PEET: Prototype Embedding and Embedding Transition for Matching Vehicles over Disparate Viewpoints

Yanlin Guo *Ying Shan* *Harpreet Sawhney* *Rakesh Kumar*

Email: {yguo, yshan, hsawhney, rkumar}@sarnoff.com

Address: Sarnoff Corporation, 201 Washington Road, Princeton, NJ 08534

Abstract

This paper presents a novel framework, Prototype Embedding and Embedding Transition (PEET), for matching objects, especially vehicles, that undergo drastic pose, appearance, and even modality changes. The problem of matching objects seen under drastic variations is reduced to matching embeddings of object appearances instead of matching the object images directly. An object appearance is first embedded in the space of a representative set of model prototypes (Prototype Embedding (PE)). Objects captured at disparate temporal and spatial sites are embedded in the space of prototypes that are rendered with the pose of the cameras at the respective sites. Low dimensional embedding vectors are subsequently matched. A significant feature of our approach is that no mapping function is needed to compute the distance between embedding vectors extracted from objects viewed from disparate pose and appearance changes, instead, an Embedding Transition (ET) scheme is utilized to implicitly realize the complex and non-linear mapping with high accuracy. The heterogeneous nature of matching between high-resolution and low-resolution image objects in PEET is discussed, and an unsupervised learning scheme based on the exploitation of the heterogeneous nature is developed to improve the overall matching performance of mixed resolution objects. The proposed approach has been applied to vehicular object classification and query application, and the extensive experimental results demonstrate the efficacy and versatility of the PEET framework.

1 Introduction

We present an approach to the problem of matching views of objects, such as vehicles, across views with pose and illumination variations. The application is that of classification and class-based querying of vehicles using exemplars as queries, in a wide area surveillance system with non-overlapping, widely separated camera viewpoints. Due to the separation of cameras, objects undergo pose, appearance, and scale change from different viewpoints as shown in Fig. 1. As a result, direct matching of views is in general not reliable. An even more important reason for avoiding direct matching is that for querying direct matching requires linear access of all the objects in the database which is not feasible when millions of objects are in the database. Even indexing with quasi-invariant features such as in [11] may not be feasible due to large appearance variations.

Figure 1. Top Row: A single object viewed by different cameras in disparate locations exhibits large appearance change. Middle & Bottom Rows: A single object viewed by multiple cameras in disparate locations and various orientations exhibits large pose change.

In this paper, we propose a Prototype Embedding and Embedding Transition (PEET) framework that is capable of matching objects over ***pose, appearance, scale, and potentially modality variations***. For example, objects that are seen from the top view (middle row in Fig. 1) and side view (bottom row in Fig. 1) can be correctly matching via PEET. Even multiple sensor data such as visible and IR can be matched although that is not the focus of this paper. In PEET, an object image is first embedded in a small representative set of model prototypes that are rendered from viewpoints similar to the camera pose since the cameras are fixed. The same set of model prototypes is used for all the cameras but each set is rendered with respect to its respective camera's viewpoint. Each object image in each camera is then embedded within the space of the respective camera's prototypes. Embeddings in the space of rendered prototypes result in low dimensional embedding vectors as representations of objects instead of the dense pixel or edge based representations in the original or processed images. Object matching problem is now

1-4244-1180-7/07/$25.00 ©2007 IEEE

formulated as the problem of matching embeddings. Embeddings represent a given vehicle not as itself but as a vector of distances for the given vehicle from the set of prototypes. Intuitively, a sedan is represented in terms of how far it is from prototypical sedans, SUVs, pickups, etc. By pre-computing the mapping between the embedding vectors disparate camera views and appearances can be matched efficiently. Another important advantage of PEET is that no training is required, and no expensive computation is involved, which makes it especially suitable for on-line classification and large database querying tasks. We review the related work in Section 2, and explain technical details in Section 3. Experimental results are given in Section 4, and we conclude in Section 5.

2 Related Work

Vetter and Poggio first proposed the idea of learning shape changes from the 2D prototype shapes of two distinctive views [1]. Under certain linearity assumptions, they decomposed the object image onto two sets of basis, one for shape and the other for texture. Instead of computing a linear projection within the space spanned by the basis shapes/textures, our approach uses an embedding process that requires only the distances of the query image with respect to each basis image.

