

Regression from Local Features for Viewpoint and Pose Estimation

Marwan Torki Ahmed Elgammal
Department of Computer Science
Rutgers University, New Brunswick, NJ, USA
{mtorki, elgammal}@cs.rutgers.edu

Abstract

In this paper we propose a framework for learning a regression function from a set of local features in an image. The regression is learned from an embedded representation that reflects the local features and their spatial arrangement as well as enforces supervised manifold constraints on the data. We applied the approach for viewpoint estimation on a Multiview car dataset, a head pose dataset and arm posture dataset. The experimental results show that this approach has superior results (up to 67% improvement) to the state-of-the-art approaches in very challenging datasets.

1. Introduction

Many problems in computer vision can be formulated as regression problems where the goal is to learn a continuous real-valued function from visual inputs. For example the estimation of the viewpoint of an object, head pose, a person’s age from face images, illumination direction, joint angles of an articulated object, limb position, *etc.* In many of these problems, a regression function is learned from a vectorized representation of the input. For example, in head pose estimation, researchers typically learn regression from vectorized representation of the raw image intensity, *e.g.*, [14, 3, 8, 25, 9].

In the last decade, there have been a tremendous interest in recognition from highly discriminative local features such as SIFT [13], Geometric Blur [5], *etc.*, Most research on generic object recognition from local features have focused on recognizing objects from a single viewpoint or from limited viewpoints, *e.g.*, frontal cars, side view cars, rear cars, *etc.* Very recently, there have been some interest on object classification from multi-view setting [6, 11, 20, 19, 12, 21]. There have been also some promising results on pose recovery (3D viewpoint estimation) from local features for generic object class [20, 19, 12, 21]. The problem of object classification from multi-view setting and pose recovery are coined together. Pose (viewpoint) recovery is a fundamental problem that has been long studied for rigid

objects with no within class variability [7]. A very challenging task is to solve for the pose for a generic object class, *e.g.*, recovering the pose of a chair instance that was never seen before in training.

Most of recent work on viewpoint estimation from local features are based on formulating the problem as a classification problem [19, 20, 12, 22, 23, 21] where the viewpoint is discretized into a few number of bins, 4, 8, or 16, and a classifier is used to decide the viewpoint. Obviously, the accuracy of such classifiers is limited by how coarse the viewpoint is discretized. Such treatment does not facilitate the continuous estimation of the viewpoint and can not interpolate between the learned views.

Viewpoint estimation is fundamentally a continuous regression problem, where the goal is to learn a regression function from the input; similarly, other problems such as posture estimation. *The question we address in this paper is how to learn a regression function from local features: their descriptors and their spatial arrangement.*

Local features are designed to have some geometric invariant properties. For example, SIFT [13] is view invariant. From two close viewpoints, we expect to see the same local features. Such local features can be useful in viewpoint estimation only if we consider apart views. If our goal is to accurately recover the viewpoint, local features’ descriptors only are not enough. It is obvious that the spatial arrangement of the features will play a more important role in this case. Recent work have addressed this though encoding the spatial information through a pyramid spatial subdivision [16], or through enforcing geometric constraints at test time [12]. Relative distances between parts have also been used [19, 20]

In this paper we introduce an approach for learning a regression function from local features. The approach is based on the feature embedding approach introduced in [24] where it was shown that an embedded representation that encodes both the features’ descriptors and their spatial arrangement can be achieved. In this paper we show how such an embedding can be used to achieve regression from local features that takes into consideration the features’

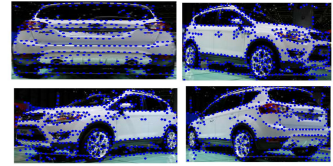
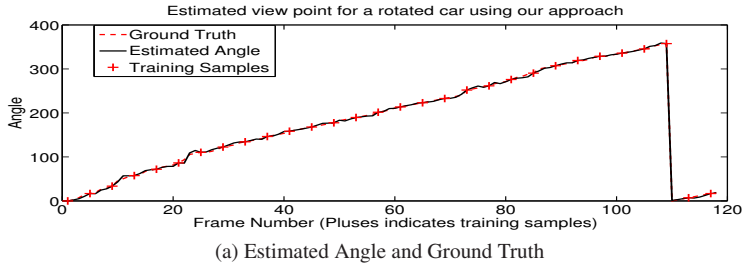


Figure 1: Regression on a single car: (Left) Absolute Error computed using our approach is plotted with the ground truth, they are very close to each other. (Right) sample views of the car with features detected on it.

descriptors and their spatial arrangement. The regression is achieved by defining a proper kernel in the embedding space. We also show how a supervised manifold constraints can be enforced in the embedding. For example, for viewpoint estimation, we can enforce that the viewpoints lie on a one dimensional manifold. In the resulting embedding space, image similarity can be measured in a way that reflect the smooth changes in the functions to be learned, *e.g.* the smooth changes in viewpoint. Therefore, we can learn a regression function from local features that can accurately estimate viewpoint from a small number of training example and a small number of features. The experimental results show that this approach has superior results to the state-of-the-art approaches in very challenging datasets (*e.g.* in a challenging multiview car data set we have 67% improvement over the state-of-the-art result [16]).

