A COLLABORATIVE ALGORITHMIC FRAMEWORK TO TRACK OBJECTS AND EVENTS

Somrita Chattopadhyay Constantine J. Roros Avinash C. Kak

School of Electrical and Computer Engineering, Purdue University

ABSTRACT

One faces several challenges when tracking objects and events simultaneously in multi-camera environments — especially if the events associated with the object require precise knowledge of the pose of the object at each instant of time. To illustrate the challenges involved, we consider the problem of tracking bins and their contents at airport security checkpoints. The pose of each bin must be tracked with precision in order to minimize the errors associated with the detection of the various items that the passengers may place in the bins and/or take out of them. Unfortunately, pose estimation of the bins is made difficult by the fact that as a bin moves away from the camera's optic axis, its appearance changes nonlinearly. This paper presents a collaborative algorithmic framework to address this complexity: The framework includes simple and fast 2D trackers when such tracking can be carried out reliably, and more sophisticated 2.5D trackers when the bins move away from the cameras' optic axes. The system decides automatically as to which tracker to trust the most. We validate the framework with a ground-truth dataset.

Index Terms— Tracking, Kalman Filter, CNN.

1. INTRODUCTION

A fundamental problem that is faced when it is necessary for a diverse set of algorithms to collaborate for solving a complex problem in real time deals with having to make a time-bounded decision about which algorithm to trust the most at each instant of time in order to achieve the highest overall precision in problem solving. We address this issue in the context of automating an important aspect of airport checkpoint security with multiple video cameras: Keeping track of the bins in which the passengers divest their belongings. Tracking the bins and and keeping tabs on their contents are considered important for airport checkpoint security.

This problem is made difficult by the fact that the pose of the bin must be known with accuracy at each moment in order to minimize the errors associated with the detection of the items being placed in the bins or being taken out of them. Note that it is possible for adjoining bins to be touching. So any errors in the estimation of each bin could result in wrong conclusions regarding where exactly an item was placed by a passenger, or from what bin an item was being taken out.

Bin pose estimation requires that a model of a bin — no matter how rudimentary — be fit to the features extracted from the camera images. Unfortunately, as a bin moves away from the optic axis of the camera, what would ordinarily be a rectangular appearance begins to suffer from perspective distortion, which makes it more difficult to fit the model features with the scene features.

This paper presents a fully automated algorithm that uses a convolutional network for detecting the bins in the camera images when they are first placed on the conveyor belt. Subsequently, our framework uses two different types of Extended Kalman Filter (EKF) based trackers – 2D and 2.5D. An important part of the framework is the Mediator that decides whether to use the 2D tracker or the 2.5D tracker depending on the severity of perspective distortion in the scene.

For validating the framework, we present results on ground-truthed datasets supplied by Northeastern University's ALERT Center of Excellence. The ground-truth consists of records of the events related to the placement of items in the bins or being taken out of them. We show the precision and recall numbers associated with these items.

2. PREVIOUS WORK

Bin tracking at airport checkpoints poses unique challenges due to complexity arising from partial and complete occlusions, complex object motion, variable illuminations, and noisy overhead images. The topic is studied in [1] in a simulated airport setting using Gaussian Mixture Models. However, a simulated environment does not capture actual real-life challenges. Most closely related work is by Islam et al. [2] who proposed an adaptive background aware correlation filter to track the bins. However, their algorithm does not provide precise bounding box segmentation making dynamic bin content analysis difficult. In literature, there are several model-based Kalman filter [3] and particle filter [4] frameworks for tracking the affine poses [5] of complex rigid objects. To the best of our knowledge, none of these works has accounted for the challenges encountered when the target experiences projective distortion due to change in perspective.

3. BIN DETECTION AND TRACKING

This section gives detailed description of our bin detection and tracking algorithmic framework, as shown in Fig. 1.

3.1. Bin Detection and Tracking Initiation

To start the tracking process, we use a Single Shot Detection (SSD) neural network architecture [6] to automatically determine when and where the bins first appear in the video. SSD is a convolutional neural network (CNN) that tries to detect objects at varying image scales. To make it easier to determine where to place a bounding box, SSD uses "default boxes" of varying aspect ratios and scales. Default boxes serve as reference regions at discrete locations throughout feature maps in SSD. SSD learns correspondences between default boxes and the ground truth bounding boxes and formulates a confidence score of how likely the object is within a default box. Taking the default boxes with the highest confidence score, it uses non-maximum suppression to select the final bounding box outputs.