Koller et al. [2] estimate a 3D deformable model of five vehicle classes from a video sequence, and use parameters for vehicle recognition and classification. Instead of using a hand crafted 3D model, our approach uses exemplars that either rendered from 3D model or automatically selected from real vehicle images. Our approach does not require precise model parameters for rendering exemplars.

Jacobs et al. [12] proposed vector of distances to other images for a given image, rather than the direct pairwise distance, as a robust measure of similarity for non-metric distance measures between images, for instance distance measures that are robust to outliers. Athitsos et. al. [3] use Lipschitz embedding to approximate the Chamfer distance for hand pose recognition with large database. Athitsos et. al. [4] propose an approach using AdaBoost to learn the embedding with the triangle inequality enforced. Grauman et. al. [5] use Locality Sensitive Hashing-based embedding (LSH-embedding) [6] to approximate the expensive Earth Mover's distance.

Shan et al. [7] also address the vehicle-matching problem using the embeddings but require a mapping function to align the matching scores. In our approach, we avoid this problem using a novel two-level structure, which normalizes the matching scores computed from different camera, without explicitly solving a non-linear mapping function.

On the application side, [8] also dealt with object matching between non-overlapping cameras and on-line learning of camera topology and path probabilities. Kettnaker&Zabih [9] and Pasula et al. [10] proposed a nice framework for object matching and feature learning. All these methods rely on directly matching vehicle objects across multiple cameras.

3 Proposed Algorithm

For the purposes of this paper, it is assumed that an in-camera tracker tracks vehicles within the field of view of a single camera and produces bounding boxes around observed vehicles. We have used our own version of a fixed camera tracker.

3.1 Overall Approach

We first describe the overall approach and then detail each step. Fig. 2. outlines our overall approach. To match objects observed in two different cameras, there are four major steps in the PEET framework:

Step 1: The image-to-model matching is performed using Prototype Embedding (PE) for camera 1. For an imaged vehicle, the best N1 matching models for camera 1 are selected based on embeddings. This step alone can be used for object classification.

Step 2: Embedding Transition (ET) is carried out through embeddings with model prototypes across the two cameras. The same set of N1 models is rendered respectively for the two cameras.

Step 3: The model-to-image matching is performed using Prototype Embedding (PE) at camera 2. This time for each of the N1 model prototypes, top N2 matched image objects are chosen.

Step 4: Finally from the potential matching set of N1*N2 matched candidates, best matches are selected through competition and combination. The results are query returns for a given object.

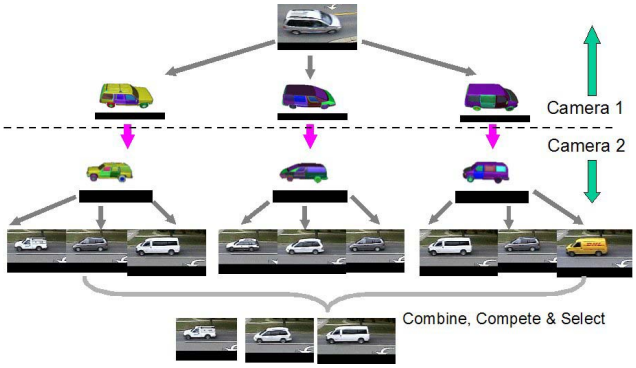

Figure 2. Overall schema of PEET.

3.2 Image Exemplar Based Embedding

PEET is largely inspired by [3, 4] and [7]. In [7] a set of representative image exemplars is first chosen for every

camera. For every object, say object i of dimension $N_i \times M_i$, viewed by every camera, say camera j, a distance $d_{i,j,k}$ is computed between the object and every exemplar k, $k = \{1,2,...,K\}$. For a specific object i and camera j, if we stack all the distances with respect to all the exemplars together, we obtain a K dimensional vector $\mathbf{d}_{i,j} = \{d_{i,j,1}, d_{i,j,2}, ..., d_{i,j,K}\}$. For the same object i seen by another camera j', we carry out the same routine and get another K dimensional vector $\mathbf{d}_{i,j'} = \{d_{i,j',1}, d_{i,j',2}, ..., d_{i,j',K}\}$. The two vectors $\mathbf{d}_{i,j}$ and $\mathbf{d}_{i,j'}$ are dubbed as embedding vectors. As can be seen from Fig. 3, if the object aspects between the two cameras j and j' are similar, $\mathbf{d}_{i,j}$ and $\mathbf{d}_{i,j'}$ are also very similar, and the L_2 distance or correlation between them can be used to decide if the two objects are similar or not. Therefore, we can effectively use a much lower dimensional feature to represent a high dimensional object (usually K << $N_i \times M_i$), and compute its similarity with respect to other objects.

