Heuristics2Annotate

Efficient Annotation of Large-Scale Marathon Dataset For Bounding Box Regression

Pranjal Singh Rajput

Supervised by:
Prof. Dr. Jan van Gemert

Co-supervised by:
Yeshwanth Napolean

Heuristics2Annotate

Efficient Annotation of Large-Scale Marathon Dataset For Bounding Box Regression

by

Pranjal Singh Rajput

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday October 20, 2020 at 10:00 AM.

Student number:	4788087	
Project duration:	January 22, 2020 – October 20, 2020	
Thesis committee:	Prof. Dr. Jan van Gemert,	TU Delft, Supervisor
	Prof. Dr. Silvia Pintea,	TU Delft
	Prof. Dr. Arjan van Genderen,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Delft
University of
Technology

Preface

This report presents the work of my master's thesis project on the topic of Efficient Annotation of Large-Scale Marathon Dataset For Bounding Box Regression. This research was conducted at Computer Vision Lab of Pattern Recognition and Bioinformatics Group in TU Delft under the supervision of Dr. Jan Van Gemert.

Foremost, I would like to thank Dr. Jan Van Gemert for his invaluable guidance and support throughout my research work and in writing of the thesis. I would also like to thank Yeshwanth Napolean for being so kind and patient with me. His continuous support and guidance was really helpful during the entire research process. I would also like to extend my sincere gratitude towards Dr. Silvia Pintea and Dr. Arjan van Genderen for taking out their valuable time to assess my work.

I would like to thank my Guruji, for his guidance and his immense blessings. Next, I would like to thank my family for their infinite love and support, especially to my Mom, Dad and my Sister for their endless sacrifices, and for always believing in me. I will always be indebted to them.

I would like to thank Shivangi for being an avid listener, with whom I benefitted from brainstorming the issues. I was fortunate to find some beautiful souls in Delft. I like to thank my friends Anurag, Sonnya, and Kanya for making the time at Delft, fun and memorable. I also thank my dear friend Parijat for always standing by me in my difficult times.

Pranjal Singh Rajput
Delft, October 2020

Contents

1

Scientific Paper

Heuristics2Annotate: Efficient Annotation of Large-Scale Marathon Dataset For Bounding Box Regression

Pranjal Singh Rajput
Computer Vision Lab
TU Delft

Supervisor: Yeshwanth Napolean
Computer Vision Lab
TU Delft

Supervisor: Jan van Gemert
Computer Vision Lab
TU Delft

Abstract

Annotating a large-scale in-the-wild person re-identification dataset especially of marathon runners is a challenging task. The variations in the scenarios such as camera viewpoints, resolution, occlusion, and illumination make the problem non-trivial. Manually annotating bounding boxes in such large-scale datasets is cost-inefficient. Additionally, due to crowdedness and occlusion in the videos, aligning the identity of runners across multiple disjoint cameras is a challenge. We collected a novel large-scale in-the-wild video dataset of marathon runners. The dataset consists of hours of recording of thousands of runners captured using 42 hand-held smartphone cameras and covering real-world scenarios. Due to the presence of crowdedness and occlusion in the videos, the annotation of runners becomes a challenging task. We propose a new scheme for tackling the challenges in the annotation of such large dataset. Our technique reduces the overall cost of annotation in terms of time as well as budget. We demonstrate performing fps analysis to reduce the effort and time of annotation. We investigate several annotation methods for efficiently generating tight bounding boxes. Our results prove that interpolating bounding boxes between keyframes is the most efficient method of bounding box generation amongst several other methods and is 3x times faster than the naive baseline method. We introduce a novel way of aligning the identity of runners in disjoint cameras. Our inter-camera alignment tool integrated with the state-of-the-art person re-id system proves to be sufficient and effective in the alignment of the runners across multiple cameras with non-overlapping views. Our proposed framework of annotation reduces the annotation cost of the dataset by a factor of 16x, also effectively aligning 93.64% of the runners in the cross-camera setting.

Index Terms — dataset, annotation, computer vision, cameras, benchmarks, object detection, object tracking, interpolation, marathon, bib detection, cross-camera alignment, person re-identification

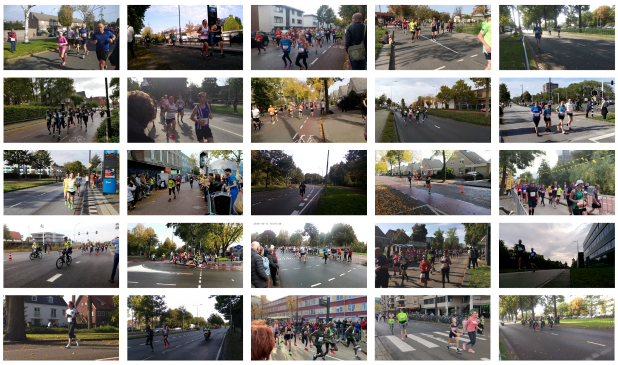


Figure 1: **Sample images of videos recorded from multiple disjoint cameras.** Different scenarios including variation in light intensity, occlusion, shadows, and different camera poses and angles make the task of runner detection non-trivial.

1. Introduction

Marathon events are gaining popularity due to cognizance towards health awareness and motivation in improving psychological well being. The professional runners always aim in improving their performances. Nowadays, athletes record their running data for their personal health monitoring. Also, some event organizers record the race along with the individual athlete's statistics for their performance tracking. Sometimes runners are interested in retrieving their moments of appearance in different time frames during the event to monitor their performance during the race. Retrieving such images and videos for thousands of participants from numerous videos is a challenging task. Individually runners can be tracked using a personalized GPS tracker, but the videos and images contain multiple athletes at a time.

Searching a runner with just a single portrait, in thousands of videos with hundred hours of recordings from disjoint cameras is a laborious task. Every runner has a unique bibId, a unique number attached in front of its uniform at the chest. If the bibId is visible, then text recognition models can be used to identify and locate the athlete. We can also use computer vision-based person re-identification models to identify the runner if a clear portrait of the runner is available. But in the real scenarios, the athletes are partially or fully occluded, having similar clothing, different poses, variable illumination in images/videos making it challenging even for state-of-the-art computer vision techniques to find all the appearances of a runner. For improving the performance of computer vision algorithms on such research problems, a huge amount of data covering the real-world scenarios is needed.

With the increase in the number of smartphones embedded with high-quality cameras, events are easily captured and shared on the internet. Due to which there is a large availability of data. The advent of datasets of increasing scale has made a significant contribution to the advancements in computer vision. However, the scaling of such large scale datasets is hindered because of the cost and difficulty in the annotation of these large datasets with complex scenes and multiple objects. This has obstructed the progress in various deep learning tasks such as person re-identification systems that currently fail to generalize to any scenario. Another example is of text-detector systems, that fails in real-world scenarios like varying illumination, poor resolution, etc.

With the fast-paced ongoing research in machine learning, the main focus is shifting towards the applications. Deep neural networks still don't have performance saturation, and they still benefit from more data (Sun et al. 2017). Annotation of objects in the image is an equivalently time-consuming task as any other machine learning topic. According to some sources, 80% of the AI project development time is spent on data preparation [3]. ImageNet [23] consists of almost 14 million sample images. It takes around 42 seconds per bounding-box by crowdsourcing using the Mechanical Turk annotation tools [25]. So, it can be imagined how much time it will take to annotate the entire dataset. Another example is of VIRAT dataset [17], consisting of recordings from surveillance cameras. It cost around tens of thousands of dollars to annotate the VIRAT dataset. It took around 20,000 hours to annotate object class labels and almost 5,000 additional hours for bounding box annotation in the COCO dataset [13].

Several other challenges persist in the collection and preparation of a large-scale representative dataset such as that of a Marathon Event, where the data is recorded using multiple cameras at different locations, containing hours of recordings of thousands of runners. Annotating such datasets with a large number of images and videos becomes difficult due to the presence of different viewpoints, varying illumination, varying camera resolution, occlusions, etc. Due to these problems, it becomes hard to annotate the runners in videos having crowded scenes and recorded at poor resolution or lighting conditions. As it is very labor-intensive to annotate such large datasets, they are generally annotated using traditional ways by crowdsourcing on platforms such as Amazon Mechanical Turk. Therefore. efficiently annotating a dataset of marathon runners is non-trivial. This area of research still needs to be explored.

Several annotation tools have been proposed in the past to accelerate the annotation process and also reducing human efforts. Almost all the tools allow the user to annotate the object of interest using a bounding box rectangle. Generally, these boxes are drawn mostly in all the video frames where the object of interest is present. Some of these tools are supported with machine learning and computer vision methods such as object detection, action recognition, object tracking, etc. to support automatic or semi-automatic annotation. However, the main drawback of such algorithms is that they are domain-specific and lacks robustness in case the complexity level in the dataset is high.

In this paper, we study the heuristics to efficiently annotate a novel in-the-wild large-scale dataset of Marathon runners consisting of 3,264 videos of almost 86 hrs of recording, covering 9,834 runners captured using 42 hand-held smartphone cameras. We explore the ways to efficiently annotate the dataset at the same time reducing the human efforts, and the cost of annotation including time as well as the budget. We investigate ways to efficiently generate bounding box annotations. We propose a novel method for aligning the identities of runners across multiple cameras. Our approach reduced the overall cost of annotation substantially by a factor of **16x** compared to the baseline method of annotation, at the same time ensuring the alignment of **93.64%** of the total number of runners, in the cross-camera setting.

In summary, our contributions are: We propose the annotation method of a novel dataset of Marathon runners, consisting of hours of recordings of thousands of marathon runners captured using 42 hand-held smartphone cameras. We followed a three-stage approach in the complete annotation process of the dataset. In the first stage, we study the effect of frame extraction rate in the overall time and accuracy of annotation. We study the video annotation at different fps rate to verify if it helps in reducing the overall cost. In the second stage, we investigate different object annotation methods to annotate a runner, which is the main object of interest in our dataset. We used different computer vision-based methods such as object detection, object tracking, and box interpolation to see if they generate efficient bounding boxes in minimum time. In the final stage, we try to find ways to align the runners' identity in the cross-camera setting. We propose a runners' dashboard for the cross-camera alignment. Additionally, we show how to intelligently add noise in achieving the cross-camera alignment by using state-of-the-art person-reid methods.

2. Related work

Bounding box annotation. There have been many methods proposed in the literature, for quickly generating bounding boxes. Adhikari et al. 2018, used state of the art object detector models to generate tight bounding boxes [2]. Importantly, their object detector models are pre-trained on MS COCO [13] dataset. Additionally, they train the model on a small subset of the dataset that is pre-labeled manually. Papadopoulos et al. 2016 [18] used the human-machine collaboration to generate high-quality bounding boxes. The idea is to use human verification for correcting the detections and use active learning for re-training the object detectors. One approach is to annotate only a sparse set of boxes and linearly interpolate the remaining boxes between them [28]. In VATIC [27], authors used shortest-path interpolation between manual annotations to generate boxes. Gygli et al. 2019 [7], proposed efficiently annotating objects by clicking at the four corners of an object and speaking its label. Manen et al. [14], proposed path supervision to annotate large-scale datasets for multi-object tracking. The authors claim that their method is efficient as it can be used to turn the watching time into annotation time. We will be investigating how we can use the object detection, box interpolation method, and object tracking techniques in efficiently generating the bounding boxes.

Marathon Datasets. To the best of our knowledge, no marathon dataset has been the introduction in the literature till now. Our dataset is the first of its own kind. A similar work has been done in the past by Napolean et al. 2019 [16], where the data is collected from a university campus marathon event. This dataset is quite a smaller version of our dataset, containing recordings of a 5km marathon race. There are a total of 262 runners captured using 9 unconstrained hand-held smartphone cameras. However, the closest resemblance of our dataset is with the person re-identification datasets [6, 9, 10, 11, 12, 29]. Like person re-id datasets, our dataset also contains recordings of a number of runners appearing in multiple cameras installed at different locations. Our dataset differs from the re-id datasets in the sense that the main object of interest in our dataset is runner, whereas, in re-id datasets, all the persons appearing in the videos are the main object of interest.

3. Eindhoven Marathon Dataset

Some datasets for marathon runners with recordings at fixed camera locations are available, however, these videos are recorded with High-Quality DSLR cameras and thus fail to generalize to real-world scenarios due to inherent domain shift. To facilitate research for in-the-wild videos, we propose a novel video dataset of marathon runners captured using handheld smartphone cameras. In total, 85 hours of videos were recorded in High Definition (HD) quality. The variability in terms of occlusion, light intensity variation, resolution, crowdedness, background clutter, and pose, etc, makes it even more challenging to learn high performing models. The dataset is also timestamped with the GPS coordinates of the location of the recording which will help keep track of the location of runners/cameras and can be useful in reconstructing the event.

Two marathons have been recorded, namely Half-marathon (21.5km) and Full-marathon (42km). The recording is done by the volunteers at 42 different locations covering the full-marathon track.

3.1. Dataset Collection

In this section, we will discuss the procedure we followed in data collection and data gathering.

3.1.1 Organizational Logistics

The Marathon event was organized in the city of Eindhoven, The Netherlands. Full-marathon and the half-marathon races are recorded as these events cover the entire marathon track, along with the maximum number of participants. We expected to cover approximately 10,000 marathon runners.

For coverage of the entire full marathon track, we recruited in a total of 42 volunteers. The volunteers were asked to use their personal *smartphone cameras* for recording the event.

3.1.2 Setup

In this section, we will discuss the recording setup and the instructions given for recording the event.

