Parcel Tracking by Detection in Large Camera Networks

Sascha Clausen, Claudius Zelenka, Tobias Schwede, and Reinhard Koch

Department of Computer Science, Kiel University, Kiel, Germany
{sac, cze, tosc, rk}@informatik.uni-kiel.de

Abstract. Inside parcel distribution hubs, several tenth of up 100 000 parcels processed each day get lost. Human operators have to tediously recover these parcels by searching through large amounts of video footage from the installed large-scale camera network. We want to assist these operators and work towards an automatic solution. The challenge lies both in the size of the hub with a high number of cameras and in the adverse conditions. We describe and evaluate an industry scale tracking framework based on state-of-the-art methods such as Mask R-CNN. Moreover, we adapt a siamese network inspired feature vector matching with a novel feature improver network, which increases tracking performance. Our calibration method exploits a calibration parcel and is suitable for both overlapping and non-overlapping camera views. It requires little manual effort and needs only a single drive-by of the calibration parcel for each conveyor belt. With these methods, most parcels can be tracked start-to-end.

Keywords: Multi-object tracking · Tracking by Detection · Instance Segmentation · Camera Network Calibration

1 Introduction

Parcel delivery is a vital part of today’s society. Parcel delivery companies process their parcels in parcel hubs, so that every parcel reaches the right recipient. Some parcels are directed to the wrong destination or simply get lost by falling off the conveyor belts. This can pertain several tenth per 100 000 parcels sent. To recover them, parcel hubs are equipped with large-scale camera networks to surveil the parcels. A camera network may consist of up to 200 cameras inside a single distribution hub. When a parcel gets lost, a human operator views videos starting from the entry scan to see where it got lost. This is a tedious, time consuming and tiring task. We propose a tracking framework to assist the operators in their work. It is important to note that our framework operates on an existing camera network and that no extra hardware such as RFID scanners are required.

Parcel hubs are highly heterogeneous: Different conveyor belt types and colors, different conveyor belt and camera topology, manual or (semi-)automatic
sorting, varying illumination and strongly varying viewpoints (Figure 1). Also
the parcels can be very diverse and occur in all materials, colors and dimensions.
It is possible to see car tires next to very small brown paper board parcels. However,
depending on operating conditions, there may also be a large number of
the same kind of parcels. Traditional approaches and trackers such as back-
ground subtraction have problems with changing illumination, occlusions and
especially with separating touching parcels, which occur in case of parcel jams.
These conditions in combination with a low frame rate of approx. 5 fps and
a high maximum object velocity of approx. 2 m s\(^{-1}\) define the setting for this
application oriented research in tracking algorithms.

1.1 Related Work

There are only few related works in the domain of camera based tracking of
parcels in multi-camera systems. [15] proposes a multi-camera system to track
parcels on a short conveyor belt with 7 m length, which is a much smaller problem
setting than a large-scale parcel hub. Their solution is based on applying
the KLT-tracker [31] on feature corner points, which is not practical for our
problem with much lower frame rates, larger intra-frame motion, variation, and
background variability. Other systems are optimized for robot vision on conveyor
belts [29] and focus more on how to grab an object with a robotic arm, than
fast tracking with low frame-rate. In [32] for a related task, tracking persons
across non overlapping camera, albeit with a higher frame rate, the Hungarian
algorithm [17] for label assignment and Kalman filter for prediction is used.
One aspect of tracking parcels is that the process of detecting parcels can be seen as separating them from background. However, background subtraction techniques [30, 26] are very fast, but cannot handle very dirty or irregular backgrounds and touching objects cannot be separated. Large progress has been made in recent years in deep learning based general object detection algorithms, examples are Faster R-CNN [28] and SSD[21]. Even more useful for our task are algorithms, which also provide instance segmentation, i.e. FCIS [19] and Mask R-CNN [9] and thus allow more accurate localization.

The second aspect is the tracking itself, but we must consider that detection quality is key for tracking performance, as shown in [2], where an increase of 18.9% in performance is made by changing the detector. In our task, we have a lower frame rate compared to benchmarks (e.g. [24]), which means that objects can change their appearance significantly between frames and their motion is not constant, but depends on the interaction of the parcels. Hence, some approaches that provide very high benchmark results, such as Tubelets [14], are clearly not suitable for our task. Also, we are strongly limited in the amounts of training data for the tracking task, especially for longer sequences. This problem of insufficient training data is even more severe for recurrent neural networks, which have recently shown good results [25].

