robotics

MDPI

Article

Reliable Real-Time Ball Tracking for Robot Table Tennis

Sebastian Gomez-Gonzalez [1,*], Yassine Nemmour [1], Bernhard Schölkopf [1] and Jan Peters [1,2]

1 Max Planck for Intelligent Systems, Max Planck Ring 4, 72072 Tübingen, Germany; ynemmour@tuebingen.mpg.de (Y.N.); bs@tuebingen.mpg.de (B.S.)
2 Technische Universität Darmstadt, Computer Science Department; Hochschulstr. 10, D-64289 Darmstadt, Germany; mail@jan-peters.net
* Correspondence: sebastian@robot-learning.de

Received: 20 August 2019; Accepted: 18 October 2019; Published: 22 October 2019

check for updates

Abstract: Robot table tennis systems require a vision system that can track the ball position with low latency and high sampling rate. Altering the ball to simplify the tracking using, for instance, infrared coating changes the physics of the ball trajectory. As a result, table tennis systems use custom tracking systems to track the ball based on heuristic algorithms respecting the real-time constrains applied to RGB images captured with a set of cameras. However, these heuristic algorithms often report erroneous ball positions, and the table tennis policies typically need to incorporate additional heuristics to detect and possibly correct outliers. In this paper, we propose a vision system for object detection and tracking that focuses on reliability while providing real-time performance. Our assumption is that by using multiple cameras, we can find and discard the errors obtained in the object detection phase by checking for consistency with the positions reported by other cameras. We provide an open source implementation of the proposed tracking system to simplify future research in robot table tennis or related tracking applications with strong real-time requirements. We evaluate the proposed system thoroughly in simulation and in the real system, outperforming previous work. Furthermore, we show that the accuracy and robustness of the proposed system increases as more cameras are added. Finally, we evaluate the table tennis playing performance of an existing method in the real robot using the proposed vision system. We measure a slight increase in performance compared to a previous vision system even after removing all the heuristics previously present to filter out erroneous ball observations.

Keywords: object tracking; multiple camera stereo; real-time robotics

1. Introduction

Game playing has been a popular technique to compare the performance of different artificial intelligence methods between themselves and against humans. Examples include board games like Chess [1] and Go [2] as well as sports like robot-soccer [3]. Table tennis has been used regularly as a robot task to evaluate the performance of ad-hoc techniques [4], imitation learning [5], reinforcement learning [6] and other techniques in a complex real-time environment.

In order to play table tennis, a robotic system needs reliable information about the ball trajectory with low latency and high sampling frequency. Commercial tracking systems like VICON can provide reliable 3D positions with high sampling frequencies, but it requires attaching IR reflective markers to the objects to track. Table tennis balls are very light, and it is not easy to attach an IR marker or even coat the ball surface with IR reflective paint without changing the physics of the ball trajectory. For this reason, robot table tennis approaches typically use software-based solutions that take images from a set of video cameras and estimate the 3D position of the ball.

Tracking systems for table tennis balls use fast heuristics to detect the ball respecting real-time constrains required by table tennis systems. These heuristics typically look for round objects and use color information of table tennis balls. Although these heuristics work well most of the time, assuming that the reported ball positions are always correct before the 3D triangulation will result in a number of outliers that increases as more cameras are used in the tracking system.

As a result, robot table tennis systems need to incorporate outlier detection [7] techniques on the reported 3D positions using for example physical models of the ball trajectory [5]. This is unfortunate, since it results in effort duplication and reduces the interest of the machine learning community to work on real robot table tennis platforms.

In this paper, we propose a simple and efficient framework for object tracking. The proposed framework is tested on a robot table tennis setup and compared with previous work [8]. Unlike previous work, we focus on the reliability of the system without the use of any strong assumptions about the object shape or the physics of the flying ball. To evaluate the performance of the algorithm in setups with different amount of cameras, we use a simulation environment. We show that adding more cameras helps to increase the robustness and the accuracy of the proposed system.

In the real system, we evaluate the error distribution of the proposed system and compare it with previous work [8]. We show that the proposed framework is clearly superior in accuracy and robustness to outliers. Finally, we evaluate the system by using a robot table tennis policy [9] that was designed to be used with the RTBlob vision system [8]. We remove all the heuristics to detect and remove outliers from the policy implemented [9] and still obtain a slight improvement of performance compared using the proposed vision system. Figure 1, shows the real robot setup used on the experiments, executing the policy proposed in [9] with the vision system proposed on this paper.

Figure 1. Robot table tennis setup used to evaluate the proposed methods. We use four cameras attached to the ceiling to track the position of the ball. The robots used are two Barrett WAM robot arms capable of high speed motion, with seven degrees of freedom like a human arm.

Although we focus on robot table tennis due to its particular real-time requirements, we use machine learning techniques for the object detection part that can be trained to track different kinds of objects. A user only needs to label a few images by placing a bounding box around the object of interest and train the system with the labeled images.

