Understanding Crowd Collectivity: A Meta-Tracking Approach

Afshin Dehghan M. Kalayeh
Center for Research in Computer Vision, University of Central Florida
adehghan@cs.ucf.edu, mahdi@eecs.ucf.edu

Abstract

Understanding pedestrian dynamics in crowded scenes is an important problem. Given highly fragmented trajectories as input, we present a novel, fully unsupervised approach to automatically infer the semantic regions in a scene. Once the semantic regions are learned, given a tracklet of a person, our model predicts the pedestrian’s starting point and destination. The method is comprised of three steps. First, the spatial domain of the scene is quantized into hexagons and a 2D orientation distribution function (ODF) is learned for each hexagon. A Time Homogenous Markov Chain Meta-tracking method is used to automatically find the sources and sinks and later find the dominant paths in the scene. In the last step, using a 3-term based trajectory clustering method, we predict the source and sink for each pedestrian. Furthermore, we introduce a 2-step trajectory reconstruction method to infer the future behavior of each individual in the scene. Qualitative and quantitative experiments on a video surveillance dataset from New York Grand Central Station demonstrate the effectiveness of our method both in finding the semantic regions and grouping of fragmented tracklets.

1. Introduction

Due to the availability of surveillance videos and increasing computational power, crowd behavior analysis has recently received significant attention. This kind of interdisciplinary study, often employs or directly involves research results in the area of social sciences in addition to machine learning and statistical methods. However, there is no single, agreed definition of ‘a crowd’. Our work follows the definition in [2]: A crowd can be defined as a gathering of people, standing in close proximity at a specific location to observe a specific event, who feel united by a common social identity, and, despite being strangers, are able to act in a socially coherent way. This definition gives us a better understanding of crowd behavior and leads us toward the goal of this paper which is to identify, model, and learn a representation of the collective behaviors of people in crowd.

One way to represent the collective crowd behaviors is through finding semantic regions [22] in the scene. Semantic regions correspond to the paths that are commonly taken by objects which share the same activities and behavior. Recent approaches for understanding collective crowd behavior can be divided into two areas based on the respective feature spaces. The first category are the so called motion pattern estimation methods [17, 7, 12, 10, 20]. These techniques usually employ instantaneous motion vectors (e.g., optical flow), to learn patterns of collective behavior in a scene. They have the advantage of bypassing object tracking where it is infeasible due to the dense crowd. The second group of methods attempt to directly organize or cluster long term object trajectories to extract meaningful regions corresponding to dominant paths in the scene [8, 16, 1, 5]. While longer trajectories or sequence of motion flow vectors are obviously more discriminative than instantaneous motion, object tracking in crowded scenes is a difficult problem and the result of existing methods is unreliable. Severely fragmented, or worse, mislabeled trajectories significantly contribute towards noise and errors in the behavior of models. In that sense, the shortest motion flow will always be more reliable albeit less discriminative. Therefore, in terms of temporal scale, these two broad categories of techniques respectively correspond to temporally instantaneous and temporally global features. In this paper we present a new method which works with sparse as well as...
extremely crowded scenes and can handle long-range, non-linear motion patterns as well as short ones. Similar to [22, 23], the input to our system is tracklets obtained by KLT keypoint tracker [15]. Our method relies on a novel 2D Orientation Distribution Function, which represents each quantized region in the scene. These descriptors are employed by a Time Homogenous Locally Markovian Meta Tracking method, which automatically detects the sources and sinks of the scene and then estimates the semantic regions. Using our approach, one can find all the potential semantic regions in a scene, which may not necessarily be taken by a pedestrian. The advantage of having all the potential semantic regions in the scene is that, one can predict the starting point and the destination of every pedestrian by just matching the partially observed trajectories with those semantic regions using any clustering technique. On the same dataset (NYC Grand Central), [22, 23] respectively discover 30 and 20 semantic regions in the scene while our proposed method finds 54 potential semantic regions. Figure 2 summarizes steps involved in our approach. The method consists of three main parts. First, scene is quantized into hexagons to achieve smoother and more accurate trajectories in spite of the discrete setting. Given a collection of tracklets, we learn a 2D Orientation Distribution Function for every hexagonal region. Second, once 2D ODFs are learned, we use our Time Homogenous Locally Markovian-Meta tracking approach to obtain potential source/sink locations in scene. Consequently those locations are used to initialize the meta-tracking for finding the candidates of semantic regions. In the final step, these candidates are smoothed using a Polynomial Regression Mixture (PRM) model. Our 3-term trajectory clustering method which is based on Perpendicular, Angular and Vicinity distances, takes the smoothed semantic paths as well as the smoothed tracklets, and cluster them according to their sources and sinks. We evaluate our proposed method on New York Grand Central Station dataset [22] and compare it with other state-of-the-art techniques. Quantitative and qualitative experiments show, that our method can better model the crowd behavior and outperform other methods in finding the potential semantic regions and predicting the sources and sinks of pedestrians. The rest of this paper is organized as follows: In Section 2, we review the related work. In Section. 3, we describe the process of scene motion representation. Section 4 explains how we find the semantic regions. We present our 3-term clustering technique in Section 5. Experimental results are presented in Section 6 and we conclude the paper in Section 7.