Figure 1 shows an example of our results in estimating the viewpoint of a car from local features. In this example we used 30 instances for learning, $\approx 12^\circ$ apart, with 200 local features, with no correspondences established. The learned regression function can estimate the viewpoint with less than two degrees error.

2. Kernel Regression Framework

The training data is a set of input images, each represented with a set of features. Let us denote the input images (sets of features) by X^1, X^2, \dots, X^K , where each image is represented by $X^k = \{(x_i^k \in \mathbb{R}^2, f_i^k \in \mathbb{R}^{\mathcal{F}})\}$, $i = 1, \dots, N_k$. Here x_i^k denotes the feature spatial location and f_i^k is the feature descriptor and \mathcal{F} denotes the feature descriptor dimension. For example, the feature descriptor can be a SIFT, HOG, *etc.* Notice that the number of features in each image might be different. We use N_k to denote the number of feature points in the k -th image. Let N be the total number of points in all sets, *i.e.*, $N = \sum_{k=1}^K N_k$.

Each input image is associated with a real-value, $v^k \in \mathbb{R}$, for example, v^k can be the angle representing the viewpoint, or the head pose of the k -th image. Therefore, the input is pairs in the form (X^k, v^k) . For simplicity, here we show how regression can be done to real numbers, extension to real-valued vectors is straight forward. Extension to real-valued vectors is necessary for problems like articu-

lated posture estimation where joint angles are estimated.

The goal is to learn a regularized mapping function $g : 2^{\mathbb{R}^2 \times \mathbb{R}^{\mathcal{F}}} \rightarrow \mathbb{R}$. Notice that unlike traditional regression, the input to such a function here is a set of features from an image with any number of features. This function should minimize a regularized risk criteria, which can be defined as

$$\sum_k \|g(X^k) - v^k\| + \lambda \Phi[g] \quad (1)$$

where the first term measures the error in the approximation, the second term is a smoothness function on g for regularization, and λ is a regularization parameter. From the representer theorem [10] we know that such a regularized regression function admits a representation in the form of linear combination of kernels around the training datapoints (or a subset of them). Therefore, we seek a regression in the form

$$v = \hat{g}(X) = \sum_j b_j K(X, X^j) \quad (2)$$

Therefore, it is suffice to define a suitable positive definite kernel $K(\cdot, \cdot)$ that measures the similarity between images that takes into consideration the local features and their spatial arrangement. Once such kernel is defined we can solve the coefficients b_j by solving a system of linear equations [17].

3. Kernel-based Regression from Local Features

3.1. Feature-Spatial Embedding

In this section we summarize the local feature embedding framework introduced by Toriki *et al.* [24]. The goal of the approach is to achieve an embedding of a collection features from training images in a way that preserves two constraints: Inter-image feature affinity and Intra-image spatial affinity. Two data kernels based on the affinities in the spatial and descriptor domains separately are assumed. The spatial affinity (structure) is computed within each image and is represented by a weight matrix $S_{ij}^k = K_s(x_i^k, x_j^k)$. Here x_i^k denotes the image coordinate of the i -th feature in the k -th image and $K_s(\cdot, \cdot)$ is a spatial kernel local to

the k -th image that measures the spatial proximity between features within each image. The feature affinity between image p and q is represented by a weight matrix \mathbf{U}^{pq} where $\mathbf{U}_{ij}^{pq} = K_f(f_i^p, f_j^q)$ that captures the similarity in the descriptor domain between the i -th feature in image p and the j -th feature in image q , using a descriptor space kernel $K_f(\cdot, \cdot)$.

To achieve a joint feature-spatial embedding space satisfying the constraints mentioned above, the following objective function in the embedding coordinates Y is minimized

$$\Phi(Y) = \sum_k \sum_{i,j} \|y_i^k - y_j^k\|^2 \mathbf{S}_{ij}^k + \lambda \sum_{p,q} \sum_{i,j} \|y_i^p - y_j^q\|^2 \mathbf{U}_{ij}^{pq}, \quad (3)$$

where k , p and q are image indices and i, j are feature indices within each image. The two terms in the objective function try to preserve the spatial arrangement and the feature descriptor similarity respectively. This objective function can be rewritten using one set of weights defined on the whole set of input points as:

$$\Phi(Y) = \sum_{p,q} \sum_{i,j} \|y_i^p - y_j^q\|^2 \mathbf{A}_{ij}^{pq}, \quad (4)$$

where \mathbf{A} is defined as a block matrix with its diagonal blocks set to the spatial affinity matrices \mathbf{S}^k and its pq block is the feature affinity matrix \mathbf{U}^{pq} . The weight matrix \mathbf{A} is an $N \times N$ symmetric matrix where N is the number of features in all images. It was shown that the objective function Eq. 4 reduces to the problem of Laplacian embedding [15] of the point set defined by the weight matrix \mathbf{A} . The result is a feature embedding space that jointly captures the similarity of the feature descriptors and local spatial arrangement. Each image is represented in the embedding space by a set of embedding coordinates corresponding to its features.