The implementation of SSD we used accepts images of size 300×300 pixels. Therefore we resize each frame of the video and feed it into the network, which then outputs image coordinates of

a bounding box for each bin detected in the frame. The bounding box coordinates of each detected bin are used in subsequent steps to obtain its contour.

In order to ensure a bin is detected, regardless of bin orientation or camera angle, the SSD network was trained on videos from cameras at various angles. For a given video, every 30 frames we manually created ground truth bounding boxes using the Sloth labelling tool [7]. During training, the AdaDelta optimization algorithm [8] was used to refine the network model parameters with a batch size of 5 annotated frames per training iteration. Since, in a given training video, the bins appear in a wide range of positions and orientations, our SSD network is able to robustly detect bins in videos that it was not trained with.

After detecting a bin with SSD, we have to obtain the bin contour (as 4 straight lines) from these bounding box regions to form a model for pose estimation. To do this, we apply the Canny operator [9] followed by a straight primitive-based grouping algorithm [10] to these regions. We compare the bin contour obtained in the current video frame with the ones in the previous 5 video frames using Intersection over Union (IoU) score to ensure the computed model is an appropriate choice for tracking the bin. Now that we have a model to represent the bin for pose estimation, we can start tracking the bin.

3.2. Bin Tracking

We use a model matching and backtracking based EKF framework to track the pose of a bin in the successive frames.

3.2.1. 2D Pose and 2.5D Pose

In our system, the pose of a bin in every frame is formulated in the form of a 3×3 homogeneous transformation from the bin's model coordinate frame (i.e. the coordinate frame with respect to which the bin pose was initially defined). In 2D tracking, a bin pose has only 3 degrees of freedom - rotation and translation along x and y coordinates. The 2D tracker assumes the outline of the bin to be purely rectangular. However, as the bin moves away from the optic axis, its shape suffers increasingly from perspective distortion. To address this issue, we inject 2 additional degrees of freedom in the homography matrix of our EKF tracker. These two additional variables represent the scaling along the X and the Y axes to account for the dimensional changes of the bin model respectively. We refer to this as the 2.5D tracker.

3.2.2. Constraint Function for Pose Error

In each frame, we update the *apriori* pose of a bin by finding matching candidates to the projected model features predicted from the previous frame. As the bin moves in the image space, for a given pose $\mathbf{p}$, each of its model features can be projected into the camera space by applying the transformation matrix ${}^{C}H_{M}(\mathbf{p})$. Say for a given pose $\mathbf{p}$, z^j is one of the potential matches for the projection $f_{m^i,\mathbf{p}}$ of the model feature m^i. Our EKF framework tries to achieve the best estimation of the transformation matrix ${}^{C}H_{M}$ such that the pose error between z^j and $f_{m^i,\mathbf{p}}$ is minimum. Thus, our EKF pose update mechanism must satisfy the following constraint equation:

$$h(\mathbf{p}, m^i, z^j) = f_{m^i,\mathbf{p}} - z^j = 0. \quad (1)$$

3.2.3. Select Matching Candidates

Tracking in each frame starts by searching for potential matching scene feature candidates surrounding the projected model features

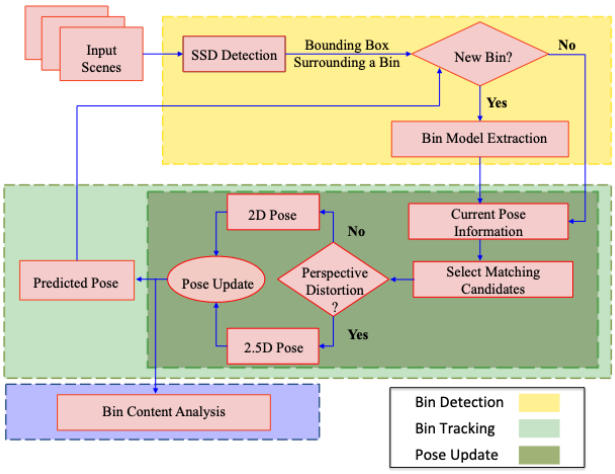

Fig. 1. Overall Framework

obtained from the previous frame's pose prediction module. Details of pose prediction is in Section 3.3. Owing to the uncertainty associated with pose prediction, we define search regions (i.e. squares) surrounding the endpoints of each model feature where we expect to find our potential matching candidates for that model feature. We assign score to each potential scene feature and projected model feature match based on the maximum distance the scene feature is from the projected model feature. The maximum distance is computed by normalizing the maximum of the perpendicular distances from the two endpoints of the scene feature onto the projected model feature. Henceforth, we would refer to this score as **distance score**. We retain only those potential matches which are within a certain distance threshold. We denote the set of matching candidate pairs as $\mathcal{C}_0$.