2. Related Work

As mentioned earlier, one of the important and prevalent types of scene motion abstraction techniques, is the one derived from object trajectories. In recent years, a number of different methods and features for trajectory and path modeling of traffic in the scene have been proposed. These methods differ by their choice of features, models, learning algorithms, applications, and training data. In following, we briefly discuss some of these methods.

A hierarchical clustering of trajectories is proposed in [4, 14], where trajectories are represented as a sequence of states in a quantized 6D space for trajectory classification. The method is based on a co-occurrence matrix that assumes all trajectory sequences to be of the same length. However, this assumption is usually not true in real sequences. Detection of sources and sinks in a scene as a pre-step allows robust tracking. In [13], a Hidden Markov Model based scheme of learning sources and sinks is presented, where all sequences are two-state long. The knowledge of sources and sinks is used to correct and stitch tracks in a closed-loop manner. Hu et al. [5] have presented an algorithm for learning motion patterns where foreground pixels are first clustered using fuzzy K-means algorithm and trajectories are then hierarchically clustered based on the results of the previous step.

Another group of motion abstraction methods are scene modeling and understanding techniques. The most commonly used features for scene modeling are low level motion and appearance features [21, 18, 17]. Wang et al [17] obtain sparse optical flow for pixels with frame differences above a certain threshold. They then use them as low level observations while Kratz and Nishino [7] use spatiotemporal gradients as the most basic features. Saleemi et al [12] represent dense Lucas-Kanade optical flow as Gaussian mixtures using hierarchical clustering. We argue that both aforementioned categories lie at the two extremes in terms of temporal scale of the underlying features. Long term object trajectories are noisy and difficult to obtain while instantaneous optical flow has little discriminative power in the absence of association to preceding or subsequent frames. Our proposed algorithm employs KLT trajectories to learn 2D Orientation Distribution Functions for each quantized spatial region, thereby resulting in a three state feature, i.e., it can be used to estimate the probability of an object going from its current location to another given its previous location. We experimentally show that this feature at an intermediate temporal scale results in a better performance compared to the instantaneous (two state) features.

3. Scene Representation

We begin by describing our proposed spatial quantization of the scene which is an important step to obtain a reliable representation. To best utilize the dynamic model of the scene, we propose to use a hexagonal sampling of the location space, rather than a rectangular one. Since hexagonal lattice has more preferred directions it has been shown to reduce aliasing artifacts compared to the regular grids.
Figure 2. The block diagram of our proposed method. Tracklets obtained by KLT keypoint tracker are used to train descriptors on a hexagonal lattice. Once descriptors are learned, we apply the meta-tracker to find the gates. The gate’s locations are further used to find the semantic paths in the scene. Given a partially observed tracklet in the scene and learned semantic paths, our similarity-based trajectory clustering method finds the belief of starting point and destination for the person associated to the tracklet.