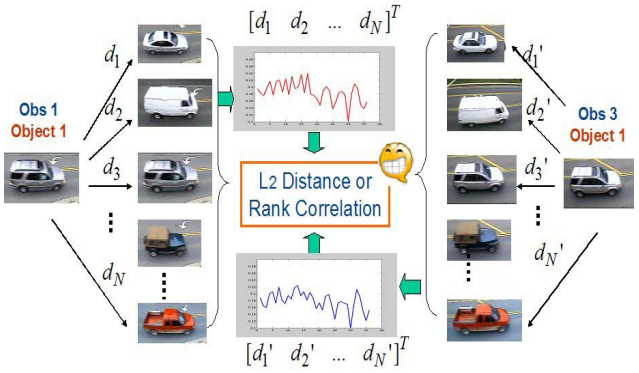

Figure 3. Image exemplar based embedding illustration. (For simplicity, subscripts denoting object and view indices are omitted in the distance representation.)

However, in the image exemplar based embedding, one needs to manually select common representative image exemplars for every camera, and not all the exemplars exist for all observers. In addition, each distance value $d_{i,j,k}$ in the embedding vector is supposed to reflect only the similarity between the object and the image exemplar, given that the illumination condition for the object instance and the exemplar from the same camera is similar. Since the tracker segmentation may not be perfect, the influence from background will be reflected in the distance computation and affect the matching.

The most significant disadvantage is that if the poses of the two cameras are very different, then the two embedding vectors for the same object will not be similar any more, as shown in Fig. 4. In this case, we need to compute a complex mapping function between the embeddings for the object instances in the two cameras. And more generally, for example, if one camera sees the front of an object, and the other camera sees the back of an object, there might not exist a continuous mapping function at all.

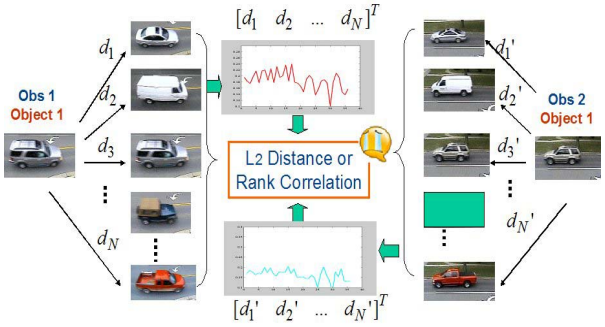

Figure 4. Exemplar Embedding cannot match objects with large pose change in this example. A complex mapping function needs to be computed between the embedding distances from the two cameras.

3.3 Prototype Embedding (PE)

To overcome the above-mentioned problems in image exemplar based embedding, we replace the image exemplars with prototype exemplars. As shown in Fig. 6, a set of representative object models are chosen and rendered at each camera location. Similar to image exemplar based embedding, if the shape of the two objects i and i' are similar, their embedding distances $\mathbf{d}_{i,j}$ and $\mathbf{d}_{i',j}$ (shown in read and blue curves) are also very similar, and the L_2 distance or correlation between them can be used to decide if the two objects are similar or not.

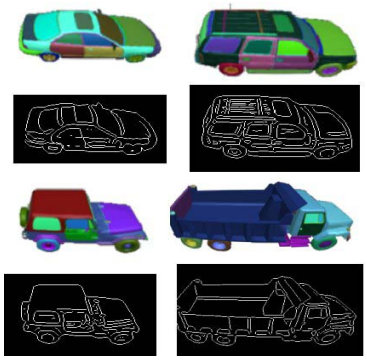

Figure 5. Some representative vehicle prototypes and their edge maps.

