location points for joint j computed by the pose estimation model through out the video frames t. Then $d^j(t)$ was defined as the curve created by the sequence of pair-wise frame's euclidean distance traveled by joint j:

$$d^j(t) = \Delta^j_{t,t-1}$$

A Savitzky-Golay filter [54] was then applied over the curve to reduce noise

created by the joint estimation model

$$Y^j(t) = \sum_{i=\frac{1-m}{2}}^{i=\frac{1+m}{2}} C_i d^j(t+i)$$

where m is an odd integer defining the interval size of the smoothing window and C_i are convolution coefficients defined based on m. To further filter out noise then a thresholding filter based on the average distance traveled was added to clearly identify all intervals between moves:

$$D^j(t) = \begin{cases} Y^j(t), & \text{if } Y^j(t) \geq \tilde{Y}^j \\ 0, & \text{otherwise} \end{cases}$$

where

$$\tilde{Y}^j = \frac{1}{T} \sum_{i=1}^{T} Y^j(i)$$

Given $D^j(t)$, then key frames t^{kj} were computed by obtaining all central frames belonging to all maximum zero-valued intervals. That is, by definition:

$$t^{kj} = \lfloor \frac{t_1 + t_2}{2} \rfloor, \forall t_1, t_2 \in [1, T] \; t_2 > t_1$$

$$\text{s.t. } \sum_{i=t_1}^{t_2} D^j(i) = 0 \text{ and } \nexists \; t_1' < t_1 \text{ s.t. } \sum_{i=t_1'}^{t_2} D^j(i) = 0 \text{ and } \nexists \; t_2' > t_2 \text{ s.t. } \sum_{i=t_1}^{t_2'} D^j(i) = 0$$

The obtained key frames (represented by red dots) can be seen in Figure 18 for the right hand joint by the above definition over all the sequence frames for one of the sample videos. The drawn curve is defined by $Y^j(t)$ instead of $D^j(t)$ to better understand the process.

4.4.2 Key frame detection evaluation

To evaluate the key frame detection approach, two metrics were defined similar to 4.2.5. *Recall* computes the ability of the process to detect key frames for all true moves in the sequence while *precision* aims to evaluate the process in detecting true key-frames and avoiding duplicate key frame pairs (key frames belonging to the same true move):

$$Recall = \frac{\#\text{ true moves detected}}{\#\text{ true moves}}$$

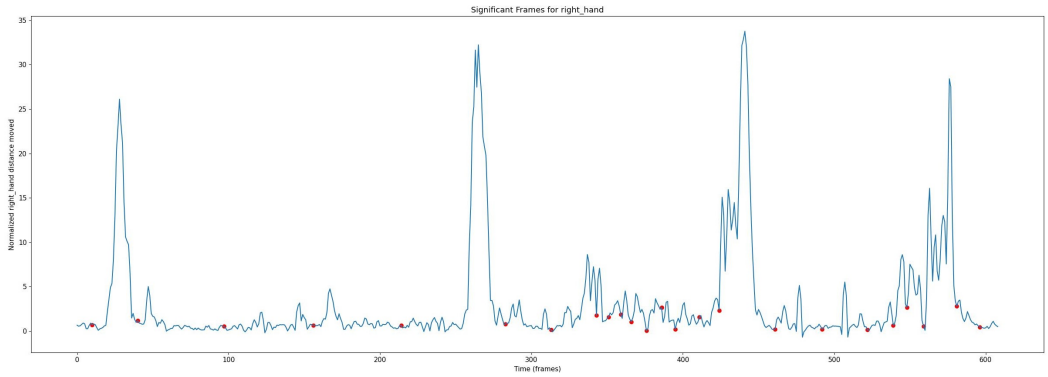

Figure 18: Key frames detected over the Sav-Gol smoothed right hand movement

$$Precision = \frac{\#\text{ unique true moves detected}}{\#\text{ key frames detected}}$$