3.2.3.1. Hypothesis Generation Framework

Once we have the set of all possible model and scene feature correspondences, we need to identify the true correspondences i.e. the best set of model and scene feature pairs. Let $\mathcal{C}_i$ be the set of matching scene features for the i^{th} model feature. We generate a matching hypothesis set $\mathcal{C}_H \subseteq \mathcal{C}_0$ by taking each model feature and a corresponding matched scene feature. If $n(\leq 4)$ is the number of model features for which correspondences are found, the cardinality of $\mathcal{C}_H$ is $\prod_{i=1}^{n} |\mathcal{C}_i|$.

Owing to the uncertainties associated with the system, the element with the lowest score may not always be the best candidate. We, thus, select an element with the distance score d_n with an exponential probability as follows -

$$p = \frac{e^{-d_n}}{\sum_{n=1}^{|\mathcal{C}_H|} e^{-d_n}} \quad (2)$$

This ensures that the element with a lower score has a higher chance to get selected.

3.2.3.2. Hypothesis Verification Framework

The selected candidate from $\mathcal{C}_H$ is then checked using our verification framework. We construct a rectangle from the chosen scene features and compare its dimensions with the corresponding dimensions of the bin found in the last frame. We retain a hypothesis only if the dimensional changes are within certain thresholds; else, we

eliminate the matching pair with the largest distance score and repeat the verification process with the remaining pairs. We continue until the updated hypothesis satisfies the threshold check or that entire hypothesis is exhausted. If the entire hypothesis is exhausted, we eliminate that entire set of correspondences, update $\mathcal{C}_H$ and select a new element from the updated $\mathcal{C}_H$ using (2). Even if we accept a hypothesis for pose update, we may reject it later based on the decision of **Nil-Map Test** (See Section 3.2.5.1).

3.2.4. 2D and 2.5D Collaboration

Once we accept a hypothesis, the next step is to check if we need to use the 2D or the 2.5D EKF for pose update. We need the 2.5D tracker only when there is significant perspective distortion in the scene and the initial bin model needs to be updated. On the other hand, the 2D tracker is much faster and comparatively more stable. This demands a collaboration. We do 2D tracking if -

1. $|\tilde{l_i} - l_i| \leq \epsilon_i$ where l_i is the length of the i^{th} feature in the initial model, $\tilde{l_i}$ is the length of the corresponding scene feature in the accepted hypothesis and ϵ_i is a small positive number.
2. The absolute difference in angles between each pair of parallel lines in the accepted hypothesis is within 5°.
3. The absolute angular difference between each pair of perpendicular lines in the accepted hypothesis is in the interval $[\ 85^\circ, 95^\circ]$.

If any one of the three conditions is violated at a frame, we resort to the 2.5D tracker.

3.2.5. Pose Update Using Extended Kalman Filter

The bin pose in each frame is updated once we decide which EKF framework to apply. If the i^{th} model feature m^i is matched with the scene feature z^{j^i} in the j^{th} frame, and the current pose obtained based on this matched pair is $\mathbf{p^i}$ with mean $\overline{\mathbf{p}}^\mathbf{i}$ and covariance $\Sigma_{\mathbf{p^i}}$, then our EKF updates the bin pose from $\mathbf{p^{i-1}}$ to $\mathbf{p^i}$ while minimizing the constraint function in (1). This iterative updation ensures a pose that minimizes the errors for all four correspondence pairs.

If $\hat{z}^{j^i}$ is the actual measurement of z^{j^i}, the implicit function $h(.)$ in (1) can be expanded around the point $(\overline{\mathbf{p}}^{\mathbf{i-1}},\ \hat{z}^{j^i})$ using Taylor's series as: $y^i = M^i\mathbf{p} + v^{j^i}$, where y^i is the new Kalman measurement vector representing the current best state estimate $\overline{\mathbf{p}}^{\mathbf{i-1}}$. v^{j^i} is the random noise associated with the new measurement, and M^i is the linearized transformation matrix obtained by taking the gradient.