Embedding the features in a new image amounts to solving an out-of-sample problem. However, this is not similar to traditional out-of-sample solutions, which learn a mapping function between the input space and the embedding space [4], since here the input is a set of features with two different kernels defined. In [24] a solution was proposed for this special out of sample problem. Given a set of feature in a new image $X^\nu = \{(x_i^\nu, f_i^\nu)\}$, the feature affinity is computed, denoted by \mathbf{U}^ν where $\mathbf{U}_{ij}^\nu = K_f(f_i^\nu, f_j^\tau)$ where f_j^τ are samples from the training features. The spatial affinity local to the new image is also computed, denoted by \mathbf{S}^ν where $\mathbf{S}_{ij}^\nu = K_s(x_i^\nu, x_j^\nu)$. The embedding can be achieved by solving an optimization problem that tries to keep the coordinate of the previously embedding coordinate, \hat{y}_i^τ 's, unchanged, *i.e.*,

$$\begin{aligned} \min \quad & \text{tr}(\mathbf{Y}^T \mathbf{L} \mathbf{Y}) \\ \text{s.t.} \quad & \mathbf{y}_i^\tau = \hat{y}_i^\tau, i = 1, \dots, N^\tau, \end{aligned} \quad (5)$$

here \mathbf{Y} is the embedding of coordinate of all points properly

stacked and \mathbf{L} is the laplacian of the matrix \mathbf{A} is defined as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}^\tau & \mathbf{U}^{\nu\tau T} \\ \mathbf{U}^{\nu\tau} & \mathbf{S}^\nu \end{pmatrix}. \quad (6)$$

3.2. Enforcing Manifold Locality Constraint

To achieve a smooth image similarity kernel from local features, we learn an embedded representation of the features and their spatial arrangement, as was described in Sec 3.1. Let $y_i^k \in \mathbb{R}^d$ denotes the embedding coordinate of point (x_i^k, f_i^k) , where d is the dimensionality of the embedding space, *i.e.*, we seek a set of embedded point coordinates $Y^k = \{y_1^k, \dots, y_{N_k}^k\}$ for each input feature set X^k .

The embedding approach as described in section 3.1 satisfies two constrains: Inter-image feature affinity and Intra-image spatial structure. Besides these two constraints, we need to add a third constraint that enforces manifold locality, we denote that by *Supervised Manifold Locality Constraint*. The idea is to enforce existing manifold structure in the data, features from images neighboring each other on the manifold should be embedded close to each other. For example, if images are labeled with viewpoints, such label can be used to define a neighborhood for each image. Since we are using the labels to define the neighborhood, this is a supervised enforcement of the data manifold constraint. Enforcing the manifold constraints have been shown to highly improve regression results in many applications [3, 18, 25, 9]. However all these applications used vectorized representations of the raw intensity.

We can enforce the manifold constraint in a supervised way from the labels v^k . This can be achieved by amending the objective function in 3 by supervised weights between images as

$$\Phi(Y) = \sum_k \sum_{i,j} \|y_i^k - y_j^k\|^2 \mathbf{S}_{ij}^k + \lambda \sum_{p,q} \sum_{i,j} \|y_i^p - y_j^q\|^2 w(p, q) \mathbf{U}_{ij}^{pq}, \quad (7)$$

where $w(p, q)$ denotes a weight function that measure the supervised affinity between images X^p and X^q as implied by their labels v^p and v^q . There are many ways to define such weights. If we set all the weights to one, we reduce to an unsupervised embedding as in Eq 3. The weights can be set to reflect labels distances, *i.e.*, $w(p, q) = \mathcal{G}(v^p - v^q)$. For example a Gaussian function can be used or alternatively, the weights can be set to reflect neighborhood structure by using a uniform window kernel. Therefore the matrix \mathbf{A} can be redefined as

$$\mathbf{A}_{ij}^{pq} = \begin{cases} \mathbf{S}_{ij}^k & p = q = k \\ \mathcal{G}(v^p - v^q) \cdot \mathbf{U}_{ij}^{pq} & p \neq q \end{cases} \quad (8)$$