1.1. Contributions

We provide an open source implementation [10] of a simple table tennis ball tracking system that focuses on reliability and real-time performance. The implementation can be used to track different objects simply by retraining the model. The provided open source implementation will enable researchers working on robot table tennis or related real-time object tracking applications to focus their efforts into better strategies or models, instead of devising strategies to determine which observations can be trusted and which can not.

We evaluate the proposed system in simulation and in a real robot table tennis platform. In simulation, we show that increasing the number of cameras results in higher reliability. On the real system, we evaluate an existing robot table tennis strategy using the proposed vision system with four cameras attached to the ceiling. The heuristics used to discard outliers on the ball observations where removed, while obtaining a slightly increase on playing performance. In addition, we provide latency times for the different experiments to show the proposed system can deliver real-time performance even with a large number of cameras.

1.2. Related Work

Ball tracking systems take an important role in almost all popular ball based sports to aid coaches, referees and sport commentators. Examples include soccer [11,12], basketball [13], tennis [14], etc. There are multiple systems designed for tracking table tennis balls, some of which include real-time considerations or were designed for robot table tennis. Table tennis is a fast game, and that makes it a hard robot problem to tackle. A smashed ball takes about 0.1 seconds to reach the other end of the table, and even at beginner level, it takes about 1 second for the ball to reach the opponent. Considering that robot arms like the Barrett WAM are much slower than a human arm, the amount of time available to make a decision of how and where to move before it is too late to reach the ball is low even to play at a beginner level. As a result, a vision system for robot table tennis needs to provide a high sampling rate with a low latency to provide as much information as possible as early as possible.

RTblob [8] was one of the first vision systems used for robot table tennis applications. It uses four color cameras to track the position of the ball. To find the position of the ball on an image, this system uses a reference orange color and convolves the resulting image with a circular pattern using the fast Fourier Transform for efficiency. Instead of using the four cameras to output one single 3D ball position, this system uses two pairs of two cameras. As a result, if all the cameras are seeing the ball, two 3D position are estimated. In this system, it is not clear how to use more cameras or how to determine which observations are reliable or not. Each table tennis policy that used RTBlob had to implement its own outlier rejection heuristics to determine which produced ball observations were reliable.

There are several other vision systems for robot table tennis, but none of them addresses the problem of how to deal with mistakes from the object detection algorithm in the images. Quick MAG 3 [15] uses a motion blur and a ball trajectory model to estimate and predict ball trajectories. In [16], a background model is used to extract the position of the ball. The detected blobs are filtered out according to their area, circularity and other factors. Finally, a ball model is used to predict the ball trajectory. In [17], the authors focus on the physical models useful to predict the ball trajectory, and use these models for humanoid robot table tennis.

A common design pattern for all the discussed table tennis vision systems, is that the object detection part consists on multiple heuristics based on background subtraction, color templates and basic shape matching on blobs. These approaches tend to work well in practice and satisfy the real-time requirements of robot table tennis. Machine learning based methods, on the other hand, have been typically not fast enough for real-time robotics, but have the potential to be easily adapted to track different objects by simply labeling new images. Heuristic based methods are hard to adapt to track different objects. We discuss two different machine learning approaches that can be used to find the position of the ball in the image, showing that it is possible to obtain the real-time performance required for robot table tennis.

2. Reliable Real-Time Ball Tracking

End-to-end systems are an appealing strategy for system design in machine learning research, because it makes less assumptions about how the system works internally. For our table tennis vision setup, an end-to-end system should receive the input images from all the cameras and output the corresponding ball location in 3D cartesian coordinates. However, such an end-to-end solution would have a number of disadvantages for our table tennis setup. For example, adding new cameras or moving around the existing cameras would require to re-train the entire system from scratch.

We divide our vision system into two subsystems. The object detection subsystem that outputs the ball positions in pixel space for each image, and the position estimation subsystem that outputs a single 3D position of the ball based on a camera calibration procedure. To add new cameras we only need to run the calibration procedure, and moving existing cameras requires only the re-calibration of the moved cameras.

First, we discuss about different methods used in the machine learning community to detect general objects in images. In particular, we discuss about object detection and semantic segmentation methods, and their advantages and disadvantages for the ball detection problem. We show that although both methods can successfully find table tennis balls in an image, the semantic segmentation method can be used with smaller models, achieving the required real-time execution requirements we need for robot table tennis.

Subsequently, we discuss how to estimate a single 3D ball position from multiple camera observations. We focus particularly on how to deal with erroneus estimates of the ball position in pixel space, for example, when the object detection method fails and reports the location of some other object. We analyze the algorithmic complexity of the proposed methods and we also provide execution times in a particular computer for setups with different number of cameras.

2.1. Finding the Position of the Ball in an Image

The problem of detecting the location of desired objects in images has been well studied in the computer vision community [18]. Finding bounding boxes for objects in images is known as object detection. In [19], a method called Single Shot Detection (SSD) was proposed to turn a convolutional neural network for image classification into an object detection network. An important design goal of the SSD method is computational efficiency. In combination with a relatively small deep network architecture like Mobilnet [20], which designed for mobile devices, it can perform real-time object detection for some applications.

