Mask R-CNN based Pedestrian Tracking

Team 18: Sahib Dhanjal (*sdhanjal*), Sumit Joshi (*jsumit*), Siddharth Ratapani Navin (*sidnav*)

***Abstract*—We present a conceptually simple design and implementation of a real-time pedestrian detection and tracking algorithm that implements computer vision based motion estimation through scene flow and pedestrian detection and instance segmentation through a Regional Convolutional Neural Network.**

I. Introduction

Pedestrian tracking systems are commonly used for visual surveillance, but are also a key component in systems with other objectives such as activity recognition and behavioral understanding. Since the advent of self-driving vehicles, a lot of focus has been laid on pedestrian detection and tracking in real-time. A variety of methods have been developed for tracking single or multiple pedestrians in static or moving cameras by exploiting different types of image information. We obtain information required for tracking an object in an image using (i) Mask R-CNN [1], (ii) Scene Flow, and (iii) Particle Filter [2].

II. Related Work

One common problem in tracking algorithms is that it is difficult to achieve both high accuracy and efficiency. More specifically, most tracking algorithms are based on the hypothesis that the object being tracked is a rigid body. Thus, some visual features are disabled when the hypothesis is invalid. Meanwhile, the existing trackers tend to fail when the target object changes poses, because such changes lead to the dramatic mismatch between the momentaneous appearance model and the learned one. Bo Zhang et.al discuss an ensemble color feature model [3]. It takes pedestrians' physiological structure into consideration, and takes advantage of color histogram information under several color spaces, including the RGB, normRGB, HSV, and Lab. Xu Fen and Gao Ming [4] present a particle filter algorithm[cite] for pedestrian tracking by extracting the target color-histogram features and calculating the similarity between particle candidates and target template region through discrete Bhattacharyya Coefficient. Particle Filters are advantageous on solving nonlinear problems with non-gaussian system noise. G.C.H.E. de Croon et.al in their work [5] discuss an optic flow vectors based vision algorithm to develop a robust landing capability for Micro Air Vehicles. In our work we use Mask-RCNN framework [1] with dense optical flow [6] and particle filtering techniques [2] for modeling pedestrian movement across frames.

III. Methodology

A. Video Feed

We use two methods for testing purposes: (i) videos downloaded from the Internet containing single/multiple pedestrians, and (ii) live video feed from a mobile camera for tracking motion of pedestrians in real-time.

The input is down-sampled from a frequency of 60 fps to 6 fps.

B. Mask RCNN based image segmentation

Convolutional Neural Networks (CNNs) have become the gold standard for image classification. In fact, CNNs have improved to the point where they are able to outperform humans on the ImageNet challenge. Regional Convolutional Neural networks (R-CNNs) take this a step further and solve the problem of detection and localization of the object in the image through bounding boxes and respective classification of objects in the image.

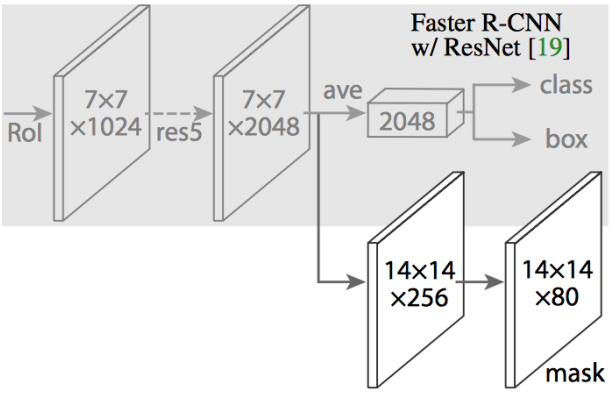

Fig. 1: In Mask R-CNN, a FCN is added on top of the CNN features of Faster R-CNN to generate a segmentation output (*source: https://arxiv.org/abs/1703.06870*)

Mask R-CNN is the newest member in the family of RCNN's which takes Faster R-CNN a step further by implementing a pixel-wise segmentation of the objects in the image along with classification and bounding boxes. Mask R-CNN does this by adding a branch to Faster R-CNN that outputs a binary mask that says whether or not a given pixel is part of an object (*Figure 1*). Hence, Mask RCNN provides us with instance segmentations for each object along with its classifications in a given frame. It adopts the same two-stage procedure, with an identical

first stage, which is RPN (Region Proposal Network (RPN), proposes candidate object bounding boxes).