Also highly successful in benchmarks, but easier to train and not as dependent on high frame rates is the approach of [18], where siamese networks are used to solve the problem of re-identifying the tracked objects and [5], where the feature vectors of the CNN backbone of the detector are used as identifying features.

1.2 Contribution

The difficult conditions are what separates our application oriented research from many of the related works optimized for benchmark data introduced above. The contribution of this article lies in the industrial large scale application and adaptation of these previously benchmark optimized algorithms.

Inspired by this recent success of deep architectures on benchmark data we propose a two-stage Tracking-by-Detection approach via [9] by detecting the parcels with a CNN and assigning the detections between frames similar to [5, 18, 32], with an additional novel feature improver network. We show that even though benchmark optimized, their approach and ideas are transferable and adaptable to tracking parcels across multiple camera in parcel distribution hubs.

2 Detection

The parcels are detected via state-of-the-art instance segmentation with Mask R-CNN [9], because the precise contour of the parcels gives additional informations to tell them apart. We use the implementation of Matterport [23] with pre-trained weights on the COCO data set [20]. The model is fine-tuned on 3306 hand-labeled images from 37 different cameras containing 14 284 parcels.
The training data is augmented by horizontal flipping, darkening the images by adjusting the gamma value and using random crops of the image.

We train the network with stochastic gradient descent and a learning rate of 0.002. The training is split in two stages: First, the RPN, the classifier and the mask heads are trained for 70,000 iterations. Then the stages 4 and 5 of the ResNet backbone are additionally trained for another 200,000 iterations. We use a weight decay of 0.0001, a momentum of 0.9 and a batch size of 2. The learning rate is smaller by a factor of 10 compared to [9] because larger learning rates lead to worse results and exploding gradients [23].

### 2.1 Feature Improver Network

Mask R-CNN uses the output of the ROIAlign layer for the classification and segmentation of each region proposal. This output can be seen as a feature vector for each region proposal. We define the similarity of two parcel feature vectors as the normalized scalar product between them to get a similarity value between 0 and 1. We observed that feature vectors of visually similar parcel images (e.g. images of the same parcel in adjacent frames) have a higher similarity than parcel images of different parcels. The similarity can thus be used to assign the detections in adjacent frames to the correct parcels (for details see Section 4).

Inspired by siamese networks [4, 5, 18], we propose a feature improver network as an additional head for Mask R-CNN (Figure 2). This head is supposed to make the feature vectors more distinguishable via the scalar product.

The feature improver network is a simple fully connected network with one layer consisting of 1024 neurons. As a proof of concept, we first train Mask R-CNN and extract the outputs of the ROIAlign layer afterwards. The feature improver network is then trained with these unimproved feature vectors in a siamese network fashion, i.e. we normalize the unimproved feature vectors and calculate the dot product of the two siamese network branches. As a result, the output lies in the interval [0, 1] and we can use binary cross entropy as loss function. The training target is to maximize the scalar product of feature vectors from the same parcel in adjacent images and to minimize the scalar product of feature vectors from different parcels. We train the feature improver network on the same data set as Mask R-CNN with the Adadelta [34] optimizer for 1000 iterations using a batch size of 200 and a learning rate of 0.01.
Fig. 3: (a) shows one recording of the calibration parcel on the conveyor belt. The manually annotated mask for the same camera view as in (a) can be seen in (b). In (c) the reconstructed surface of the conveyor belt is displayed. The different colored pieces are the planar elements, the surface is constructed of. The yellow circles are the projected positions of the calibration parcel and describe its trajectory during the calibration.

3 Calibration

For the tracking across different cameras, we require the exact topology of the conveyor belts, the cameras and their viewpoints. Therefore, before discussing the tracking component of our system, we will briefly introduce our calibration method.

Our system is designed to be in large parts automatic without requiring additional external knowledge of the network. Our approach consists of a calibration parcel, which is filmed as it moves across every conveyor belt, as in Figure 3a. The calibration parcel has an ArUco [7] marker on each side to provide at least one calibration target from each viewing direction.

The marker can be easily detected and used as a reference to intrinsically calibrate each camera. A further result of the calibration are relative 3D positions of the calibration parcel to each camera. We can approximate the conveyor belt’s surface with piecewise planar segments, see Figure 3c. Furthermore, we capture the calibration images with their timestamps. The 3D positions combined with their respective timestamps results in the calibration parcel path. With the help of this path we can predict the position of other parcels.