3.3. Feature Embedding based Regression

Since each image is represented in the embedding space by a set of Euclidean coordinates in that space, the similarity in the embedding space can be measured by a suitable

set kernel that measures the distance between any two sets of embedded features representing two images. There are a variety of similarity measures that can be used. For robustness, we use a percentile-based Hausdorff distance, defined as

$$H_l(X^p, X^q) = \max\{\max_j^{l\%} \min_i \|y_i^p - y_j^q\|, \max_i^{l\%} \min_j \|y_i^p - y_j^q\|\} \quad (9)$$

where l is the percentile used. Since this distance is measured in the feature embedding space, it reflects both feature similarity and shape similarity. However this distance is not a metric and, therefore, does not guarantee a positive semi-definite kernel. Therefore we use this measure to compute a positive definite matrix \mathbf{H}^+ by computing the eigen vectors corresponding to the positive eigenvalues of the original $\mathbf{H}_{pq} = H_l(X^p, X^q)$. The regression problem now can be solved by using kernels based on matrix \mathbf{H}^+ in the embedding space, *e.g.*, Radial Basis Function (RBF) kernels are used. Therefore, we can solve for the regression parameter in Eq. 2.

Given the learned regression function, it can be applied to any new image. However, the features in that new image has to be mapped first to the embedding space. Therefore, the regressor for a new test image X will be in the form

$$v = \hat{g}(X) = \sum_j b_j K(\mathcal{O}(X), Y^j) \quad (10)$$

where $\mathcal{O}(X)$ is a function that maps the features in a test image X into a set of coordinates in the embedding space, *i.e.*,

$$\mathcal{O}(X) : \{(x_i, f_i)\} \longrightarrow \{y_i\}$$

The out of sample solution described earlier can be used to obtain such a function in a closed form.

3.4. Image Manifold based regression:

The regression can be also learned from an image manifold embedding space, which can be obtained using the similarity kernel defined on the feature embedding space. This is a second embedding where each image is represented by a single point in a Euclidean space. However the problem with this approach is that for any test image two out of sample problems have to be solved: First, out of sample on the features should be used to map them to the feature embedding space. Second, the embedded set of features has to be used to achieve the image coordinate in the image embedding space using a second out of sample. The advantage of learning a regressor from the image embedding space is that enforcing manifold constraints on the images can be easier in that space. However, a two stage embedding and two out of sample problems discourages this approach. Our experiments showed that learning the regression from the feature embedding space with the manifold locality constraint enforced produces similar results to the image manifold based regression.

Table 1: Regression on a single car

Train	Supervised	Dim	MAE ^o	std(AE)
30	Yes/30°	20	2.34	1.99
30	Yes/30°	40	2.06	1.65
30	Yes/30°	80	2.04	1.64
30	Yes/30°	100	1.94	1.63
30	Yes/30°	160	1.93	1.63
30	Yes/30°	200	1.95	1.59
15	Yes/30°	100	5.47	4.21
30	Yes/30°	100	1.94	1.63
40	Yes/30°	100	1.84	1.66
30	Yes/45°	100	1.94	1.5
30	Yes/60°	100	2.09	1.66
30	No/∞	100	2.16	1.83

4. Experiments

4.1. Regression on a single car example

We use a single car sequence (first car) from the dataset introduced by [16] to demonstrate the different setups for our approach and to show the effect of the different parameters. The sequence contains 118 views of a rotating car. We changed the following parameters: 1) The number of training images to learn the feature embedding, which are also used as RBF centers: 15, **30**, and 40. 2) The dimensionality of the embedding space: 20, 40, 80, **100**, 160, and 200. 3) Manifold supervision neighborhood size: **30°**, 45°, 60°, and ∞, where ∞ means unsupervised embedding. We change one parameter at a time while we fix all other parameters with a default value (shown in bold above). In all experiments we fix the RBF scale to 0.05 of the median Hausdorff distance in the data. We measure the mean and standard deviation of the absolute error (MAE, std(AE)), between the estimated and the ground truth viewpoints. Table 1 shows the obtained results for various settings. Fig 1 shows the estimated and ground truth angles for the default base case: 30 training samples, 100 dimensions, 30° neighborhood. The MAE in this case is 1.94°. From the table we can see that, in general, the accuracy in the regression does not change much with the change in the parameters. We can see that when the number of training images increased from 15 to 30 the mean absolute error dropped to half of its value, increasing the training size after that does not change the accuracy much. Also we can see that the dimensionality of the embedding space is insignificantly affecting the MAE. Notice that there is an error in the ground truth itself of the same order as the error in the estimation. So, this experiment basically shows that we can achieve accurate regression on a single object from local features from a small number of sparse training samples. *Additional plots and sequences can be seen in supplemental material.* In the next experiment we show results on the whole dataset.