Let V^{j^i} be the associated noise covariance and K^i be the Kalman Gain. The final pose update equation is -

$$\overline{\mathbf{p}}^{\mathbf{i}} = \overline{\mathbf{p}}^{\mathbf{i-1}} + K^i(y^i - M^i\overline{\mathbf{p}}^{\mathbf{i-1}}) = \overline{\mathbf{p}}^{\mathbf{i-1}} - K^i h(\overline{\mathbf{p}}^{\mathbf{i-1}}, m^i, \hat{z}^{j^i}),$$

where

$$K^i = \Sigma_{\mathbf{p^i}} {M^i}^T (M^i \Sigma_{\mathbf{p^{i-1}}} {M^i}^T + {V^{j^i}})^{-1},$$

$$M^i = \frac{\partial h(\overline{\mathbf{p}}^{\mathbf{i-1}}, m^i, \hat{z}^{j^i})}{\partial p} \text{ and } \Sigma_{\mathbf{p^i}} = (I - K^i M^i)\Sigma_{\mathbf{p^{i-1}}}\ .$$

3.2.5.1. Updated Pose Verification using Nil-Map Test

There may not be any matching scene feature corresponding to one or more model features in the chosen hypothesis. If the updated pose is the correct one, we may now find matching scene features corresponding to those projected model features using the updated pose (cf. Section 3.2.3). If we still do not find any matching scene feature surrounding a projected model feature, we assign a nil-map.

If the number of nil-maps is above a certain threshold, we again reject the hypothesis. A lower threshold on the number of nil-maps implies higher confidence associated with the updated pose. However, this may result in a null hypothesis in presence of significant occlusion. If we reject a hypothesis, we remove the element from $\mathcal{C}_H$. We backtrack and discard the pose. Subsequently, we select a new hypothesis from $\mathcal{C}_H$ with the probability described in (2).

3.3. Pose Prediction Module

After updating pose in current frame, EKF predicts an initial estimate of the bin pose for the next frame. Two different update modules are used for pose prediction. If a matching hypothesis is accepted in the previous frame **k**, the pose estimate for frame **k+1** is given by -

$$\mathbf{p^{k+1}} = \begin{cases} \mathbf{p^k} + (\mathbf{p^k} - \mathbf{p^{k-1}}) + 0.5 * (\mathbf{p^k} - \mathbf{p^{k-1}} - (\mathbf{p^{k-1}} - \mathbf{p^{k-2}})), & \text{if } k \geq 2 \\ \mathbf{p^k} + (\mathbf{p^k} - \mathbf{p^{k-1}}), & \text{otherwise} \end{cases}$$

$(\mathbf{p^k} - \mathbf{p^{k-1}})$ has the same notion as the velocity component; whereas, $(\mathbf{p^k} - \mathbf{p^{k-1}}) - (\mathbf{p^{k-1}} - \mathbf{p^{k-2}})$ has a notion similar to acceleration. $\mathbf{p^{k+1}}$ serves as the initial pose $\mathbf{p_0}$ for frame **k+1**.

The corresponding covariance matrix for frame **k+1** is updated as follows -

$$\Sigma_{\mathbf{p^{k+1}}} = \begin{cases} (\frac{5}{2})^2\ \Sigma_{\mathbf{p^k}} + 2^2 \Sigma_{\mathbf{p^{k-1}}} + (\frac{1}{2})^2 \Sigma_{\mathbf{p^{k-2}}} + \Sigma_v, & \text{if } k \geq 2 \\ 4\Sigma_{\mathbf{p^k}} + \Sigma_{\mathbf{p^{k-1}}} + \Sigma_v \end{cases}$$

Note that $\Sigma_{\mathbf{p^{k+1}}}$ is the initial $\Sigma_{\mathbf{p_0}}$ for frame **k+1**, which we use to find the matching candidates surrounding the projected model features. $\Sigma_\mathbf{v}$ is the variance of the maximum velocity that can be encountered in each of the state variables.

