A Machine Learning Approach to Improve Automated Kinematics Tracking of Non-Human Primates

Sarah Pratt

Advisors: David Borton and James Tompkin
Reader: Thomas Serre

A thesis submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Applied Mathematics- Computer Science

Brown University
Spring 2018

Abstract

The goal of this research project is to track the positions of hind-limb joints of a non-human primate from 2D video of the animal. One of the current methods for converting the video recordings into coordinates of the animal's position is SIMI Reality Motion Systems, a commercial computer program. While this software has the ability to localize and track the movement of the animal automatically, the results of the automatic tracking are inaccurate. Achieving accurate annotations of the limb positions of the animal through SIMI requires a user to mark each joint of the animal by hand which is inefficient. This paper aims to create a pipeline which allows for accurate unsupervised tracking. We achieve this by utilizing videos previously hand-annotated with SIMI to train a neural network to localize the position of the joints on the limb of the primate. We then use the relative location and velocity of the joints in adjacent frames to track the position of the primate over time.

Contents

1 Introduction

1.1 Motivation

Making accurate predictions about the intended movement of the body given signals from the brain allows for advances in the medical field. Accurate kinematics allows for a more complete understanding of the bio-mechanics of movement, allowing for improved research in this field. Additionally, kinematics can be used to track a patient's gait. This could improve the identification of irregular gaits and make the diagnosis of injury more accurate. This knowledge could also improve quality of life for patients who have lost limbs or are partially paralyzed. If their intended motion could be deciphered from their neural signals, then a prosthetic could be designed to respond to the needs of its user. The ability to predict motion from neural signals requires a thorough understanding of the relationship between neural output (intended motion) and actual limb movements. Grasping this relationship necessitates large amounts of neural and kinematic data. Currently, the collection of kinematic data is slow and creates a bottleneck in the ability to correlate neural signals and motion.

1.2 Data Acquisition

In order to analyze the accuracy of the methods for localizing and tracking joints throughout the recordings, the results must to be measured against ground truth annotations. This paper uses six recordings of a primate walking on a treadmill.

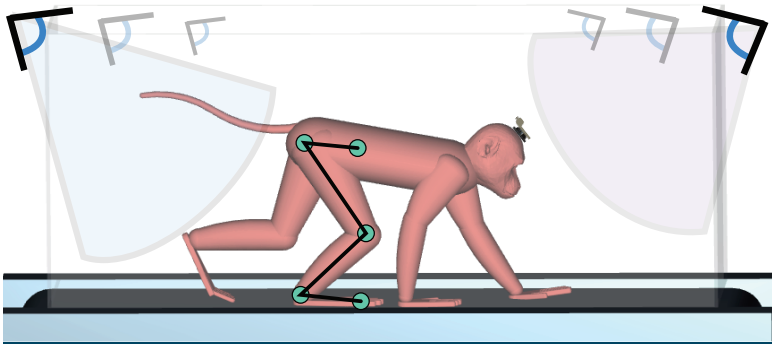

Figure 1.1 [1]

Each of the six recordings is taken from a different angle, with three cameras on each side of the animal. Two of the cameras face the side of the animal, two face the front of the animal, and two face the rear. This setup is illustrated in Figure 1.2.

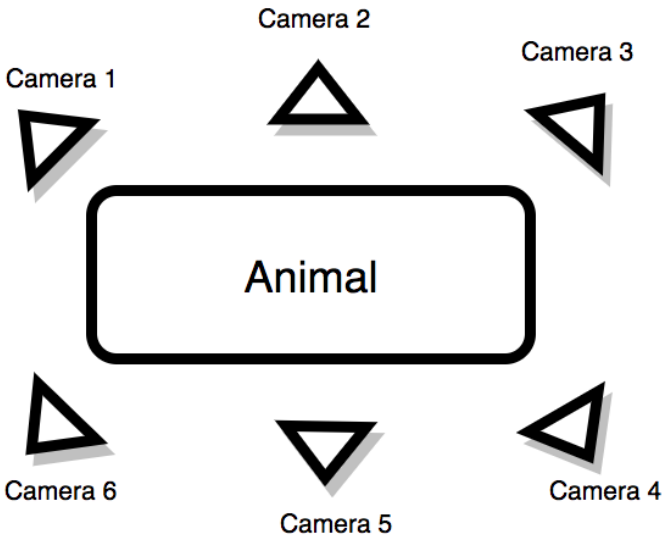

Figure 1.2

Each recording is approximately 10,000 frames of video and is captured at 100 frames per second. This paper describes a method to track the location of the joints on the hind-limb of the animal, which have been marked with yellow ultra-violet paint (as in Figure 1.3).

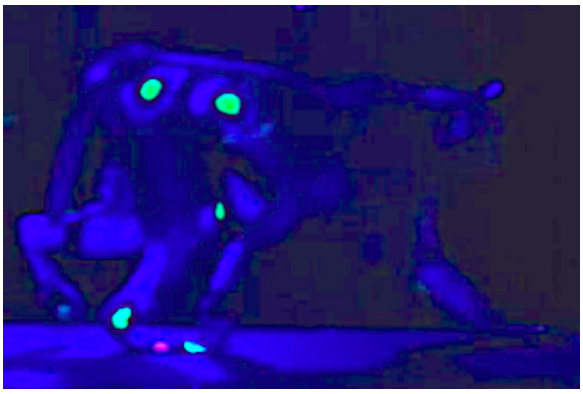

Figure 1.3

For each of the six camera angles, the centers of the joint markers have been hand annotated. Using these hand annotations, bounding boxes are created around each joint. These bounding boxes act as the true location for analyzing the accuracy of joint localization. For analyzing the effectiveness of tracking and localization, the true joint locations were separated from the video data. This paper attempts to recreate these annotations using machine learning and computer vision methods on the unannotated recordings.

1.3 Current Methods

The current method for collecting the kinematic data from the video recordings of the primate is SIMI Reality Motion Systems. This software allows for the joints of the primate to be tracked in two different ways. The first is semi-automatic tracking. This setting allows the user to select the initial location of the markers they wish to track. SIMI then finds the center of the markers by taking the average location of the pixels it identifies as part of the marker. SIMI categorizes pixels as part of the marker using color and light thresholds which can

be adjusted by the user. This is effective while supervised by the user, but very time intensive. Each frame of video takes a minimum of 10 second to correctly annotate.