We modified the naive Mask R-CNN network such that it classifies based on two classes -person or background. The frames from the video feed serve as the input to the network. For each frame that is an input to the network, we obtain the following outputs-

- Masks- This a 3 dimensional array, each 2 dimensional instance of this array corresponds to the masked image of the pedestrian it is associated with.
- ROI's- Regions of Interest or Bounding boxes are a minimum area rectangle that are built around each pedestrian in the frame.

C. Dense Optical Flow Tracking

Scene flow (*or Optical Flow*) is the pattern of apparent motion of objects, edges or other features in a visual scene caused by the relative motion between an observer and a scene. The basic starting point for optical flow is the following assumption -

$$I(\hat{x}, t) = I(\hat{x} + \hat{u}, t + 1) \tag{1}$$

where $I(x, t)$ is image intensity as a function of space $x = (x, y)^T$ and time t, and $u = (u_1, u_2)^T$ is the 2D velocity. To derive an estimator for 2D velocity u , we first consider the Taylor series approximation of the equation as shown below

$$I(\hat{x} + \hat{u}, t + 1) = I(\hat{x}, t) + \hat{u}\nabla I(\hat{x}, t) + I_t(\hat{x}, t) \tag{2}$$

where $\nabla I \ \ (I_x, I_y)$ and I, t denote spatial and temporal partial derivatives of the image I, and $u = (u_1, u_2)^T$ denotes the 2D velocity. The above equation can be reduced to this form-

$$\nabla I(\hat{x}, t)\hat{u} + I_t(\hat{x}, t) = 0. \tag{3}$$

The above equation is the basis for the optical flow velocity estimation between frames.

In our implementation, a dense optical flow computation is done based on Gunner Farnebacks algorithm. The first step is to approximate each neighborhood of both frames by quadratic polynomials. Afterwards, considering these quadratic polynomials, a new signal is constructed by a global displacement. Finally, this global displacement is calculated by equating the coefficients in the quadratic polynomials' yields.

We set the first captured frame at startup as the previous frame to kick start the algorithm. The flow object returned by the algorithm contains the estimated displacement along x and y directions for each pixels between two frames. We then calculate the mean displacements for each ROI, which gives us the the sensed motion for the 'pedestrian' with it's covariance. The output from this layer is fed into the particle filter action model for estimation (*Figure 2*). In our implementation, we select the region that is detected as a person with maximum probability and use nearest neighbor data association between consecutive frames for tracking its position.

Fig. 2: The frame of the image sequence taken and the result of dense optical flow (at 60 fps) based on Gunner Farneback's algorithm

D. Particle filtering on image sequences

The particle filter is a nonparametric version of the **Bayes filter**. Particle filters approximate the **posterior** by a finite number of parameters. However, they differ in the way these parameters are generated, and in which they populate the state space. The key idea of the particle filter is to represent the posterior $bel(x_t)$ by a set of

random state samples drawn from this posterior. Instead of representing the distribution by a parametric form, particle filters represent a distribution by a set of samples drawn from this non parametric distribution. Such a representation is approximate, but it is nonparametric, and therefore can represent a much broader space of distributions than other Parametric forms.

In the particle filter, each particle p consists of three estimates $p.x$,$p.y$, and $p.weight$. $p.weight$ is the weight associated to each particle in the distribution. This is later normalized prior to the importance re-sampling which is explained later. The two steps in Bayesian Estimation is given below:

1) Action Model: The optical flow based velocity estimation, is used to predict the motion between frames in the feed. We use each frame and its earlier frame to predict the pixel motion between images. The dense flow algorithm that we have implemented gives us the pixel motion between frames, We compute an average displacement between frames in the x and y directions and add them to each particle along with a Gaussian noise as follows:-

$$\begin{pmatrix} p.x \\ p.y \end{pmatrix} = \begin{pmatrix} p.x \\ p.y \end{pmatrix} + \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} + \mathcal{N}(0, \sigma_{motion}). \quad (4)$$

The particle here p has attributes $p.x, p.y.p.weight$ which correspond to the x, y position in image space and the weight of the particle. δ_x and δ_y are the displacements of each particle in image space between frames. This is calculated through the optical flow algorithm discussed previously. The noise between frames associated with the movement of the particles is modeled using a zero mean white Gaussian noise with a standard deviation equal to the standard deviation of the optical flow in the regions of interest in the image determined by the masks generated by Mask R-CNN.