4.2. Multi-View Car Dataset

In this experiment we used the ‘Multi-View Car Dataset’ that was introduced recently in [16], which captures 20 rotating cars in an auto show. The dataset is very challenging since the cars are accompanied with much clutter even within the detected bounding boxes. It has large class variation in appearance, shape, and texture of the cars in this dataset. We use this data set since it provides finely discretized viewpoint ground truth, the discretization varies in each car sequence. However, there are some drawbacks and challenges in this dataset: 1) The high within-class variation makes it hard for a regressor or classifier to generalize. 2) Ground truth accuracy problems: The viewpoint is calculated using the time of capturing assuming a constant velocity, which affects the ground truth. There are some frames of the same car that have the same time stamp but there is a slight change in the pose. Also, in few frames, the cars are partially occluded by passing people. 3) Some cars are highly symmetric from a side view, that makes classifiers subject to 180° reflection error in some views. Such reflection error exist in other datasets as well and reported in the results of [16, 19, 22]. 4) Some cars are very odd, and it is very hard, even for humans, to discriminate between whether the car front or rear is facing the camera.

The dataset has been used for viewpoint classification in [16] where the viewpoint was discretized into 16 bins. In [16] their goal was to classify the car pose using a bag-of-words technique that is based on a spatial pyramid of histograms. They build 16 SVM classifiers for the 16 bins to cover the 360 range of rotation (i.e., bin size is 22.5°). We use the results of [16] as a baseline since it incorporates both the features and their spatial arrangement through the spatial pyramid structure. *The approach proposed in [16] resulted in 41.69% viewpoint classification accuracy from bounding box input. In contrast, given a similar 16 bin setting, our approach results in 70% accuracy using the same bounding box as inputs, that is over 67% improvement over the state of the art result.*

In our regression experiment, we use the same split of training and test sets as [16]. The dataset contains 20 sequences for 20 rotating cars. The total number of images is 2137, the first ten cars are used for training (1103 images) and the last ten cars for testing (1034 images). We used only 135 images (sampled randomly from 4 sequences of train data) to learn an initial feature embedding. Each image is represented using 50 geometric blur features [5]. The initial feature embedding is then expanded using out-of-sampling to include all the training images with maximum of 350 features per images (the number of features extracted per image varies).

We learn our regression model using Radial Basis Functions (RBF) as described in section 2. We examine the effect of “supervision”, i.e., enforcing the view manifold con-

straint on the initial embedding by defining a neighborhood for each image not to exceed 45° difference. For quantitative evaluation, we use the Mean Absolute Error (MAE) between the estimated and ground truth viewpoint. In addition we also used the MAE of 90% percentile of the absolute errors and the 95% percentile of the absolute errors. These are used because, typically, a very small percentage of the test data produces very large error (180°) due to reflection, which biases the MAE. While MAE is a good measure for validating regression accuracy, it is not suitable for comparison with classification-based viewpoint estimation approaches which uses discrete bins, such as [16, 19, 22]. Therefore, we also used the estimated viewpoint to compute the error in discretized viewpoint classifier. For example, to achieve an equivalent of a 16 bin viewpoint classifier, we compute the percentage of test samples that satisfies $AE \leq 22.5$, where the absolute error $AE = |Est.Angle - GroundTruth|$. With this measure we can compare to the 16 bin classifier used in [16]. To achieve an equivalent of an 8 bin viewpoint classifier, we also compute the percentage of test samples that satisfies $AE \leq 45$

For comparative evaluation we evaluate different supervised and unsupervised setting within our framework as described in section 3.1, in addition we used the results from [16] as a baseline. We also evaluated a support vector regressor (SVR) based on our framework. For each setting we evaluated the 10/10 split as described above and also a leave one out split (learn on 19 cars and test on 1).

We show our results in table 2, we observe the following: – The MAE is ranging from 33.9° to 40.60°, which seems to be a large error. However, this might be misleading because if we compare reported results in [22] in which they learn classifiers for 8 bins, the reported average accuracy on the diagonal of the confusion matrix is 66%. In this case this means only 66% of the testing set is recovered within the bins and any error adds at least 45°. The last two columns of table 2 show that around 65% of testing samples are giving $AE < 22.5^\circ$ and around 80% or more are giving $AE < 45^\circ$. The source of the higher MAE is then coming from few instances with large reflection errors (around 180 degrees), this also clear in the percentiles MAEs. Comparing our results to the reported confusion matrices in [16, 22]¹ we can find that our approach has a lower reflection effect in the estimated angles. This can be noted in Fig. 3, only few test samples lies in the last bin of the histogram.

– As we can see the supervised setting is giving the best results for this dataset. This confirms that enforcing the neighborhood constraint on the manifold is in fact boosting the regression results.

¹The confusion matrix in [16] was shown without the actual numbers in it, we obtained the actual values from the authors of [16].