SIMI also has an option for fully automatic tracking. This is very fast, but does not produce accurate results. SIMI automatic tracking depends on the color and shape of the markers remaining consistent. However, the color values of the painted markers vary as the animal moves. Additionally, the algorithm that SIMI uses to identify markers without user supervision relies heavily on the markers appearing as circles in each frame. This is possible to achieve only with round markers that remain directly in line with the plane of the camera, or with spherical markers that appear round from multiple different angles. Because the primate is able to move freely on the treadmill and change the angle between itself and the plane of the camera, the markers on the body of the animal often do not appear circular. Spherical markers are impractical for marking the location of the animals joints, as the animals are prone to remove anything that adheres to their skin.

Similarly, the unpredictable movements of the animal cause inaccurate annotations when using SIMI automatic tracking. Frequently, the joints of the animal will be occluded because of the angle of the camera or by a forelimb of the animal. When the joint reappears, it is often not detected or it is relabeled as a new joint rather than a continuation of the same joint. This paper explores automated methods to localize and track the hind-limb joints of the animal that are robust to varying shapes and pixel values of the markers, including during or throughout momentary occlusions.

1.4 Workflow

In order to accurately extract the hind-limb positions of the primate from the 2D video, the recordings were processed with multiple algorithms. Each stage of the pipeline, visualized in Figure 1.4, produced outputs that were then utilized in the following process.

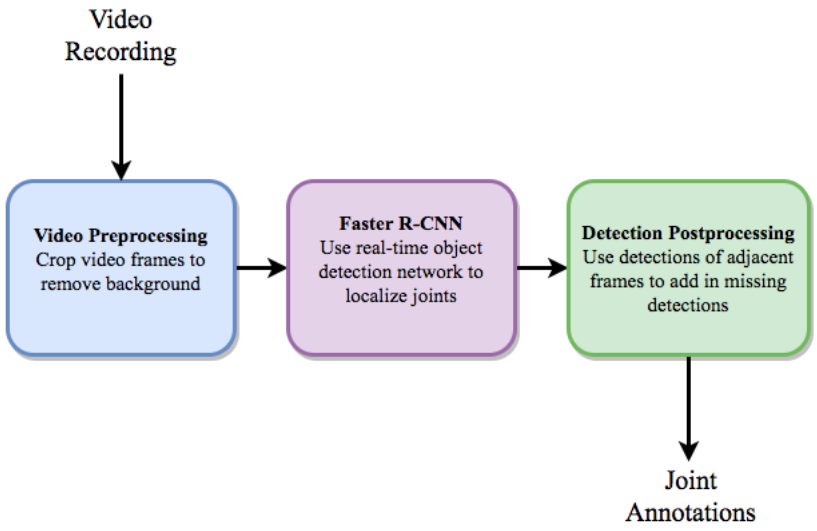

Figure 1.4

The first stage is to preprocess the recordings to remove the portion of the frame which do not contain the animal. The second step trains a convolutional neural network on these cropped images to create initial predictions for the locations of the joints on a test recording. Finally, the detection post-processing uses prior knowledge about the location of the joints on the animal and the movement of the animal in order to remove

incorrect detections and insert detections for joints that have not been captured by the neural network. The follow sections describe the detailed methods and evaluation of each of these stages.

2 Video Preprocessing

2.1 Motivation

In this section we explore methods to preprocess the recordings to remove areas of the frames which do not contain the primate. Later in this paper, we will explore the usage of a Region-Based Convolutional Neural Network as a tool to detect markers that vary in color and shape. This is a powerful tool which categorizes and localizes joints of the animal. Removing large regions of background from the initial image allows the algorithm to more quickly and accurately identify the region of the video frames which contain the joints of the animal. Additionally, it normalizes the scale of the hind-limbs across the frames, allowing the size of the joint to act as a potential feature which the Neural Net can use to identify the markers.

2.2 Methods

The goal of the preprocessing of the recordings is to crop each frame to contain only the areas which are likely to contain the animal. A large part of each frame contains the background of the room. Removing this part of the frame has several advantages. Firstly, it makes joint localization more efficient. Narrowing the region in which to search for the joints decreases the time and memory required during the joint localization step of this project. It also makes the joint localization more accurate. The distinguishing feature between the markers on the joint of the animal versus the rest of the frame is the color and light values of these pixel. Removing the area of the frame which captures the rest of the room and the lab equipment surrounding the primate removes other lights and markers which look similar to the markers on the animal. The video frames are cropped using two steps. The first is Background Subtraction, which separates the primate from its surroundings. The second is the Mean Shift Algorithm which rapidly localizes the center of mass of the animal.

Background Subtraction

The primary distinction between the foreground objects and the stationary background is the variance in the pixel values between frames. The pixels of the objects in the foreground change frequently as the object moves. The values of the pixels of the background of the image stay relatively constant. Relying solely on the difference in color value of pixels between frames can make incorrect predictions. For example, the shadow of a person or an animal may be cast on the background of the image, but causes the color value of these pixels to change quickly. OpenCV (Open Source Computer Vision Library) [2] uses a method to combat these potential mislabeling by modelling each pixel as multiple Gaussian distributions. The weights of these distributions adjust based on the frequency that the pixel changes value as well as the time since it last changed value, giving a more accurate labeling of the background [3].

Mean Shift

The mean shift algorithm provides an efficient way to locate the center of mass of the foreground once these pixels are separated from the background pixels. Mean shift begins with an initial region of interest and then finds the center of mass of all the selected pixels within that region. The process is then repeated, however the new region of interest is shifted so that the new center is now the center of mass of the pixels of interest in the previous step. This is repeated until the center of mass of the selected pixels move by an amount that

is less than a specified threshold, or until the algorithm is repeated a specified number of times. This allows the center of mass of the primate to be rapidly calculated as it changes location during the video [4].

2.3 Results

An image containing the animal with the background cropped is extracted for each frame of the recording. These cropped images acted as the training and testing independent variable data for the following section. A sample of the extraction of the image of the animal is illustrated in Figure 2.1.