Some of the representative rendered vehicle prototypes and their edge maps are shown in Fig. 5. Note that each semantic part of a vehicle is rendered with a different color, so that the edge between different parts can be easily extracted. Since our object matching is based on geometric

properties of vehicles, the distance is defined to reflect the geometric aspect of vehicles. For example, Chamfer distance, after robust alignment and outlier rejection, between the edge maps of images or an image and a prototype is a good shape similarity measurement.

Compared with image exemplar based embedding, prototypes are easily rendered for all observers, and perfect segmentation of the object from the background is not needed. But simple replacement of images with prototypes does not solve the problem of reliably matching objects viewed from drastically different poses shown in Fig. 4. We resolve this problem with Embedding Transition (ET) described in Section 3.4.

3.4 Embedding Transition (ET)

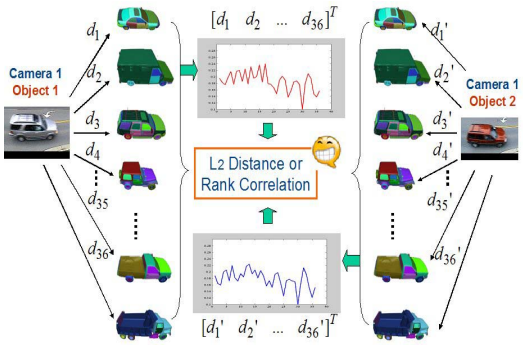

Figure 6. A Schematic of Prototype Embedding.

For matching vehicles across cameras, we need to employ an ***embedding transition*** scheme. First, as with object images, we treat each rendered model as an image i_M, and embed each model itself in the space of the K representative prototypes and obtain a K-dimensional embedding vector for the model $\mathbf{d}_{i_M,j} = \{d_{i_M,j,1}, d_{i_M,j,2}, ..., d_{i_M,j,K}\}$, as shown in Fig. 7. The set of embedding vectors $\mathbf{d}_{i_M,j}$, $i_M = \{1,2,...,K\}$ are pre-computed and stored in the database. Given the embedding vectors for each object image, and those for each model prototype, we can compare any image embedding vector $\mathbf{d}_{i,j}$ with respect to each of the model embedding vectors $\mathbf{d}_{i_M,j}$, $i_M = \{1,2,...,K\}$, and compute the following, as shown in Fig. 8:

- Given an image, return the best matched images;
- Given an image, return the best matched models (prototypes);
- Given a model (prototype), return the best matched images.

With this knowledge, we can now robustly and accurately match vehicles across cameras, as demonstrated in Fig. 2. It is easy to see that within each camera, images and prototypes share similar pose and modality. And across cameras, the pose and modality change is naturally handled by the rendering the same set of pre-chosen prototypes with corresponding pose and modality in difference camera locations.

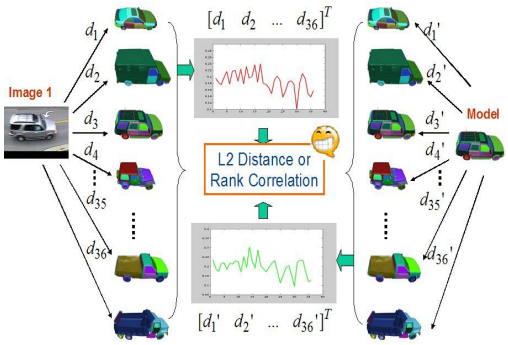

Figure 7. A Schematic of Model embedding.

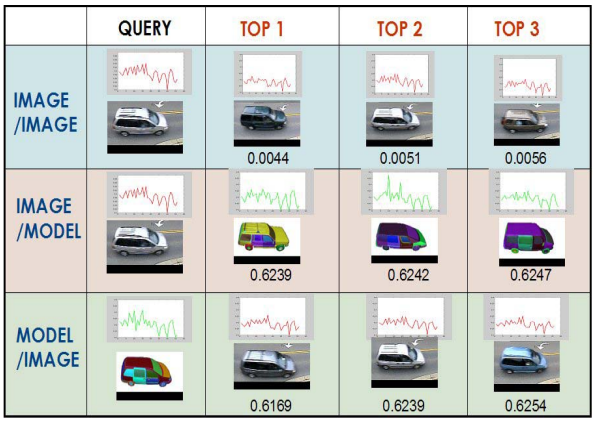

Figure 8. Model-Image embedding transition example.