Once the hexagonal lattice and tracklets are generated, each hexagon is assigned a two dimensional orientation distribution function (2D-ODF). All the tracklets which lie in a hexagon (partially or fully) contribute to the ODF of that hexagon. Trajectories have more valuable information compared to the optical flows, since they represent the behavior of a person for a short period of time. Our 2D-ODF uses these informations. Given the entry direction of a trajectory in a hexagon, it allows computation of all the transition probabilities for that trajectory going to the adjacent hexagons. Figure 3 (c) shows an example of a 2D-ODF indicating that if a person enters the hexagon via its third edge, he/she would most likely choose the first edge as the outgoing direction. We should note that 2D-ODF assigns non-zero probabilities to other transition choices.

\[ P_{h_i}(\theta_{out}^{l} | \theta_{in}^{k}) = \frac{1}{N} \sum_{i=1}^{N} f(T_i), \]

\[ f(T_i) = \begin{cases} 
1 & 60^\circ (k-1) < T_i(1) < 60^\circ k \text{ and } 60^\circ (l-1) < T_i(n) < 60^\circ l \\ 
0 & \text{others} 
\end{cases} \] (1)

where \( P_{h_i}(\theta_{out}^{l} | \theta_{in}^{k}) \) shows the transition probability for a person who enters \( h_i \) from the \( k \)-th edge and leaves the hexagon from the \( l \)-th edge.

4. Extracting Semantic Regions

Individuals in crowded scenes are observed to usually come together and share common goals and interests in a coherent manner which defines their collective behavior.

The semantic regions in a scene represent these behaviors. Basically, semantic regions are the paths which are more likely to be taken by pedestrians. One intuition behind finding the semantic regions in the scene is that, once all the potential paths are found, it will be easier to estimate the source and sink for a person using any clustering method given its partially observed trajectory. In order to find the semantic regions, we first introduce our meta-tracking approach. It is later used to automatically find the entry and exit locations and potential semantic regions in the scene.

4.1. Time Homogenous Locally Markovian Meta Tracking (THLM Meta Tracking)

Once every hexagon is assigned a 2D-ODF, THLM-meta tracking can be launched. The intuition behind our tracking method is coming from the 2D-ODF. Each value in the ODF defines a transition probability between two hexagons, given the incoming direction. This means that, one can lay a grid of particles in any hexagon and have them advected...
in the scene using these transition probabilities in an iterative fashion. We define a track $T$ as an ordered list of observed hexagons, i.e. $T = \{h_1, h_2, ..., h_n\}$ where $h_i \in H$. Given the transition probabilities between two neighboring hexagons in the lattice, obtained from the ODFs, we can compute the transition probability between any two hexagons by sequentially multiplying such transition probabilities. More details are provided in Algorithm 1. Since the transition probability between two adjacent hexagons is independent from the previous states and does not change through time, We call it Locally Markovian and Time Homogenous.

One should note that the transition probability between non-neighboring hexagons depends on all the hexagons in the track.

4.2. Automatic Gate Detection

In order to find the semantic regions in the scene we need to know where the gates are located. In [22, 23] the gates are manually annotated, however given the 2D-ODF representation of the hexagons and our THLM-Meta-Tracking approach we can accurately infer the gates’ locations. Since the 2D-ODFs are learned from the real data, we can make sure that they will mostly guide the particle toward the gates’ location. Consequently, we initialize some particles, considering random incoming directions, in some random hexagons and have them advected in the scene. Once they reach the hexagons located in the borders, we increment the vote for that hexagon belonging to a gate. These observations can now be considered as a 3D space $(x, y, v)$. Where $(x, y)$ defines the spatial location of a hexagon and $v$ shows the vote for that hexagon. We aim to learn this 3D distribution model as a mixture of Gaussian components. The theoretically straightforward way to do this is random/approximate initialization of a chosen number of components followed by a parameter optimization algorithm like EM. We can then write, $X \sim \sum_{i=1}^{N} w_i N(X|\mu_i, \Sigma_i)$, where $w_i$, the weight of each cluster is the percentage of the tracklets that end in the $i^{th}$ cluster, e.g vote for that cluster. $N$ represents a Gaussian distribution, and the Gaussian parameters $\mu$ and $\Sigma$ represent the center and size of the gate respectively. An example of our Gaussian mixture model as well as the manually annotated gates can be seen in Figure 4. It can be observed that some of the gates are merged due the size of the hexagons used in our experiments. In other words, a smaller hexagon size would find the location of the gates more accurately.