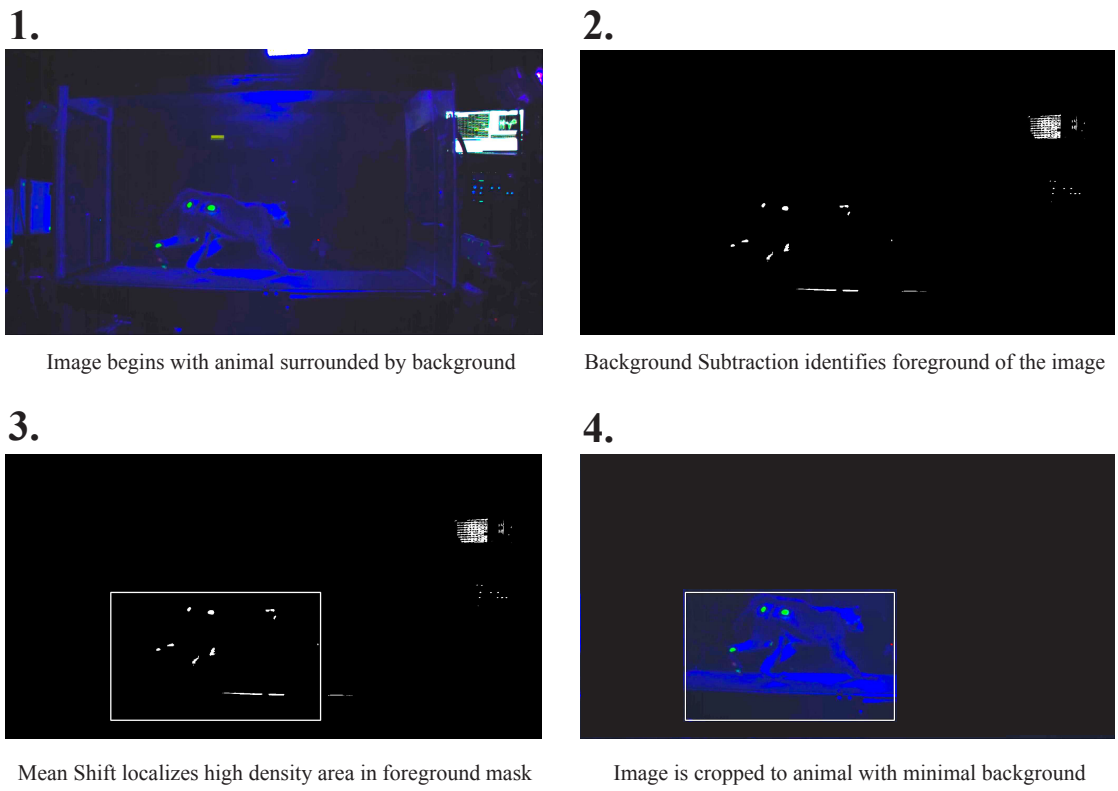

Figure 2.1

For the video recording used in the following two sections, this was able to crop the image to 15% of its original size. Additionally, 96.2% of the joints of the animal remained within these crops. When crops of the same size are taken randomly in the image, only 30% of the joints are captured. Therefore Background Subtraction combined with the Mean Shift Algorithm provides an effective method for removing the background of the image without cropping the hind-limbs of the animal.

3 Localization of Hind-Limb Joints

3.1 Motivation

The reason that the current method for tracking the joints of the animal is not effective is the automated tracking is not accurate for markers that change shape or color. Because the markers must be painted onto the primate, the eccentricity of the circular joint markings varies with the angle of the animal relative to the camera. In order to accurately capture these non-uniform markers, it is important that the tracking software

functions despite adjusting position of the primate. In order to ensure that the new method for joint localization is able to maintain a high precision even as the markers change shape, the localizer was trained on example images of the joints, rather than programmed to search for a specific shape and color.

3.2 Background

There are several methods which allow localization of joints even as vary in shape and color. This section explores two of these methods.

Convolutional Neural Network and Sliding Window Algorithm

This method of object localization can be described in two parts. The first is the Convolutional Neural Network (CNN). This network takes in an image and is able to classify it as either a joint or background. It works by convolving an image with a number of filters. This creates a matrix of the dot products which is given a score of either background or joint. The error of these predictions are measured using a loss function. The weights used to make the classifications are then updated to minimize the loss on a training set. That is, a CNN is given access to a set of images that contain the object or do not contain the object. It makes a prediction and then adjusts the values it used to make that prediction based on whether the guess was accurate.

The second part of this method is the Sliding Window Algorithm. Given a CNN which can categorize an image as a joint marker, the Sliding Window Algorithm is able to detect where the markers are located in the image. This algorithm works by running a binary categorization algorithm on many different crops (or 'windows') of the image. The algorithm then returns the coordinates of the regions that were categorized as a marker with the highest probability. This algorithms is very computationally expensive. Each area of the image must be categorized just to identify a small number of regions that contain the joints of the animal. Additionally, this method gives an estimation of the position, rather than an accurate position. Because this project requires a high degree of accuracy and needs to run quickly, the Sliding Window Algorithm is too inefficient for this project.

Faster R-CNN

Faster Region-based Convolutional Neural Net (Faster R-CNN) [5, 6] is a real time object detection network. This has some similarities to the the sliding window algorithm, but is more complex and more efficient. Rather than categorizing objects as joints or not joints, it first finds regions which are likely to contain joints of the animal and then searches those regions at a finer grain. This can be summarized in four steps.

1. Propose regions of different size and shape throughout the image.
2. Use a CNN in order to classify these regions as likely or unlikely to contain an object. Discard the regions which are unlikely to contain an object.
3. Use the remaining regions to refine bounding boxes around joints. These boxes should be a tight bound around the joint, containing the entire joint but minimal background.
4. Use a CNN to classify each one of these bounding boxes as a joint or not a joint. Output the bounding boxes which are classified as a joint with a high probability. The threshold for this probability is discussed later in this section.

This is an effective tool for identifying the joint markers of the animal because it classifies the markers with high accuracy, but also employs a Region Proposal Network in order to quickly detect the markers within

the frame. The Region Proposal Network (RPN) is described by the first two steps of the above algorithm and allows for image localization without the computational expense of the sliding window. Rather than examining many small windows in order to determine the location of the markers, the RPN identifies regions of the frames that are likely to contain the hind-limb of the animal. The algorithm then searches these regions with a much smaller window in order to determine the actual location of the joints.

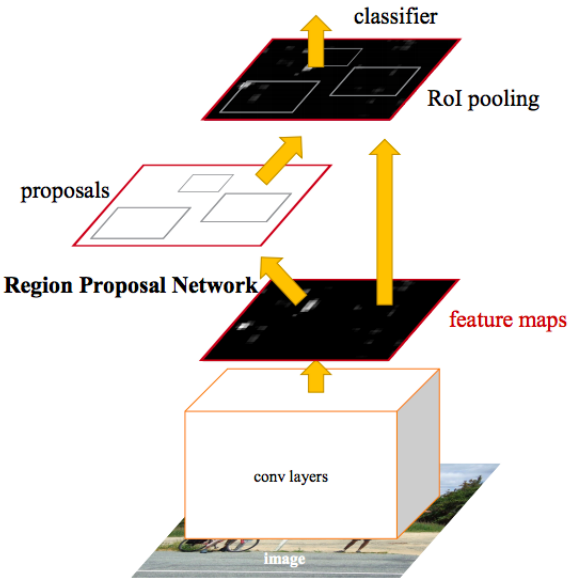

Figure 3.1 - Reproduced from [5]

The network was trained and tested on six video recordings of the animal using VGG-16 as the underlying model [7]. Each recording contained 10,000 frames. The recordings were each taken at a different angle, but recorded at the same time, and therefore captured the same motion of the animal. The parameters for the network were adjusted using six-fold cross-validation. That is, one of the six recordings was held out as testing data while the remaining five of the recording were used to train the model. This was done with each of the six recordings to prevent overfitting to just one angle.