1. Only smartphones were used for the recording and no professional cameras, DSLR's or GoPro's.
2. Total memory space available in the devices used for recording was at least 20-30 GB
3. Video recording resolution: 720p
4. Frame rate: 30fps
5. GPS enabled during recording
6. Recording mode: LANDSCAPE
7. Camera holders/Tripods used: None

All the 42 volunteers were asked to record the event. A map was created and a location was assigned to each volunteer, as shown in Figure 2 .The recording is done at the start point and the finish point along with all the intermediate locations separated by 1 km from each other. Thirty volunteers were assigned only one coordinate, denoted by 'Point-X' and 12 volunteers were assigned two

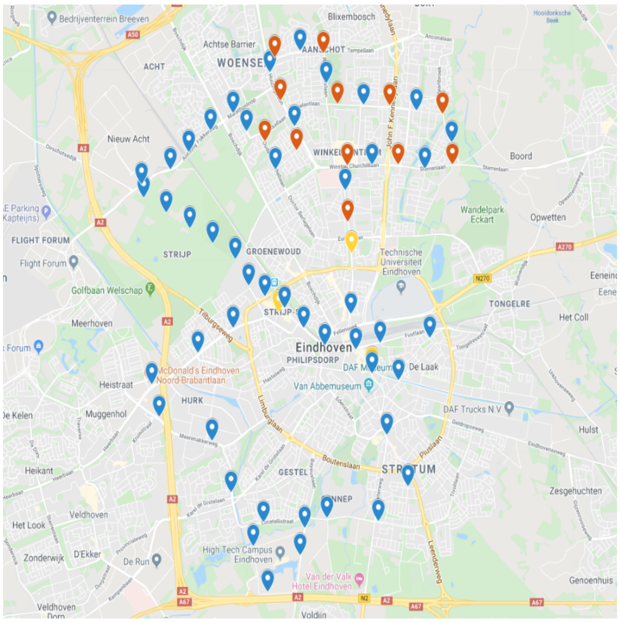


Figure 2: **A map showing the coordinates of recording locations in the marathon track.** Each Full-marathon (Blue marker) and Half-marathon (Red marker) point is at a distance of 1km from its next point of the same color. Yellow markers represents the start and the end point of the event.

points namely 'Point-X' and 'FMP-Y'. The first 12 volunteers after recording the full-marathon at 'Point-X', shifted to their next assigned coordinate 'FMP Y' after they are done recording at the former one. The map and the assigned coordinates can be seen in Figure 2. Point-X is represented by blue markers, whereas FMP-Y is represented by red-marker. The start and the end coordinates are represented by yellow markers.

Almost all the recorded videos are timestamped and also have GPS information at the time of recording. Cameras were hand-held during the recording, and no stands/tripods are used. Therefore, the recordings are shaky rather than static.

3.1.3 Execution

The recording was started whenever a runner was visible for the first time. The recording was stopped when no runner was visible in the frame. The recording was done from the front side of the runner so that bibId was visible and could be recorded in the video. The backside of the runner was not recorded as there was no bib-number on the back of the runner. Our setup in terms of camera position and the athletes resembles Figure 3, in which the runners ran towards the camera. Initially, there was a considerable distance between the camera and the runners, but the runners progressively got closer to the camera.

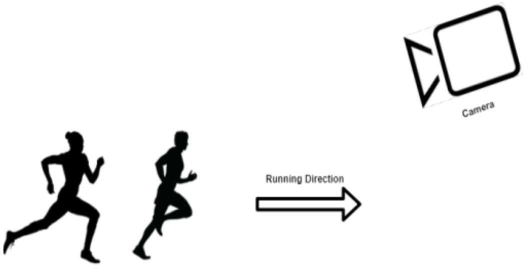


Figure 3: **Illustration of how video recording was configured.** The recording is done from the front side to capture the bibId attached to the uniform of the runner. The recording is started when a runner is visible for the first time and is done until the runner leaves the camera frame [16].

3.2. Data Gathering

3.2.1 Collecting the recorded data

After the videos have been recorded, they needed to be collected at one single storage point. For that, we used a webserver with a storage capacity of 4TB. A website was created for the volunteers to upload their collected data onto the webserver. The participating athletes were also asked to upload the strava details, that is activity and performance tracking details of athletes, that they have collected during their run.

3.2.2 Web Scraping

Figure 4: **Sample images of marathon-event scraped from official website [5].** These images are not part of the dataset, and are publicly available on the website.

To collect the detailed information of all the participants of the full-marathon and half-marathon, web-scraping was done from the official website of the Marathon Eindhoven event [5]. Information such as runner name, runner bib-id, age, country, finish times after distances(in meters) 5k, 10k, 15k, 20k, 25k, 30k, 35k, 40k, and 42k are collected. A complete list of field names is mentioned in Table 7. Event im-

Video Metadata	Value
FileName	VID_20191013_114355.mp4
FileSize(MB)	21.92
FileType	MP4
Duration	14.34
VideoFrameRate	30
ImageSize	1280x720
TrackCreateDate	2019:10:13 09:43:55
GPSCoordinates	51.4839 5.4642

Table 1: **A sample of collected video metadata.** Video metadata such as FileName, FileSize, FileType, Duration VideoFrameRate, ImageSize, TrackCreateDate, GPSCoordinates are collected.

ages were also collected from the official website [5], sample images can be seen in Figure 4. More information on data scraping is provided in B.

3.2.3 Collecting Video Metadata

After recording the data and collecting it on a common platform, the metadata of all the videos is read using a python library *exifTool* [19]. Using exifTool [19] we collected information of a video such as *FileName, FileSize, FileType, Duration VideoFrameRate, ImageSize, TrackCreateDate, GPSCoordinates*. A sample of metadata of a collected video is shown in Table 1.

3.3. Data Analysis

Statistics	
Total no. of cameras	42
No. of videos recorded	3,264
Mean duration of videos	94.13 sec
Standard Deviation	246.78
Total duration of recording	85.34 hrs
Total frames in dataset	9,216,813
Runners in full-marathon	2,423
Runners in half-marathon	7,411
Total runners	9,834

Table 2: **Statistics of the Eindhoven Marathon dataset.** Total 9,834 runners are captured using 42 cameras. The dataset consists of 3,264 videos of 85.34 hrs of recording. The mean duration of the videos is around 95 seconds.

The collected metadata is analyzed and mentioned in Table 2 and Figure 5. As can be seen in Figure 5, most of the videos are short in length with average duration around 95 seconds. All videos are recorded at 30fps and at High-Definition (HD) resolution. Therefore, there are a total of 9,216,813 frames in the dataset. The total video duration is of 85.34 hrs. A total of 9,834 runners are covered including full-marathon and half-marathon.

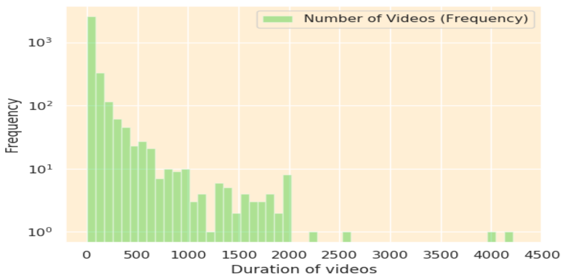


Figure 5: **Analyzing video duration vs the total number of videos of that duration (frequency).** It can be seen from the figure that the mean duration of the videos is centered around 95 seconds

3.4. General Data Protection Regulation (GDPR)

The data is recorded and collected, in adherence to the GDPR guidelines [1]. In general, the main focus of data recording was the marathon event which is organized publicly every year. None of the data subjects is focused or recorded individually. Although, the participants gave their consent of data recording and sharing to the MyLaps [15], an official organizer of the Eindhoven Marathon event, that is also working in collaboration with the Computer Vision Lab [4] at the university. We adhered to the privacy regulations, by blurring/hiding the faces of the people appearing in the images. Also, the data is not uploaded or shared publicly.

4. Methodology

In this section, we will discuss the general annotation procedure and the pipeline that is used in our experiments. Then we will cover the method used to derive a sample dataset, followed by a discussion over the data collection procedure and the Inter-Camera Alignment (ICA) tool. Lastly, we will elaborate on the metrics that are used in evaluating the performance of the methods.

4.1. Annotation Procedure

Video Annotation. Annotating videos is similar to image annotation. It involves mainly two steps: i) Extracting the frames from videos, ii) Annotation of individual frames.

The overall annotation pipeline is shown in Figure 6. Firstly, frames are extracted at a specific extraction rate. Then, each runner in individual frame are annotated using

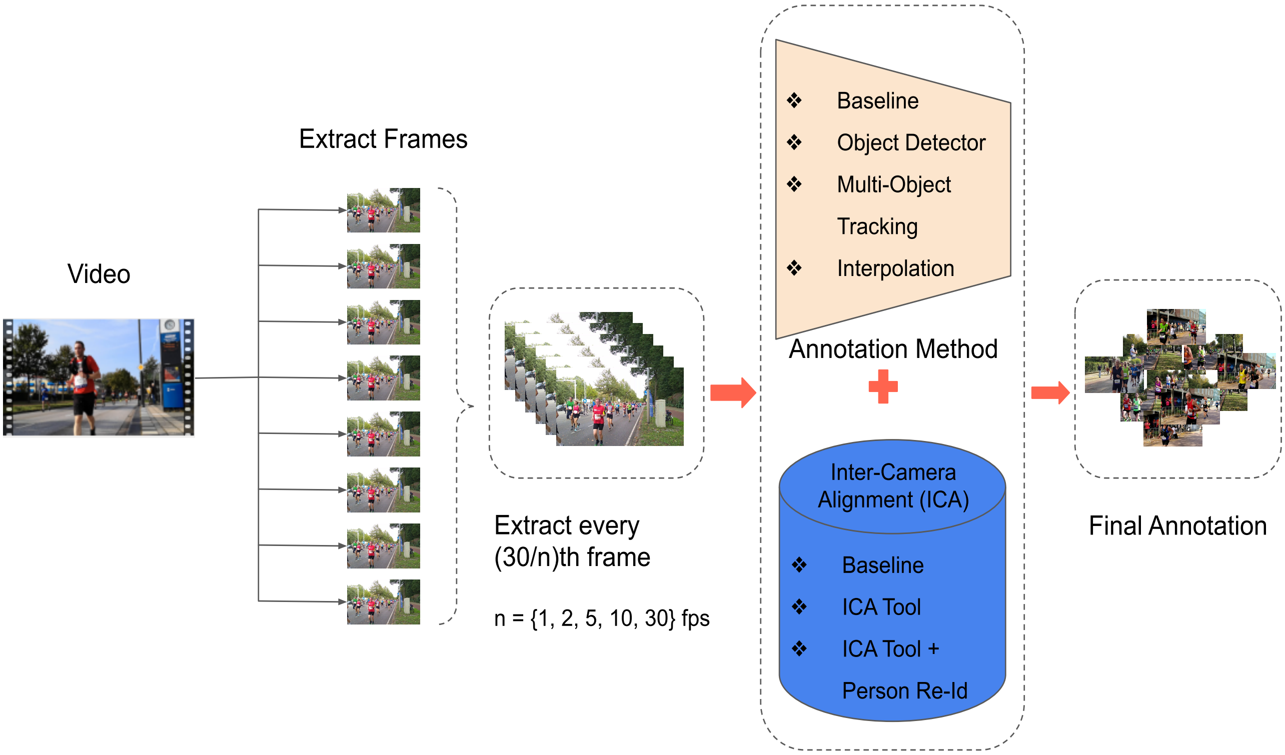


Figure 6: **Overall annotation pipeline.** The videos are recorded at the rate of 30 frames per second (fps). Firstly, all the frames are extracted from videos. Then, we analyze the effect of *fps* on annotation time, by sampling frames at different fps. The sampled data is then annotated using bounding box regression, and the runners are aligned across multiple disjoint cameras, to get final annotation.

the proposed annotation method. Later, the identities of the runners are aligned using the proposed method for inter-camera alignment, to get the final annotations.

Manually annotating runners. The main object of interest here is the runner. We used the basic bounding box annotation around the runners. Every runner is assigned a unique runner id (*bibId*) which is attached on the runner's chest. The bibId of the runner is used as the label which is manually added by the annotator. Firstly, a runner is selected, then we try to find a frame in which the runner's bibId is visible. We then move back to the frame number where the runner is visible for the first time in the camera. We start putting bounding boxes around the runner in all the frames in which he/she is visible until the runner leaves the camera frame. We followed runner-wise annotation instead of frame-wise annotating all objects of interest.

4.2. Sample Dataset Creation.

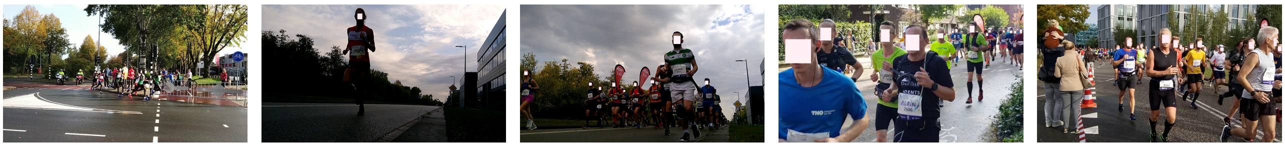
The large size of the unannotated dataset makes it difficult for validating the experiments. Hence, a sample dataset is required, that is a true representative of the actual dataset. A smaller version of the dataset is created by down-sampling the original one and covering different scenarios present in different video locations.

Scores Assignment for Locations. Firstly, variations in different scenarios are analyzed at all the recording locations. The scenarios are divided into the following five categories: *i) the resolution, ii) lighting condition, iii) recording angle, iv) occlusion, v) number of crowded videos*. Some examples of these scenarios are shown in Figure 7.

Different scenarios and their score distribution is shown in Table 3. The total score of a location is the sum of scores in individual categories of different scenarios. Location score $S_{location}$ is given by,

$$\begin{aligned} S_{\text{location}} = S_{\text{lighting}} + S_{\text{resolution}} + S_{\text{recordingAngle}} \\ + S_{\text{occlusion}} + S_{\text{noOfCrowdedVideos}}. \end{aligned} \tag{1}$$

Consider an example location 'X' having videos recorded in poor lighting conditions(2), with good resolution(5), recording angle is front(5), moderate occlusion(3) & number of videos having too many crowded runners is



(a) Resolution (b) Lighting Condition (c) Recording Angle (d) Occlusion (e) Crowdedness

Figure 7: **The five main scenarios present in the dataset** are i) resolution, ii) lighting condition, iii) recording angle, iv) occlusion, v) number of crowded videos.

Scores	Lighting	Resolution	Recording Angle	Occlusion	# Crowded Videos
1	Very poor	Very poor	Front+Side+Static+Upside-Down	Very high	Very large
2	Poor	Poor	Side	High	Large
3	Moderate	Moderate	Front+Downside+Static	Moderate	Moderate
4	Good	Good	Front+Side	Low	Less
5	Very good	Very good	Front	Very low	Very less

Table 3: **Different scenarios in the dataset and their score distribution.** All video locations are analyzed and penalized as per the scenarios of the video recording at that location. A location score is the sum of scores in different scenarios.

moderate(3), then the total score is $S_X = 2+5+5+3+3 = 18$.

All the 42 recorded locations are given overall score S = [0, 25], where for each of the above-mentioned categories we can have a score in the range [0, 5]. For more details, refer to A. The frequency vs score distribution of the entire dataset is shown in Figure 8. There are a total of 16 different score values ranging from 8 to 24. Higher scores imply better recording scenarios in videos.