Knowledge about the conveyor belt region is needed to identify and ignore parcels besides it. The external contour of the conveyor belt is manually annotated, resulting in a mask as in Figure 3b. This cannot be done automatically, because the calibration parcel path is not sufficient to decide which of the visible conveyor belts is active. Calibration is performed only once before the active phase of the system.
Listing 1: Pseudocode for the intra-camera tracking

```plaintext
Input: parcels - the parcels from the last frame
      contours - the detected contours from the current frame

costMatrix = array[size(parcels)][size(contours)]
unassignedContours = set()

for p ∈ parcels {
    prediction = predictionForCurrentFrame(p)
    vp = getFeatureVector(p)
    for c ∈ contours {
        unassignedContours.add(c)
        vc = getFeatureVector(c)
        distance = computeDistance(prediction, c)
        // The value 0.0027 was chosen empirically
        s = max(0, 1 - (distance * 0.0027))
        similarity = scalarProduct(vp, vc)
        costs = 1 - (s * similarity)
        costMatrix[p][c] = costs
    }
}

// Matching is a p × 1 vector that contains the assigned
// contour for every parcel.
matching = hungarianMatching(costMatrix)

for p ∈ parcels {
    // -1 defines that no matching is possible
    if (matching[p] != -1) {
        addContourToParcel(matching[p], p)
        unassignedContours.remove(matching[p])
    }
}

for c ∈ unassignedContours {
    createNewParcel(c)
}
```

4 Tracking

An investigation process is started when a parcel gets lost. This process tracks the lost parcel from the last known position, which is marked by a human operator. The tracking consists of the tracking inside camera views (intra-camera tracking) and the tracking between overlapping and non-overlapping camera views (inter-camera tracking). The intersection points of different camera views are determined by the calibration.

In the following, we explain the steps of our intra-camera tracking as specified with pseudo code in Listing 1. The intra-camera assignment can be seen as a weighted bipartite matching problem where every parcel in frame $f$ has a corresponding one in frame $f+1$. Therefore, we compute a cost matrix for all possible matchings of the detections in both frames. The matching cost of two detections is defined as the scalar product of their feature vectors. The feature vectors are computed by the neural network proposed in Section 2.1. To improve the matching, we predict the positions of the detections in frame $f$ for the frame $f+1$ with the optical flow algorithm proposed in [16]. The similarity is then scaled with the distance between the predicted and detected positions. From a
cost matrix, the matching with the lowest costs is computed using the Hungarian algorithm [17]. Besides matching the contours of the current and the next frame as outlined in pseudo code, we implement an occlusion handling technique, which also handles spurious missed detection. We include the unassigned contours in the current frame, shifted by the optical flow, as additional matching candidates in the next frame.

A parcel can only be tracked in this way a limited amount of times, without getting assigned to a detected contour. After that it is marked as lost and not further handled in this camera. This allows us to continue tracking a parcel even if it has not been tracked in some frames in between for various reasons.

In contrast, the inter-camera assignment can not be done by the visual appearance of the parcels because neighboring cameras can have very different poses, as can be seen in Figure 4. The reasons for the differences between two view include destination labels, which are only on one side of the parcel, and asymmetric applied tape. Therefore, the inter-camera assignment is performed by projecting the parcels from one camera to the next by using their extrinsic calibration. Although the contour of a parcel is known from the detection, we do not have the 3D shape or height. Therefore, we use the conveyor belt surface as an intermediate step for the projection.
Fig. 5: Illustration of the different steps for the inter-camera tracking. (a) shows the approximation of a triangle (red) for the visible bottom part of a contour (yellow). In (b) the estimated parallelogram for the parcel base (blue) based on the previous approximated triangle can be seen. The schematic projection onto the conveyor plane surface and the time adjustment can be viewed in (c). In this drawing the time difference between both cameras images is defined by $\Delta t$. The calibration parcel movement in the time $\Delta t$ is depicted in orange. Furthermore, the positions of the quadrangle before and after the time adjustment are shown. (d) shows the projection from the conveyor plane surface to the image plane of the second camera.
Fig. 6: These images show the tracking of multiple parcels in one camera view. The color of the contours depends on the parcel they are assigned to, whereas every parcel has its own unique color. The images (a) to (d) show subsequent moments of the tracking.