The Savitzky-Golay filter was defined with a sliding window of $m = 5$ and the coefficients C_i belonging to the quadratic polynomial. Pose estimation was applied to the sample videos in their original resolution [1080, 1920]. After video analysis of all produced pose estimation results, some joint location estimations were considered wrong enough to be considered actual moves. Key frames detected for this fake moves were not considered in the evaluation process. This cases were annotated and revisited in 6.0.1 Other estimations, on the other hand, were not assessed wrong enough to be understood as an actual move. Therefore, it was considered the key-frame algorithm should have treated them as noise and, on an evaluation basis, avoid detecting them as key frames. The analysed climber joints for key frame detection were both hands and both feet. The results of the evaluation across the sample videos can be seen in Table 5.

Recall	1.0
Precision	0.29

Table 5: *Recall* and *precision* for key frame detection

4.4.3 Discussion

As one can observe on Table 5 the key frame detection algorithm successfully detects at least one frame per important move across all video examples. The *precision* of the approach, while not great, does not present breaking problem errors. On average, roughly two extra frames per important move are detected. That is, only ≈ 1 in each 3 detected frames is necessary. The extra 2 frames

are usually generated by noise introduced by the pose estimation model or by small motions on the climber joints that do not constitute a proper move. The algorithm in those specific cases fails to distinguish them from an actual move.

Going back to the original problem posed in 4.4, the goal was to reduce the search space for further inferences and analysis. In this case, despite *precision* being low, the solution successfully detects at least one frame per important move (*recall*). Furthermore, on average, the search scope of frames gets reduced approximately **25x**. That is, of a video sample with 700 frames only 28 are detected as key (this is an estimate and the actual number depends highly on the number of moves performed on the indoor climbing problem). The solution, therefore, is deemed good enough to make 4.5 optimally feasible while detecting each key frame necessary.

4.5 Final inference

The final inference step aims to use all outputs produced by previous steps to jointly give answer to the questions posed in 2. By combining 4.1 and 4.2 we have already seen how one can identify which holds constitute the problem in the video subject to analysis by a predefined selection on a reference image. The task that the final analysis step tries to solve is how to combine the matched holds in the video, the pose estimation prediction and the key frame selection to output an estimate of the "beta" the climber took to climb the problem. To recall the term "beta" from before, the sequence of moves and holds used (and the way they are used) to try to reach the final hold with both hands in an indoor climbing problem pre-defined by a group of holds is called commonly the "beta". To do that, a process is defined for which in each key frame t^{kj} for joint j an inference step calculates if said joint is using a hold.

4.5.1 K-D Tree search

To efficiently retrieve if joint j is using hold h a K-D tree [9] was used. A K-D tree is a binary search tree structure that organizes points in k-dimensional space (2 dimensions in our case) for efficient range searches and nearest neighbor queries. For each matched hold h in the video frames, the points defining the polygon created by the segmentation prediction model introduced in 4.1 were obtained and organised in a global K-D tree. Given the restrictions introduced in 2.0.2, the tree needed to be built only once in the beginning of the inference step, as the camera viewpoint, and therefore the segmentation masks for holds, were static throughout all key frames. The goal of using a K-D tree was to optimally find the closest hold to each joint location estimation

belonging to key frames. The problem was reduced to finding the closest point belonging to an estimated segmentation mask out of the matched holds. Compared to a naive brute force approach ($O(n)$), a balanced (one segmentation point per leaf) 2-D tree takes on average $O(\log_2 n)$ for each nearest-neighbour search (where n is the number of all points belonging to segmentation masks of the selected holds). Combining this optimised search with the key frame detection algorithm successfully reduces the computational cost of the final inference step.