3.5 Asymmetry Between Image and Model Prototype Matching through Embeddings

Even though we treat images and rendered models similarly in PEET, we have to point out that the accuracy of retrieving the best-matched models for a given image is different from the accuracy of retrieving the best matched images from a given model. The total number of models is usually in the order of 10 – 40, and for each class the total number of sub-class models is even smaller, usually 3-4. The total number of image objects is usually in the order of hundreds. Therefore, given an image object, most likely all the corresponding models will be selected as the top matched models, and very rarely a non-corresponding model will be chosen as the top match, if the image resolution is good enough. While given a rendered model, there will be many image chips that will potentially match the model, and more often than not, a non-corresponding image chip can be chosen. As a remedy, in the third step of PEET, i.e., the model-image Prototype Embedding (PE)

step, we introduce the forward-backward matching verification strategy. Given a model, for every returned image chip through PE, we also compute the matched models for the image chip, and require that the top K' (K' = 2 to 4) matched models be in the same class as the query model. This backward matching step greatly mitigates the mistake of selecting the non-corresponding image chips, and hence largely increases the overall cross-sensor matching accuracy. In other words, when retrieving image objects through a model prototype query, we require mutually consistent matching between the model embedding and the image embeddings.

3.6 Un-supervised Learning with PEET

Another aspect of PEET based matching is that if the resolutions of the two cameras are very different, the matching performance is different in the two directions. For example, if camera 1 is of higher resolution than camera 2, then retrieving the matched images in camera 2 for a given query image in camera 1 is better behaved than retrieving the matched images in camera 1 for a given query image in camera 2. This is because prototype embedding is susceptible to errors in alignment and distance computation if the image resolution is poor. As a result the "represent-ability" of the corresponding embedding vector is not reliable, thus leading to mistakes in model retrieval.

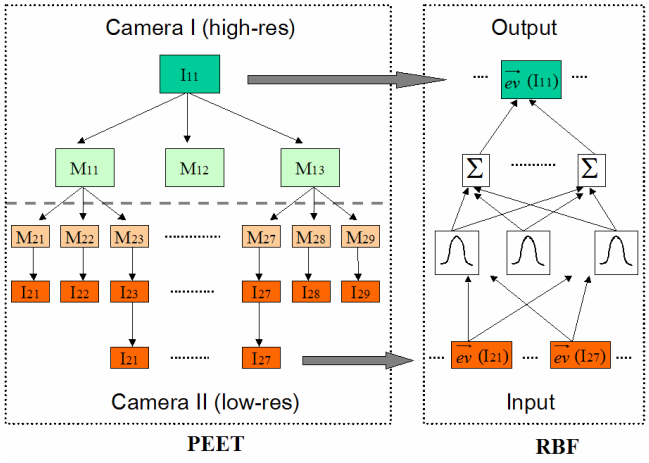

Figure 9. Un-supervised Learning with PEET.

To improve the relatively worse performance in the low-resolution ➔ high-resolution matching direction, we can take advantage of this heterogeneous matching behavior for the two different directions. Specifically, since matching images from high-resolution ➔ low resolution is good, for each high-resolution image in camera 1 (hi-res), we can generate K (usually 3-5) matched low-resolution images in camera 2 (lo-res) using PEET matching. Suppose we establish such a match between N1 image objects in hi-res to the lo-res camera. This results in N1*N2 pairs of matched image objects between hi-res and lo-res. Now we can establish a mapping between these N1*N2 embedding vectors, but this time we specify the input vectors of the mapping function to be the embedding vectors of images in camera 2 (low-res), and the output vectors to be those of images in camera 1 (high-res). With these N1*N2 embedding vector pairs, we use a Radial Basis Function (RBF) mapping to learn the mapping between them, as shown in Fig. 9.

Our intuition to choose RBF arises from the fact that vehicular objects usually consist of limited number of planar surfaces, and the total number of vehicle classes is also limited, therefore a weighted sum of some basis functions of the embedding vectors should be able to characterize the transformation between the two sets of embedding vectors in the two view points if the pose change is not drastically different.