3.3 Tuning Parameters

Faster R-CNN can be made more accurate by tuning a number of parameters that the algorithm uses to determine the most likely locations of the markers.

Classification Probability

One such parameter was the probability threshold for the classification of a joint versus the background of the image. Each region of the image was given a probability of containing a joint of the animal or not containing the joint of the animal. If a proposed region did not reach the given probability threshold of joint classification it was disregarded. However, this value did not have a large affect on the average precision of the detections.

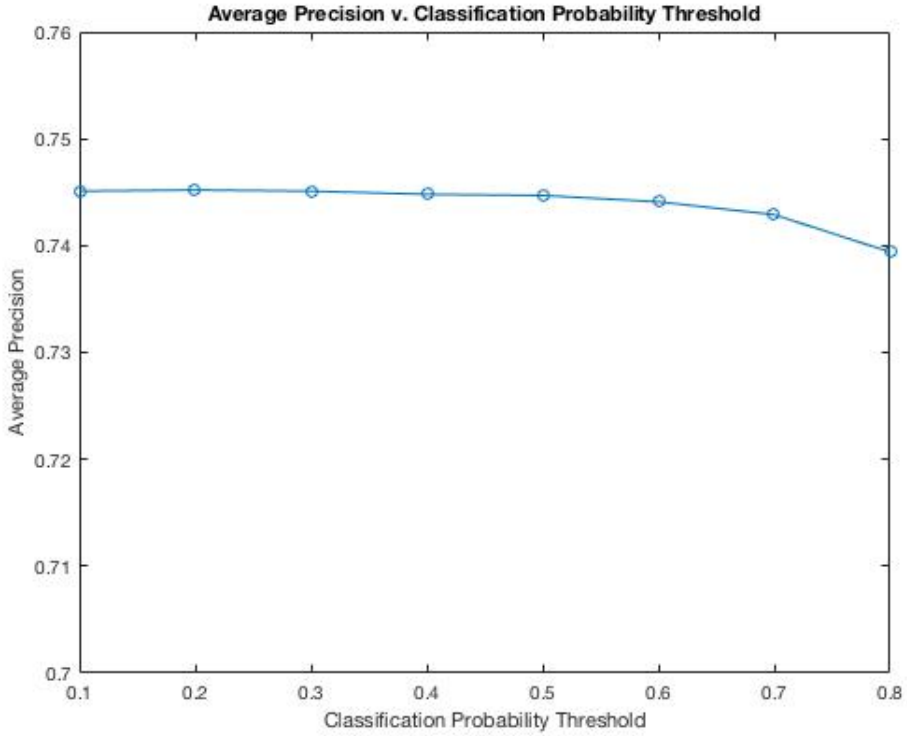

Figure 3.2

A probability threshold of 0.1 had a very similar Average Precision as a threshold of 0.8. This demonstrates that if a region did contain a joint, it was classified correctly with a very high probability and if it did not contain a joint it was not likely to have more than a 0.1 classification probability as a joint. This parameter did not have a significant impact on the outcome of the algorithm.

Non-Maximum Suppression

Non-Maximum Suppression (NMS) was a parameter whose value had a large effect on the accuracy of the detections made by Faster R-CNN. NMS reduces redundancy of detections by removing overlapping detections. The value for NMS is set as a maximum threshold for shared area of two detections. If the regions or detections overlap by this threshold, or by a value greater than this threshold, then only the most likely detections will be kept and the rest will not be considered. The most likely detection is determined by the classification of R-CNN. NMS was applied twice during Faster R-CNN, once after the Region Proposal Network to remove duplicate regions and once after the bounding predictions were created.

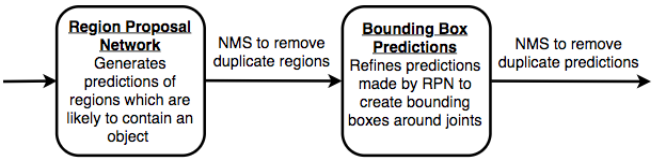

Figure 3.3

Each one of these NMS steps had a different threshold for what was considered a duplicate region and what was considered a unique prediction, and could be tuned separately.

3.4 Results

Ground Truth

The joints in each frame of the training and testing recording had been previously tracked by a human annotator. These annotations had to be adjusted, as they did not represent a perfect ground truth for R-CNN. The hand-annotations were imperfect as a ground truth because they contained some frames of video which were unannotated. They also contained some joints which were annotated, but not visible as the position of occluded joints were estimated using linear interpolation. This was corrected by using the initial hand-tracked locations as a guideline for possible locations of joints. These locations were used to take an initial crop which contained the joint if it was visible at that location. This crop was then masked using a green color mask and the bounding box was created using the minimum enclosing circle around the result of this mask. If the minimum enclosing circle did not exist, or was too small to be a joint, this detection was rejected. Additionally, all frames that had three or fewer joints annotation were considered incomplete and were not used in training or testing.

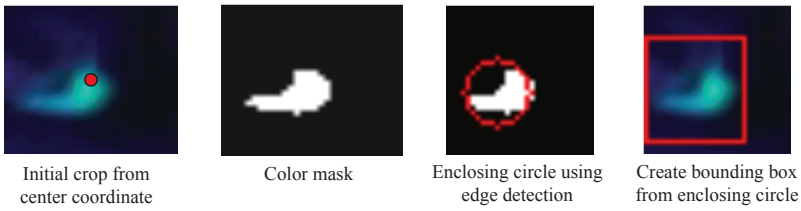

Figure 3.4

Accuracy Metrics

Detections from the R-CNN were considered accurate if their intersection over union (IoU) with the processed ground truth bounding box reached a predetermined threshold. Intersection over union measures the size of the overlap between the two detections while taking into account the size of the region that was guessed. It is calculated by finding the area of the overlap between the ground truth and then dividing it by the total area of the ground truth bounding box and the predicted bounding box. For the purposes of this paper, an IoU value of at least 0.4 was considered to be 'close' to the correct location and a value of 0.65 or higher was considered correct. Figure 3.5 illustrates the threshold for a 'close' detection and an correct detection, where ground truth is in red and the predicted boxed are in green.