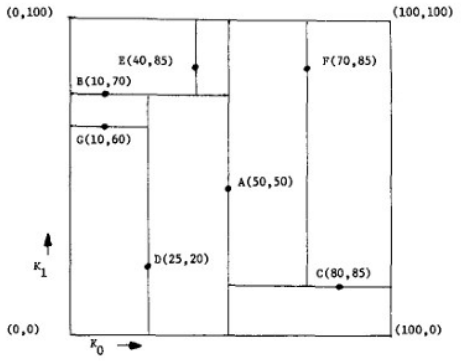

Figure 19: Example of records stored as nodes in a 2-d tree. Image from [9]

4.5.2 Move description

The process of obtaining a basic description is the following

Algorithm 1 Move description process

$J \leftarrow$ [left hand, right hand, left foot, right foot]
$threshold \leftarrow$ foot threshold
for $j \in J$ **do**
  $K^j \leftarrow$ list of key frames for join j
  for $t^{kj} \in K^j$ **do**
    $x, y \leftarrow joint\_position(j, t^{kj})$
    $dist, id \leftarrow tree.search(x, y)$
    if j `is` $hand$ `OR` $dist \leq threshold$ **then**
      $h \leftarrow get\_hold(id)$
      if j, h `is new` **then**
        save move $\Rightarrow$ j, t^{kj}, h
      end if
    end if
  end for
end for

That is, for each interesting joint j and each key frame detected for said joint t^{kj}, we obtain the location x, y of joint j in said frame and query the built 2-d tree to obtain the nearest neighbour. Given the pose estimation model specifically tracks the ankle for each foot and climbing uses mostly the tip of it, a threshold for nearest neighbour distance was established to avoid recording incorrect moves for feet. The threshold is defined as half of the distance between the ankle and the knee in a frame where pose estimation confidence is high (to estimate the maximum distance that could exist between the ankle and the tip of the foot, that is, when said distance is completely parallel to the viewpoint plane). Hands in rest position are usually using holds so thresholding is not applied on that case. Once the valid nearest neighbour point is obtained, the process recovers the hold related to it. If the move of joint j to hold h is new, then the process saves it. This check avoids saving duplicate moves given the precision of 4.4 at the expense of not saving possible actual repetitions (which is uncommon in indoor climbing).

When the algorithm is finished processing all joints J, the sorted (by frame) sequence of saved moves can be interpreted as a description of the goal question: *Which "beta" did the climber take to complete the problem?*. One example output could be

1. Right hand to hold a
2. Right foot to hold f
3. Left foot to hold o
4. Left hand to hold b
5. Left hand to hold g
6. ...

One example of the output visual representation of the description computed by the algorithm can be seen in Figure 20

5 Results

The output of executing the process implementing the methodology presented in 4 is the following:

1. An image indicating the selected holds in the provided reference image
2. An image corresponding to the first frame indicating the matched holds in the video
3. A video representation of the output of the pose estimation model for joints J
4. A key frame plot for each joint $j \in J$ as the one showed in Figure 18.
5. A visual representation of the sequence of computed moves

The main output of the process that can be considered the final result is the visual description of the "beta". We can see a subset of this visual representation in Figure 20 for one of the sample videos, with each frame highlighting the used hold.