Figure 2, shows example predictions of a Mobilnet architecture trained with the SSD method in a ball detection dataset. Each picture shows a section of the image with the corresponding bounding box prediction. The resulting average processing speed using a GPU NVidia GTX 1080 was 60.2 frames per second on 200×200 pixel resolution images. For a 4 camera robot table tennis setup, this would result in about 15 ball observations per second. Unfortunately, for a high speed game like table tennis, a significantly higher number of ball observations is necessary. However, we consider important to mention the results we obtained with fast deep learning object detection techniques like the SSD method, because it can be used with our method for a different application where the objects to track are more complex and the required processing speeds are lower.

Figure 2. Ball detection with a Mobilnet deep network architecture using the Single Shot Detection (SSD) method. Note that the SSD method finds the location of the ball in all the images with relatively good accuracy. However, we obtain an average of 15 ball observations per second on a four camera setup, not efficient enough for a highly dynamic task like robot table tennis.

An alternative approach to find objects in images is to use a semantic segmentation method, where the output of the network is a pixelwise classification of the objects of interest or background. For example, [21] uses deep convolutional neural networks to classify every pixel in a street scene as one of 20 categories like car, person and road. To track the ball we only need two categories: Ball and Background. We consider background anything that is not a table tennis ball. Let us denote the resulting probability image as a matrix $\boldsymbol{B}$, where B_{ij} is a scalar denoting the probability that the pixel (i, j) of the original image corresponds to a ball pixel or not.

In order to find the actual set of pixels corresponding to the ball, we need some kind of threshold based algorithm that makes a hard zero/one decision of which pixels belong to the object of interest based on the obtained probabilities. We used a simple algorithm that consists of finding the pixel position (a, b) with maximum probability and a region of neighboring pixels with a probability higher than a given threshold.

Algorithm 1 shows the procedure to obtain the set of pixels corresponding to the ball from the probability image $\boldsymbol{B}$. The procedure receives two threshold values T_h and T_l, that we call high and low thresholds, respectively. In Line 1, we find the pixel position (a, b) with maximum probability on the probability image $\boldsymbol{B}$. If the maximum probability is lower than the high threshold value T_h we consider that there is no ball in the image and return an empty set of pixels. Otherwise, Lines 5 to 15 find a region of neighboring pixels O around the maximum (a, b) with a probability larger than the low threshold T_l using a Breadth First Search algorithm. The center of the ball is computed by averaging the pixel positions in O.

Algorithm 1 Finding the set of pixels of an object.

Input: A probability image $\boldsymbol{B}$, and a high and low thresholds T_h and T_l.
Output: A set of object pixels O

1: $(a, b) = \arg\max_{(a,b)} B_{ab}$
2: **if** $B_{ab} < T_h$ **then**
3:     **return** $\emptyset$
4: **end if**
5: $O \leftarrow \{(a, b)\}$
6: $q \leftarrow$ Queue($\{(a, b)\}$)
7: **while** q is not empty **do**
8:     $\boldsymbol{x} \leftarrow$ pop(q)
9:     **for** each neighbors $\boldsymbol{y}$ of $\boldsymbol{x}$ **do**
10:         **if** not $\boldsymbol{y} \in O$ and $B_{\boldsymbol{y}} > T_l$ **then**
11:             push($q, \boldsymbol{y}$)
12:             $O \leftarrow O \cup \{\boldsymbol{y}\}$
13:         **end if**
14:     **end for**
15: **end while**
16: **return** O

The computational complexity of Algorithm 1 is linear on the number of pixels. If N_t represents the total number of pixels in the image and N_o the number of pixels of the object to track, the computational complexity of Line 1 alone in $O(N_t)$ and the complexity of the rest of the algorithm is $O(N_o)$. However, Line 1 can be efficiently implemented in a GPU, whereas the rest of the algorithm is harder to implement on a GPU due to its sequential nature. Given that $N_t \gg N_o$, we decided to use the GPU to execute Line 1 and implemented the rest of the algorithm in the CPU. In combination with the semantic segmentation approach using a single convolutional unit, we obtained a throughput about 50 times faster than the SSD method for our ball tracking problem.

Figure 3 shows the semantic segmentation results for the table tennis problem using a single convolutional unit with a 5×5 pixels filter size. The picture on the left shows a section of the image captured with our cameras. The picture on the center shows the probability image $\boldsymbol{B}$ assigned by the model to each pixel as being the ball, where white means high probability and black low probability. The picture on the right shows a bounding box that contains all pixels in O returned by Algorithm 1. Note that all the objects in the scene that are not the ball are assigned by the model a very low probability of being the ball, and most of the pixels of the ball are assigned a high probability of being the ball. Actually, the only object that can still be seen not completely dark in the probability image is the human arm, because it has a similar color to the ball in comparison with the rest of the scene.