2) Sensor Model: After the Action model mentioned above is implemented on each particle, only the pose of each particle is updated while the weight of all the particles is still the same. The sensor model is used to determine how likely the pose of a particle is and assign a weight to it indicating the probability of its pose closely approximating the ground truth. After the particles are propagated through the action model, they increase in uncertainty due to the inherent noise in the odometry value between frames. To reduce the uncertainty of the posterior distribution, we weight each particle by a number corresponding to how closely it associates with the sensor reading.

$$weight \propto p(z/x, y) \quad (5)$$

The center of the bounding box output given by the Mask R-CNN is used as the sensor reading "z"for our particles. The pose of each particle is equal to the predicted sensor reading for the particle

$$\hat{z} = (p.x, p.y) \quad (6)$$

We calculate the innovation[cite thrun] in the sensor reading for each particle which given by the following equation

$$innovation = z - \hat{z} \quad (7)$$

The innovation is the difference between the predicted sensor reading and actual sensor reading for each particle, as mentioned before a particle is weighted higher if its predicted sensor reading is closer to the actual sensor reading i.e

$$weight \propto 1/innovation \quad (8)$$

The weights are assigned to each particle as follows:-

$$weight = \hat{p}(innovation, Q) \quad (9)$$

$\hat{p}$ is a zero mean normal distribution with a covariance of Q which is modeled by the uncertainty in the bounding box prediction by Mask R-CNN and is proportional to the size of the bounding box[cite Semantic slam paper]. The weight is equal to the probability of this innovation in this distribution.

After propagating the particle through the motion and sensor model we re sample the particles[cite algorithm] for the next iteration using an importance re sampling algorithm, which samples particles with a probability proportional to its weight. This re-sampled distribution is used as the prior hypothesis for the next hypothesis. The hypothesis for the pedestrian position after each iteration is the normalized weighted mean of the position of all the particles.

E. Data Association

To ensure that the same pedestrian instance is tracked across all frames, we use a nearest neighbor data association algorithm in each new frame to predict which instance is the same as the instance that was tracked in the previous frame. The algorithm returns the sensor reading from the Mask R-CNN output that has the least Mahalanobis[cite ie] distance to the current coordinates of our particle hypothesis.

$$\textit{Mahalanobis distance} = \sqrt{(z - \hat{z}) \times Q \times (z - \hat{z}^T)}$$

IV. Results

A. Pedestrian Detection

The Mask R-CNN implementation was capable of detecting almost all pedestrian in a given frame and even did well in occluded cases. The classifier in our network used two classes i.e pedestrian and background.

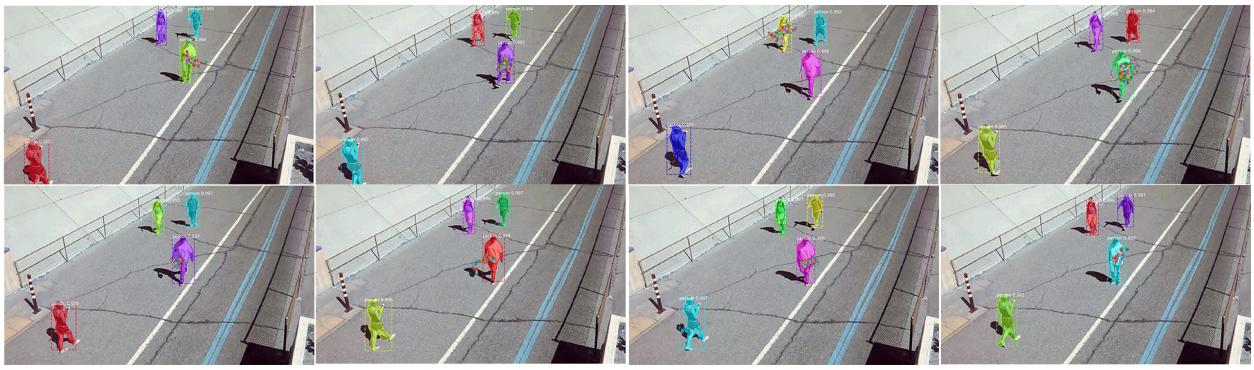

Fig. 3: Motion of the Pedestrian captured by optical flow across 8 frames (*top left to bottom right*)

We assessed the quality of the detections through running Mask R-CNN on images with sparse and dense environments . We observed that the classifier worked well in mildly cluttered environments, while in cluttered environments with a lot of people in it, the detector missed a few instances as shown below.