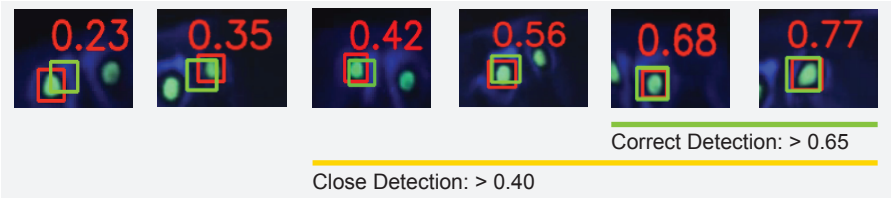

Figure 3.5

While tuning parameters, the aim was to optimize Average Precision of the predictions. Average Precision incorporates both recall and precision. Recall is defined as the number of correct predictions divided by the total number of ground truth detections. Precision is defined as the number of correct predictions divided by the total number of predictions. Average Precision is given as follows, where f is the frame number, R_f is the recall at frame f and P_f is precision at frame f [8]:

$$\text{Average Precision} = \sum_{f=0}^{\text{Total Frames}} (R_f - R_{f-1}) P_f$$

This was used as the accuracy metric because both precision and recall are important for useful results, as good results should contain a high number of accurate detections and a low number of false positives. Optimizing Average Precision allowed for concurrent optimization of both precision and recall.

Average Precision

Figure 3.6 represents Average Precision for varied Non-Maximum Suppression thresholds. The left plot represents a constant NMS threshold of 0.5 on the bounding box refining layer, while the NMS threshold on the Region Proposal Network is varied. The right plot is a constant value of 0.5 for the NMS value on the Region Proposal Network while the NMS value for the bounding box predictions is varied. Both plots are six-fold cross-validated using the six different recordings. That is, the Average Precision was calculated using each of the six recording as the held-out test recording, while Faster R-CNN was trained on the remaining five recordings. These results were then averaged together to find the Average Precision of each configuration of NMS values.

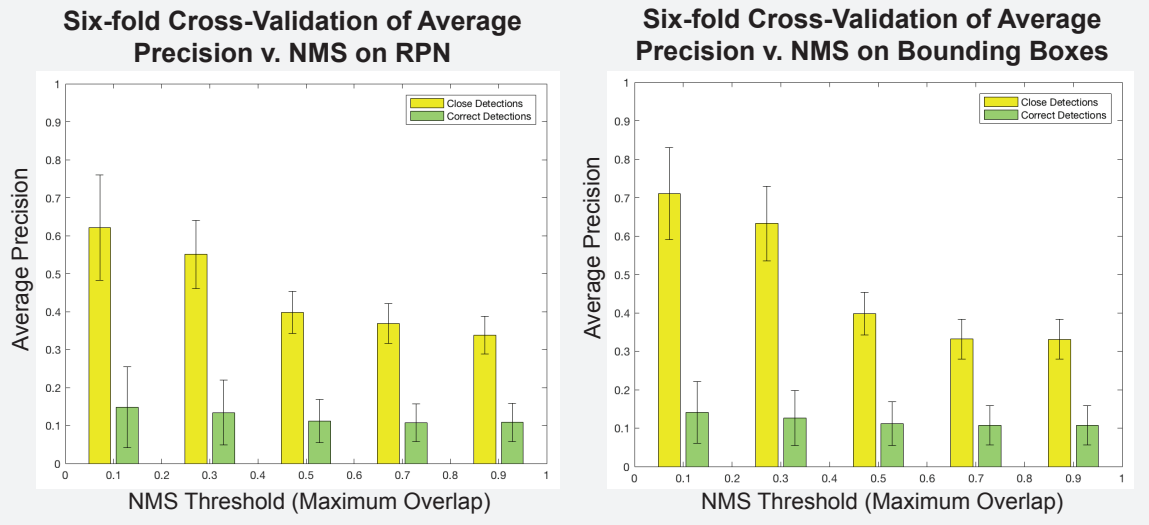

Figure 3.6

For both NMS on the Region Proposal Network and on the bounding box predictions, a lower threshold yields a better result when the other value is held constant at 0.5. For the remainder of this paper, the detections used are those generated when the NMS on the RPN is 0.1 and the NMS value on the bounding box prediction is set at 0.5, as this was the best result found, with an Average Precision over 0.7.

4 Detection Post-Processing

4.1 Motivation

Prior knowledge about the video recording can be used to improve the accuracy of the detection. Faster R-CNN can use the location and images of joints in the training videos in order to make predictions when presented with novel videos. Each frame is analyzed independently without information from the previous or subsequent frames. Accuracy can be improved by utilizing several pieces of information that are not used in the previous section.

Firstly, the positions of the joints follow the structure of the skeleton of the animal. This allows the removal of inaccurate joint detections which are not possible given this constraint. Secondly, the detections

can be made more accurate using information about the approximate speed at which the joint is moving. The video is captured at 100 frames per second, so this provides a tight constraint when considering adjacent frames. Detections from previous and subsequent frames can then be used to improve accuracy.

4.2 Background

Removing Duplicate Detections

During the localization stage of the algorithm, one joint might be detected more than once. Non-Maximum Suppression, described in the previous section of this paper, was one tool designed to prevent this, but this was not completely effective. However, these errors could be corrected utilizing knowledge about the likely position of the joints in relation to each other. While the animal was walking in a constant direction, the joints maintained a relatively constant distance from each other. The closest joints were the ankle joint and the toe joint, but these remained more than 30 pixels away from each other when the monkey was recorded such that the plane of the camera was parallel to its motion. The first layer of post-processing was to find detections that were within 20 pixels away from each other. Multiple detections whose euclidean distance were within this threshold were considered to be detections of the same joint, and their values were averaged resulting in one detection.

Predicting Locations of Undetected Joints

Information about the movement of the animal allowed joints that had not been detected by the localization step to be identified based on their likely location. The video recordings capture one frame every hundredth of a second. Therefore the location of the same joint in two adjacent frames would be confined to remain within a certain distance of each other, given some maximum speed that the joint is able to move.

Thus the locations of joints that are momentarily undetected can be estimated if their location is known in adjacent frames. Missing joint detections were inserted using the following algorithm:

1. For a given frame, iterate through each detection from the previous frame. If there is a joint that was detected in the previous frame, that is not close to any detection in the current frame, add it to a list of undetected joints.

2. For each joint in this list, check the following frames. If there is a joint in the subsequent frames that is close to this 'missing joint', then consider it momentarily lost.

3. For each joint that is momentarily lost, estimate its position in the current frame by linearly interpolating between its last and next known position.