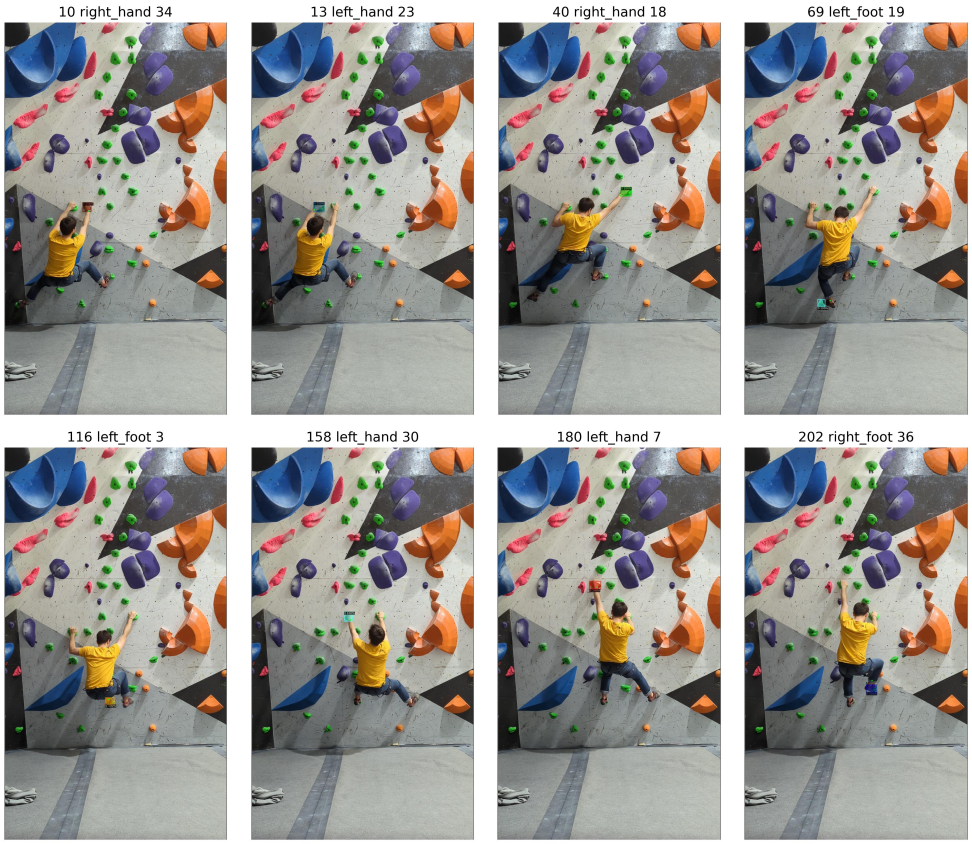

Figure 20: Subset of the computed sequence of moves of the "beta" for a sample video with a header indicating *frame-joint-hold*

To execute the process these previously mentioned inputs need to be provided:

1. A video for analysis
2. A reference picture of the wall (if not provided the first frame of the video is used)
3. Start and finishing frame number for climbing action

5.1 Results evaluation

To evaluate the final results the videos presented in 2.0.3 are used. Each evaluation so far has been individually applied to each sub-problem but running the entire process creates possible chains of errors. To estimate how correctly the implementation answers specifically the "beta" problem, the process was run for each one of the samples. The best configuration for each one of the steps outlined in 4 was used. To better measure the correctness of the implementation all inputs were used at their highest resolution: reference frame images at [2262, 4038] and sample videos at [1080, 1920]. To evaluate the correctness of the visual description of the problem resolution each video sample was annotated with its correct joint j - hold h pairs. The outputted sequence of joint-hold moves was compared to the truth labels by defining the following evaluation metrics

$$Recall = \frac{\#\text{ true } j-h \text{ moves detected}}{\#\text{ true } j-h \text{ moves}}$$

$$Precision = \frac{\#\text{ true } j-h \text{ moves detected}}{\#\ j-h \text{ moves detected}}$$

The results of the evaluation across the 4 videos examples presented in 2.0.3 and 85 true moves were

Recall	0.81
Precision	0.75
Recall**	0.87
Precision**	0.80

Table 6: *Recall* and *precision* for "beta" description estimation process

As in 4.2, the symbol ** indicates the evaluation disregarding the sample indoor problem composed by grey holds. Having acknowledged the over-reliance of the hold-matching step on colour, the errors introduced in said step for said sample makes it difficult to generalise the evaluation metrics to the rest the process. Figure 21 shows some correct and incorrect "beta" visual move descriptions generated by the evaluated process.

Figure 21: *Left, Middle-left*: Correct "beta" visual move descriptions. *Middle-right*: Wrong move description by incorrectly matched hold. *Left*: Wrong move description by a combination of imprecise key-frame detection and hold segmentation mask.

6 Discussion

The results presented in 5 and specifically the evaluation metrics showed in Table 6 demonstrate a sufficient correctness of the implemented process on answering two out of the three posed questions in 2. One of the challenges of the set goal is the mentioned chaining of errors after each step. On that note, each one of the steps conforming the process is able to solve its individual problem with reasonable evaluation metrics in order to obtain satisfactory evaluation metrics for the overall process. For the reasons mentioned, we consider the implementation a sufficiently successful one.