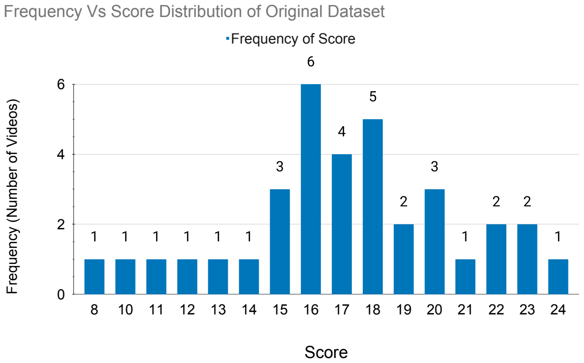


Figure 8: **Score distribution of the entire dataset.** Frequency represents the number of locations with a given score. The distribution of the dataset is a Gaussian.

Kolmogorov-Smirnov (KS) Test. The Kolmogorov–Smirnov test is used to test the similarity of two underlying one-dimensional probability distributions.

The two-sample K–S test is one of the most useful and general non-parametric methods for comparing two samples, as it is sensitive to differences in both location and shape of the empirical cumulative distribution functions of the two samples.

For larger sample sizes, the approximate critical value D_α is given by the Equation 2,

$$D_\alpha = c(\alpha)\sqrt{\frac{n_1 + n_2}{n_1 n_2}}, \tag{2}$$

where, n_1 and n_2 are the sample sizes of the two distributions and α and $c(\alpha)$ are the coefficients given by Table [8]. More information can be found in Appendix A.

How to select locations? To select the desired number of locations for the sample dataset, we use Kolmogorov-Smirnov Test for distribution similarity matching. The score distribution of both the sample dataset and the original dataset is assumed to be Gaussian. Here, we randomly select 6 scores out of a total of 16 and matches the similarity of the selected scores with the original dataset scores. The K-S test calculates the distance between the two distributions, and the sample distribution of the randomly selected 6 score values, having the minimum distance from the actual data distribution is selected.

We selected score values $\{11, 15, 16, 17, 19, 23\}$ for the sample dataset. The distribution of the selected values and the original dataset is shown in Figure 9. Now, 6 locations having these scores are selected. In all, 60 videos are collected, corresponding to the top 20 and middle 3 runners. It is made sure that all these runners appear in all the 6 sam-

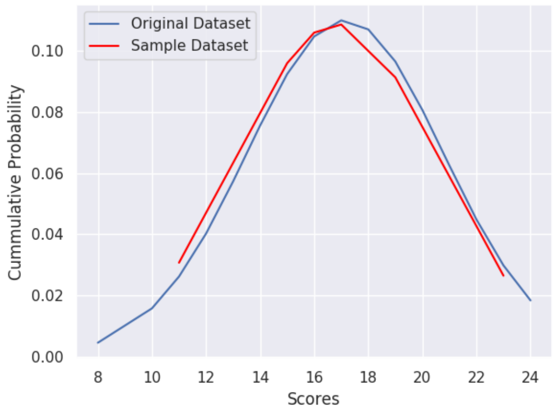


Figure 9: **The graph shows the probability distribution of the scores of the original and the sampled dataset.** The sample score values [11, 15, 16, 17, 19, 23] is having distribution that fits nicely with the actual dataset score distribution.

pled video locations as it's necessary for the validation of the cross-camera alignment. However, due to unavoidable circumstances, there can be runners other than the selected ones, appearing in each sampled location.

4.3. Data Collection

For achieving the cross-camera alignment of runners, our methods require the information of every runner that finished the race. For this, we scraped the images and information of every runner from the official website of marathon [5].

Scraping runners' information. The data is scraped from the official website [5] of the event using Beautiful soup [22] and Selenium Web-driver [24]. For every participant, information such as name, bibId, gender, country, finish times, etc is retrieved. More information is mentioned in appendix B. Apart from this, we also have information corresponding to the time taken by every runner to reach every 5 km distance intervals.

Scraping runners' images. We also scraped images of all the runners available on the website [5]. The image resolution is 233 × 350 pixels. On average, 40 images per runner are scraped. Sample images can be seen in Figure 10.

4.4. Inter-Camera Alignment (ICA) Tool

This section outlines the computation of runners' timeline and the creation of an interactive Inter-Camera Alignment tool for alignment of runners in cross-camera setting.

Computing the runner's timeline. For ICA, it is important to have the athlete's trajectory, to know the whereabouts of the runner at different locations. As discussed in Section 4.3, we have scraped the data corresponding to every runner who finished the race, which also includes the time taken by a runner to reach distances of 5 km interval. The event is recorded at 42 different locations, each separated by 1 km distance, covering the full-marathon track. So we need time values of every runner to reach at these 42 locations. These time values are calculated using *variable average speed concept*.

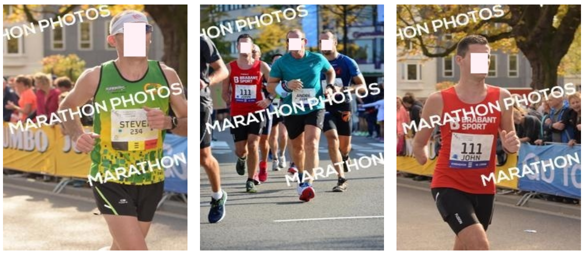


Figure 10: **Sample images of marathon-event scraped from official website [5].** All the scraped images contains a watermark text.

Variable average speed. In general, the average speed of a runner is given by Equation 3,

$$V_{avg} = \frac{\text{Total distance travelled}}{\text{Total time taken}} = \frac{42}{t_{finish}}. \tag{3}$$

But the average speed concept is valid in case the runners is always running with constant speeds in different segment. But as can be seen in Figure 11, only top runners are running with more continuous pace than middle and bottom runners. Therefore, we use variable average speed concept.

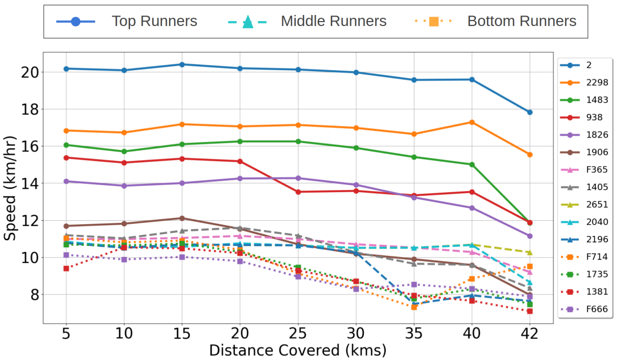


Figure 11: **Speed variations of top 5, middle 5, and bottom 5 runners, chosen randomly from top 800, middle 800, and bottom 800 full-marathon runners respectively.** It can be inferred that runners' speed is not constant throughout the run. Also, the top runners run at a more constant pace than middle and bottom runners.

It means, using different average speeds for a different segment of the race. As shown in Figure 12, the time taken (in green) by the runner to reach every 5km distance is retrieved from the official website [5]. There are cases where the time value is missing for some intermediate checkpoints (in red). In this case, the variable average speed is calculated as follows:

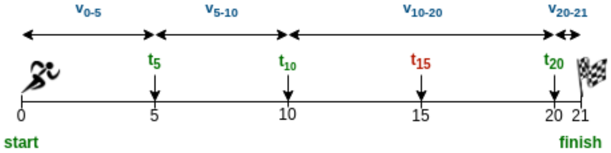


Figure 12: **Using the variable average speed concept to compute the complete timeline of a runner.** Time values in green color are retrieved from the official website, whereas the value in red is missing.

Consider a segment of a track i → j, where i,j are checkpoints and i < j. A segment is defined as the two consecutive checkpoints for which the time values are available. Let's calculate the time (t_x) taken by a runner to reach a point x, where $i < x < j$. For a given segment the variable average speed is given by Equation 4,

$$\mathrm{V_{i\text{-}j}} = \frac{\mathrm{d_{i\text{-}j}}}{\mathrm{t_{i\text{-}j}}} = \frac{\mathrm{d_{0\text{-}j}} - \mathrm{d_{0\text{-}i}}}{\mathrm{t_{0\text{-}j}} - \mathrm{t_{0\text{-}i}}}, \qquad \text{where } i < j. \tag{4}$$

Now, $\mathrm{V_{i\text{-}j}}$ is used to compute the time (t_x) taken by runner to reach point x, where $i < x < j$,

$$\mathrm{t_x} = \frac{\mathrm{d_{i\text{-}x}}}{\mathrm{V_{i\text{-}j}}} = \frac{\mathrm{d_{0\text{-}x}} - \mathrm{d_{0\text{-}i}}}{\mathrm{V_{i\text{-}j}}}, \tag{5}$$

Then the time (t_x) taken by a runner to reach the point x is given by Equation 5. This way, for every runner we calculate the time taken by him/her to reach all the 42 locations.

Runners' dashboard. For aligning the identities of runners across different camera locations, a dashboard of runners is created. Runners' information such as name, bibId, and timeline of the entire event are some of the main features of the tool. It also incorporates a complete timeline of every runner computed using the variable average speed concept. The interactive tool provides features for searching and sorting over all the fields, view timeline diagrams of runners, thereby helping in quickly finding a runner's location at a particular point in time. There are separate tabs for full-marathon and half-marathon runners. There's also a tab containing rules of inter-annotator agreement.

The images of runners included in the tool are scraped from the official website [5] of the event. Only two images per runner are included, to help the annotators in visually recognizing the runner. A runner can also be searched using the name or part of a name, using its bibId or part of a bibId. This partial name or bibId searching facility helps in reducing the search space. The partial searching feature is also valid for time values. All the runners crossing a specific location at a specific range of time can be searched. This way the tool helps in quickly spotting multiple runners running in groups, thereby reducing the time to perform the sequential search. The tool also has the feature to sort over different fields such as name, bibId, and time values. This way, we can find runners in ascending order of the time they crossed a specific location. An outlook of the ICA tool is shown in Figure 13.

4.5. Performance Evaluation Metrics

In this section, we discuss the metrics used in performance evaluation for our experiments.

Terminology:

True Positive (TP). True positive is when the positive class is correctly predicted. Here, a runner is referred to as the positive class.

False Positive (FP). False positive is when the positive class is incorrectly predicted. Here, the incorrect detection of a runner or detection of a non-runner is referred to as the FP.

False Negative (FN). False negative is when the positive class is present but not predicted. Here, FN is when if a runner is present in the frame and is not detected.

An example is shown 14, explaining the cases of TPs, FPs, and FNs.

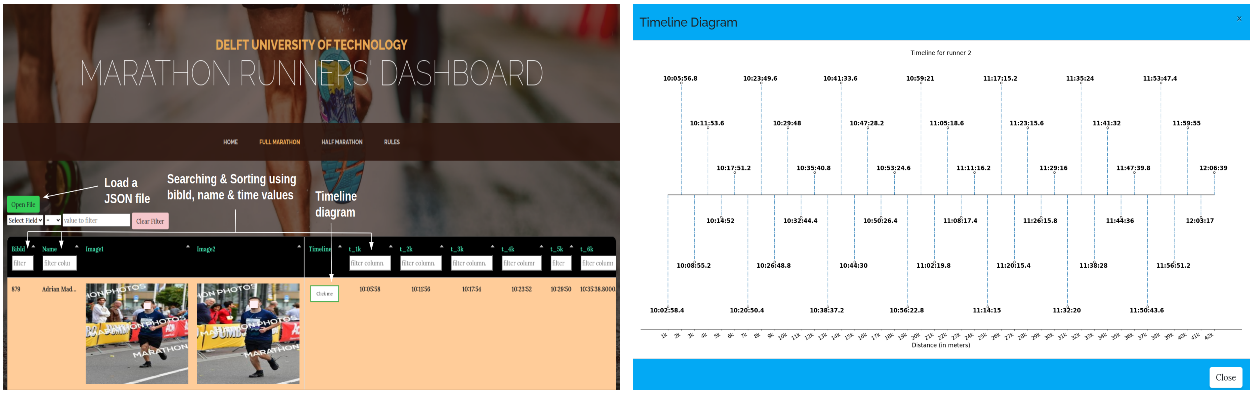
Frame-wise Metrics. As the video annotation is done by extracting the frames and annotating them individually, so we need a frame-wise metric to evaluate the annotation accuracy at the frame level. To determine the accuracy of the bounding box annotation, Intersection Over Union (IoU) is used as the metric. This is given by Equation 15

$$\mathrm{IoU} = \begin{cases} 0, & \textit{if overlap} < 0 \\ \dfrac{\text{Area of Overlap}}{\text{Area of Union}}, & \textit{otherwise.} \end{cases} \tag{6}$$

We define TP as given by Equation 7. If IoU is atleast 0.8, it's TP, else it's FP. If a runner remains undetected then it's a FN.

$$\text{if, } \begin{cases} \mathrm{IoU} \geqslant 0.8, \rightarrow \textit{TP} \\ 0 \leqslant \mathrm{IoU} < 0.8, \rightarrow \textit{FP} \\ \text{No detection}, \rightarrow \textit{FN.} \end{cases} \tag{7}$$

Video-wise Metrics. We use Precision, Recall, and F1-Score to measure the performance of our methods on a given video sample. Precision is defined as,



BibId	Name	Image1	Image2	Timeline	t_1k	t_2k	t_3k	t_4k	t_5k	t_6k
879	Adrian Mad...			Click me	10:05:58	10:11:56	10:17:54	10:23:52	10:29:50	10:35:38.8000

Figure 13: **An outlook of the Inter-Camera Alignment tool developed to align the runners across different camera locations.** The tool incorporates features such as runners' names, bibId, time values, timeline diagram, etc. The timeline diagram helps the annotator in quickly checking the time taken by the runner to reach different locations.

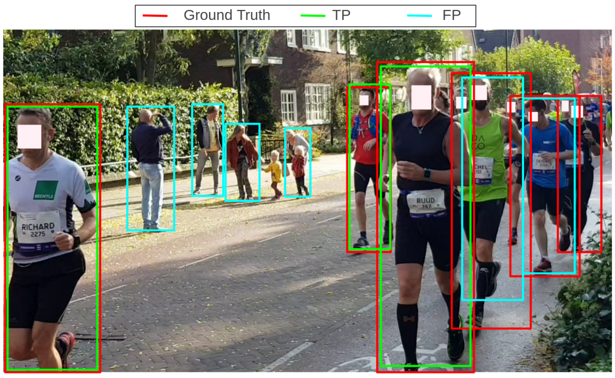


Figure 14: **An example of True positive (TP), False positive (FP), and False negative (FN) case.** The boxes in *red* color represents the ground truth. whereas boxes in *green* are TPs and in *blue* are FPs. Runners that didn't get detect are FNs, represented by only *red* box.

$$\text{Precision } (p) = \frac{\#\,\text{TP}}{\#\,\text{TP} + \#\,\text{FP}}, \tag{8}$$

Recall is defined as,

$$\text{Recall } (r) = \frac{\#\,\text{TP}}{\#\,\text{TP} + \#\,\text{FN}}, \tag{9}$$

F1-Score is defined as,

$$\text{F1-Score} = \frac{2pr}{p + r}. \tag{10}$$

Workload Estimation. The total workload consists of four kinds of manual operations:

1. Removal of False Positives (Detection of non-runners)
2. Addition of False Negatives (Missed detection of runners)
3. Addition of labels
4. Adjustment of boxes (Improper detection of runners)

Total workload = (Remove FP) + (Add FN) + (Add Labels) + (Adjust detections)
And therefore, total annotation time is given by Equation 11,

$$\mathrm{t}_{\text{annotation}} = \mathrm{t}_{\text{removal}} + \mathrm{t}_{\text{addition}} + \mathrm{t}_{\text{adjustment}}. \tag{11}$$

As we have to ensure that all the runners are identified across different cameras, we use percentage of unidentified runners(UR) as the evaluation metric given by 12,

$$\text{UR } \% = \frac{\#\text{ Total runners - }\#\text{ Identified Runners}}{\#\text{ Total runners}} \times 100. \tag{12}$$

5. Experiments

In this section, we will discuss different experiments corresponding to different problems and analyze the results.