4.3. Finding Semantic Regions

We employ our meta-tracking approach to find the semantic regions. We lay a grids of particles in the hexagons which belong to the gates considering all the six possible incoming directions and have them advected in the scene. The particles will start propagating in the scene and each will be assigned a probability accordingly. We see in Figure 3 that the outgoing directions which are along with incoming directions are more preferable for pedestrians and will have a higher transition probability $P_{h_i} (\theta^m | \theta^m)$, but one should note that we allow them to propagate in any direction while each will have a probability assigned to them. Since the 2D-ODFs are learned on the tracklets, time series observation of pedestrians dynamics, our simulation favors the paths which are more likely to be taken in the scene by assigning a higher probability to them. Algorithm 1 summarizes the THLM meta-tracking procedure for a grid of particles. The tracking will stop whenever one of these criterias is satisfied: (1) if a particle takes a sharp turn between $h^\tau$ and $h^{\tau+1}$ (in our case more than 120 degree), (2) if particles reach a hexagon which belongs to one of the gates, and (3) when the probability of the path goes below the threshold $\alpha$. Once the meta-tracking is done for all the hexagons belonging to the gates. The meta-tracks which start at one gate and end at another gate are selected to be potential semantic regions. Due to the quantization step of our approach the semantic paths that we obtain in this step are not smooth. Therefore,

![Figure 4. The output of our gate detection method (left) and the manually annotated ones (right). The center of the gates ($\mu$) along with their sizes ($\Sigma$) are defined using the Gaussian mixture model.](image-url)

**Algorithm 1:** Finding potential semantic regions

**Procedure** Compute Meta-Track ($T$)

**Input:** $h^\tau, \theta^m, N_p$

**Output:** metaTrack $T$

StopFlag $\leftarrow$ false, $\tau \leftarrow 1$, $T \leftarrow \emptyset$

**while** StopFlag is false do

**for** each edge $m$ in $h^\tau$ do

if $P(h^\tau | \theta^m) \neq 0$, then

$h^{\tau+1} = h^\tau + \delta \times \theta^m$

$N^\tau_p = P(h^\tau | \theta^m) \times N^\tau_p$

$T(\tau) = h^{\tau+1}$

$\theta^m = \theta^m$

$\tau \leftarrow \tau + 1$

if Stopping Criteria Satisfies then

StopFlag $\leftarrow$ true;

end

end

end
in order to smooth the tracklets, handle the noise, reducing the computation cost and at the same time increase the accuracy of our similarity based clustering method, we utilize a Polynomial Regression Mixture (PRM) \[\text{(3)}\] to estimate each tracklet with a fixed number of points. Eqn. \(2\) shows how a \(p\)-th order Vandermonde matrix is used to map \(x_i\) to \(y_i\) where \(y_i\) is the observed tracklet (semantic path) which has \(n_i\) points and each point is represented by \(x_i\). In our case \(x_i\) corresponds to the 2D coordinate of the point.

\[
y_i = x_i\beta + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2 I),
\]

\[
X_i = \begin{bmatrix}
1 & x_{i1} & x_{i1}^2 & \cdots & x_{i1}^p \\
1 & x_{i2} & x_{i2}^2 & \cdots & x_{i2}^p \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_{in_i} & x_{in_i}^2 & \cdots & x_{in_i}^p
\end{bmatrix}
\]