Figure 3. Ball detection using a single convolutional unit in a semantic segmentation setting. The image on the left shows a section of a table tennis scene. The image on the center shows the probability image $\boldsymbol{B}$ representing the probability assigned to each pixel of being the ball. Dark means low probability and bright means high probability. The image on the right shows the detected ball position. This simple model can successfully find the ball in the image, and it is around 50 times faster than the SSD method.

The throughput of the single 5×5 convolutional unit is about 50 times higher that the throughput of the SSD method on the same hardware with our implementations. As a result, we decided to use the single convolutional unit as the ball detection method, achieving the necessary ball observation frequency and accuracy for robot table tennis. In Section 3, we analyze in detail the performance and accuracy of the single convolutional unit. In addition, we compare the accuracy of our entire proposed system with the RTBlob vision system [8] and evaluate the playing performance of an existing robot table tennis method [9] using the proposed system.

2.2. Robust Estimation of the Ball Position

Once we have the position of the ball in pixel space in multiple calibrated cameras, we proceed to estimate a single reliable 3D ball position. The process to obtain an estimation of the 3D position of an object given its pixel space position in two or more cameras is called stereo vision. For an overview in stereo vision refer to [22].

Previous work on vision systems for robot table tennis focused on providing real-time 3D ball positions without considering the possibility of errors in the ball detection algorithm. As a result, robot table tennis systems like [5] had to include outlier detection techniques on the 3D observations using for example physics models. When an observation deviated significantly from the position predicted with the physics model, the observation was discarded. The main drawback of this approach is that we have to discard a complete 3D observation even if only one camera made a mistake localizing the ball in pixel space and the rest of the cameras provided a correct observation. Adding more cameras to such a system would end up in discarding more ball observations instead of improving the entire system reliability. Instead, our approach consists on detecting outliers on the 2D pixel space ball positions individually for each camera. If a small set of cameras detect the ball position incorrectly, but a different set of cameras detect the ball correctly, we can still provide a 3D observation using the correct set of cameras only.

Outlier detection algorithms have received a lot of attention by the scientific community. The more common approaches consist of detection of atypical values in a unsupervised or supervised fashion [23], modeling a distribution of the typical values in the former case or detecting outliers as a classification problem on the latter. These kind of approaches are not usuful in our case, since the distribution of the ball position in the image space is not expected to be different from the position of other objects that could be mistaken by the ball. Instead, we focus on outlier detection by consensus [24]. Examples of algorithms for outlier detection by consensus include RANSAC [25], MLESAC [26], NAPSAC [27] and USAC [28]. All these algorithms consist on taking a random subsample of the observation set, fitting the parameters of the model of our interest only on the subsample set, and keeping best model parameters according to some optimality criteria. The differences between all these algorithms lies on the criteria to select the subsets of observations and to decide what is the best model found so far.

The RANSAC (Random Sample Consensus) algorithm [25] is probably the most popular method of outlier detection by consensus used in the computer vision community [24]. The random subsamples are typically taken uniformly at random. A model hypothesis is generated by fitting the model with the random subsample. An observation is considered consistent with the hypothesis if the error is smaller than a parameter ϵ, and the hypothesis with the largest number of consistent observations is selected as the best one. The MLESAC [26] algorithm, maximizes the likelihood of the inlier and outlier sets under a probabilistic model instead of maximizing the size of the consistent set. The NAPSAC [27] algorithm uses the observation that inliers often are closer to each other than outliers. Instead of selecting the subsample set uniformly at random, the NAPSAC algorithm generates a subsample set taking into consideration the distance between the observations.

We propose a consensus-based algorithm that maximizes the size of the support set like RANSAC does, but instead of selecting the subset of observations at random, we try all possible camera pairs, guaranteeing to find a solution if it exists. In the rest of this section, we explain in detail our consensus-based algorithm to find a single the 3D position from several 2D pixel space observations containing outliers. We assume we have access to two functions project and stereo available from an stereo vision library, as well as the projection matrices P_i for each camera i. Given a 3D point X, the function $x_i = \text{project}(X, P_i)$ returns the pixel space coordinates x_i of projection of X in the image plane of camera i. For the stereo vision method, we are given a set of pixel space points $\{x_1, \ldots, x_k\}$ from k different cameras and their corresponding projection matrices $\{P_1, \ldots, P_k\}$, and obtain an estimate of the 3D point X by

$$X = \text{stereo}(\{x_1, \ldots, x_k\}, \{P_1, \ldots, P_k\}).$$

Intuitively, the function stereo finds the point X that minimize the pixel re-projection error given by

$$L(X) = \sum_k \text{dist}(x_k, \text{project}(X, P_k)),$$

where dist is some distance metric like euclidean distance.

We assume that from a set S of pixel space ball observations reported by the vision system, some of the observations $\hat{S} \in S$ are correctly reported ball positions and the rest of the reported observations $\bar{S} = S - \hat{S}$ are erroneously reported ball positions. We call $\hat{S}$ the inlier or support set and $\bar{S}$ the outlier set. We would like to find the 3D ball position X that minimizes $L(X)$ using only the support set $\hat{S}$.