6.0.1 Pose estimation evaluation revisited

As mentioned in 4.3.2, having now implemented all the steps in 4, the results of the pose estimation model can better contextualised. To do that, an error rate is computed both for 4.4 and 4.5 directly caused by errors in the estimation. In the case of key frame detection, the error rate was 21.77%. That is, approximately 1 in out 5 key frames detected were caused by wrong estimation of joints that were very similar to actual moves. In the case of obtaining a move sequence for the "beta", only 2 out of 99 identified moves were incorrect because of the pose estimation output (error rate of 2.02%). One example can be seen in Figure 22. Overall, one can deem the pose estimation approach as satisfactory as other steps given the similar effect it had at introducing errors in the process chain. While the error rate in key frame detection can be considered high, the algorithm was able to still detect all important frames and at the same time significantly reduce the number of frames to be analysed.

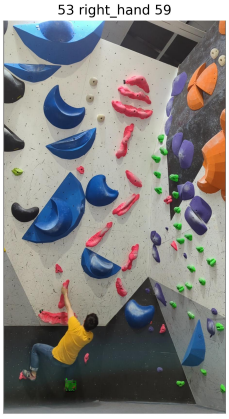

Figure 22: Incorrect move description caused by wrong estimated pose.

6.1 Areas of improvement

Despite obtaining satisfactory results, the output of the process does not produce completely accurate answers for the video samples it has been indirectly tailored too. Furthermore, one of the questions posed in 2 remains unanswered. Therefore, a list of possible areas of improvement are proposed below.

6.1.1 Climbing solution validity

The implemented process fails to check if the computed "beta" is a valid solution for the indoor climbing problem. By analysing the description of moves generated it could be checked if both hands reach the top hold. The solution right now does not provide a process to define such hold or to compute it. For computational reasons, feet tracking imprecision and time constraints presented in 2.0.2, the process fails too in incorporating a robust and tested solution to check if invalid holds were used in the climbing action (that is, holds not belonging to the problem being solved). Checking for correct problem finishing and move validity could be two areas of improvement given 2.

6.1.2 Hold detection and segmentation

The created dataset in 4.1.4 produces good enough results using fine-tuning, but a bigger dataset of holds would greatly improve the performance shown in Table 2. The current model does not work as effectively in high complexity situations such as holds over volumes, holds that are really close together and scenes with a high number of individual instances. Adding more samples to the training dataset could help the current model generalize better in such situations.

6.1.3 Hold matching

Hold matching was the step in the implementation that introduced errors that highly affected final inference. The step could be improved by continuing to built on color information, such as using CSIFT. Another technique that could produce better results would be leveraging the features created by vision transformers after their success in other areas. By building a small dataset of matched pairs of holds, one could train or fine-tune a classification model on the generated features from the holds bounding boxes. Siamese networks based on triplet loss [41, 61] have been very successful at different types of pair matching. Finally, similar to computing a RANSAC homography model for matched key-points in SIFT, one could take advantage of the consistent spatial relation holds belonging to the same route have. That is, by using each hold's spatial relations to other holds and comparing said relations to the ones

computed in the reference image, one could identify holds that do not conform with estimated spatial model similarly to how RANSAC detects outliers.

6.1.4 Pose estimation

Given the success of fine-tuning a COCO trained model in 4.1, one technique for improving HPE for climbing would be to create a small dataset of annotated videos similar to [43] and tune the current approach. Furthermore, applying a robust pose estimation model that had the ability to provide detailed estimation of joints such as hands and feet (Figure 23) could improve the results of 4.5. One possible solution could be to continue to use performance-oriented pose estimation models such as YOLO to first identify key-frames and then apply a more accuracy-oriented models such as the mentioned OpenPose or Mask-RCNN on the detected key-frames.