Parameters \(\beta\) and \(\sigma^2\) are learned through an EM framework. We used second order PRM with 5 EM iterations for smoothing both tracklets and semantic regions. Our approach returns all the possible paths between two gates. In order to have fewer representative paths for each pair of gates, we cluster the path according to their corresponding probability into several groups. Consequently, each cluster is represented with one path which is the average of the cluster’s paths and is assigned a probability equal to the average probability of all the paths within that cluster. This gives us our final semantic regions between each pair of gates. Figure 5, in the top row, shows the smoothed semantic paths that the meta tracker has found and the bottom row shows the final semantic regions that we get after clustering. Then by employing our approach, we are able to find 54 out of 64 the potential semantic paths that a person might take in the scene as well as its corresponding probabilities.

5. Trajectory Clustering

Our goal is to cluster trajectories according to their belief of starting point and destination and not necessarily their spatial location and shape alone. Therefore, we prefer to compute the similarity between two trajectories, considering their sub-trajectories. Later the contribution of different sub-trajectories are fused together to obtain the final similarity. This would give more weight to the similarity scores for partitions that play a more important role in shaping the belief of a trajectory. Inspired by Lee’s [8] work, we propose our PAV-based (Prependicular-Angular-Vicinity) trajectory clustering method in which we borrowed the partition-grouping idea and modified version of three terms distance function from TRACLUS [8]. We aim to satisfy the conditions of belief estimation problem while reducing its computation cost. TRACLUS formalizes trajectory partitioning using minimum description length (MDL) principle followed by an approximate solution in order to palliate the high cost of computing the optimal trajectory partitioning solution. Instead, we use Polynomial Regression Mixture (PRM) which not only makes the trajectories smooth but also satisfies the aforementioned desired properties of partitioning. This modification drastically lowers the computation cost of partitioning. Once the partitions are found, for each sub-trajectory, we search a neighboring area around it to find partitions of semantic paths that fall within that area. The radius of neighboring area is selected adaptively according to the hexagon which the partition of tracklet lies in. In hexagons with low-density of semantic paths, passing through, the radius increases while the searching area would shrink when the density is high. The density of hexagons is shown in Figure 7. Later, the similarity between a partition of query tracklet and each of those sub-semantic paths that fall within neighboring area is computed using three different measures: Perpendicular, Angle distance and Vicinity density.

In contrast to [8], instead of using parallel distance, we use vicinity density measure. The reason behind this is that the more a query tracklet stays in vicinity of a semantic path, the higher the chance of belonging to that semantic path should be. Our experiments show that such hypothesis is true. By giving more weight to vicinity density, we observed a considerable improvement in results. Finally we fuse three similarity measures using a weighted linear summation to determine the similarity score of a partition of query tracklet belonging to a particular semantic path. Similarity scores of different partitions of the query tracklet are summed up in a weighted manner so that scores of those partitions that are closer to the gates, play a more important role in generating the final probability of to which semantic path the query tracklet belongs to. This helps us to emphasize on similarity measures that matter more in determining the belief of tracklets. Algorithm 2 summarizes the PAV-based trajectory clustering algorithm.

6. Experiments

We evaluate our proposed method on the publicly available New York Grand Central Station dataset [23]. The sequence is 33 minutes long, frame rate is 24, and the resolu-
Figure 5. The semantic regions found for 4 gates. The top row shows the smoothed path that meta-tracking achieves. Each figure in top row shows the semantic regions corresponding to one of the gates, e.g the top left one shows all the possible paths that a person who enters the scene from gate 1 is likely to take. The bottom row shows the final semantic regions that we get for those gates after clustering and averaging.

Figure 6. Trajectories assigned to different semantic paths found by (a) our model, (b)[23] and (c) [22]. Colors of each trajectory is randomly assigned.