We define a set of pixel space observations as consistent if there is a 3D point X such that $L(X) < \epsilon$, where ϵ is a pixel space error tolerance. We estimate $\hat{S}$ by computing the largest subset of S that is consistent. The underlying assumption is that it should be hard to find a single 3D position that explains a set of pixel observations containing outliers. On the other hand, if the set of observations contains only inliers, we know it should be possible to find a single 3D position X, the cartesian position of the ball, that explains all the pixel space observations.

Algorithm 2 shows the procedure we use to obtain the largest consistent set of observations. Note that we need at least two cameras to estimate a 3D position. Our procedure consists in trying all pairs of cameras (i, j), estimating a candidate 3D position only with those two observations, and subsequently counting how many cameras are consistent with the estimated candidate position. If c represents the number of cameras reporting a ball observation, the computational complexity of this algorithm is $O(c^3)$.

For a vision system of less than 30 cameras, we obtained real-time performance even using a sequential implementation of Algorithm 2. Nevertheless, it is easy to parallelize Algorithm 2. Note that the outermost two for loops can be run independently in parallel. In Section 3, we evaluate the real-time performance and accuracy of the 3D estimation simulating scenarios with different number of cameras and probability of outliers. Unlike previous robot table tennis systems that discard entire 3D observations using physics based models [5], we obtain improved accuracy and less dropped observations as the number of cameras on the vision system is increased. We also evaluate the error in the real system and compare it with the RTBlob method using the same experimental setup with four cameras, obtaining a higher accuracy and reliability with the vision system proposed in this paper.

Algorithm 2 Remove outliers by finding the largest consistent subset of 2D observations for stereo vision.

Input: A set of 2D observations and camera matrix pairs $S = \{\{x_1, P1\}, \ldots, \{x_k, P_k\}\}$, and pixel error threshold ϵ.
Output: A subset $\hat{S} \subset S$ of maximal size without outliers.

$\hat{S} \leftarrow \emptyset$
for $i \in \{1, \ldots, k-1\}$ **do**
  for $j \in \{i+1, \ldots, k\}$ **do**
    $candidate \leftarrow \text{stereo}(\{P_i, P_j\}, \{x_i, x_j\})$
    $S_{ij} \leftarrow \emptyset$
    for $k \in \{0, \ldots, k\}$ **do**
      $\hat{x}_k \leftarrow \text{project}(candidate, P_k)$
      $p\_err \leftarrow \|x_k - \hat{x}_k\|_2$
      if $p\_err < \epsilon$ **then**
        $S_{ij} \leftarrow S_{ij} \cup \{x_k, P_k\}$
      end if
    end for
    if $|S_{ij}| > |\hat{S}|$ **then**
      $\hat{S} \leftarrow S_{ij}$
    end if
  end for
end for
return $\hat{S}$

3. Experiments and Results

We evaluate the proposed system in a simulation environment and in a real robot platform. In simulation, we measure the accuracy of the system as we increase the number of cameras and when we change the probability of obtaining outliers. We use the real robot platform to evaluate the interaction of all the components of the proposed system. In particular, we measure the accuracy and robustness of the proposed system and compare it with the RTBlob method. In addition, we evaluate the success rate of a method proposed in [9] to return balls to the opponent's court with the proposed vision system. We have a slightly higher success rate using the proposed vision system than using the RTBlob system even after removing all the outlier rejection heuristics implemented in [9].

3.1. Evaluation on a Simulation Environment

To evaluate the proposed methods in scenarios that include different number of cameras and probability of outliers, we use a simulation scenario. The advantage of evaluating in simulation is that we have access to exact ground truth data and we can easily test the robustness and accuracy of the system. In this section, we evaluate the robustness of the introduced procedure to find the 3D position of the ball from several unreliable pixel space observations. First, we want to evaluate the performance of Algorithm 2 independently of the rest of the system. In addition, we want to test the accuracy and running time of the algorithm for different amount of cameras and outlier rates.

The simulation for a scenario with c cameras and a probability of outlier p_o consists of the following steps: First, we generate randomly a 3D ball position X in the work space of the robot and project it to each camera using the calibration matrices. We add a small Gaussian noise with a standard deviation of 1.3 pixels to the projected pixel space position, because that is the average re-projection error reported by the camera calibration procedure. For each camera, we replace the obtained pixel space position by some other random position in the image plane with probability p_o. Subsequently, we attempt to obtain the 3D ball position with Algorithm 2. If it fails to obtain any position at all, we count it as a failure. Otherwise, we measure the error of the obtained position with the ground truth value X.

Table 1 shows the results for scenarios with a number of cameras ranging from four to 30 and probability of outliers ranging from 1% to 50%. For every combination of number of cameras and probability of outliers, we report the failure rate (F) and the error (E) between the ground truth position and the reported ball position. As the probability of outliers increases the error and failure rate increases as it is expected. Similarly, as more cameras are added to the system, the robustness of the system increases, obtaining smaller errors and failure rates. There are few entries in Table 1 that seem to contradict the trend to reduce the error as more cameras are introduced or the outlier rate drops. For example, for an outlier rate of 50% the error with four cameras is 4.67 cm whereas the error for eight cameras is 6.84 cm. Note however that the failure rate for four cameras is much higher than for eight cameras in this case.