How the overall cost of annotation can be reduced in terms of time & budget?

In this section, we will discuss different ways with which we can reduce the cost of annotation.

The budget of the annotation is directly proportional to the time of annotation. Therefore, we define the cost of annotation as the total time taken to annotate the dataset. Annotation time highly depends on the type of annotation we need to perform. In our case, video annotation is performed by extracting the frames. So, there are two ways in which we can reduce the time of annotation: i) *Frame level annotation without bounding box annotation. ii) By intelligently reducing the number of frames to annotate.*

Setup. Due to the massive amount of data, it is hard to perform evaluations on the entire dataset. So, we used four sample videos for the experiments. Each video is of 95 sec duration which is the mean duration of videos in the complete dataset. The videos are randomly selected from 4 different locations, such that they are the true representative of the entire dataset, covering various scenarios as mentioned in Figure 7. Following are the four selected locations with location number 3, 15, 38 and 41 and scores 10, 20, 20 and 23 respectively.

5.1. Exp 1: Frame level annotation

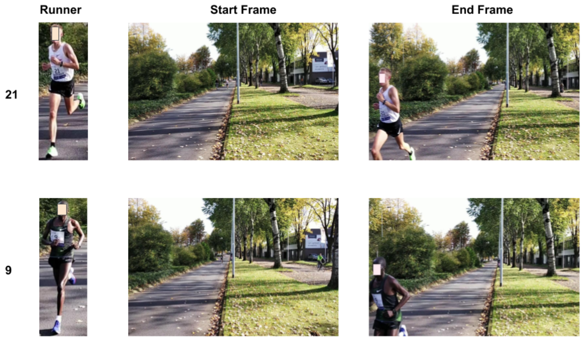


Figure 15: **An example of frame-level annotation.** The annotation is done with the target runner appearing within a range of frames. The frame range is defined as the start frame where the runner appears for the first time and end frame where a runner exists in the camera view.

There are different levels of details to annotate an object. The cheapest method is video-level and frame-level annotations. In frame-level annotation, video frames are manually analyzed and visualized. A range of frames is defined for each runner where they are recognizable by the human eye. A range is defined as the start frame and end frame for each individual runner in a video. *Start frame* is the frame number where the runner is recognizable for the first time in the video, whereas *End frame* is the frame number where a runner exits the camera's field of view. For example, if a runner 'X' is recognizable for the first time in frame number 'n1', and leaves the camera view at frame number 'n2'. Then for runner 'X', we annotate the frame 'n1' as the *Start frame* and 'n2' as the *End frame*. An example is shown in Figure 15,

We evaluated the method on the four sample videos. The result of the experiment is shown in Figure 16. On average it takes around 12 min to annotate a video of mean duration.

There are in total 3,268 videos in the dataset. Then,

Time to annotate all videos = $12 \times 3268 = 39,216$ min.

Analysis. Frame level annotation takes less time to annotate. However, the average time of annotation may vary depending on the number of runners in the video. That is, if there are hundreds of runners running together, then most of the runners would be occluded and therefore, it will take more time to spot and annotate the runner.

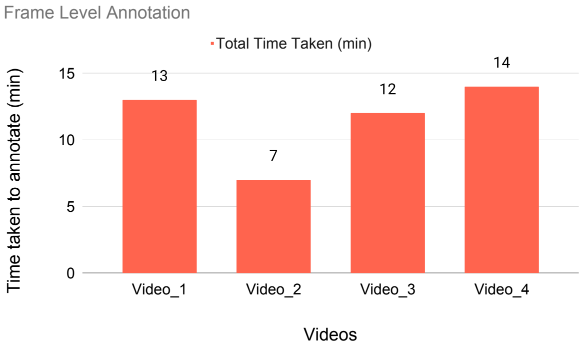


Figure 16: **Results of Frame Level Annotation.** On average, it takes around 12 min to annotate a video of 95 sec duration (mean duration).

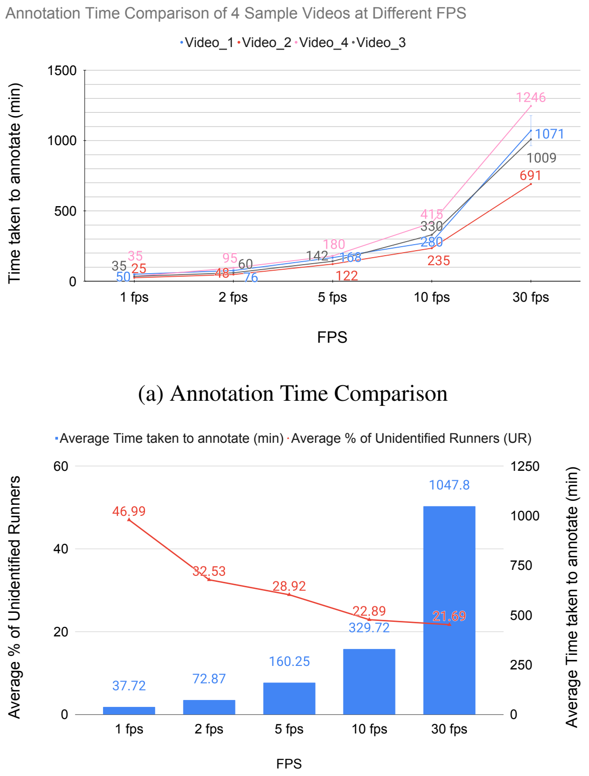
Overall, frame-level annotation is an easy and naive way to annotate which helps to reduce the cost of annotation. However, in this type of annotation, the use cases of the dataset will be limited, as we don't have pixel-level annotation.

5.2. Exp 2: Analysing the effect of FPS on annotation time

The main question behind this experiment is to investigate whether we can obtain the same amount of information from the videos if we extract the frames at some different extraction rates. If we can reduce the frame rate per second, then the total number of frames will be reduced, leading to the reduced time and effort of annotation.

Setup. We evaluate the method on 4 sample videos. The videos are recorded at *30fps*. The experiment is performed at five different *fps* values, $n = \{1, 2, 5, 10, 30\}$, where n = frames per second. In each case, every $(30/n)^{\text{th}}$ frame is extracted. For example, if $n = 2$, then we take every 15th frame and remove the others, or in other words, there are 2 frames in 1 second. We used LabelImg [26], for manually creating the bounding boxes. A runner's bibId is used as the label.

Analysis. Results in Fig17(a) shows that it takes more time to annotate a video if it's recorded at a higher frame rate. Also, from Fig17(b), it is clear that as the fps increases, the number of unidentified number decreases. Also, there is very little change in the slope of the curve after 5fps. Most of the runners' identities visible in the 30fps case are also visible in the case of 5 fps videos. On an average, we are able to identify 91.8% of the identifiable runners in 5fps videos. We conclude that it's better to annotate at a lower fps rate (in this case 5 fps), as it takes a lesser amount of time to annotate, and at the same time derives most of the information from the data.



(a) Annotation Time Comparison

(b) Analysing the average annotation time and percentage of unidentified runners

Figure 17: **Results of the FPS experiment are shown in figure.** (a) Comparison of time taken to annotate four sample videos at different fps rates is shown in the figure. It is clear that annotation time increases with the increase in the number of frames per second. (b) In the figure, as the frame extraction rate increases, more number of runners are identified. But the slope of the curve doesn't change much after 5fps. Also, the annotation time increases with the increase in fps, as the number of frames to be annotated also increases.

How can the efficient bounding-box annotation be done?

Another problem in the annotation is to generate accurate ground truth. The performance of most of the machine learning model highly depends on the accuracy of the ground truth labels. So, it becomes important to generate tight bounding boxes around the runners for the ground truth labels.

There are several ways to do the bounding box annotation. We have analyzed and compared three different approaches namely *object detector, object tracking and box interpolation,* alongside the baseline method. For all the experiments, we use annotation time as a common unit to compare the efficiency of one approach with other approaches.

5.3. Exp 3: Baseline: Manually generating the bounding boxes

We consider the manual annotation performed at 5 fps extraction rate in the FPS experiment, as our main baseline method. Results of the baseline method is shown in Figure 18. On average, it takes around *153 min* to generate bounding boxes in a video of 95 sec duration (mean duration).

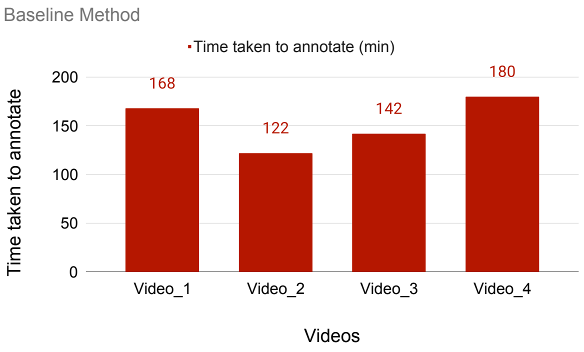
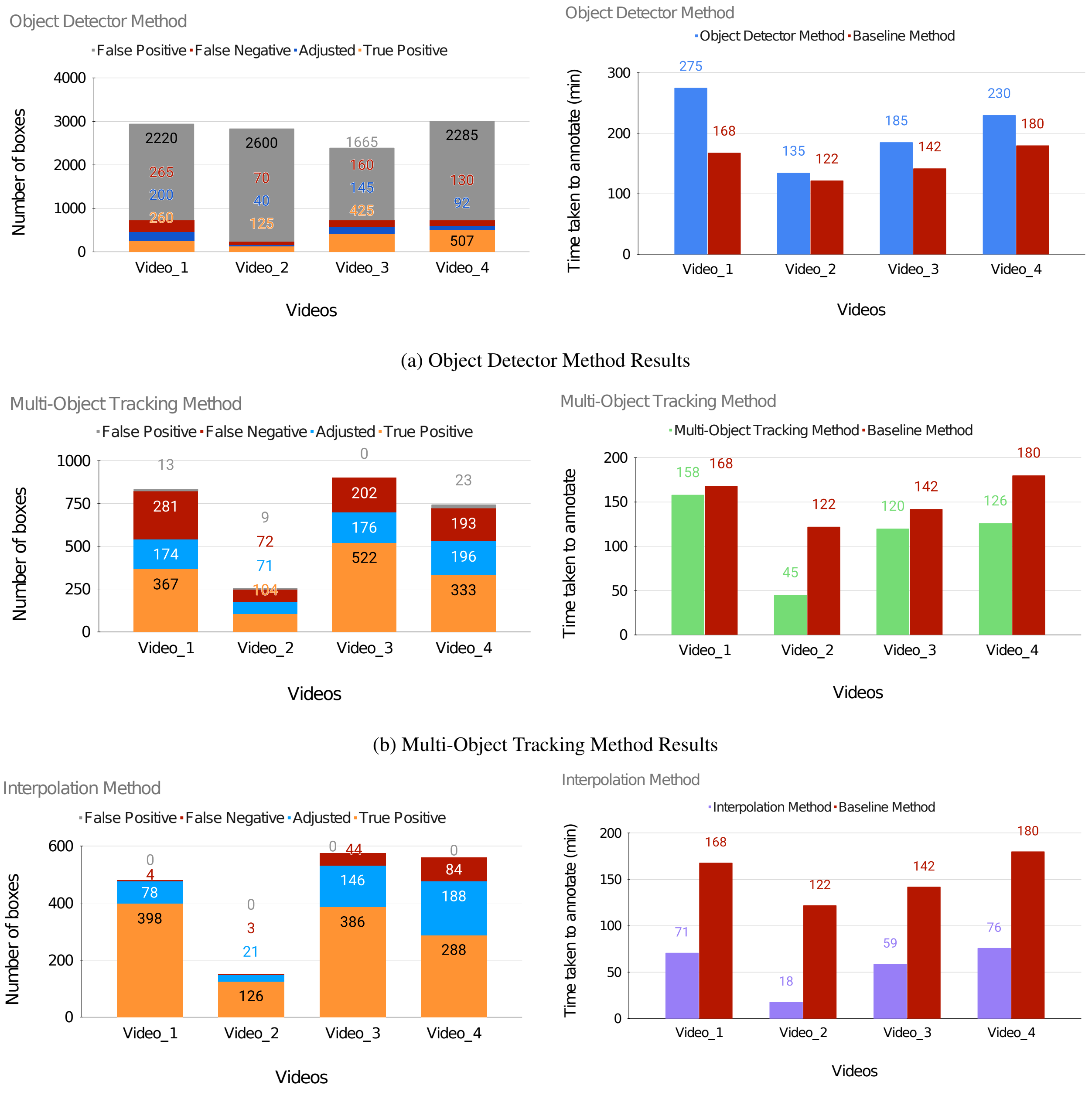


Figure 18: **Results of the baseline method.** Manual bounding box annotation is performed on four sample videos. On average, it takes 153 minutes to annotate a video of mean duration.

5.4. Exp 4: Faster bounding box annotation using Object Detector

Our interest is to validate the hypothesis if the automatic generation of bounding boxes using object detectors could save the time and effort of annotation.