Fig. 4: Performance of Mask R-CNN in progressively denser populations (*from top to bottom*)

B. Optical Flow

The optical flow calculated between frames gave us noisy estimates when the frame rate was higher than 20 fps, the magnitude of the optical flow at high frame rates was lower and was indistinguishable from the noise in the optical flow estimate. We decreased the frame rate of the images to 6 fps. With the increase in the apparent motion between frames we observed that the optical flow worked better for objects moving faster in the frames compared to slower objects between frames. This was a qualitative observation which resulted in better tracking of the pedestrians.

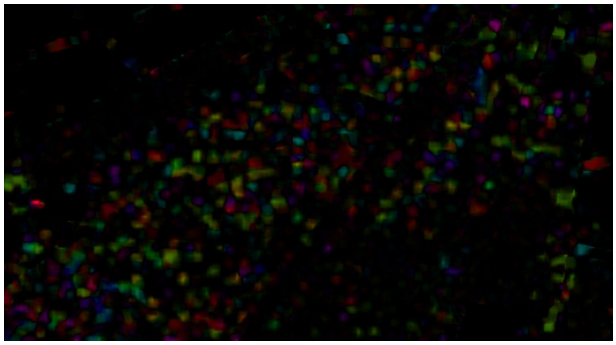

Fig. 5: Noisy Optical Flow performance at high frame rates

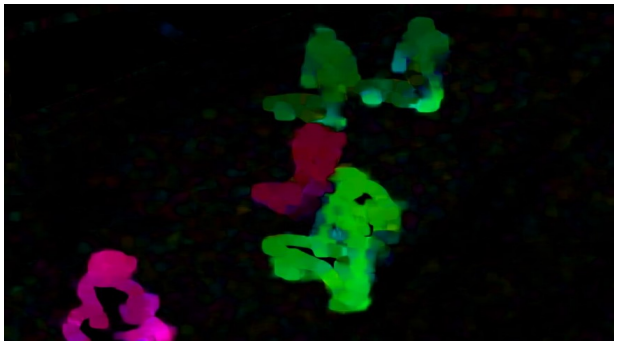

Fig. 6: Optical Flow performance at 6fps

C. Pedestrian Tracking

We use the particle filter algorithm described previously to track the pedestrians in the image. We use the particle filter for a single hypothesis tracking about multiple frames. The results of which are shown in *Figure 7*. We observe that the particle filter works well when the pedestrian instances in the image are sparsely located. Frames with pedestrian close to each other causes divergence in our particle filter estimates. We have observed that this occurs when we have wrong data associations in the instance we are tracking. this is shown in Figure 8. We observed that the occurrence of wrong data association decreased with more particles.