Table 1. Estimation error (E) and failure probability (F) of the 3D position estimation procedure in the presence of outliers. A failure means that the system does not report any ball position at all because the maximum consistent set returned by Algorithm 2 consisted of less than two ball observations. Otherwise, the system return an estimated ball position and we report the distance in centimeters to the ground truth position. We simulate multiple scenarios with a different number of cameras and different probability of outliers. Note that as the number of cameras increases and the probability of obtaining outliers decreases the system becomes more reliable.

c		Probability of Outliers p_o				
		1%	**5%**	**10%**	**25%**	**50%**
4	E	0.71 cm	0.85 cm	0.84 cm	0.79 cm	4.67 cm
	F	0.1%	0.5%	2.0%	9.7%	37.7%
8	E	0.52 cm	0.53 cm	0.59 cm	0.94 cm	6.84 cm
	F	0.0%	0.0%	0.0%	0.1%	4.5%
15	E	0.35 cm	0.36 cm	0.37 cm	0.41 cm	4.72 cm
	F	0.0%	0.0%	0.0%	0.0%	0.02%
30	E	0.24 cm	0.25 cm	0.25 cm	0.28 cm	0.35 cm
	F	0.0%	0.0%	0.0%	0.0%	0.0%

Adding more cameras to the system improves accuracy and robustness. However, it also increases the computation cost. The cost of the image processing part grows linearly with the number of cameras, but can be run independently in parallel for every camera if necessary. Therefore, we will focus on the cost of the position estimation procedure as the number of cameras grows. As discussed in Section 2.2, the cost of the position estimation procedure in $O(c^3)$. Table 2 shows the run time in milliseconds of a sequential implementation of Algorithm 2 in C++ in a Lenovo Thinkpad X2 laptop. For a target frequency rate of 200 observations per second we need a processing time smaller than 5 milliseconds. Note that even the sequential implementation of Algorithm 2 has the required real-time performance for systems up to 30 cameras. In addition, Algorithm 2 can be easily parallelized if necessary as discussed in Section 2.2.

Table 2. Run time in milliseconds of a sequential implementation of Algorithm 2 with respect to the number of cameras. For a system of up to 30 cameras, the sequential implementation of Algorithm 2 provides real-time performance for more than 200 ball observations per second.

Cameras	4	8	15	30	50
Run time (ms)	0.001	0.012	0.015	3.02	11.46

It is important to note that on a real system not all the cameras might be seeing the work space of the robot. For example, in the real robot setup we used four cameras but there are many parts of the work space that are covered only by two cameras, reducing the effective robustness of the system on those areas. However, the outlier rate of the image processing algorithms is below 1% in practice, and good results can be obtained using a small number of cameras as we discuss in the next section.

3.2. Evaluation on the Real Robot Platform

We evaluate the entire proposed system in the real robot platform and compare the performance to the RTBlob system presented in [8]. The evaluation on the real robot platform consisted of two experiments. First, we attach a table tennis ball to the robot end effector. We move the robot and use its kinematics to compute the position of the ball and use it as ground truth to compare against the ball positions obtained with the vision system. Finally, we evaluate the playing performance of the robot table tennis strategy introduced in [9] if we remove all the heuristics used to remove vision outliers.

We compare the performance of the proposed vision system with RTBlob [8]. The RTBlob system has been used for robot table tennis experimentation [5,7]. In order to compare the accuracy of both systems, we need access to ground truth positions. We use the joint sensors of the robot and the robot kinematics to compute the Cartesian position of the robot end effector. We attach the ball to the robot end effector and use the Cartesian position computed with the joint measurements as ground truth.

Figure 4 shows a histogram of the error of the RTBlob method and the method proposed in this paper. We called the proposed system RT^2 in the figure, standing for Real-Time Reliable Tracking. The error is computed as the distance between the position reported by the vision and the ground truth computed with the robot kinematics. Note that the proposed vision system outperforms the RTBlob method in terms of accuracy, but specially in terms of outliers. The distribution of errors for RT^2 concentrates the probability mass between 0 and 5 cm error. On the other hand, the error distribution of the RTBlob method is multimodal. The first mode corresponds to the scenario where all the cameras detected the ball correctly, and in this case the error mass is also concentrated below a 7 cm error threshold. The second mode shows a high probability of error between 25 and 30 cm, and it is likely to correspond to a scenario where one of the four cameras reported an incorrect ball position.