The different steps of the inter-camera tracking are shown in Figure 5. In the first step, we regard the upward surface normal direction and select those parcel contour points, which are located in the opposite direction. This results in the bottom part of the parcel contour, which is in-plane with the conveyor belt. Then we approximate it with a triangle, shown in Figure 5a. The second step is to estimate the parcel base by extending the triangle to a parallelogram, as shown in Figure 5b. In the third step, we project the parallelogram onto the conveyor belt surface. The projected position is adjusted using the time difference between both cameras and the calibration parcels velocity and direction, for this point. Lastly, the quadrangle is projected from the conveyor belt surface into the image plane of the second camera, as shown in Figure 5d. All relevant contours in a frame are projected in this way into the next camera. Then these projected contours have to be matched with the detected ones. This is done by defining the costs $c$ as $c = 1 - \text{IoU}$ and solving the problem with the Hungarian algorithm. Examples of the tracking can be seen in Figure 6 and Figure 7.
Fig. 7: These images show the tracking of one parcel over multiple cameras. The orange contour highlights the tracked parcel. The images (a) to (d) show the tracked parcel from different camera views over time, as it progresses along the conveyor belt.

5 Evaluation

5.1 Detection

If not stated otherwise, the training is conducted with a training set of 3306 images with a ResNet-101 backbone. For evaluation purposes a sequence of 857 images is used. The threshold confidence for the detections is set to 0.8.

Usually, the performance of object detectors is quantified with the average precision (AP) [28, 9, 21]. The AP emphasizes accurate bounding boxes while neglecting classification accuracy [27] by ordering all detections above a given ground truth IoU threshold by their detection confidence. If all ground truth objects of an image are detected with a higher confidence than the false positive detections, the AP gets misleading: It reaches 1 and is not affected by any low-confidence false-positive detections. Because false positive detections could confuse our tracking algorithm, we propose a different metric to compare our results. Instead of calculating the scores for each image independently and averaging the values over the whole validation set, we count the true positives, the false positives and the ground truth annotations for all validation image and calculate one global precision, recall and F1 score. These global scores allow us to select the best training result for the requirements of our tracking algorithm.
Different training and network configurations have been tested (Table 1):

- **Backbone network**: A deeper architecture improves the accuracy, but results in a slightly longer inference time of \(\sim170\text{ ms}\) instead of \(\sim150\text{ ms}\) per image on a Nvidia Titan Xp.
- **Data augmentation by darkening the training images**: Because some parcel hubs have darker areas, we darken some training images to simulate bad lighting conditions. This augmentation does not change the F1 score significantly, but increases the precision at the cost of lower recall.
- **Data augmentation by using random crops of the training images**: Random crops of the training images are used to simulate only partly visible parcels at the image borders.

### 5.2 Tracking

Because tracking is conducted in two phases (inter-camera tracking and intra-camera tracking), we also have two different tests to evaluate the tracking performance. The first test aims at evaluating the intra-camera tracking performance, while the second test evaluates the combined performance of inter- and intra-camera tracking.

The test set for the first test contains data of three different cameras. The first two cameras have a frame rate of 8 and the third of 5 fps. The data set contains 236, 233 and 161 images for the cameras respectively. For each tracking test we compare different configurations of the intra-camera tracking. We test the tracking as described in Section 4 with the improved feature vector, unimproved feature vector and IoU of the prediction for the similarity computation of parcels. Furthermore, we test the tracking with the improved and unimproved feature vector without the optical flow prediction and the distance based scaling. All configurations are compared by computing the MOTA-score [1]. For the computation of this score, detected contours are matched with ground truth contours, if their IoU ratio is not less than 0.5.

In the second test, the parcels are tracked across four cameras up to a specific point. We evaluate the tracking performance end-to-end. If for a track the initial parcel is the same as the tracked object at the specified point, the track is labeled successful. This also applies if the tracked parcel was mismatched in the tracking process, but was recovered to the correct parcel. The data set for the test contains the start positions of 38 parcels. The cameras have an average frame rate of 5 fps.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference ResNet-101</td>
<td>0.92</td>
<td>0.812</td>
<td>0.863</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>0.927</td>
<td>0.78</td>
<td>0.847</td>
</tr>
<tr>
<td>Without darkening</td>
<td>0.952</td>
<td>0.794</td>
<td>0.866</td>
</tr>
</tbody>
</table>
Table 2: Results of the intra-camera tracking and combined tracking test. The segmentation model for intra-camera tracking achieves 111 false positives and 455 false negatives for the 2742 annotated parcels. For the combined test, single parcels are tracked across multiple cameras up to a specific point.