Table 2: Regression on Multi-View car dataset, baseline and different variants of our approach

Method	MAE 90% percentile	MAE 95% percentile	MAE	AE<22.5	AE<45
Results [16] (Baseline)	–	–	46.48	41.69%	71.2%
Unsupervised(RBF) - 50% split	27.17	32.65	39.2	50.09%	73.6%
Unsupervised(RBF) - Leave One Out	22.57	27.12	35.87	63.73%	76.84%
Supervised (RBF) - 50% split	19.4	26.7	33.98	70.31%	80.75%
supervised (RBF) - Leave One Out	23.13	26.85	34.9	55.83%	76.65%
Unsupervised(SVR) - 50% split	29.52	34.44	40.60	41.19%	70.12%
Supervised (SVR) - 50% split	25.23	30.63	36.07	57.9%	78.6%



Figure 2: Regression on a Multi-view car dataset: Top left corner shows how the arrows reflect the estimated angle. The ground truth is shown along with the estimated angle. Yellow arrows for ground truth and Magenta for our results, features are shown as blue dots(Best viewed in color)

– Also we can observe that using the leave one out settings for regression is not improving beyond few degrees over the split settings. This means that our approach is generalizing well since it does not gain much by including as many training samples.

4.3. Face Pose Estimation in Uncontrolled Environment

In this experiment we used ‘Face Pose’ dataset that was introduced recently in [1, 2]. It has been used in inferring the face pose of freely downloaded faces from the web. The pose ranges from -90° to 90° , the ground truth is manually labeled for 11900 images, 10900 of them were used for training and 1000 for the testing. The images that were used in [1] experiments are 60×60 bounding boxes that were normalized using a Euclidean warp. The dataset is a real world challenging set that exhibits much variation in controlling factors like illumination, scale, expression and pose as well as partial occlusion and background clutter. However, we want to mention the drawbacks of the dataset. First the distribution of the pose degrees is very biased and only few examples are beyond the range $[-50^\circ, 50^\circ]$ which affects the regression we learn. Second as mentioned in [1] the manual

labeling is not so accurate since four subjects were asked to label every image and the pose is then averaged. The correlation of the manually labeled poses between different subjects was $\approx .75$ [1].

In our regression experiment, we use the same training set and same test set as [1], and we compare our results in terms of the MAE and Pearson Correlation Coefficient (PCC) as they were provided in [1]. We used 250 images (sampled randomly from train data) to learn the initial feature embedding of dimensionality 50 for each feature. Each image is represented using 24 geometric blur features. The initial feature embedding is then expanded using out of sample to include all the features from training images with maximum of 72 features per images (the number of features extracted per image is not equal). The dimensions of the images is the reason for the small number of extracted features per image when compared to the cars dataset.

We learn our regression model as we did in the cars dataset. We examine the effect of supervision on the initial embedding by defining a neighborhood for each image not to exceed 15° difference. The histogram of absolute error in Fig. 3 show that in around 86% of the case the estimated error is less than 20° .

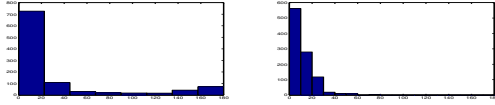


Figure 3: Histogram of absolute error: Left: for Multi view car dataset. Right: for face dataset.

We achieved an MAE error of 10.92° and 11.15° for the unsupervised and supervised cases respectively and PCC of .81 and .79 respectively. In [1] the reported results are MAE=13.21 and PCC=.76. We have better MAE for both the supervised and unsupervised settings. This shows that from sparse local features we can achieve better results in regression in this example. The most noticeable point is that the unsupervised is behaving better than the supervised setting. Although this might seem strange, but the distribution of poses of the training samples and the testing samples is very biased towards the region $[-50^\circ, 50^\circ]$ and actually in the 10900 training samples there is not a single image with pose in the interval $[-80^\circ, -90^\circ]$. Under this condition enforcing the neighborhood in the region that have few samples in the training will result in a poor generalization. Fig. 4 shows example of the estimated head pose.

4.4. Regression with Sparse Training Samples

Many datasets for viewpoint estimation use sparse training samples on the view circle, e.g. [19, 22]. In particular we used the car subset from the dataset **3DObjects** that have been used in [19, 22] to compare our regression approach to the methods that use binary classification namely [19, 22]. This data set contains only eight sparse views. We follow the same setup by using five sequences for training and testing on the other five sequences, which yields 160 images for training and 160 for testing. According to [22] the accuracy reported is 66.625%, where the accuracy for [19] is 52.5%. The accuracy is measured as the sum over the diagonal of the confusion matrix. In our case, in order to make the numbers comparable we report our regression results in terms of $AE \leq 45$. Using our regression approach with the supervised setting we achieve 77.5% for $AE \leq 45$, where for unsupervised setting we achieve 75%. This shows the ability of our approach to be used even with sparse view point classes.