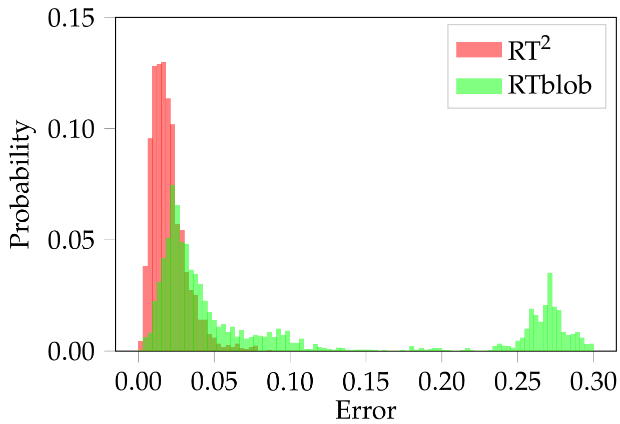

Figure 4. Histogram of the error of the presented vision system and RTBlob. We call the proposed vision system RT^2, depicted in red in the histogram. The ball is attached to the robot end effector and the end effector position computed with the kinematics is used as ground truth. The accuracy of the proposed vision system is superior to the RTBlob system. In terms of robustness to outliers, the proposed system (RT^2) outperforms the RTBlob system as expected. The error distribution for our system is unimodal, whereas the RTBlob system error is multimodal, reflecting the sensitivity of the RTBlob system to the presence of outliers.

The total error of the proposed system on the real robot experiments is 1.99 cm $\pm$ 1.15 cm, which is comparable with the radius of the ball of 2 cm. During the execution of this experiment, the proposed system never reported any ball position whose error was larger than 10 cm. On the other hand, the RTBlob system reported errors on the order of tens of meters with probability around 0.1%. As a result, the table tennis strategies that use the RTBlob method have to incorporate strategies to filter outliers to work properly.

In the last part of this section, we present a final experiment where we use the proposed vision system to return table tennis balls with the robot to the opponent's court. We use a method presented in [9], that is based on Probabilistic Movement Primitives (ProMPs) and learning from a human teacher. The system presented in [9] was originally designed to use the RTBlob method as the vision system. To detect and filter outliers, the RANSAC algorithm was used on a set of initial observations fitting a second order polynomial. Once a set of candidate positions is found, a Kalman filter is used to predict the ball trajectory and subsequent ball observations are rejected if they are more than three standard deviations away from the mean position predicted by the Kalman filter.

We decided to remove the heuristics to filter outliers, accepting all ball observations as valid, and test the method with the proposed vision system. We define "success" as the robot hitting the incoming ball and sending it back to the opponent's court according to the table tennis rules. The average success rate using the RTBlob vision system and all the outlier rejection heuristics was 68%, whereas using the proposed vision system and no outlier rejection heuristics the average success rate was 70%. Given the variability of the table tennis performance between multiple experiments, we can not say that the improvement with the new vision system is statistically significant. However, we find it remarkable that the success rate of table tennis strategy presented in [9] did not decrese after the outlier rejection heuristics were removed. We think that the slight improvement on the success rate by using the proposed vision system is due to the improved frame rate, that is about three times as high as that of the RTBlob implementation provided by the authors [8].

4. Conclusions and Discussion

This paper introduced a vision system for robot table tennis focused on reliability and real-time performance. The implemented system is released as an open source project [10] to facilitate its usage by the community. The proposed vision system can be easily adapted for different tracking tasks by labeling a new dataset and training the object detection algorithm. For the object detection part, this paper evaluates two different approaches used commonly in the computer vision community that are known for obtaining real-time performance. We decided to use the simpler approach for tracking table tennis balls due to its high throughput.

For the position estimation procedure, we proposed an algorithm that focuses on reliability by assuming that some times the object detection methods will report wrong ball positions on the provided images. We evaluated the proposed method thoroughly in simulations and in the real robot platform. In simulation, we tested the accuracy of the system under different probability of outliers and number of cameras. In the real system, we evaluated the complete proposed system in a four-camera setup and compared it with the RTBlob vision system. We showed that our system provides higher accuracy, and outperforms the RTBlob system in robustness to outliers. Finally, we tested an existing technique to return table tennis balls to the opponent's court with our vision system. We removed all the outlier detection techniques used by the table tennis algorithm and obtained a small increase in success rate compared to the RTBlob system with all the outlier detection techniques present. We believe that the proposed approach will help future research in robot table tennis by allowing researchers to focus on the table tennis policies instead of techniques to deal with an unreliable vision system.

Author Contributions: Conceptualization, S.G.-G., B.S. and J.P.; methodology, S.G.-G.; software, S.G.-G. and Y.N.; validation, S.G.-G.; formal analysis, S.G.-G.; investigation, S.G.-G.; resources, B.S.; data curation, S.G.-G.; writing–original draft preparation, S.G.-G.; writing–review and editing, S.G.-G.; visualization, S.G.-G.; supervision, J.P. and B.S.; project administration, B.S.; funding acquisition, B.S.

Funding: This research was funded by the Max Planck Institute for Intelligent Systems in Tübingen, Germany.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study, the interpretation of the results, in the writing of the manuscript, or in the decision to publish the review findings.