<table>
<thead>
<tr>
<th></th>
<th>Intra-camera</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mismatches</td>
<td>MOTA-score</td>
</tr>
<tr>
<td>Without optical flow prediction:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature vector</td>
<td>78</td>
<td>0.7651</td>
</tr>
<tr>
<td>Improved feature vector</td>
<td>54</td>
<td>0.7739</td>
</tr>
<tr>
<td>With optical flow prediction:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IoU</td>
<td>59</td>
<td>0.7721</td>
</tr>
<tr>
<td>Feature vector</td>
<td>30</td>
<td>0.7826</td>
</tr>
<tr>
<td>Improved feature vector</td>
<td>22</td>
<td>0.7856</td>
</tr>
</tbody>
</table>

As seen in Table 2, using the improved feature vectors without optical flow decreases the mismatches from 78 to 54. Using optical flow prediction further decreases to 30 and 22 respectively. Causes for remaining errors are sudden changes in the velocity or direction or occlusion by body parts or held parcels and other objects. If the matching is done with the IoU ratio, 59 mismatches are encountered, caused by relatively large parcel movements between following frames induced by the low rate of the cameras. Furthermore, the usage of feature vectors allows rediscovering similar parcels, even if the prediction does not overlap with the contour in the next frame.

The results of the second test in Table 2 are similar. The optical flow prediction results in tracking more than twice as many parcels correctly. The usage of the improved feature vector raises performance slightly compared to the unimproved one. Surprisingly, there are cases in which the tracking is successful with the unimproved feature vector, but fails with the improved one. In these cases both trackers switch targets to a wrong parcel. However, the tracker with the unimproved feature vector switches back one or two frames later. We reason that this is caused by the high similarity of the unimproved feature vectors to each other. This allows mismatches between similar parcels to occur more often and also makes it easier to switch back to the correct parcel after mismatching.

Another interesting aspect is the comparison between the IoU tracking and the improved feature vector tracking without the optical flow prediction. Although the numbers of mismatches are similar, the IoU tracking tracks twice as many parcels correctly. We conclude that the differences in performance are caused by the sensitivity of the feature vector matching to lighting changes and similar parcels.

Tracking Baseline Comparison For baseline comparison we conduct the tracking evaluation on the test data with the OpenCV implementations of the following tracking algorithms: Boosting [8], KCF [11, 6], MedianFlow [12], TLD
Fig. 8: These images show a comparison between our tracking algorithm and the tracking algorithms of OpenCV. The change of direction and the shadow lead to failure of the other tracking algorithms.

[13], GOTURN [10], MOSSE [3] and CSRT [22]. The tracker’s parameters were optimized for the best results on the test set.

A challenging part of the test set can be seen in Figure 8. All tracking algorithms get initialized with the same bounding box (Figure 8a). Most trackers work well as long as the target parcel does not change direction. When the parcel changes direction and reaches the shadow, only CSRT manages to stay on target (Figure 8b), but loses the parcel five frames later (Figure 8c). A similar tracking behavior can be seen for all other test parcels, which results in no correctly tracked parcel in the test set.

6 Conclusion

We presented a multi-object, multi-camera tracking system for parcels, designed for large parcel hubs. The system is designed to work in large scale industrial environments. It profits from the automatic calibration process, which reduces the calibration effort significantly. With this system and our novel feature vector improver network, we are currently able to track about 81% of the parcels correctly. While in most cases the tracking works smoothly, especially cases of
human interaction with parcels cause the tracking to fail. In these cases human operators are still needed. Furthermore, a human operator is additionally needed to initialize the tracking process by selecting the parcel in the image.

In the future we plan to automate the tracking initialization by using timestamps and compulsory barcode scans at the entrance of the parcel hub. A possibility to improve the runtime performance is to use SSD [21] or YOLO [27] for the detection and only compute segmentations when they are needed. Another direction could be the exploration of more dedicated architectures or specialized loss functions like triplet loss [33] for the feature improver head, to minimize the number of mismatches. Considering the current need to capture all possible paths to neighboring cameras, the calibration effort can be reduced by changing the routine for the inter-camera tracking, so that it does not rely on the path of the calibration parcel. Another possible avenue of improvement is of course increasing our set of training data for both tracking and detection.

Our evaluation data set was assembled to provide a realistic impression of overall tracking performance. We are planning to assemble different evaluation data sets with only common or only particular difficult situations, to be able to analyze tracking behavior in more detail.

Even if our system is not able to track all parcels fully automatic yet, we have done a major step in this direction. We are confident that this research helps reducing the occasions human operators are needed to manually track parcels in video data.

Acknowledgments This work was supported by the Central Innovation Programme for SMEs of the Federal Ministry for Economic Affairs and Energy of Germany under grant agreement number 16KN044302.

References