In other words, if a joint does not appear in a frame, the algorithms looks ahead to see if that joint appears in the immediately following frames. If it does, its position in the next and last frame is used to estimate its position in the current frame. This is illustrated in the below diagram.

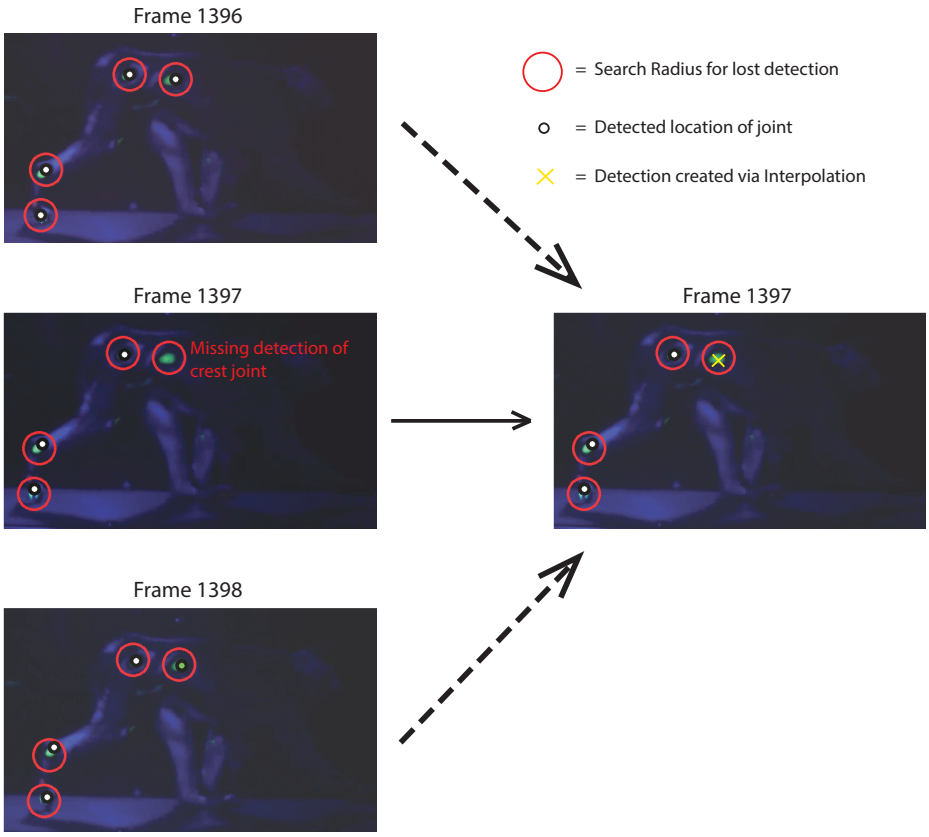

Figure 4.1

While the crest joint of the animal is undetected in frame 1397 of the recording, it is captured in both the previous and following frames. This joint is identified as missing by comparing it to the detections in frame 1396. There is no joint in frame 1937 within the distance threshold (indicated in red) of the detection in the previous frame. However, a joint does appear within this threshold in the following frame. The algorithm then inserts a detection into the frame using the location of the crest joint from the previous and subsequent frames.

4.3 Results

Ground Truth

As with training the R-CNN, the hand-tracked data from SIMI had to be modified before it could be used as an accurate ground truth. This is because the SIMI data stores the location of the joints by limb. That is, the location of all the joints on a given limb are stored together and contains the coordinated of each joint in a given frame. However, the detection from R-CNN are stored by camera angle. Each camera angle may see both hind-limbs, as the primate rotated multiple times throughout the recordings. During these rotations, the hand-annotations gathered are incomplete or inaccurate. The R-CNN detections may capture one or both of the hind-limbs.

The result is that the precision of the detections is poor when the primate is rotating. R-CNN may have detection of both limbs, while the hand-tracked data has neither. For the purposes of post-processing, frames that have not been fully annotated presented an obstacle, as they caused an artificially high precision for post-processing algorithms that resulted in a small number of detections in these intervals. Thus only frames with 4 or 5 visible joints were included when calculating precision and recall. These frame were identified

using the same masking process as in the previous section, represented in Figure 3.4. This section measures accuracy using the processed ground truth. Initial experimental detections are obtained from R-CNN's output given one of the video recordings. To obtain these detections, R-CNN was trained on the other five videos.

Accuracy Metrics

In the previous section of this paper the metric for accuracy is a threshold on intersection over union (IoU) on the bounding boxes of the joints. A value of the IoU which exceeded this threshold is considered an accurate result. This metric is useful for measuring the effectiveness of Faster R-CNN because R-CNN is concerned with the position of the joints, as well as the area that contained the joint. It is necessary to examine bounding boxes surrounding joints rather than just the coordinates of the center of a joint because locations must be classified as joint or background. In order to classify a section of an image as a joint, the section of the image had to be correctly identified, which requires a bounding box around the relevant pixels, rather than just coordinates estimating the center of the marker.

However, the algorithm for processing the output of R-CNN utilizes the position of the joint, not the pixel values of the joint. Thus the bounding box is not a valuable metric, but rather the distance between the true position of the center of the joint and the predicted position of the center of the joint. For the remainder of this section, an accurate detection is one where the center of the joint is predicted to be within 15 pixels of where is actually appears in the recording. This is visualized below, where the true joints are painted with yellow UV paint, and a circle with radius of 15 pixels have been marked around the center of the joint in red.

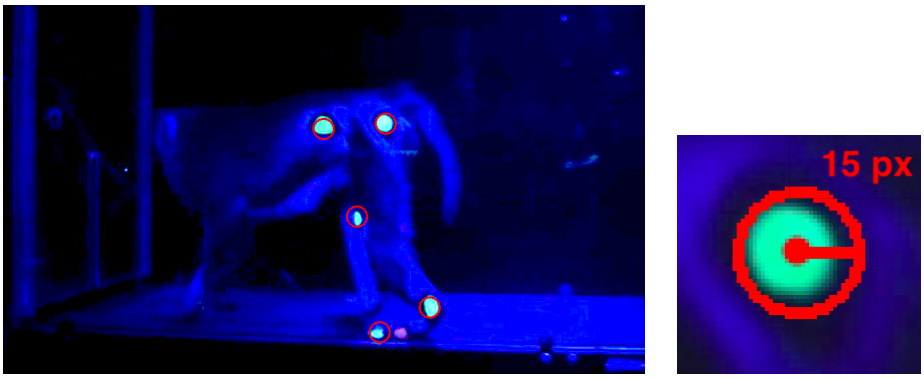

Figure 4.2

Detections that fall within this threshold are considered accurate and all others are considered false-positives. If more than one falls within this threshold, only one is considered correct and the rest are marked as false positives.