References

1. Campbell, M.; Hoane, A.J., Jr.; Hsu, F.H. Deep blue. *Artif. Intell.* **2002**, *134*, 57–83. [CrossRef]
2. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
3. Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; Osawa, E. Robocup: The robot world cup initiative. In Proceedings of the first International Conference on Autonomous Agents (AGENTS '97), Marina del Rey, CA, USA, 5–8 February 1997; pp. 340–347.
4. Mülling, K.; Kober, J.; Peters, J. A biomimetic approach to robot table tennis. *Adapt. Behav.* **2011**, *19*, 359–376. [CrossRef]
5. Gomez-Gonzalez, S.; Neumann, G.; Schölkopf, B.; Peters, J. Using probabilistic movement primitives for striking movements. In Proceedings of the International Conference of Humanoid Robots, Cancun, Mexico, 15–17 November 2016; pp. 502–508.
6. Mülling, K.; Kober, J.; Kroemer, O.; Peters, J. Learning to select and generalize striking movements in robot table tennis. *Int. J. Robot. Res.* **2013**, *32*, 263–279. [CrossRef]
7. Huang, Y.; Schölkopf, B.; Peters, J. Learning optimal striking points for a ping-pong playing robot. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 4587–4592.
8. Lampert, C.; Peters, J. Real-time detection of colored objects in multiple camera streams with off-the-shelf hardware components. *J. Real-Time Image Process.* **2012**, *7*, 31–41. [CrossRef]
9. Gomez-Gonzalez, S.; Neumann, G.; Schölkopf, B.; Peters, J. Adaptation and Robust Learning of Probabilistic Movement Primitives. *arXiv* **2018**, arXiv:1808.10648.
10. Open Source Implementation of the Ball Tracking System. Available online: https://gitlab.tuebingen.mpg.de/sgomez/ball_tracking (accessed on 20 October 2019).
11. Seo, Y.; Choi, S.; Kim, H.; Hong, K.S. Where are the ball and players? Soccer game analysis with color-based tracking and image mosaick. In *International Conference on Image Analysis and Processing*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 196–203.
12. Tong, X.F.; Lu, H.Q.; Liu, Q.S. An effective and fast soccer ball detection and tracking method. In Proceedings of the International Conference on Pattern Recognition, Cambridge, UK, 26 August 2004; Volume 4, pp. 795–798.
13. Chen, H.T.; Tien, M.C.; Chen, Y.W.; Tsai, W.J.; Lee, S.Y. Physics-based ball tracking and 3D trajectory reconstruction with applications to shooting location estimation in basketball video. *J. Vis. Commun. Image Represent.* **2009**, *20*, 204–216. [CrossRef]
14. Pingali, G.; Opalach, A.; Jean, Y. Ball tracking and virtual replays for innovative tennis broadcasts. In Proceedings of the International Conference on Pattern Recognition, Barcelona, Spain, 3–7 September 2000; Volume 4, pp. 152–156.
15. Liu, H.; Li, Z.; Wang, B.; Zhou, Y.; Zhang, Q. Table tennis robot with stereo vision and humanoid manipulator II: Visual measurement of motion-blurred ball. In Proceedings of the International Conference on Robotics and Biomimetics, Shenzhen, China, 12–14 December 2013; pp. 2430–2435.
16. Li, H.; Wu, H.; Lou, L.; Kühnlenz, K.; Ravn, O. Ping-pong robotics with high-speed vision system. In Proceedings of the International Conference on Control Automation Robotics & Vision, Guangzhou, China, 5–7 December 2012; pp. 106–111.
17. Chen, X.; Huang, Q.; Wan, W.; Zhou, M.; Yu, Z.; Zhang, W.; Yasin, A.; Bao, H.; Meng, F. A robust vision module for humanoid robotic ping-pong game. *Int. J. Adv. Robot. Syst.* **2015**, *12*, 35. [CrossRef]
18. Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S.; et al. Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
19. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 21–37.
20. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861

21. Pohlen, T.; Hermans, A.; Mathias, M.; Leibe, B. Full-resolution residual networks for semantic segmentation in street scenes. *arXiv* **2017**, arXiv:1611.08323
22. Heyden, A.; Pollefeys, M. Multiple view geometry. *Emerging topics in computer vision*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2005; pp. 45–107.
23. Hodge, V.; Austin, J. A survey of outlier detection methodologies. *Artif. Intell. Rev.* **2004**, *22*, 85–126. [CrossRef]
24. Chum, O. Two-View Geometry Estimation by Random Sample and Consensus. Ph.D. Thesis, Czech Technical University, Prague, Czech Republic, 2005.
25. Fischler, M.A.; Bolles, R.C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
26. Tordoff, B.; Murray, D.W. Guided sampling and consensus for motion estimation. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 82–96.
27. Nasuto, D.; Craddock, J.B.R. Napsac: High noise, high dimensional robust estimation-it's in the bag. In Proceedings of the British Machine Vision Conference, University of Cardiff, September 2002; pp. 458–467.
28. Raguram, R.; Chum, O.; Pollefeys, M.; Matas, J.; Frahm, J.M. USAC: A universal framework for random sample consensus. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *35*, 2022–2038. [CrossRef] [PubMed]