3.7 Function Approximation & Mapping

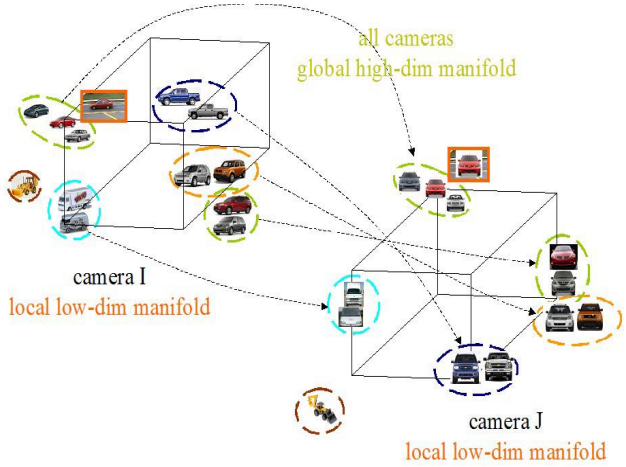

Figure 10. Space tessellation using prototype models.

The relationship between object classes and image features under arbitrary conditions (lighting, pose, and modality) is highly non-linear. However, for nearly fixed conditions, the relationship becomes tractable. As shown in Fig. 10, for each individual camera, objects lie in a low-dimensional manifold (as shown in the upper left or bottom right corner). If we tessellate the low dimensional space with enough objects, then for a new object (shown with a red bounding box), its class can be easily obtained by a proper function approximation scheme, i.e., prototype embedding in this case. To handle the highly complex function form for classification under different camera conditions (for example, across camera 1 & 2 in Fig. 10), PEET uses the same set of object prototype to tessellate all the local low dimensional space, and object classification becomes a non-parametric discrete function mapping problem.

As shown in Fig. 11, function mapping between local spaces can be simple continuous or discrete parametric mapping (first two rows), and it can also be a discrete non-

parametric mapping (third row). In Embedding Transition, we simply replace each date point in discrete non-parametric mapping with object model prototype (fourth row), and the non-parametric mapping can learned if enough data points are available in each local space.

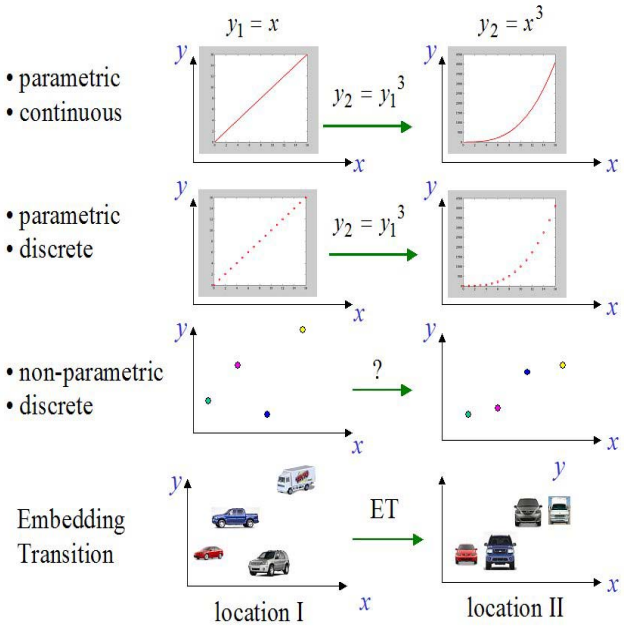

Figure 11. Embedding Transition (ET) as non-parametric discrete function mapping.

4 PEET Applications and Experimental Results

We apply the PEET for vehicle classification and querying in a large area visual surveillance system. The system consists of a network of non-overlapping cameras for tracking vehicles over an area of roughly 4 square kms. in modest to heavy traffic. Each camera covers typically 2 and at some locations 3 lanes of traffic; both the near and far lanes are covered. Vehicles typically appear larger in the near lane cameras and usually both top and side views are visible. Far lane cameras see mostly side views and vehicle resolution is smaller (usually 0.5 – 0.9 times smaller) (Fig. 1). We selected 12 representative cameras and collected data for around 30 minutes, with each camera seeing around 200 different vehicles passing through during the period of time. We use a frame-to-frame tracker to obtain image object chips with their approximate bounding boxes as shown in Fig. 1. We also manually created the ground truth for vehicle classes for all the observed vehicles in each camera.

4.1 Object Classification with Relatively High-Resolution Camera

We have tested our classification algorithm extensively on the above mentioned live surveillance system. We choose two representative data sets for our experiment. The first data set (DS1) was collected in the morning with moderate shadows, and the second data set (DS2) was collected in the afternoon with moderate to heavy shadows.