Precision and Recall

Two parameters which affect the precision and recall of the smoothed detections are Search Radius and Frame Value. Search Radius is defined by the area in which the algorithm will search for a missing joint in subsequent frame. For example, if this value is set to 40, then for a given frame, the algorithm will check that there is a joint within 40 pixels of all the detections in the previous frame. If this is not the case, then that joint is considered missing. The algorithm will look ahead to subsequent frames to check if there exists a detection within 40 pixels of the position of the missing joint in its last know location. The Search Radius parameter is represented in Figure 4.1 by the red circle surrounding each joint. A larger threshold means that finding a joint that falls within that radius in future frames is more likely. The second parameter which affects

precision and recall is the Frame Value. The Frame Value is defined as the number of frames following the current frame in which the algorithm will search for a detection of a missing joint. If it is set to 1, then the algorithm will look just in the subsequent frame to find a detections that falls within the Search Radius of a missing joint. Like with Search Radius, a larger frame value means that the probability of finding the future location of a missing joint is higher. However, the probability of finding a false positive is also higher.

Figure 4.3 represents the Recall and Precision for varied Search Radii given a fixed Frame Value. The figure illustrates that a larger Search Radius yields both a higher precision and a higher recall.

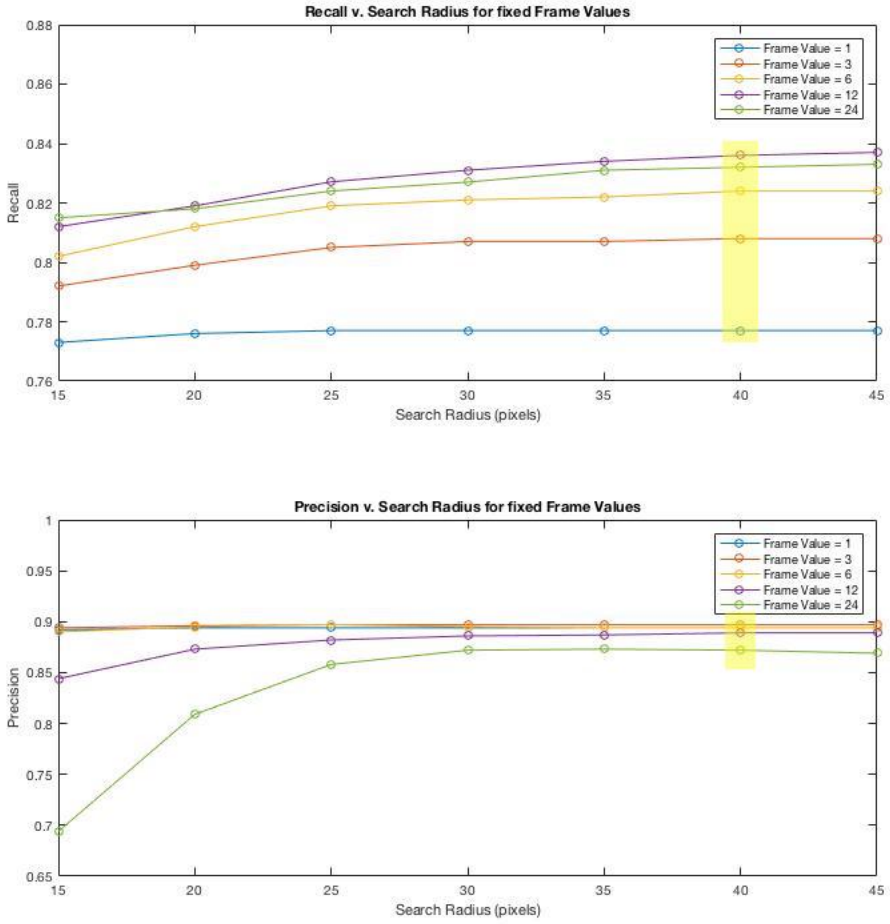

Figure 4.3

A larger value of this threshold is more lenient, as detections do not have to be as close to their original position in order to be considered adequately close to the last known location. However, this leniency yields more accurate detections than false positives. Using the above plots, the ideal Search Radius is 40 pixels (indicated in yellow), as this maximizes recall without losing precision.

Unlike with Search Radius, larger Frame Values yield a significantly worse precision. Thus, while the Search Radius performs better for larger values over both recall and precision, the Frame Value represents a trade-off where larger values have better recall, but smaller values have better precision.