Figure 7 shows some statistics of the tracklets associated with the New York Grand Central Station dataset. One sees that most of the tracklets are observed only for a short period of time. Moreover, illustrated in Figure 7 (right), most of the tracklets with a length shorter than a hundred frames are located close to the center of the scene. This makes the prediction and clustering even more challenging. Our hexagonal lattice consists of 285 hexagons of 20 pixels radius. We use all the 48,000 tracklets to learn our 2D-ODF descriptors. To have a fair comparison with [23], we used 8 manually annotated gates shown in Figure 4. The meta-tracking approach is applied at every hexagon that belongs to the gates. Each hexagon is initialized using $10^6$ particles and the tracker is launched 6 times considering one of the incoming directions each time. After running the tracker for all of the hexagons of a gate we obtain all the potential paths that a person who leaves that gate is more likely to take. We later post-process these paths by...
Algorithm 2: PAV-based trajectory clustering

| Input: query tracklet ($Q$), hexagon weights, gate proximity weights |
| Output: belief of query tracklet |

for all partition of query tracklet $Q_i$
do
    Find in which hexagon $Q_i$ falls in;
    Compute neighboring area according to the hexagon weights;
    Find sub-semantic path in neighboring area ($SP_{mn}$);
    for each $SP_{mn}$
do
        Compute perpendicular (P) distance with $Q_i$
        Compute angular (A) distance with $Q_i$
    end
    Find average of P and A distances
    Compute number of frames ($N_f$) that a SP is in vicinity of $Q_i$
end
Compute weighted sum of similarity scores
Normalize the scores by $N_f$
pick the semantic path corresponds to the highest probability as the belief of $Q$

Figure 8. 2-step trajectory reconstruction. (a) shows the partially observed tracklet, hexagon which the track ends at, the entry direction and the 2D-ODF for that hexagon. The groundtruth is shown in (b). The meta-tracks are presented in (c). (d) shows the semantic region which is matched with the tracklet. Final path which is the highest scored path that end at the gate which is defined by the matched semantic path.

non-linear regression to make them smooth. Most of the times for each pair of gates we obtain more than 20 semantic paths. In order to have fewer representative paths, we cluster all the paths into 5 groups based on their scores. Each group is represented via a path which is the average path of the members and a score that is the average scores of all the members. These paths represent our final semantic regions. An example of the semantic regions are shown in figure 5. Compared to 20 and 30 semantic paths, respectively found by [23] and [22], we find a total of 54 semantic paths out of all the 64 possible paths. These paths describe the dynamics of the crowd. Figure 6 shows trajectories that our method, [23] and [22], respectively, assign to different semantic paths.

6.1. Prediction Semantic Regions

We compared our method with [23] as well as two motion pattern approaches [6, 19] in terms of finding the semantic paths in the scene. Figure 9 illustrates the outcome of different methods. The first row shows the semantic paths which we obtain using our proposed method. Total 36 semantic paths out of 54 are shown for 6 gates. The second row shows the output of multi-agent model proposed by Zhou et. al [23]. The third and fourth rows, show the outputs of motion pattern approaches proposed in [6] and [19], respectively. Motion pattern approaches which are based on noisy low-level motion features, e.g optical flow, fail in some scenarios, especially those in which the motion is sparse. Moreover, when two semantic regions overlap in some areas and share similar motion features, these methods fail to correctly distinguish between two paths. This is not the case for our method.

6.2. 2-Step Trajectory Prediction

Once the 2D-ODFs and Semantic Regions are learned, given partially observed trajectory of a person, our model can predict the most likely path that the pedestrian takes. The motivation is that we want to construct a potential trajectory from the current object position and its past behavior which acts as the predicted behavior of the person. Thus, we employ a 2-Step Trajectory Prediction approach which uses both the local (current location) and global (the whole tracklet) information to predict the belief of destination for a given pedestrian. First, we initialize the meta-tracker using the current state of the target, while considering the incoming direction to the hexagon. Once the tracker is initialized, it gives us all the possible paths that a person might take as well as their corresponding joint probability. Second, we find the closest semantic path to the observed tracklet using our 3-term clustering method. The semantic paths help to find the most probable destination for the pedestrian. Later, we select the path with the highest probability which ends at the predicted gate as our final prediction. Figure 8 shows our trajectory prediction approach. Figure 8(a) shows the hexagon that the current target state lies in along with the incoming direction and its 2D-ODF. The ground truth is shown in Figure 8(b). In Figure 8(c) all the possible meta-tracks are shown as well as the one which is selected as our final prediction by matching the semantic path shown in Figure 8(d). Table 6 provides a quantitative comparison with [9] and [23] for the task of trajectory clustering.