We classify all the vehicles (S) into five classes: Sedan (S1), SUV/Mini-Van (S2), Delivery Van/Bus (S3), Pickup Truck/Truck (S4), and none-of-the-above. We compute three types of classification performance:

1) TD: True Detection Rate. TD measures for a certain class, out of all detected vehicles belonging to the class, how many vehicles truly have the correct class label.
2) MD: The Mis-Classification Rate. MD measures for a certain class, out of all the vehicles that have that class label according to ground truth, how many are not classified as belonging to that class.
3) TD_all: Overall Detection rate for all classes.

Tables 1 & 2 show the classification performance (using the first step in PEET) for DS1 for four cameras -- 1, 3, 7, and 11. Tables 3 & 4 show the classification performance of DS2 for four cameras -- 1, 3, 7, 11, and 15. These are all near lane cameras, hence the resolution is relatively high, typically 200x100 pixels per vehicle. Note that there are much less pickup trucks and delivery vans in the datasets than sedans, SUVs, and mini-vans, therefore the performance on classes S3 & S4 is not as statistically meaningful.

Table 1. True & Overall Detection Rates for DS1

	TD(S1)	TD (S2)	TD (S3)	TD (S4)	TD_all (S)
cam1	87.65%	96.55%	62.50%	93.33%	88.82%
cam3	86.59%	94.92%	47.37%	100.00%	85.96%
cam7	82.80%	89.06%	72.73%	100.00%	85.39%
cam11	92.59%	85.51%	83.33%	100.00%	89.56%

Table 2. Miss Detection Rate for DS1

	MD (S1)	MD(S2)	MD (S3)	MD(S4)
cam1	0%	13.85%	9%	39.13%
cam3	2.74%	13.85%	0.00%	54.17%
cam7	0.00%	18.57%	33.30%	50.00%
cam11	1.32%	7.81%	25.00%	36.00%

Table 3. True & Overall Detection Rates for DS2

	TD(S1)	TD (S2)	TD (S3)	TD (S4)	TD_all (S)
cam1	97.52%	86.09%	46.67%	87.80%	90.06%
cam3	94.37%	86.21%	42.86%	96.55%	89.34%
cam7	97.35%	83.33%	63.64%	96.30%	90.12%
cam11	95.19%	76.53%	50.00%	85.71%	84.62%
cam15	94.23%	84.21%	58.33%	95.24%	88.73%

Table 4. Miss Detection Rate for DS2

	MD (S1)	MD(S2)	MD (S3)	MD(S4)
cam1	7.65%	10.81%	30.00%	12.20%
cam3	2.90%	15.73%	0.00%	28.21%
cam7	5.98%	6.59%	0	31.58%
cam11	16.10%	15.73%	0.00%	14.29%
cam15	2.00%	13.51%	0.00%	37.50%

4.2 Object Classification with Relatively Low-Resolution Camera

For the far lane cameras, the image resolution is much smaller, typically around 70x30 pixels per vehicle. If the same algorithm (the 1st step in PEET) is used for classification, the performance is poor because of the limited resolution. However, since we are confident about the classification performance with near lane, high-resolution cameras, we can utilize the heterogeneous nature in matching different resolution image objects as discussed in Section 3.6. Specifically, as described in Section 3.6, we learn a mapping between the embedding vectors of low-resolution and high-resolution objects, and convert the embedding vectors of far lane objects to those of near lane objects. Then we treat these converted embedding vectors as if they were obtained from the high-resolution cameras, and use the standard classification scheme (the 1st step in PEET) to perform classification. The performance comparison of with-mapping (NEW) and without-mapping scheme (OLD) for two cameras (2 & 4) is shown in Table 5. We can see that the learning based mapping scheme has greatly improved the true detection rate and reduced the miss-classification rate for all the classes of the far lane objects, including pickup trucks and delivery vans.