4.5. Arm Posture Estimation

As we mentioned earlier, our approach is general and can be used in different regression problems, not only viewpoint estimation. We show here articulated body posture estimation for a subject who moves his arms freely. We used the sequence from [18]. The local features are affected very much by the clutter. The ratio of features on the hands to the extracted features is about 10%, all the features in each frame are used in the regression. The sequence contains 200

frames, 25 equally spaced are chosen for training (12.5% of the sequence). Initial Embedding: 150 features from 20 training frames, dimensionality 250. We then compute out of sample embedding for all 25 training frames, with 450 features each. Then we learn the regressor parameters for the hands and elbows joints positions from the 25 training frames. The regressor was used to estimate the position of the hands and elbows joints in the rest of the frames. We evaluated the estimation using 75 frames marked with ground truth and the error is 18 pixel in average per estimated parameter (image size is 640x480). Sample results are shown in the figure. See the supplementary materials for more results and videos.

5. Conclusions

We presented a novel and promising direction for regression from local features. The approach integrates the strengths from manifold learning as a global constraint on the data and the discriminative power of the local features and their arrangement. Through embedding we learn a representation for features and their spatial arrangement where regression is possible. Experiments showed that our framework gives superior results in very challenging datasets for viewpoint and face pose estimation. The approach is general and can be used within any regression problem that can benefit from local features such as age estimation or articulated body pose estimation.

Acknowledgments: This research is partially funded by NSF CAREER award IIS-0546372.

References

- [1] J. Aghajanian and S. Prince. Face pose estimation in uncontrolled environments. In *BMVC09*, 2009. 6, 7
- [2] J. Aghajanian, J. Warrell, S. J. Prince, P. Li, J. L. Rohn, and B. Baum. Patch-based within-object classification. In *ICCV09*, 2009. 6
- [3] V. Balasubramanian, J. Ye, and S. Panchanathan. Biased manifold embedding: A framework for person-independent head pose estimation. In *CVPR07*, pages 1–7, 2007. 1, 3
- [4] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *NIPS'04*. 3
- [5] A. C. Berg. *Shape Matching and Object Recognition*. PhD thesis, University of California, Berkeley, 2005. 1, 5
- [6] H. Chiu, L. Kaelbling, and T. Lozano Perez. Virtual training for multi-view object class recognition. In *CVPR07*, pages 1–8, 2007. 1
- [7] D. Forsyth and J. Ponce. *Computer Vision, a modern approach*. Prentice Hall, 2002. 1
- [8] Y. Fu and T. Huang. Graph embedded analysis for head pose estimation. In *FGR06*, pages 3–8, 2006. 1
- [9] G. Guo, Y. Fu, C. Dyer, and T. Huang. Head pose estimation: Classification or regression? In *ICPR08*, pages 1–4, 2008. 1, 3
- [10] G. S. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41:495–502, 1970. 2
- [11] A. Kushal, C. Schmid, and J. Ponce. Flexible object models for category-level 3d object recognition. In *CVPR07*, pages 1–8, 2007. 1