We used the same four sample videos for experimentation as used in the baseline method. The frames are extracted at 5FPS rate. Next, we feed the frame into the



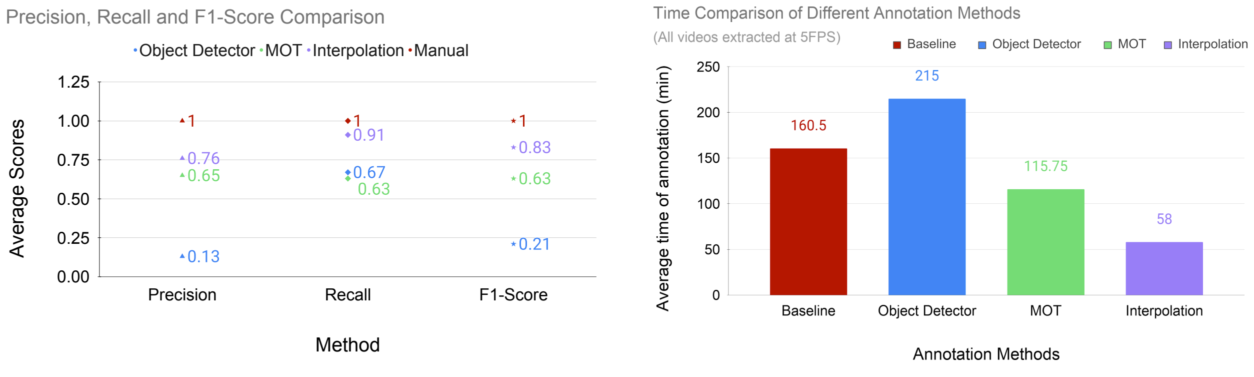
(a) Object Detector Method Results

(b) Multi-Object Tracking Method Results

(c) Interpolation Method Results

Figure 19: **Comparison of results of different methods of bounding box generation.** (a) Time taken in the annotation method using Object Detector is more than the baseline method. This is due to the presence of too many FPs in the form of non-runners, which needs to be removed manually. (b, c) Time taken in annotation using MOT and Interpolation is less than the baseline method. It can be observed that the number of FPs are very less. Also, the number of TPs is high, resulting in a reduced time of annotation.

deep learning based object detector models for the generation of bounding boxes. Any SOTA object detector can be used. We used YoloV3 [21] pre-trained model, trained on the COCO dataset. As our main target is to predict a runner,



(a) Precision, Recall and F1-Score comparison

(b) Annotation time comparison of different methods

Figure 20: **Accuracy and Time Comparison of different annotation methods.** (a) It is clear that the annotation using interpolation has maximum precision and recall values. It is because the number of false positives and false negatives in the interpolation method is less than any other method. (b) The annotation time using the interpolation method is least compared to any other method.

therefore we removed all the classes except the person class. This way we prevent the generation of unwanted bounding boxes.

Manual correction of inferred predictions. The incorrectly predicted boxes are adjusted or removed by a human annotator. The annotator has to go through all the proposed boxes and check if the predictions are correct or need an adjustment. If IoU is greater than 0.8, then it is TP, else it is an FP. All the FPs over the runners are adjusted, whereas the ones including the non-runners are removed manually. A runner's bibId is used as the label and is manually added for each prediction by the annotator. The total time of annotation is calculated using Equation 11.

Analysis. The results of the experiment are shown in Figure 19(a). It can be seen in Fig.19(a), that the number of FPs are too much. The FPs are due to the detection of the non-runners which are present in every frame. These FPs are the unwanted detections which are unavoidable, and it needs to be removed manually from every frame. This results in an increase in the overall annotation time. Hence, the total time in the case of the semi-automatic annotation using an object detector is more than the baseline method. The average precision, recall and F1-score is shown in Table 4. The lower precision value is due to a higher percentage of false positives.

We conclude that using an object detector to automatically generate the bounding boxes saves time in the case when only the object of interest is present in the frame. In the case of our dataset, the method completely fails, due to the presence of too many false positives, as a result, it takes more time than the baseline method.

Precision	**Recall**	**F1-Score**
0.13	0.67	0.21

Table 4: **Average Precision, Recall and F1-Score of Object Detector Method.** The lower precision value is due to a higher percentage of false positives due to the presence of non-runners in most of the frames.

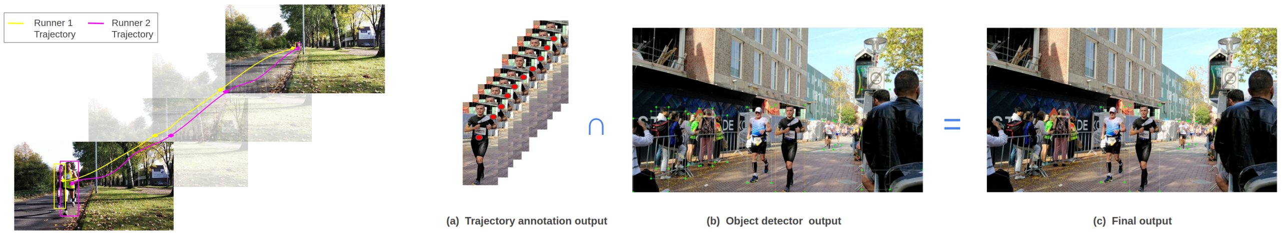
5.5. Exp 5: Faster bounding box annotation using Multi-Object Tracking

A typical annotation pipeline would involve an annotator watching the videos or images while doing annotations. So, our main idea is to verify if we can make use of this watching time of video by efficiently turning it into annotation time. Also, in this way, we can overcome the problems of object detector experiment, by avoiding the unwanted predictions of non-runners.

The idea is based on the paper *Pathtrack* [14], where the authors use path supervision to generate dense box trajectory annotations for Multi-Object Tracking (MOT) datasets.

Setup. For experiments, we use the same set of four sample videos. Firstly, the frame rate of videos is decreased from 30fps to 5fps. Then we use the Pathtrack Tool [14], to annotate these videos.

The tool has an interface to play a video. The user can control the playback speed by speeding up or down, as per the requirement. For annotation of runners, firstly a segment of the video where the target runner is visible is played once, to observe the trajectory of the runner. Then, the user



(a) Trajectory annotation

(b) Linking Trajectory and Box annotations

Figure 21: **The figure shows the annotation procedure of the MOT method.** (a) The trajectory of runners is followed using the mouse cursor. The colored lines show the mouse movement during the path supervision. (b) The intersection of the trajectory annotation and the object detector detection outputs the final annotation. It helps in removing the false positives detected in the object detector.

rewinds the video back to the point where the runner is visible for the first time. The user then changes the playback speed of the video and starts following the runner's trajectory by hovering the mouse cursor near the center of the runner. In the final trajectory, the annotator provides three bounding boxes for the first and last appearance of the runner and one in between the two.

Steps. The annotation using path supervision is performed mainly in two stages:

Trajectory annotation with path supervision. Annotations using path supervision is efficient & intuitive and is obtained by watching each runner independently and tracking it using a mouse cursor. Annotation path of a runner *r* consists of an (x, y)-coordinate point p_r(f) that lies inside it's location boundaries at frame id f.

Bounding box generation. The trajectory annotation will provide the coordinates of a runner in every frame. Now, we need to generate bounding boxes over the runners. We use an automatic way to predict the bounding boxes by using deep learning-based object detector models. We used YoloV3 [21] pre-trained model, trained on the COCO dataset [13], to predict boxes only for *person* class. The object detector will predict the boxes, both over runners and non-runners.

Linking Trajectory and Box annotations. The main task now is to remove the unwanted FPs and generate box over the trajectories of runners. Given the set of path annotations p_r(f) and object detection D, the intersection of the two annotations will remove the FPs. In Equation 13, in a given frame f, if the detection D_i contains at least one point p_r(f), such that point p_r(f) lies inside or over D_i, then the detection survives, else it is eliminated.

If, detection D_i has coordinates x_{min}, y_{min}, and x_{max}, y_{max}, then

$$D_i = \begin{cases} \text{Accepted}, & \text{if } \mathrm{x}_{min} \leqslant \mathrm{x}, \leqslant x_{max} \; \& \\ & \quad \mathrm{y}_{min} \leqslant \mathrm{y} \leqslant \mathrm{y}_{max} \\ \text{Rejected}, & \text{otherwise.} \end{cases} \tag{13}$$

This way we get bounding boxes generated over the runners along their trajectories. Each runner's detection is then labeled manually, by assigning the runners' bibId as the label. Thereafter, the unavoidable false positives are removed and adjusted, whereas FNs are added manually by the annotator. Figure 21 describes the annotation procedure using MOT method.

Workload estimation. The total time of annotation includes mainly 5 time-consuming components: 1) Watching the video every time to check a runners' trajectory once, 2) Following each runners' trajectory using mouse hovering 3) Time to adjust and create the bounding boxes 4) Time to remove the false positives, 5) Time to add the labels for each box prediction.

Analysis. The results of the experiments are shown in Figure 19(b). It is clear from Fig 19(b), the semi-automatic annotation using MOT takes lesser time compared to the baseline annotation. The method generates a quick path trajectory annotation of runners. It generates bounding boxes 1.36x times faster than the baseline method. Also, it reduces the number of false positives mainly due to the avoidance of non-runners detection, see Figure 19(b). There's also an improvement in precision, recall, and F1-score, compared to the object detector experiment, see Table 5.

However, the method fails to generate boxes in case the runner is far away from the camera frame. It is due to the inaccurate prediction of the object detector. Also, it is difficult to track runners in a crowded scene, as the runner's visibility is not constant throughout the video segment because of occlusion. Due to this, the number of false negatives is high resulting in decreased recall value.

Precision	Recall	F1-Score
0.65	0.63	0.63

Table 5: **Average Precision, Recall and F1-Score of Multi-Object Tracking Method.** The recall value is lower because of higher number of FNs. It is difficult to track runners in crowded scenes, hence the number of FNs is high.

5.6. Exp 6: Faster bounding box annotation using Bounding Box Interpolation

In the baseline annotation method, the annotator has to put a bounding box around every runner in every frame. As the position of the runner doesn't vary much in consecutive frames, it could be useful if boxes can be interpolated between frames. Our main hypothesis is that interpolating the boxes between keyframes can save a lot of time, as we can assume that the trajectory of the runner is mostly linear.

The interpolation technique is intuitive, efficient, easy to implement, and produces compelling results. In our approach, we annotate a sparse-set of bounding boxes known as *keyframes*, and linearly interpolate between them. Keyframes are defined as the frames where the bounding boxes if created, would help in filling the boxes in intermediate frames. We used *Darklabel* [20] tool that uses a linear interpolation technique to propagate the boxes for the intermediate frames in between the given set of keyframes.

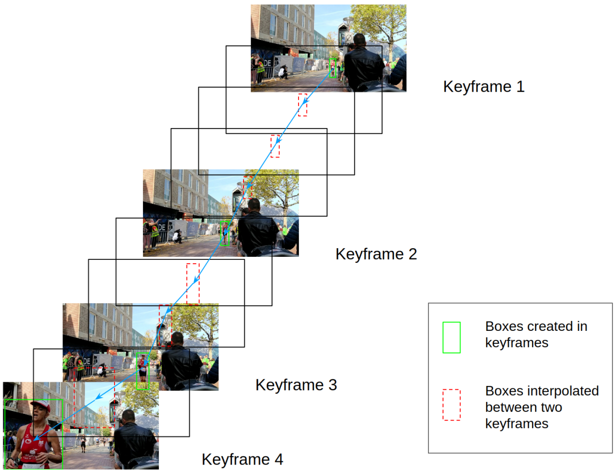

Keyframe 1
Keyframe 2
Keyframe 3
Keyframe 4
Boxes created in keyframes
Boxes interpolated between two keyframes

Figure 22: **The workflow of the interpolation method is shown in Figure.** The bounding boxes are created only in keyframes, and the rest of the boxes are linearly interpolated in the frames in between consecutive keyframes. As can be noticed, the size of the runner increase as it approaches near to the camera, and therefore the keyframe interval is decreased.

Setup. The four sample videos are used for the experiment. Firstly, the frame rate of videos is decreased from 30fps to 5fps. We used Darklabel [20] annotation tool for interpolating the boxes. The annotator selects a runner and finds his bibId. Then a frame is selected where the runner is visible for the first time in the camera. The annotator starts with putting the bounding box around the runner and labels it with it's bibId. The next four frames are skipped and the next box is created in the 5th frame. The steps are repeated until the runner comes closer to the camera. As it is more difficult to adjust a box than to create a new one. So, the keyframe interval is decreased as the runner approaches towards the camera, to avoid as many FPs. It is because the size of the runner increases as it comes closer to the camera, and therefore the boxes can't be linearly interpolated across distant keyframes. The overflow workflow of the method is explained in Figure 22.

Workload. The total time of annotation includes the time to create boxes in keyframes, adjustment of FPs, the addition of FNs, and time to label the detections.

Analysis. The results of the experiment are shown in Fig.19(c). The method clearly outperforms the baseline method by a large margin. The method generates bounding boxes almost 3x times faster than the baseline method. In Fig.19(c), the number of FPs are very low, and hence the precision is very high, as shown in Table 6. The number of FNs is also very less compared to any previous method used, and hence the method has the highest recall value of 91%. The number of adjustments in Video_3 and Video_4 are high due to the the presence of shakiness of the camera. Due to shakiness, the runner location shifts drastically between consecutive frames. This is where linear interpolation incorrectly propagates the box, and therefore adjustment is needed.

Precision	Recall	F1-Score
0.76	0.91	0.83

Table 6: **Average Precision, Recall and F1-Score of Interpolation Method.** The precision and recall values are high because of very less FPs and FNs respectively.

How can Inter-Camera alignment be achieved in minimum time and effort?

In inter-camera/cross-camera alignment, we need to ensure that every runner is assigned the same identity across all his appearances in different camera locations. For example, if a runner has the label 'X' in camera location 1, then it should have the same label in all his/her appearances in the remaining camera locations, an example of which is shown in Figure 24.

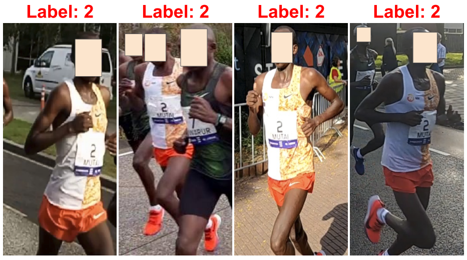


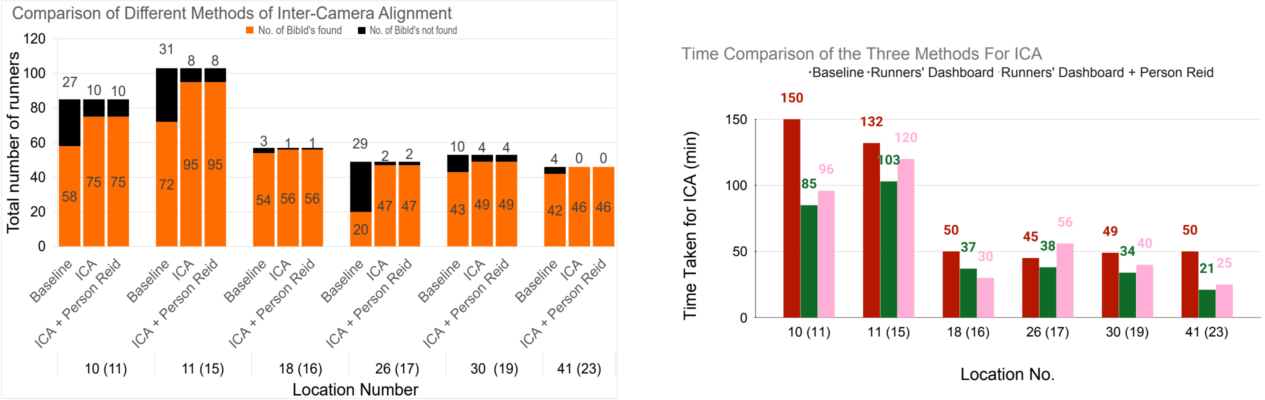
Figure 24: **An example of inter-camera alignment.** All the appearances of the runner in different non-overlapping cameras should have the same label.