Figure 23: Detailed hand pose estimation by OpenPose. Source: [17]

6.1.5 Key Frame detection

To greatly reduce the amount of false positives in the key frame detection algorithm and to avoid defining the movement threshold based on the biggest moves of the joint throughout the action, a moving average thresholding function could be implemented to better detect small moves or moves at lower speeds and to decrease the detection of pose estimation noise as key frames.

6.1.6 Final inference

All presented areas of improvement could greatly improve the results of the final inference step. The "beta" move identification and description step could be one of the most difficult steps to improve given the need of very specific climbing related context and rules. To improve said step from a very basic rule based approach, one could incorporate context knowledge using priors to shift to a more probabilistic based solution. Research in action recognition in other sports or other human actions [32] could a source of possible better solutions.

7 Conclusion

The general aim of this dissertation was to investigate how one could make use of computer vision to generate a more descriptive analysis of indoor climbing. To that end, the methodology and results successfully show a proposal which leverages vision techniques to generate significantly good answers to two out of the three posed questions, taking into account certain limitations. The different methodology steps successfully use different traditional and modern techniques to solve its specific problem in a way that allows the entire process to cohesively integrate them. While the evaluation metrics shown in Chapter 5 demonstrate a sufficiently good enough accuracy in obtaining a good analysis of indoor climbing, the biggest conclusion of the project is that it shows how computer vision can bring viable solutions to areas where other approaches have failed or have been insufficient. The ability of computer vision techniques to understand visual information and context approximates human eye vision and reasoning for specific enclosed and defined tasks. Going back to the problem in hand, different areas of improvement are proposed to anyone wanting to contribute on the project. On an effort to do that, the author open-sources all code, data samples and trained models to contribute to the public body of knowledge of indoor climbing analysis, the use of computer vision in sports analytics and overall the field of computer vision that has helped so much in the implementation of this very project.

8 References

[1] Alaa E Abdel-Hakim and Aly A Farag. Csift: A sift descriptor with color invariant characteristics. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1978–1983. Ieee, 2006.

[2] Mammut (Advertisement). Mammut climbax: The world's first climbing tracker. *UKClimbing*, June 2021.

[3] Ankur Agarwal and Bill Triggs. 3d human pose from silhouettes by relevance vector regression. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[4] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *IEEE transactions on pattern analysis and machine intelligence*, 28(1):44–58, 2005.

[5] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *2009 IEEE conference on computer vision and pattern recognition*, pages 1014–1021. IEEE, 2009.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Lecture notes in computer science*, 3951:404–417, 2006.

[7] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking. *arXiv preprint arXiv:2006.10204*, 2020.

[8] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade: Second Edition*, pages 437–478, 2012.

[9] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[10] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9157–9166, 2019.

[11] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[12] Bonnie de Bruijn. Climbalytics: Data-tracking tech for climbers, route setters and gym owners. *Gripped Magazine*, May 2019.

[13] Roberto Brunelli. *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[14] Michelle Bruton. Interest in climbing and gym memberships have spiked following sport's tokyo olympics debut. *Forbes*, Dec 2021.

[15] Matija Buric, Miran Pobar, and Marina Ivasic-Kos. Ball detection using yolo and mask r-cnn. In *2018 International conference on computational science and computational intelligence (CSCI)*, pages 319–323. IEEE, 2018.

[16] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[17] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[18] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[21] Vladislavs Dovgalecs, Jérémie Boulanger, Dominic Orth, Romain Hérault, Jean François Coeurjolly, Keith Davids, and Ludovic Seifert. Movement phase detection in climbing. *Sports Technology*, 7(3-4):174–182, 2014.

[22] Sarah Ekaireb, Mohammad Ali Khan, Prem Pathuri, Priyanka Haresh Bhatia, Ripunjay Sharma, and Neha Manjunath-murkal. Computer vision based indoor rock climbing analysis.