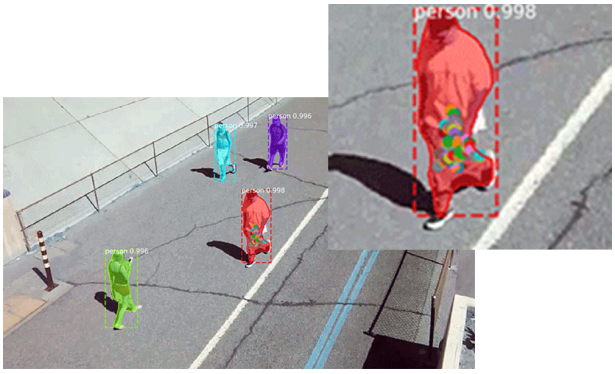

Fig. 7: Particle filter tracking the motion of the person

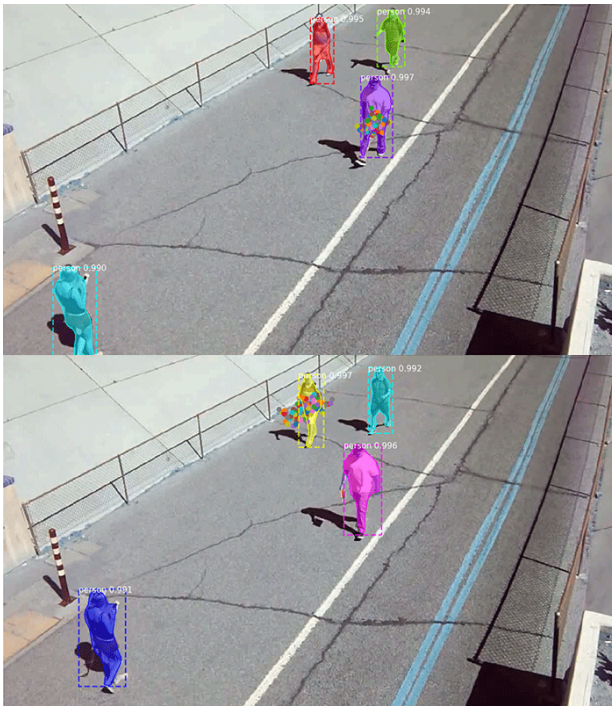

Fig. 8: Wrong data association - Tracking Different Instances

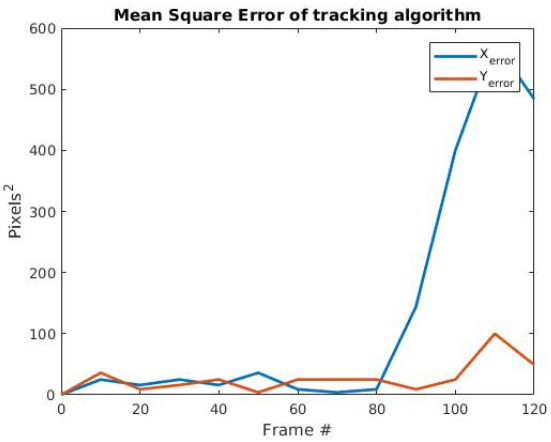

Fig. 9: Mean Square Error (MSE) in tracking for every 10 frames

As shown in Figure 9 we observed that the mean square error (MSE) in estimation was low for the first 80 frames. At that instance, we observed a wrong data association which caused the MSE to increase rapidly.

V. Discussion

The Mask RCNN detection algorithm requires a powerful GPU to achieve real time results in the detection pipeline while using live video feed. As we were not able to obtain the depth information of the image, we restricted our tracking algorithm to the image frame of reference. With the depth information we can use it along with the camera matrix and camera pose to estimate the position of the pedestrian in world space. The dense estimation optical flow algorithm exhibits noise which is not closely approximated by the Gaussian noise that we implemented in the action model. This might have been the reason why the particles in our estimation would exhibit unpredicted behavior and divergence at later frames even through the bounding box predictions remained fairly accurate. We believe that we can make the optical flow estimate perform better through filtering the output using a low pass filter, although we have not tried experimenting with different filters to substantiate this claim. In terms of data association, the nearest neighbor data association is not stable in even mildly cluttered environments, this might be due to the high covariances in our estimate and euclidean proximity in cluttered environments that causes the nearest neighbor association to pick the wrong pedestrian to track. We could improve on this using more stable data association techniques such as Joint Compatibility Branch and Bound [7]. For multi-hypothesis tracking, advanced probabilistic filters such as the Probability Hypothesis Density Filter can be used [8], [9].

REFERENCES

[1] K. H. et al, "Mask r-cnn," https://arxiv.org/pdf/1703.06870.pdf.

[2] S. Thrun, in *Probabilistic Robotics*, http://www.probabilistic-robotics.org/.

[3] B. Zhang, Y. Xu, and X. Yang, "Online pedestrian tracking using ensemble color feature," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Oct 2015, pp. 276–282.

[4] X. Fen and G. Ming, "Pedestrian tracking using particle filter algorithm," in *2010 International Conference on Electrical and Control Engineering*, June 2010, pp. 1478–1481.

[5] G. de Croon, H. Ho, C. D. Wagter, E. van Kampen, B. Remes, and Q. Chu, "Optic-flow based slope estimation for autonomous landing," http://www.bene-guido.eu/guido/SlopeEstimationIJMAVDraft.pdf.

[6] G. Farneback, "Two-frame motion estimation based on polynomial expansion," http://www.diva-portal.org/smash/get/diva2:273847/FULLTEXT01.pdf.

[7] J. Neira and J. D. Tardos, "Data association in stochastic mapping using the joint compatibility test," http://webdiis.unizar.es/~jdtardos/papers/Neira_TRA_2001.pdf.

[8] K. Panta, B.-N. Vo, S. Singh, and A. Doucet, "Probability hypothesis density filter versus multiple hypothesis tracking," http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.7757&rep=rep1&type=pdf.

[9] B.-N. Vo and W.-K. Ma, "The gaussian mixture probability hypothesis density filter," http://www.ee.nthu.edu.tw/~wkma/publications/gmphd_2col.pdf.