Our main objective is to find ways to ensure cross-camera alignment for all the runners, at the cost of minimum time and effort. There are a number of issues that make the ICA problem difficult. Firstly, there are too many camera locations and runners, which makes the problem more laborious. Secondly, due to crowdedness, occlusion, poor resolution, and illumination in videos, it becomes difficult to see the runners' bibId. In that case, we have to find ways, to search for the identity of the runner in a different camera location. It is tiresome to search a runner amongst thousands of other runners in a pool of videos with hours of recording. Therefore, there's a need for some better way to do the inter-camera alignment of the runners.

Setup. In all the experiments for the current problem, we have used the sample dataset already discussed in Section 4.2. As the main objective of the experiment is to align the identities of the runners in all the 6 cameras of the sample dataset, we are only interested in the bibId of the runners and the frame number in which the runner's bibId is clearly visible. In case, there is no bibId attached on the runner's chest or in case it's taking too long or if it is not possible to find the bibId of the runner, then a unique ID is assigned in the pattern 'LiRj', where 'L' stands for location, 'R' stands for the runner, and 'i' and 'j' represents the location and the runner number respectively.

5.7. Exp 7: Baseline

In the baseline experiment, the annotator has to find the frame where the runners' bibId is recognizable. In case of a specific runner, if it is not possible to find the bibId due to discrepancies mentioned in Section 4.2, or if the tag of the runner is missing, then the annotator will try to find the runner in videos from a different camera location. The new camera location can be randomly selected from the locations having location number lower than the current location number, as it is assumed that the runner has not evaded any checkpoint and has passed through all the previous locations before reaching the current location. If the annotator can locate the runner in the videos of randomly selected lo-



(a) Comparing the number of bibId's found and not found using three different methods of ICA

(b) Annotation time comparison of different methods of ICA

Figure 23: **Comparing the results of three methods for the Inter-Camera Alignment(ICA) of runners.** The numbers shown in brackets on the horizontal axis are the scores of each location, indicating the difficulty level of the scenarios present in videos, refer to Table 3 for more details. (a) The number of runners with bibId not recognizable are higher in case of baseline method. Using ICA tool (runners' dashboard), most of the runners' bibId are identified. A few number of unidentified runners using tool reflects the number of those runners that did not finish the race, and hence were unidentifiable using the tool. (b) It is clear in the figure that ICA using runners' dashboard is the quickest of all the methods. Integration of person re-id in runners' dashboard doesn't decrease the time, however, it is easier and is recommended to use in finding unidentified runners.

cation then he will assign the runners' bibId as it's label, and if he fails to find the runner after spending a few minutes then he will assign it a unique Id as mentioned in Section 5.6.

Analysis. The results of the experiments are shown in Figure 23. It can be observed in the Figure that the number of unidentified runners is quite high, if we try to find the runners naively in other video locations. Only 60.71% runners are identified. It took almost 8 hrs to perform the inter-camera alignment in the sample dataset. Also, many runners are still unidentified. Therefore, a better solution is needed for the inter-camera alignment of the runners.

5.8. Exp 8: Creating an Inter-Camera Alignment Tool

To ease the problem of aligning the runners across multiple cameras, we have created a dashboard of runners as mentioned in Section 4.4. Therefore, in this experiment, the task is performed in a similar manner as in the baseline method, except we try to locate an unidentified runner using the ICA tool. To find a runner, we can use a partial name search or bibId search option in the tool. If no part of the name or bibId is visible, then we select a nearby runner whose bibId or name is visible. Firstly, we find the time 't' at which this specific runner reaches the current location. We use this time to find all the runners crossing the current location, in time duration t-δt $\leqslant$ t $\leqslant$ t+δt, where δt = 1 min. This way we reduce the search space, and can quickly identify the runner using images provided in the tool. If the runner's identity is unknown even after spending a few minutes searching it, then assign a unique id to the runner as mentioned in Section 5.6.

Analysis. The results of the experiment are shown in Figure 23. It can be noticed that there's a big drop in the overall annotation time when the runners' dashboard tool is used for the inter-camera alignment of runners. The tool is almost 1.5x faster than the naive baseline method. Also, the number of unidentified runners is also very less in comparison to the baseline method. Total 93.64% runners are identified using the ICA tool. The small number of unidentified runners' can also be explained. The tool comprises the information of runners' who finished the race. It doesn't contain the details of runners that abandoned the race in the middle. That's why it is not possible to find the identity of such runners using the dashboard tool.

Overall, the tool is user friendly and quick in locating unidentified runners. But, it is a bit tricky to use the tool, and hence needs some time to get the expertise. Also, a manual search operation is performed every time to find the runner. So, we will try to find some semi-automatic way to reduce the number of manual search operations.

5.9. Exp 9: Intelligent use of noise for Inter-Camera Alignment

To reduce the effort of manually searching the runners in the dashboard tool, we try to automatize the process. To do this, we intelligently use the noisy results of the person re-identification system. Person re-identification is a well-known research problem in computer vision, applications of which ranges from tracking persons appearing across multiple non-overlapping cameras. The main goal of a person re-id is to retrieve all appearances of a person from a large gallery of images captured from cameras with non-overlapping views. There is a lot of work done in the past on supervised and unsupervised person re-identification. Though supervised methods can attain good performance, they need large-scale labeled datasets.

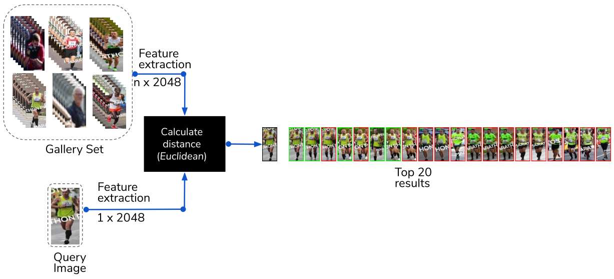
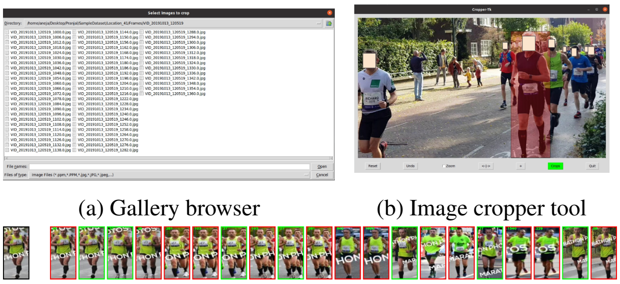


Figure 25: **Workflow diagram of the person re-id model.** Firstly, the feature vector of the gallery images and the probe image is calculated. Euclidean distance is calculated between the gallery images and probe image feature vectors and the 20 closest images are displayed as the final output.



(a) Gallery browser

(b) Image cropper tool

(c) Visualization of top 20 results

Figure 26: **An outlook of the person re-identification tool integrated with runners' dashboard.** (a) The tool offers the option to quickly browse images from the gallery set. (b) The target runner can be cropped from the selected image. After cropping, the person re-id system will be activated for the cropped person's image. (c) The top-20 matches closest to the probe images are displayed as the final output.

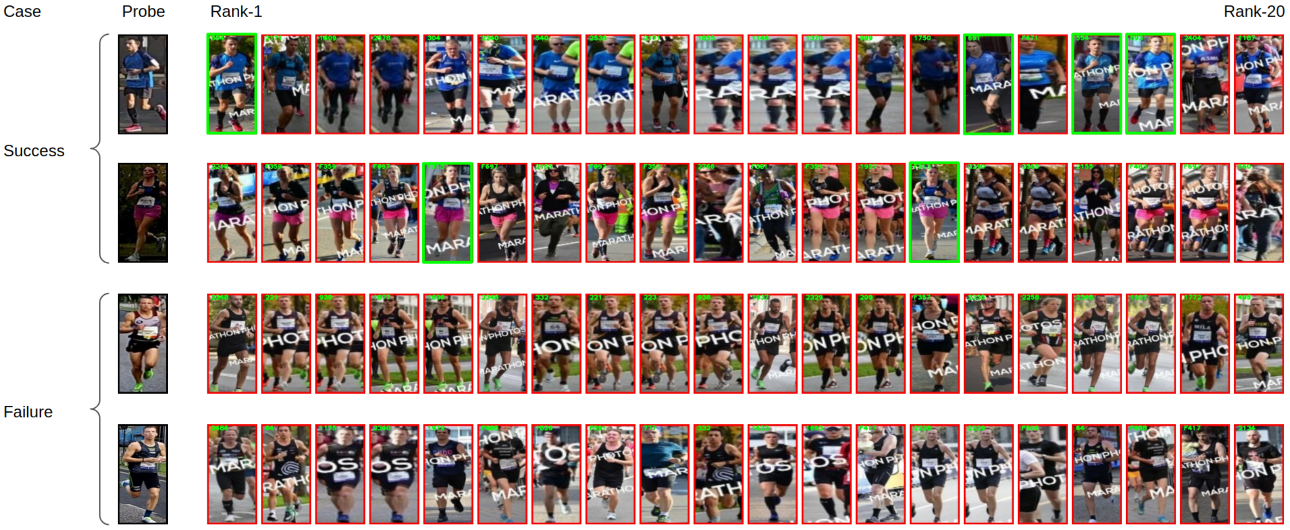


Figure 27: **Results of the inter-camera alignment experiment using person re-id model.** The upper two rows correspond to the cases where the tool successfully finds at least 1 image of the target runner. The lower two rows correspond to cases where the tool fails to find any image correct image of the target runner in the top-20 results. In a successful case, the matching images of the runner have different labels. It is because the target runner may have appeared in the background of an image of some different runner, and the identity of the other runner is displayed in the result.

Setup. In our experiment, we use an off-the-shelf person re-id [30], to identify the runner among the gallery of images of all the participant runners. In the person re-id task, we used the dataset of runners, scraped from the official website, as mentioned in Section 4.3. The dataset consists of images with a watermark text in every image. The text isn't removed as it was not affecting the re-id model results. As, we used an object detector [21], to crop runners from the images, the dataset also contains random images of non-runners appearing in the background. Hence, the dataset is noisy and contains some garbage images as well. Our main interest is to see if we can use the noisy results of the person re-id model to find the unidentified runner. We used the pre-trained model, trained on market1501 [29]. In [30], the dataset is divided into a gallery set and a query set. In our experiment, the gallery set consists of at-most 20 images per runner, and minimum 2 images per runner, whereas the query set consists of images, used as probe images to find all the appearances of the runner in the gallery set.

A person re-id tool is developed and integrated into the runners' dashboard. An outlook of the tool is shown in Figure 26. The tool helps in quickly browsing the images from the gallery. It also allows the user to crop the person of interest from the image. The tool takes the probe image of the runner as input and process out the top 20 images closest to the probe image. Firstly, the feature vector of the probe image and all the images in the gallery set is created. Then, the euclidean distance between the feature vector of the probe image with the feature vector of all the gallery images is calculated. The top-20 images having the minimum distance with the probe image are selected as output. In the end, top-20 results are displayed as shown in the workflow diagram in Figure 25.

In the experiment, the same steps are followed as in the previous Experiment 5.8, except that for any unidentified runner whose bibId or name is not visible, instead of finding the runner using the dashboard, we first find the runner using the person re-id tool integrated into runners' dashboard. The runners' dashboard is only used when the runner's name or bibId is visible, or the person re-id tool fails to find the runner.

Analysis. The results are mentioned in the Figure 23. The method takes very little time in comparison to the baseline method. However, it is observed that the integration of person re-id is taking more time than solely using the runners' dashboard tool. The reason is that the person re-id takes around 10-15 seconds per query image, to process the results. As the deep learning models' performances are still not saturated, they tend to give false results, as can be seen in Figure 27. Due to which, we had to perform multiple runs in the person re-id to reach the correct result, and in case re-id fails, we had to use the runners' dashboard at the end to find the identity of the runner. As a result, the overall time of annotation is increased. However, it is more flexible and easy to identify runners using person re-id, so we propose to use this method for the inter-camera alignment task.

6. Conclusion

In this work, we proposed the heuristics to annotate a large-scale in-the-wild video dataset of marathon runners. We discussed the problems that arise in the annotation of the marathon dataset covering real-world scenarios. We demonstrated how to reduce the overall cost of annotation by reducing the frame extraction rate. Additionally, we investigated different ways to generate efficient tight bounding boxes. Our study shows that using box interpolation is the most effective way of generating bounding boxes in such datasets. We also proposed a novel method of aligning the runners in the cross-camera setting of multiple non-overlapping cameras. We introduced an inter-camera alignment tool integrated with state-of-the-art deep learning person re-identification method, to help in quickly and efficiently aligning unidentified runners across multiple disjoint cameras.

Even though our methods significantly reduce the human effort and total cost of annotation, more research into cross-camera alignment can incentivize the annotation time and accuracy. In this paper, we used a noisy dataset for the person re-identification task that contained images of non-runners. Due to this, the person re-id system is not very accurate. The dataset can be more refined and augmented to improve the accuracy. Also, for bounding box regression, more alternatives can be explored in the future. One alternative could be the combination of extreme clicking [7], and box interpolation, to further reduce the annotation time. In future, we will look forward to investigative these incentives.

References

[1] General data protection regulation, 2018.

[2] Bishwo Adhikari, Jukka Peltomaki, Jussi Puura, and Heikki Huttunen. Faster bounding box annotation for object detection in indoor scenes. pages 1–6, 11 2018.

[3] Cognilytica. Data preparation time. `https://www.cognilytica.com/2019/04/19/infographic-data-prep-and-labeling/`, 2020.

[4] TU Delft. Computer vision lab, delft university of technology, 2020.

[5] Marathon Eindhoven. Marathon eindhoven 2019. `https://www.marathoneindhoven.nl/`, 2019. Official Website.

[6] Mengran Gou, Srikrishna Karanam, Wenqian Liu, Octavia Camps, and Richard J. Radke. Dukemtmc4reid: A large-scale multi-camera person re-identification dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[7] Michael Gygli and Vittorio Ferrari. Efficient object annotation via speaking and pointing, 2019.