[23] Hao-Shu Fang, Jiefeng Li, Hongyang Tang, Chao Xu, Haoyi Zhu, Yuliang Xiu, Yong-Lu Li, and Cewu Lu. Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[24] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[25] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[26] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[27] Ben Gray. You can now track single pitch, indoor climbing and bouldering with the coros vertix 2 adventure gps watch. *ADAPT Network*, Dec 2022.

[28] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7297–7306, 2018.

[29] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2315–2324, 2016.

[30] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[31] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[32] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. Going deeper into action recognition: A survey. *Image and vision computing*, 60:4–21, 2017.

[33] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, January 2023.

[34] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[35] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[37] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[38] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019.

[39] Debapriya Maji, Soyeb Nagori, Manu Mathew, and Deepak Poddar. Yolopose: Enhancing yolo for multi person pose estimation using object keypoint similarity loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2637–2646, 2022.

[40] Mehrtash Manafifard, Hamid Ebadi, and H Abrishami Moghaddam. A survey on player tracking in soccer videos. *Computer Vision and Image Understanding*, 159:19–46, 2017.

[41] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 378–383. IEEE, 2016.

[42] Banoth Thulasya Naik, Mohammad Farukh Hashmi, and Neeraj Dhanraj Bokde. A comprehensive review of computer vision in sports: Open issues, future trends and research directions. *Applied Sciences*, 12(9):4429, 2022.

[43] Mahdiar Nekoui, Fidel Omar Tito Cruz, and Li Cheng. Falcons: Fast learner-grader for contorted poses in sports. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 900–901, 2020.

[44] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, pages 483–499. Springer, 2016.

[45] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1*, pages 128–144. Springer, 2020.

[46] Dominik Pandurevic, Paweł Draga, Alexander Sutor, and Klaus Hochradel. Analysis of competition and training videos of speed climbing athletes using feature and human body keypoint detection algorithms. *Sensors*, 22(6):2251, 2022.

[47] Thomas Pock, Daniel Cremers, Horst Bischof, and Antonin Chambolle. An algorithm for minimizing the mumford-shah functional. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1133–1140. IEEE, 2009.

[48] Vignesh Ramanathan, Jonathan Huang, Sami Abu-El-Haija, Alexander Gorban, Kevin Murphy, and Li Fei-Fei. Detecting events and key actors in multi-person videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3043–3053, 2016.

[49] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[50] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[51] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[52] Angie K Reyes, Juan C Caicedo, and Jorge E Camargo. Fine-tuning deep convolutional networks for plant recognition. *CLEF (Working Notes)*, 1391:467–475, 2015.

[53] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[54] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[55] Shakhnarovich, Viola, and Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 750–757. IEEE, 2003.

[56] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[57] FTA Stapel. A heuristic approach to indoor rock climbing route generation. B.S. thesis, University of Twente, 2020.

[58] Cheng-Hao Tai, Aaron Wu, and Rafael Hinojosa. Graph neural networks in classifying rock climbing difficulties. *Student project report, CS*, 230, 2020.

[59] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[61] Alexandre Vilcek, Seth Mottaghinejad, Steven Shi, Ketki Gupte, Sujitha Pasumarty, Linsey Pang, and Prakhar Mehrotra. Transformer-based deep siamese network for at-scale product matching and one-shot hierarchy classification. 2018.

[62] Marie-Sophie von Braun, Patrick Frenzel, Christian Kading, and Mirco Fuchs. Utilizing mask r-cnn for waterline detection in canoe sprint video analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 876–877, 2020.

[63] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023.

[64] Xinchao Wang, Vitaly Ablavsky, Horesh Ben Shitrit, and Pascal Fua. Take your eyes off the ball: Improving ball-tracking by focusing on team play. *Computer Vision and Image Understanding*, 119:102–115, 2014.

[65] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[66] Whipper. Mbientlab launches the whipper, a wearable climbing tracker that brings tech to the crag. *Explorersweb*, Apr 2016.

[67] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *arXiv preprint arXiv:2204.12484*, 2022.

[68] Chong Zhou. *Yolact++ Better Real-Time Instance Segmentation*. University of California, Davis, 2020.