If none of the hypothesis ($\mathcal{C}_H$) satisfies the the matching consensus test for frame **k**, we can not rely on the pose $\mathbf{p^k}$. In this scenario, the predicted pose is updated as -

$$\mathbf{p^{k+1}} = \alpha \mathbf{p^k} + (1 - \alpha)\mathbf{p^{k-1}}$$

$$\Sigma_{\mathbf{p^{k+1}}} = \alpha^2 \Sigma_{\mathbf{p^k}} + (1 - \alpha)^2 \Sigma_{\mathbf{p^{k-1}}} + \Sigma_v$$

The weight α is kept non-zero to ensure that if the bin is undergoing any motion, we would not lose track of it.

4. BIN CONTENT ANALYSIS

Our final objective is to analyze the contents of the bins dynamically (cf. Fig. 2). We need to find precise locations of the bins containing the items divested by the passengers and detect the frames where transfer events (like, passengers divesting items in the bins and taking items out of the bins) occur.

To detect bin locations, background subtraction using the information of an empty bin is applied on the bins in the current frame. We conclude whether a bin is empty or not based on the number of non-zero in the binary image after background subtraction.

For detection of transfer events, we keep track of the bin state transitions (from empty state to non-empty state or vice-versa). However, momentary state transitions may occur which may not represent an actual transfer of event. We conclude a transfer event has occurred only if the bin state remains constant for x frames after a detected state transition.

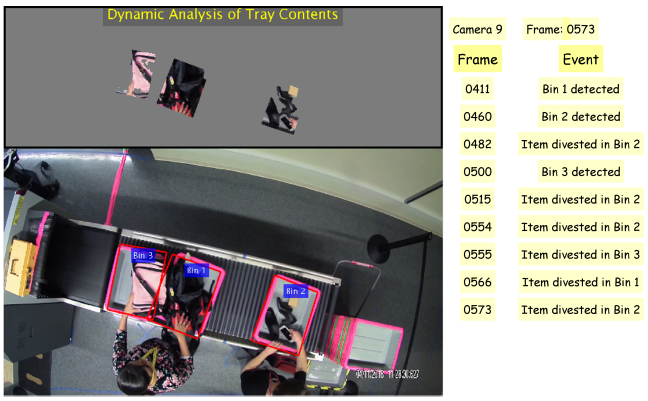

Fig. 2. News feed shows bin content analysis in Camera 1.

5. EXPERIMENTAL ANALYSIS

We evaluate our proposed bin tracking method on datasets emulating real airport security checkpoint scenarios. The datasets are collected from multiple cameras at a mock checkpoint located at Kostas Research Institute (KRI) for Homeland Security in Burlington, MA, USA. Each dataset consists of 3 synchronized videos from overhead cameras (say,1, 2, 3) focused on different regions of the roller-section area of an airport checkpoint. Camera 1 focuses on the conveyor belt section area when the bins enter the x-ray machine. Camera 2 captures the roller section area where the bins come out of the x-ray machine. Finally, Camera 3 is used to cover the exit area of the security checkpoint. Each camera captures videos at 30 fps with 1080p resolution. In this paper, we report results for two different datasets, namely, A and B, each having videos of duration $\approx$ 135 seconds.

We validate the performance of our algorithm on bin location events and transfer events with respect to precision and recall metric. The ground-truths for bin location events contain annotated bounding boxes outlining bin location in random chosen frames of the videos. A true bin location event is said to occur if the algorithm generated bounding box has greater than 50% IoU with the ground-truth bounding box at that frame; otherwise a false alarm is created. The ground-truths for transfer events consist of the frame numbers where the events occur. Frames within $\pm$ 1 seconds are searched for a transfer event with respect to the ground-truth.

Based on visual inspection, it is evident that Cameras 2 and 3 have significant perspective distortion throughout the field of view (FOV). Thus, we compare our collaborative tracker with respect to the 2D tracker used in Camera 1, and 2.5D tracker used in Cameras 2 and 3. Tables 1 and 2 show that both precision and recall values are better for the bin location as well as transfer events when we use the collaborative tracker. This results from the fact that though camera 1 does not suffer from severe perspective distortion throughout its FOV, but has significant distortion along the frame boundaries. This reduces the accuracy of bin detection. On the other hand, the 2.5D tracker is comparatively less stable, as evident from the low recall value in Camera 2 as it fails to track all the locations.