[8] Patrick M. Hartigan. Two sample ks test. `https://sparky.rice.edu//astr360/kstest.pdf`.

[9] Srikrishna Karanam, Mengran Gou, Ziyan Wu, Angels Rates-Borras, Octavia Camps, and Richard J. Radke. A systematic evaluation and benchmark for person re-identification: Features, metrics, and datasets, 2016.

[10] Wei Li and Xiaogang Wang. Locally aligned feature transforms across views. In *CVPR*, 2013.

[11] Wei Li, Rui Zhao, and Xiaogang Wang. Human reidentification with transferred metric learning. In *ACCV*, 2012.

[12] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *CVPR*, 2014.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[14] Santiago Manen, Michael Gygli, Dengxin Dai, and Luc Van Gool. Pathtrack: Fast trajectory annotation with path supervision, 2017.

[15] MYLAPS. Mylaps official website. `https://www.mylaps.com/`, 2020.

[16] Yeshwanth Napolean, Priadi Teguh Wibowo, and Jan van Gemert. Running event visualization using videos from multiple cameras, 2019.

[17] Sangmin Oh, Anthony Hoogs, A. G. Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Lee, Saurajit Mukherjee, J. Aggarwal, Hyungtae Lee, Larry Davis, Eran Swears, Xioyang Wang, Qiang Ji, Kishore Reddy, Mubarak Shah, Carl Vondrick, Hamed Pirsiavash, Deva Ramanan, Jenny Yuen, and Mita Desai. A large-scale benchmark dataset for event recognition in surveillance video. pages 3153–3160, 08 2011.

[18] Dim Papadopoulos, Jasper Uijlings, Frank Keller, and Vittorio Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. 02 2016.

[19] Phil Harvey. Exiftool. `https://exiftool.org/`, year = 2016.

[20] Dark Programmer. Darklabel-video/image labeling and annotation tool. `https://darkpgmr.tistory.com/16`, 2020.

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

[22] Leonard Richardson. Beautiful soup documentation. *April*, 2007.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[24] Selenium. Selenium webdriver. `https://selenium.dev/documentation/en/`, lastchecked = 10.11.2019.

[25] H. Su, Jia Deng, and Li Fei-Fei. Crowdsourcing annotations for visual object detection. In *HCOMP@AAAI*, 2012.

[26] Tzutalin. Labelimg. `https://github.com/tzutalin/labelImg`, 2015. Git code.

[27] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101, 01 2012.

[28] Jenny Yuen, Bryan Russell, Ce Liu, and Antonio Torralba. Labelme video: Building a video database with human annotations. pages 1451–1458, 09 2009.

[29] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. pages 1116–1124, 12 2015.

[30] Kaiyang Zhou and Tao Xiang. Torchreid: A library for deep learning person re-identification in pytorch. *arXiv preprint arXiv:1910.10093*, 2019.

A. Kolmogorov-Smirnov (KS) Test

For larger sample sizes, the approximate critical value D_α is given by the equation,

$$D_\alpha = c(\alpha)\sqrt{\frac{n_1 + n_2}{n_1 n_2}} \tag{14}$$

where, n_1 and n_2 are the sample sizes of the two distributions and α and $c(\alpha)$ are the coefficients given by table mentioned [8].

Original dataset location scores from the table 8 = $S_{originalDataset}$ = {8, 10, 11, 12, 13, 14, 15, 15, 15, 16, 16, 16, 16, 16, 16, 17, 17, 17, 17, 18, 18, 18, 18, 18, 19, 19, 20, 20, 20, 21, 22, 22, 23, 23, 24}

Sample size of $S_{originalDataset}$ = 35

As we want to select 6 scores for the sample dataset,

Sample size of $S_{sampledDataset}$ = 6,

$c(\alpha)$ = 1.63, α = 0.01

$D_\alpha = 1.63\sqrt{\frac{35+6}{35\times 6}} = 0.7202.$

The KS test gives statistics and p_{value} as output.

Null hypothesis is that the two distributions are different.

$$\text{if,} \begin{cases} \text{statistics} < D_\alpha & \\ \text{and p-value} > \alpha, & \textit{null hypothesis rejected} \\ & \\ \text{else,} & \textit{null hypothesis accepted.} \end{cases} \tag{15}$$

After thousands of iterations of random sampling the score values, we selected 6 best scores that have the distribution similar to the original dataset score distribution.

Sampled dataset location scores = $S_{sampled_{D}ataset}$ = {11, 15, 16, 17, 19, 23}

B. Webscraping

For scraping the data, python libraries namely Selenium web-driver [24] and Beautiful-soup [22] are used. The data is available on the website of Eindhoven Marathon [5]. The request was first intercepted using the selenium web-driver and the data available in JSON format on the website was retrieved corresponding to all the page requests containing the data. Only data corresponding to the required fields is read and later saved into a *xls* file.

Data Type	**Field name**
Full marathon data	id, eventId, raceId, bib, bibForUrl, category, rank, genderRank, categoryRank, gunTime, chipTime primaryDisplayTime, speedInKmh, name, countryCode, activityType, gender, city, cumulativeTime_5k, name_5k, cumulativeTime_10k, name_10k, cumulativeTime_15k, name_15k, cumulativeTime_20k, name_20k, cumulativeTime_25k, name_25k, cumulativeTime_half, name_half, cumulativeTime_30k, name_30k, cumulativeTime_35k, name_35k, cumulativeTime_40k, name_40k, cumulativeTime_finish, name_finish, gunTimeInSec, chipTimeInSec, customValues, displayDistance, qualified
Half marathon data	id, eventId, raceId, bib, bibForUrl, category, rank, genderRank, categoryRank, gunTime, chipTime primaryDisplayTime, speedInKmh, name, countryCode, activityType, gender, city, cumulativeTime_5k, name_5k, cumulativeTime_10k, name_10k, cumulativeTime_15k, cumulativeTime_finish, name_finish, gunTimeInSec, chipTimeInSec, customValues, displayDistance, qualified

Table 7: **Field names corresponding to which the data of full-marathon and half-marathon is scraped from the official website.**

Location	Occlusion	Lighting	Recording Angle	Resolution	#Crowded Videos	Score
5	1	3	2	1	1	8
3	1	3	2	3	1	10
10	3	1	1	4	2	11
2	1	2	4	4	1	12
4	1	4	4	3	1	13
9	2	3	3	4	2	14
25	2	4	2	5	2	15
11	2	3	5	2	3	15
35	3	3	5	2	2	15
8	2	3	5	4	2	16
28	5	3	3	2	3	16
29	3	4	3	3	3	16
33	4	4	2	2	4	16
22	3	2	5	4	2	16
30	3	3	5	3	2	16
37	4	4	4	2	3	17
40	4	4	4	2	3	17
26	3	4	4	3	3	17
14	2	4	5	4	2	17
12	3	5	3	4	3	18
13	3	4	4	3	4	18
31	4	4	4	3	3	18
14	3	5	4	4	2	18
17	3	4	5	4	2	18
20	4	4	3	4	4	19
18	4	3	5	4	3	19
15	4	4	5	3	4	20
38	4	4	5	4	3	20
39	4	4	5	3	4	20
36	4	4	5	4	4	21
34	4	4	5	5	4	22
42	4	5	5	5	3	22
27	5	5	5	4	4	23
41	4	5	5	5	4	23
24	5	5	5	4	5	24

Table 8: **Score distribution of recording locations in original dataset.** Lower score means bad recording scenarios and higher score means good recording scenarios.

2 Introduction

The popularity of Marathon events has increased in recent times, due to the health awareness amongst the individuals. Videos of the marathon runners could also be used in monitoring the performance of the athletes. A lot of videos of marathon events are available online in the raw format but cannot be used to train the deep learning models. Therefore, we collected a large-scale in-the-wild video dataset of marathon runners, that can be used for research such as the physiological estimation of the marathon runners, health monitoring of athletes, runner re-identification, etc. But annotating such a large-scale in-the-wild dataset is a non-trivial task, due to the presence of variability such as illumination, crowdedness, occlusion, resolution, etc. Therefore, we explore ways to efficiently annotate a large-scale dataset of marathon runners with minimum effort and cost of annotation.

2.1. Research Questions

Main Research Question

How can we efficiently annotate a large-scale video dataset?

Three sub-research questions are derived from the main question:

1. How can we reduce the overall cost of annotation in terms of time as well as budget?
2. How can we do efficient bounding-box annotation?
3. How can we achieve Cross-Camera alignment of runners in minimum time and effort?

2.2. Supplementary Report Outline

In chapter 3, we have covered the pre-requisite knowledge that is required for the understanding of the thesis. We will discuss about the data annotation, its different types, and the basics of the person re-identification method. Chapter 4 details the results of the experiments in the tabular form.

3

Pre-requisite Knowledge

In this section, some background information will be discussed related to the research topic. We will discuss the basics of annotation and it's different forms, about the cross camera alignment, person re-identification technique, etc.

3.1. What is Data Annotation?

The task of labeling and preparing the data in such a way that it can be used to train different machine learning algorithms to memorize and learn patterns for predictions is called Data Annotation. The data can be in any format such as images, video, audio, or text. Data annotation is a vital part of data pre-processing in supervised machine learning. A machine learning algorithm after training sufficiently on annotated data can recognize the same patterns when provided with unseen and unannotated data.

Nowadays, AI is everywhere and has penetrated through every section of human life. Today AI stories are all the rage, the data narrative is not entailed. But the real story is not about the AI revolution but it's more about the quality of data that is fed into the algorithms that actually makes AI powerful.

According to some sources, 80% of the AI project development time is spent on data preparation [3]. Data annotation becomes important as even a small error could prove disastrous in machine learning. To have better accuracy and performance of a certain algorithm, good quality annotated data is necessary.

3.2. Different types of data annotation

There are several ways to annotate data for computer vision research projects. Images can simply be bucketed into classes, tight bounding boxes can be drawn around the object of interest, or every individual pixel can be labeled in an image.

Type of data annotation depends on many different use cases, primarily of which falls into the following categories [11]:

1. Image classification / tagging
2. Speech and text tagging and labeling speech and text
3. 3D Point Cloud tagging
4. Sentiment Analysis

There are various types of annotation which vary as per the requirement of the project. Some of the traditional data annotation tasks are:

3.2.1. Image and Video Annotation

One of the most important tasks in computer vision is image annotation. Image annotation involves the addition of the metadata to the image so that machines can use and learn from it. The metadata refers to the labels and tags, which could be a bounding box around a certain part of an image or could be a segmentation mask where every pixel of an image is annotated.

Video annotation is similar to the image annotation where the annotation is done precisely frame-by-frame, where each frame is an image. Image and video annotation is used by various computer vision algorithms for various tasks varying from simple classification to object detection, object tracking, autonomous driving, etc.

Following are the most common type of image annotation tasks:

Image Classification

In this, the entire image is associated with a tag or a label. For example, images of different fruits would be given a different label (an image of an orange would be tagged as 'orange'). On feeding this type of annotated data, the classification algorithms learn to categorize the image accurately based on the type of fruit.

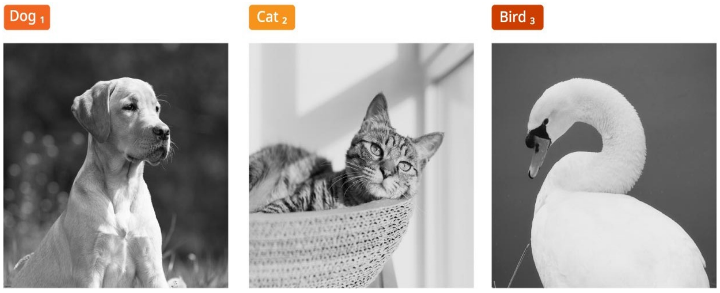
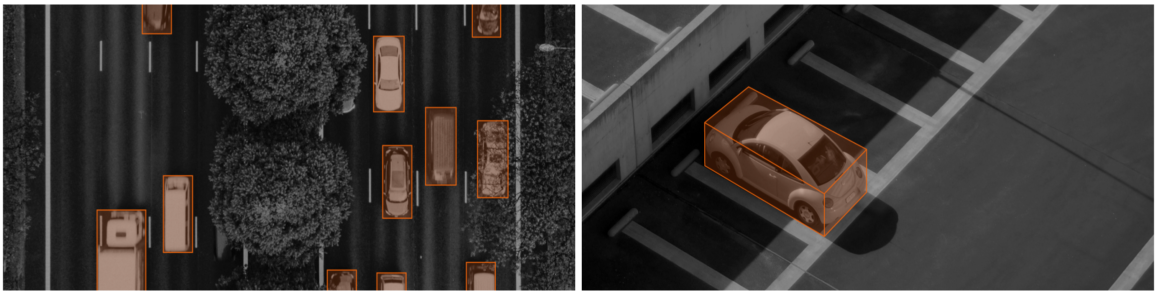


Figure 3.1: An example of image classification annotation. [1]

Bounding box annotation

In this type of annotation, a box is drawn around the object of interest. In the case of 2D tasks, a rectangular box is drawn, whereas, for 3D tasks, a cubical box is drawn around the target object. A label is then assigned to the object inside the box.



(a) 2D bounding box annotation

(b) 3D bounding box annotation

Figure 3.2: Bounding box annotation [1] (a) Rectangular boxes are placed around the object of interest. (b) Cubical boxes are placed around the object of interest.

Semantic Segmentation

It involves dividing an image into different segments and each segment is then labeled pixel-wise. Each labeled pixel in the image belongs to a specific class.

It is mainly used for perception model training. A good use case of Semantic segmentation involves autonomous driving, where various objects in the traffic images are labeled pixel-wise, thereby helping the computer vision to make accurate predictions.

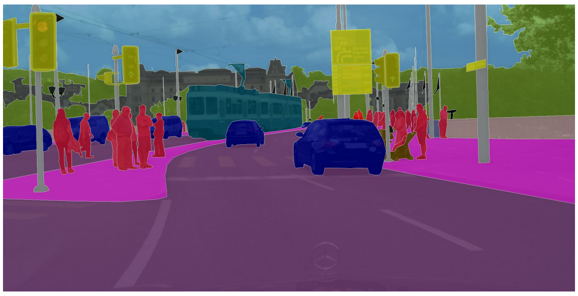


Figure 3.3: An example of Semantic Segmentation annotation [1]. Different objects in the image are labeled pixel-wise.

Polygons

Sometimes the objects to be annotated could be of irregular shapes and therefore it's difficult to annotate the object using rectangular bounding boxes. Therefore, we annotate the object using polygons which allows the annotators to annotate the exact shape of the object. A label is then assigned to the object inside the box.