Table 5. Far Lane Object Classification Performance Comparison w/ & w/o Learning Based Mapping

	Cam 2		Cam 4	
	NEW	OLD	NEW	OLD
TD (S1)	90.54%	60.56%	88.75%	91.43%
MD (S1)	6.94%	18.87%	2.74%	33.33%
TD (S2)	80.00%	57.14%	90.00%	63.64%
MD (S2)	15.79%	48.94%	16.67%	6.67%
TD (S3)	100.00%	100%	70.00%	75.00%
MD (S4)	50.09%	86.96%	25.00%	52.63%
TD (S4)	80.95%	58.33%	100.00%	87.50%
MD (S4)	26.09%	46.15%	25.00%	22.22%

4.3 Object Querying Across Disparate Cameras

Figure12. Demonstration of object querying. The leftmost column shows the vehicle images used as queries. Each of the corresponding rows on the right show the vehicle objects returned as matches ordered from best to worst.

Another application of PEET is object querying in a large database, where we need to retrieve all the objects that belong to the same class as the query independent of the camera in which they were observed. We use the same data sets (DS1 & DS2) for testing. The query performance is illustrated in Fig. 12. The image chips in the left most column are query objects viewed from one camera, and chips in each row are all the retrieved objects viewed from other cameras for that particular query (only top six returns are shown because of limited space). Quantitative results are shown in Table 6, where the accuracy is computed as the total number of correctly matched objects within the top three returns over the total number of query objects.

Table 6. Object Query Performance for Both Same and Different Side Objects

Cross Camera Query for Same Side Lanes		Cross Camera Query for Different Side Lanes	
	Accuracy		Accuracy
cam001-003	97.63%	cam001-002	93.60%
cam001-007	97.25%	cam008-011	88.00%
cam011-015	97.87%	cam003-016	94.44%
cam004-002	95.18%	cam004-007	91.02%
cam012-008	95.79%	cam001-012	94.06%

Note that the query and retrieved objects can be seen on either the same side or the opposite side cameras. Also in our application appearance querying is similarity querying and not exact matching.

5 Conclusions

We have demonstrated that Prototype Embedding is a computationally expedient way to handle vehicle classification and querying in a system with cameras that view vehicles over a large area with disparate pose and illumination conditions. Furthermore, we also demonstrated that Embedding Transition could be used to associate vehicle appearances across cameras with significantly different resolutions. Results on representative datasets collected over a large area show encouraging results.

The idea of Embedding Transition needs to be further investigated for disparate view matching. Furthermore, embeddings can be defined as vectors of distances for parts of objects rather than whole objects as described in this paper. Finer scale classification and querying may be afforded by such a part based representation.

6 Acknowledgement

This work was performed under a DARPA/DOI Contract No. NBCHC030085.

Approved for Public Release, Distribution Unlimited.

References

[1] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI), 19(7):733–742, 1997.

[2] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. International Journal of Computer Vision (IJCV), 10(3):257–281, 1993.

[3] V. Athitsos and S. Sclaroff. Estimating 3D hand pose from a cluttered image. In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR03), page 432, 2003.

[4] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: A method for efficient approximate similarity rankings. In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR04), pages 268–275, 2004.

[5] K. Grauman and T. Darrell. Fast contour matching using approximate Earth Mover's Distance. CVPR04, volume I, pages 220–227, 2004.

[6] P. Indyk and N. Thaper. Fast image retrieval via embeddings. In International Conference on Computer Vision (ICCV03), 2003.

[7] Ying Shan, Harpreet S. Sawhney, Rakesh Kumar, "Vehicle Identification between Non-Overlapping Cameras without Direct Feature Matching", IEEE International Conference on Computer Vision (ICCV05)

[8] O. Javed, Z. Rasheed, K. Shafique, and M. Shah. Tracking across multiple cameras with disjoint views. In International Conference on Computer Vision (ICCV03), volume 2, pages 952–957, 2003.

[9] V. Kettnaker and R. Zabih. Bayesian multi-camera surveillance. In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR99), 1999.

[10] H. Pasula, S. J. Russell, M. Ostland, and Y. Ritov. Tracking many objects with many sensors. In International Joint Conferences on Artificial Intelligence (IJCAI99), pages 1160–1171, 1999.

[11] D. Nistér and H. Stewénius, Scalable Recognition with a Vocabulary Tree, CVPR 2006.

[12] D. Jacobs, D. Weinshall, Y. Gdalyahu, ``Class Representation and Image Retrieval with Non-Metric Distances". IEEE Trans. on Pattern Analysis and Machine Intelligence. 22(6):583-600, 2000.