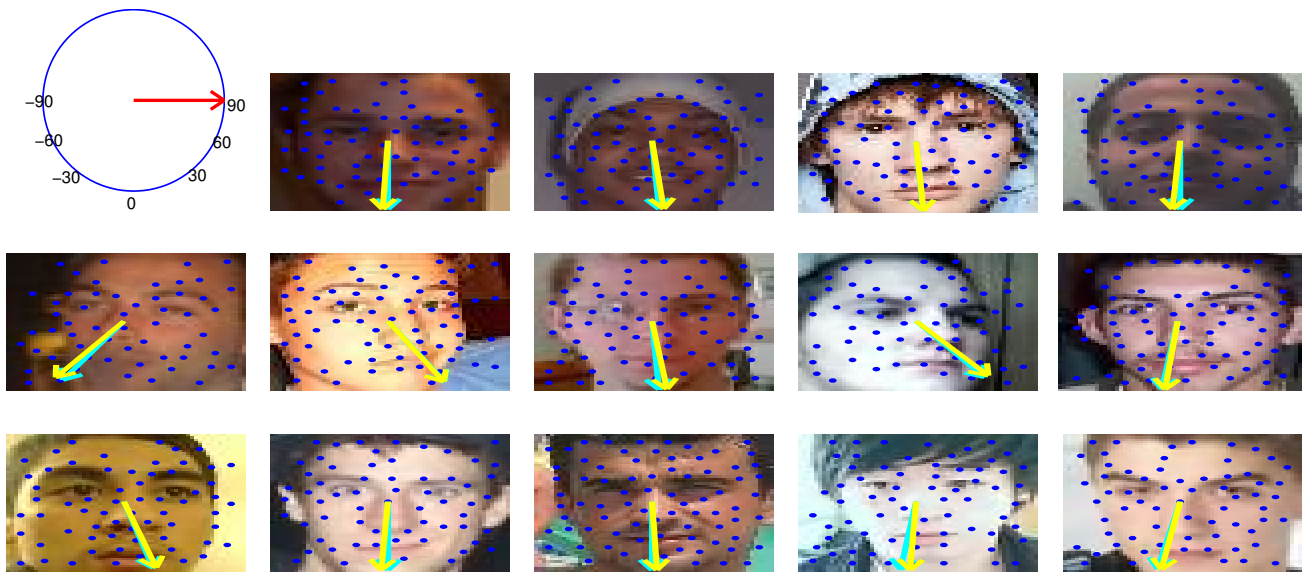


Figure 4: Regression for head pose estimation: Top left corner shows how the arrows reflect the estimated angle. The ground truth is shown along with the estimated angle. Green arrows for ground truth and Yellow for our results, features are shown as blue dots (Best viewed in color). More results are available in the supplementary materials.

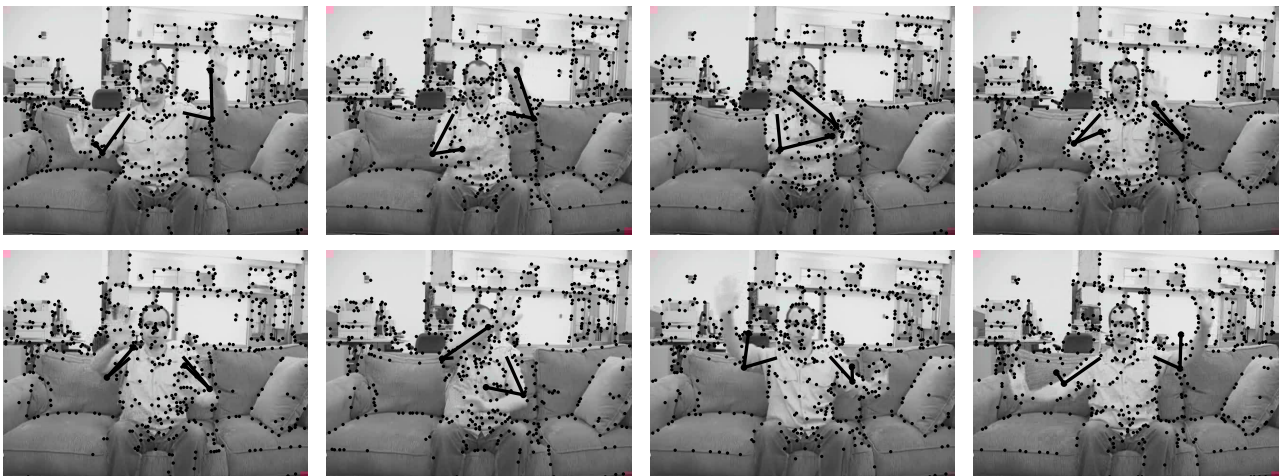


Figure 5: Regression example for articulated body posture estimation: shown are frames 20,40,60,80,100,120,140,160

- [12] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3d feature maps. In *CVPR08*. 1
- [13] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1
- [14] E. Murphy Chutorian and M. Trivedi. Head pose estimation in computer vision: A survey. *PAMI*, 31(4):607–626, April 2009. 1
- [15] P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 2003. 3
- [16] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *CVPR09*. 1, 2, 4, 5, 6
- [17] T. Poggio and F. Girosi. Network for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990. 2
- [18] A. Rahimi, B. Recht, and T. Darrell. Learning appearance manifolds from video. In *CVPR05*, pages I: 868–875, 2005. 3, 7
- [19] S. Savarese and F. Li. 3d generic object categorization, localization and pose estimation. In *ICCV07*, pages 1–8, 2007. 1, 5, 7
- [20] S. Savarese and F. Li. View synthesis for recognizing unseen poses of object classes. In *ECCV08*, pages III: 602–615, 2008. 1
- [21] H. Su, M. Sun, L. Fei Fei, and S. Savarese. Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *ICCV09*, 2009. 1
- [22] M. Sun, H. Su, S. Savarese, and L. Fei Fei. A multi-view probabilistic model for 3d object classes. In *CVPR09*, pages 1247–1254, 2009. 1, 5, 7
- [23] A. Thomas, V. Ferrar, B. Leibe, T. Tuytelaars, B. Schiel, and L. Van Gool. Towards multi-view object class detection. In *CVPR06*, pages II: 1589–1596, 2006. 1
- [24] M. Torchi and A. Elgammal. Putting local features on a manifold. In *CVPR*, 2010. 1, 2, 3
- [25] X. Wang, X. Huang, J. Gao, and R. Yang. Illumination and person-insensitive head pose estimation using distance metric learning. In *ECCV08*, pages II: 624–637, 2008. 1, 3