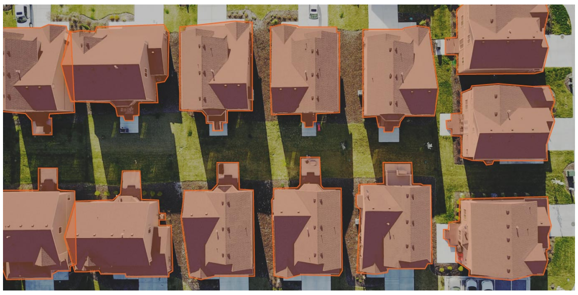
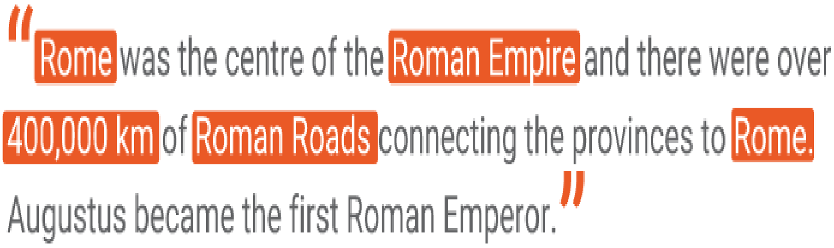


Figure 3.4: An example of Polygon annotation [1]. It is similar to 2-D bounding box annotation, where instead of rectangle we use polygons.

3.2.2. Semantic Annotation

The task of annotating a text for different concepts such as objects, people, organization, topic, or product is called Semantic Annotation. Semantic annotation is used for training chatbots and for improving search relevance.



"Rome was the centre of the Roman Empire and there were over 400,000 km of Roman Roads connecting the provinces to Rome. Augustus became the first Roman Emperor."

Figure 3.5: An example of Semantic annotation. Sentences are split to identify different concepts such as events, place, people, things, etc. [10]

3.2.3. 3D point cloud annotation

3D point clouds are the set of points that define the structure of an object. These point clouds are generated by LiDAR sensors

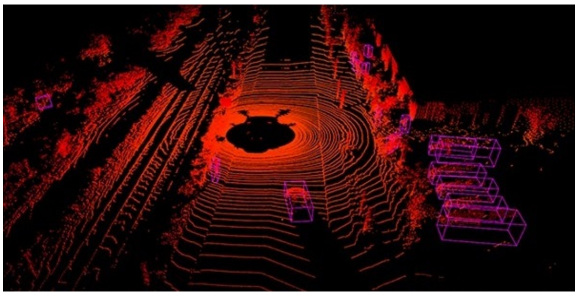
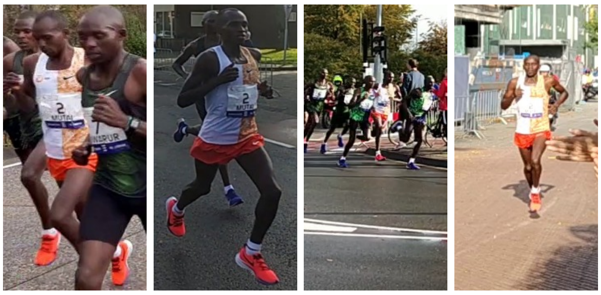


Figure 3.6: Using point clouds to detect objects with 3D boxes [2]. 3D point clouds are the set of points generated by sensors like LiDAR that define the 3D structure of an object.

The 3D point cloud is the set of points generated by sensors like LiDAR that define the 3D structure of an object in space. We are compatible with both 3D annotation techniques and point cloud handling for processing quality sensory training data.

3.3. Person Re-identification



(a) Occlusion (b) Poor Lighting (c) Viewpoint (d) Poor Resolution

Figure 3.7: Person Re-ID is a problem of identifying an individual in different cameras. Major problems in person re-id are 1) Occlusion 2) Light intensity Variation 3) Camera Viewpoint 4) Poor Resolution

Person re-identification is defined as the problem of identifying a person of interest captured in different locations and times over several camera views amongst a large number of other candidates. Due to the increasing number of surveillance cameras everywhere from streets to university campuses and public parks, person re-id has become imperative in current times pertaining to the urgent requirement of public safety.

Though the research in this area is going on for several years, still there are many problems that need to be solved. Major problems arise due to similar clothes of different individuals, poor camera resolution, object occlusion, varying poses, varying illumination, and different camera viewpoints.

3.3.1. Basic Design Flow of a Simple Person - Reid System

A simple person re-id system design is shown in the figure 3.8. It consists of five main steps [12]:

1. *Raw Data Collection [12]*: The first major step is to collect a raw video dataset from cameras at different locations. This data comprises of videos of different scenarios as mentioned earlier.
2. *Generating bounding boxes [12]*: The next step is to extract bounding boxes containing the images of persons in the collected data. The images can be cropped either manually or automati-

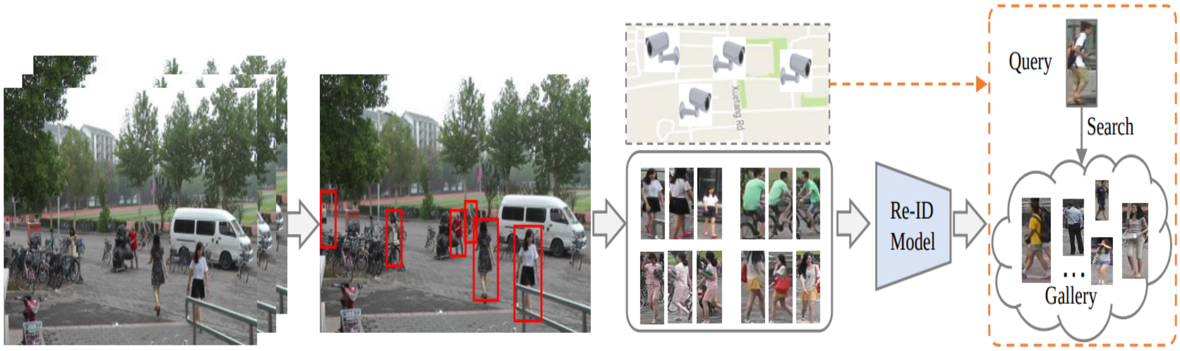


Figure 3.8: Five major steps of person re-id model. 1) Raw Data collection 2) Generating bounding boxes 3) Training data annotation 4) Model training 4) Querying person of interest [12]

cally using some object detector or object tracking algorithms.

3. *Training data annotation [12]*: It is one of the most difficult but crucial steps in a person-reid system. Annotating each and every individual identity with labels and also aligning each individual across different cameras at a different location. The cross camera alignment becomes difficult due to cross camera-variations.

4. *Model training [12]*: After the annotation of the data, the next crucial step is to train the re-id model. Several re-id models have been studied and developed in the past to handle different challenges.

5. *Querying person of interest [12]*: This step falls into the category of the testing phase. The query image or the test image of the person to be searched is matched across all the images in the gallery set by extracting the feature representations using the re-id model learned in the training phase. The query to gallery similarity is calculated and a sorted ranking list is generated at the end.

3.4. Some famous Person-Reid datasets

Several datasets have been introduced over the years for person re-id tasks. Some of the most famous are VIPeR[5], CUHK01[8], CUHK02[7], CUHK03[9], Market1501[13], DukeMTMC4ReID[4] and Airport[6]. More information about these datasets is mentioned in the table 3.1.

Dataset	# people	# BBox	# cam	label
VIPeR[5]	632	1,264	2	hand
CUHK01[8]	971	3,884	2	hand
CUHK02[7]	1,816	7,264	10	hand
CUHK03[9]	1,360	13,164	10	DPM/hand
Market1501[13]	1,501	32,217	6	DPM
DukeMTMC4ReID[4]	1,852	46,261	8	Doppia
Airport[6]	9,651	39,902	6	ACF

Table 3.1: Statistics of some famous person re-id benchmark datasets

4 Experimental Results

In this section, we document the tabular results of the experiments performed for validating the following research questions:

4.1. How can we reduce the overall cost of annotation?

4.1.1. Exp: Frame Per Second (FPS) Analysis

Videos	Video_1			Video_2			Video_3			Video_4		
FPS	#IR	#UR	Annotation Time	#IR	#UR	Annotation Time	#IR	#UR	Annotation Time	#IR	#UR	Annotation Time
1 fps	22	12	50	5	3	29.32	3	15	35	14	9	36.54
2 fps	25	9	76	5	3	56.3	5	13	60	21	2	99.18
5 fps	26	8	168	5	3	143.09	5	13	142	23	0	187.91
10 fps	30	4	280	5	3	275.62	6	12	330	23	0	433.24
30 fps	31	3	1071	5	3	810.43	6	12	1009	23	0	1300.77

Table 4.1: **Results of the FPS analysis experiment. IR** denotes the 'Identified Runners' and **UR** denotes the 'Unidentified Runners'. It can be observed from the table that the annotation time increases as the fps value increases. Also, more number of runner are identifiable as the fps value increases. It is because with more number of frames the probability of a runner occuring in at least 1 frame increases.

4.2. How can we do efficient bounding-box annotation?

4.2.1. Exp: Faster bounding box annotation using Object Detector

Videos	# FPs (Non - runners)	# FNs	#Boxes removed	# Boxes adjusted (FPs)	# TPs	Total annotation time (min)
Video_1	2220	265	150	200	260	275
Video_2	2600	70	45	40	125	158.33
Video_3	1665	160	15	145	425	185
Video_4	2285	130	0	92	507	240.11

Table 4.2: **Results of the Object Detector experiment for bounding box generation. FP, FN, TN** denotes false positives, false negatives and true positives respectively. It can observed that the number of false positives is very high, due to the detection of non-runners in most of the frames. Therefore, it takes a lot of time to remove these FPs, causing increase in total time of annotation.

4.2.2. Exp: Faster bounding box annotation using Multi-Object Tracking

Videos	#Runners tracked	#Detections in Object Detector	#FPs (Non-runners)	#FNs	#Boxes adjusted (FPs)	#TPs	Total annotation time (min)
Video_1	28	554	13	281	174	367	158
Video_2	8	184	9	72	71	104	45
Video_3	19	698	0	202	176	522	120
Video_4	23	552	23	193	196	333	126

Table 4.3: **Results of the Multi-Object Tracking experiment for bounding box generation. FP, FN, TN** denotes false positives, false negatives and true positives respectively. It can observed that the number of FPs is very less in this case in comparison to the object detector experiment. Due to which the total annotation time is also less.

4.2.3. Exp: Faster bounding box annotation using Interpolation

Videos	#FPs (Non - runners)	#FNs	#Boxes adjusted (FPs)	#TPs	Total annotation time (min)
Video_1	0	78	249	4	71
Video_2	0	21	88	3	18
Video_3	0	146	198	44	76
Video_4	0	188	253	84	59

Table 4.4: **Results of the Interpolation experiment for bounding box generation. FP, FN, TN** denotes false positives, false negatives and true positives respectively. It can be observed that the total number of FPs is very less due to zero detections for non-runners. Also, the number of FNs are very less in this case in comparison to other methods. As a result the total annotation time is also less.

4.3. How can we achieve Cross-Camera alignment in minimum time and effort?

4.3.1. Exp: Inter-Camera Alignment

Method		Baseline			ICA			ICA + Person Re-Id		
Location	Score	#IR	#UR	Annotation Time	#IR	#UR	Annotation Time	#IR	#UR	Annotation Time
10	11	58	27	150	75	10	85	75	10	96
11	15	72	31	132	95	8	103	95	8	120
18	16	54	3	50	56	1	37	56	1	30
26	17	20	29	45	47	2	38	47	2	56
30	19	43	10	49	49	4	34	49	4	40
41	23	42	4	50	46	0	21	46	0	25
Total		289	104	476	368	25	318	368	25	367

Table 4.5: **Results of the inter-camera alignment (ICA) experiments. IR** denotes the identified runners and **UR** denotes the unidentified runner. Scores represents the difficulty level of scenarios in the particular camera location. It can be observed that the annotation time is minimum using the ICA tool. Integration of person re-id system helps in automatizing the annotation process, however it takes more time than solely using the ICA tool.

5

Conclusion

In this thesis work, we discussed the collection of a novel large-scale in-the-wild video dataset of marathon runners. We discussed about the major real-world scenarios covered in the dataset. We also described the problems that occurs in the annotation of such large-scale dataset. We proposed the heuristics to annotate such datasets efficiently in terms of time as well as budget. Our proposed techniques helps in reducing the overall cost of annotation by a factor of 16x compared to the naive way of baseline method. We investigated several ways of efficiently generating bounding box annotation of marathon runners. Our results show that using box interpolation is the most effective way of bounding box regression. We also introduced a novel inter-camera alignment tool for the alignment of the runners across multiple cameras with non-overlapping views, in minimum time and effort. Our method reduces the time for the inter-camera alignment by a factor of 1.5x compared to the baseline method, along with efficiently aligning 93.64% of the total number of runners in the video.

Bibliography

[1] Limarc Ambalina. Common image annotation services, 2019. URL https://hackernoon.com/what-is-image-annotation-an-intro-to-5-image-annotation-services-yt6n3xfj.

[2] Cogito. 3d point cloud labeling services for lidars, 2020. URL https://www.cogitotech.com/lidar-3d-point-cloud-annotation/.

[3] Cognilytica. Data preparation time, 2020. URL https://www.cognilytica.com/2019/04/19/infographic-data-prep-and-labeling/.

[4] Mengran Gou, Srikrishna Karanam, Wenqian Liu, Octavia Camps, and Richard J. Radke. Dukemtmc4reid: A large-scale multi-camera person re-identification dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[5] Douglas Gray and Hai Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, pages 262–275, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-88682-2.

[6] Srikrishna Karanam, Mengran Gou, Ziyan Wu, Angels Rates-Borras, Octavia Camps, and Richard J. Radke. A systematic evaluation and benchmark for person re-identification: Features, metrics, and datasets, 2016.

[7] Wei Li and Xiaogang Wang. Locally aligned feature transforms across views. In *CVPR*, 2013.

[8] Wei Li, Rui Zhao, and Xiaogang Wang. Human reidentification with transferred metric learning. In *ACCV*, 2012.

[9] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *CVPR*, 2014.

[10] OnToText. How does semantic annotation work? https://www.ontotext.com/knowledgehub/fundamentals/semantic-annotation/, 2020.

[11] Ron Schmelzer. Different annotation use cases, 2020. URL https://estidevelopers.com/2020/02/07/the-human-powered-companies-that-make-ai-work/.

[12] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven C. H. Hoi. Deep learning for person re-identification: A survey and outlook. *arXiv preprint arXiv:2001.04193*, 2020.

[13] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. pages 1116–1124, 12 2015. doi: 10.1109/ICCV.2015.133.