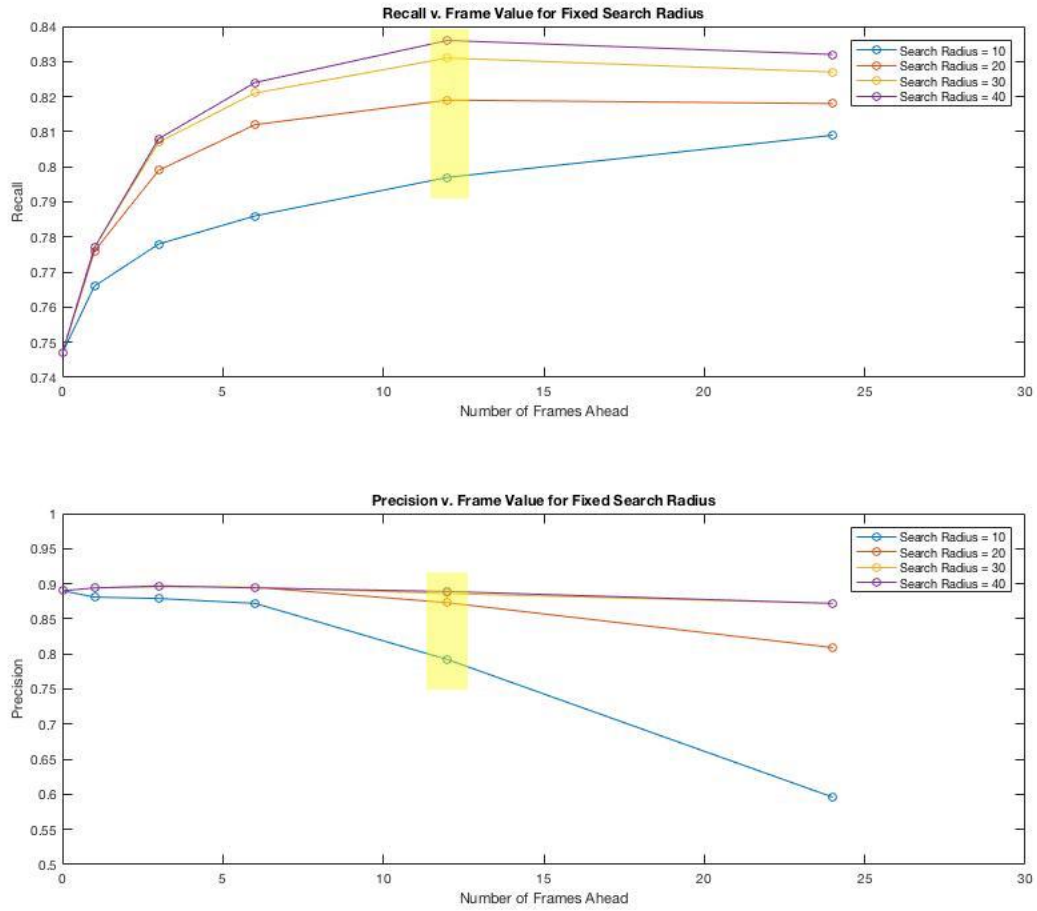

Figure 4.4

This trend is consistent with the joints of the animal being far apart from each other in any given frame, but moving through the same coordinates over time. Searching a larger radius is not likely to find the incorrect joint because each of the joints are far from each other relative to the size of the Search Radius. Furthermore, the algorithm only accepts the closest detection, so even if more than one detection is within the Search Radius, the closer one is likely to be the correct one.

However, when looking over many frames it is likely that one joint will move into the position previously occupied by a different joint. For example, the hip joint of the animal is the same height as the crest of the animal, so as the animal moves the hip joint may occupy the same space as the previous position of the crest joint. It will then be selected as the next location of the missing joint, as it is the 'closest' to the missing joint's last known location. Using Figure 4.4, the recommended Frame Value is 12 frames ahead. At this value, the recall has begun to plateau, but precision has not yet begun to fall drastically for a fixed Search Radius of 30 or 40.

All of the above plots can be visualized in a single figure, plotting recall and precision for each Search Radius and each Frame Value.

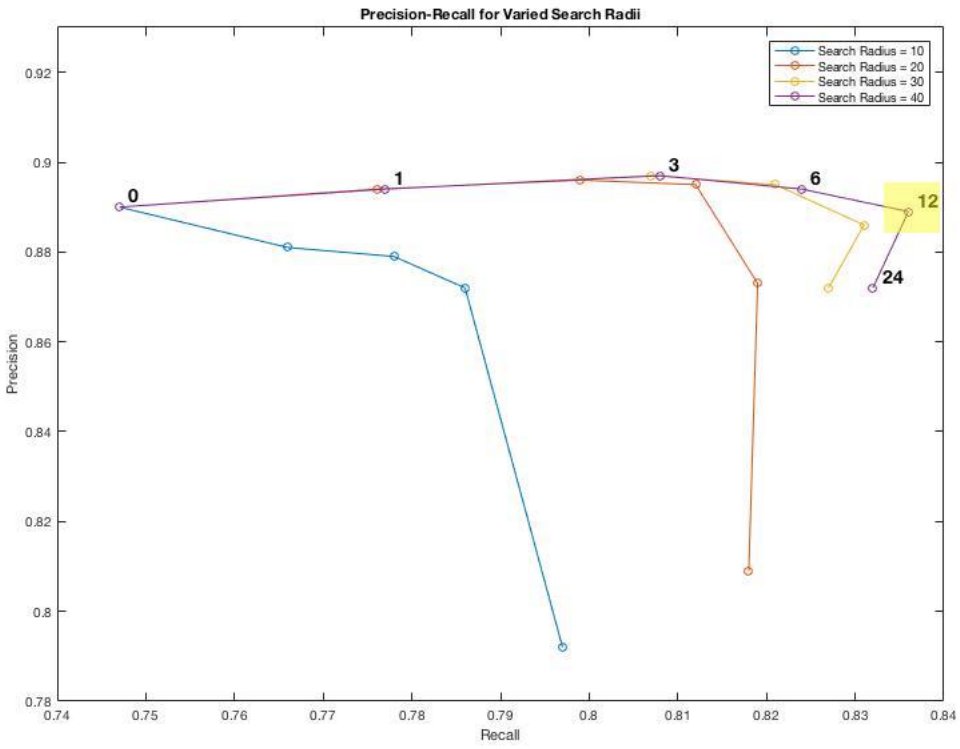

Figure 4.8

Again, this confirms the selection of 40 as a Search Radius and 12 as a Frame value, as this data point has the highest recall value, and one of the highest precision values. Additionally, we can see that both the precision and recall decrease for a frame value of 24 given a Search Radius of 20, 30 or 40. This is consistent with the joints of the animal remaining far apart in a given frame, but moving through the same space over time. Within 24 frames, the closest detection to the last known location of a joint may be the location of a different joint. Thus the interpolation between these two joints does not yield an accurate result. However, this issue does not arise as frequently within 12 frames of video.

5 Conclusion

The three steps of this pipeline (pre-processing, localization and post-processing) are able to produce detections at a much faster pace than SIMI semi-automatic tracking and a much higher accuracy than SIMI fully automatic tracking. SIMI semi-automatic tracking is able to produce near perfect results, where the only error is that of the human annotator. However, each frame takes at least 10 seconds to annotate completely. It takes over 27 hours to fully mark every joint on a video of 10,000 frames. SIMI fully automatic tracking is a much faster algorithm. The predictions are generated in under 10 minutes (plus additional time to label each joint). However, these predictions are extremely inaccurate. SIMI's precision is 0.048 and its recall is 0.040 when tested on 100 frames of the recording used in the previous section.

The pipeline discussed in this paper is able to produce a recall of 0.836 and a precision of 0.889 in under an hour. The preprossessing step crops 10,000 frames in approximately 10 minutes. The localization step

makes initial predictions of the joint position in 0.25 second per frame. This is approximately 40 minutes for one video of 10,000 frames. Finally the post-processing step removed duplicated detections and interpolates missing detections in under one minute for all 10,000 frames. This offers an alternative to the fully or semi-automatic tracking with SIMI that is rapid, but still accurate.

References

[1] David Xing David Borton. Diagram created for borton lab, 2016.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Sen ching S. Cheung and Chandrika Kamath. Robust techniques for background subtraction in urban traffic video. *Proc*, SPIE 5308, 2004.

[4] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface, 1998.

[5] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[6] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[8] Scikit Learn. Average precision score. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html.

[9] Jonathan Taylor statsmodels-developers Josef Perktold, Skipper Seabold. Vector autoregression. https://github.com/statsmodels/statsmodels, 2017.