Bin Location Events: Both the recall and precision values for bin location detection are more than 90%. The recall value for detection of bins is improved over 10 % when the collaborative tracker is used. This shows the collaboration approach can detect bins accurately over the entire image space irrespective of perspective distortion by reducing the false negatives. The precision values are more than 80% even when we do not use the collaborative tracker. Thus, the precision value improvement is smaller compared to the recall value in the collaborative tracker.

Camera #	1		2		3	
Tracker	2D	Collaborative	2.5D	Collaborative	2.5D	Collaborative
Precision (Bin Location)	89.7 %	92.9%	88.3 %	95.6 %	88.9%	100.0%
Recall (Bin Location)	87.5 %	97.5 %	75.7 %	91.4 %	84.2 %	100.0%
Precision (Transfer Event)	83.3 %	100.0%	80.0 %	92.9 %	100.0%	100.0%
Recall (Transfer Event)	83.3 %	88.9 %	75.0 %	81.2 %	80.0 %	80.0%

Table 1. Results on Dataset A

Camera #	1		2		3	
Tracker	2D	Collaborative	2.5D	Collaborative	2.5D	Collaborative
Precision (Bin Location)	88.5 %	91.4%	85.1 %	91.4 %	100.0%	100.0%
Recall (Bin Location)	65.7 %	91.4 %	76.9 %	93.5 %	66.7%	83.3%
Precision (Transfer Event)	87.5 %	93.8%	92.3 %	93.3 %	100.0%	100.0%
Recall (Transfer Event)	87.5 %	93.8 %	75.0 %	87.5%	80.0 %	100.0%

Table 2. Results on Dataset B

Transfer Events: As a result of higher bin detection accuracy, the precision and recall values for detecting transfer events also improve in the collaborative approach. Our proposed collaborative tracker has precision value over 90% even in detecting transfer events. The significant improvement stems from the fact that the number of false positive drastically reduces as our collaborative tracker increases stability (in Camera 2) and increases the tracker's range of operation (in Camera 1). The recall value also improves by reducing the false negatives compared to the non-collaborative tracker.

6. CONCLUSION

We proposed a collaborative tracker which estimates the bin poses accurately and produces tighter bounding boxes around the bins as opposed to other deep learning based bounding box detection. Using a more accurate pose we can track the bins and the contents within it with a higher precision. The quantitative analysis of our method shows that the tracking performance is more robust when we use the collaborative approach. Our approach is also computationally inexpensive and requires minimal amount of training data, as compared to other machine learning and deep learning approaches. In the future, we plan to incorporate a 3D EKF tracker in our collaborative framework in order to handle the scenarios where the bins undergo unpredictable 3D motion caused by the humans.

7. ACKNOWLEDGEMENT

This material is based on the work supported by the U.S. Department of Homeland Security. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

8. REFERENCES

[1] Ziyan Wu and Richard J Radke, "Real-time airport security checkpoint surveillance using a camera network," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. IEEE, 2011, pp. 25–32.

[2] Ashraful Islam, Yuexi Zhang, Dong Yin, Octavia Camps, and Richard J Radke, "Correlating belongings with passengers in a simulated airport security checkpoint," in *Proceedings of the 12th International Conference on Distributed Smart Cameras*. ACM, 2018, p. 14.

[3] Youngrock Yoon, Akio Kosaka, and Avinash C Kak, "A new kalman-filter-based framework for fast and accurate visual tracking of rigid objects," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1238–1251, 2008.

[4] Pedram Azad, David Münch, Tamim Asfour, and Rüdiger Dillmann, "6-dof model-based tracking of arbitrarily shaped 3d objects," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5204–5209.

[5] Eric Marchand, Patrick Bouthemy, François Chaumette, and Valérie Moreau, "Robust real-time visual tracking using a 2d-3d model-based approach," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. IEEE, 1999, vol. 1, pp. 262–268.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[7] Martin Bäuml, *A Universal Labeling Tool: Sloth*, 2014, `https://cvhci.anthropomatik.kit.edu/~baeuml/projects/a-universal-labeling-tool-for-computer-vision-sloth/`.

[8] Matthew D Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[9] John Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, , no. 6, pp. 679–698, 1986.

[10] Dmitry Lagunovsky and Sergey Ablameyko, "Straight-line-based primitive extraction in grey-scale object recognition," *Pattern Recognition Letters*, vol. 20, no. 10, pp. 1005–1014, 1999.