<table>
<thead>
<tr>
<th>Method</th>
<th>ConVelocity</th>
<th>[9]</th>
<th>[23]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Error</td>
<td>73.74</td>
<td>~ 61</td>
<td>~ 45.6</td>
<td>38.05</td>
</tr>
</tbody>
</table>

Table 2. Trajectory prediction comparison
Table 1. Trajectory clustering comparison

<table>
<thead>
<tr>
<th>Measure</th>
<th>2−1</th>
<th>2−8</th>
<th>3−4</th>
<th>6−7</th>
<th>7−4</th>
<th>7−5</th>
<th>7−6</th>
<th>7−8</th>
<th>8−2</th>
<th>8−7</th>
<th>Avg</th>
<th>[22]</th>
<th>[16]</th>
<th>[1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp.</td>
<td>93.5</td>
<td>75.1</td>
<td>99.1</td>
<td>94.3</td>
<td>76.1</td>
<td>86.0</td>
<td>100</td>
<td>94.0</td>
<td>81.1</td>
<td>90.7</td>
<td>90.0</td>
<td>71.9</td>
<td>20.63</td>
<td>35.0</td>
</tr>
<tr>
<td>Corr.</td>
<td>97.3</td>
<td>93.0</td>
<td>86.4</td>
<td>91.2</td>
<td>87.8</td>
<td>91.5</td>
<td>90.7</td>
<td>95.6</td>
<td>96.8</td>
<td>96.5</td>
<td>92.7</td>
<td>89.5</td>
<td>88.36</td>
<td>92.0</td>
</tr>
</tbody>
</table>

Figure 9. First row shows semantic regions learned by our model. Second row represents the output of [23]. Third and forth rows show the outputs of [6] and [19] respectively.

6.3. Trajectory Clustering

We evaluate our proposed method for the task of clustering using the correctness and completeness measures. Correctness is the accuracy that two tracklets, which belong to different semantic regions based on the ground truth, are also grouped into different clusters. Completeness is the accuracy that two tracklets, which belong to the same activity category, are also grouped into the same cluster by the algorithm. A good clustering algorithm should have both high correctness and completeness. Table 6 shows the correctness and completeness for 11 different clusters, each corresponds to a semantic region starting from one gate and terminating in another. There are totally 4,332 annotated tracklets. None of the annotated tracks is a complete one, i.e, start from one gate and ends at another gate. For 1,850 of those annotated tracklets, we do not know either starting or ending gate. We compare our method with [22], [16] and [1] as is reported in [22]. It is important to note that those methods use different annotations, which are not publicly available and impossible to regenerate. However, it is necessary to compare with them since they constitute the state-of-the-art on the Grand Central Station. Moreover, since it is not clear which semantic path each cluster belongs to, we just compare the average completeness and correctness over all the clusters.

7. Conclusion

In this paper, we proposed a novel approach to automatically mine the most dominant paths taken by pedestrians in a crowded scene. We divide the scene via a hexagon lattice and compute the transition probabilities for each hexagon. Estimated 2D-ODF allows us to assign a probability to different transition scenarios which passes through a particular hexagon. Then, we use our THLM meta-tracker to find all the possible paths that a pedestrian might take in the scene, given all the 2D-ODFs. Finally, we cluster these paths into fewer representative semantic paths that are highly probable to be taken. We show how our method can be used in trajectory prediction in addition to identifying the source and sink of movement patterns. To evaluated the effectiveness of our proposed method, we compared our approach with other state-of-the-art techniques both qualitatively and quantitatively.

References


[16] X. Wang, K. Ma, G. Ng, and W. Grimson. Trajectory analysis and semantic region modeling using a non-parametric bayesian model. 2008. 1, 8


