Discriminative Learning based Visual Servoing across Object Instances

Harit Pandya, K. Madhava Krishna and C. V. Jawahar

***Abstract*— Classical visual servoing approaches use visual features based on geometry of the object such as points, lines, region, etc. to attain the desired camera pose. However, geometrical features are not suited for visual servoing across different object instances due to large variations in appearance and shape. In this paper, we present a new framework for visual servoing across object instances. Our approach is based on a discriminative learning framework where the desired pose is estimated using previously seen examples. Specifically, we learn a binary classifier that separates the desired pose from all other poses for that object category. The classification error is then used to control the end-effector so that the desired pose is attained. We present controllers for linear, kernel and exemplar Support Vector Machine (SVM) and empirically discuss their performance in the visual servoing context. To address large intra-category variation in appearance, we propose a modified version of Histogram of Oriented Gradients (HOG) features for visual servoing. We show effective servoing across diverse instances over 3 object categories with zero terminal velocity and acceptable camera pose error at termination.**

I. INTRODUCTION

Visual servoing guides the robot to attain a desired pose with respect to the given object using image based feedback [1]. Visual features are extracted from the acquired image and matched against their desired configuration resulting in a residual error. This error is regulated to zero by generating appropriate control commands to the robot. Zero residual error in visual features signifies that the desired pose of the robot with respect to the given object is attained [1]. However, if the given object is different compared to that captured by the desired configuration of visual features, a large residual error may remain for any pose.

Practical scenarios require robots to servo a variety of objects with different shapes and appearances. For example, consider the problem of capping a soda bottle by a robotic arm where the task is to place a cap on top of the given bottle (desired pose). The general requirement is a robotic arm that is able to cap any bottle irrespective of its appearance and shape. A naive approach is to capture the desired pose with respect to every bottle and use classical visual servoing for capping. However, capturing the desired pose for every bottle becomes highly expensive as the number of unique bottles become large. Thus, classical visual servoing is not an optimal strategy for such scenarios and a new servoing strategy is required that could servo any given instance of an object category to the desired pose.

The problem of servoing across object instances in a category was initially addressed in [2] where the authors used keypoint features that semantically encode the location of parts of an object (for example, location of handle of a cup in the image) to make the approach robust to variation in object's appearance. They used a linear combination of available 3D models for a servoing iteration. However, the semantic features were computed manually that makes the approach laborious for large number of object instances. Moreover, the procedure requires a search over all models in all pre-rendered poses for every visual servoing iteration, which makes the approach computationally very expensive.

Harit Pandya is supported by TCS Reserach PhD fellowship.
International Institute of Information Technology, Hyderabad, India.
{harit.pandya@research.iiit, mkrishna@iiit, jawahar@iiit}.ac.in

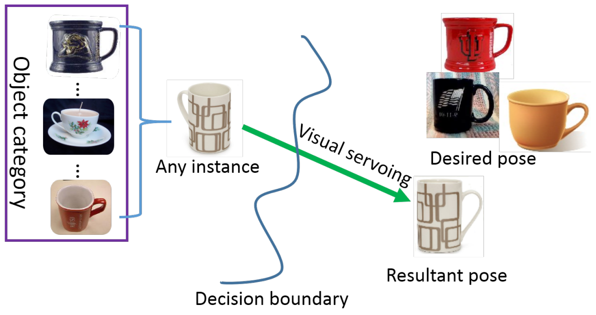

Fig. 1. **Visual servoing using discriminative learning.** From previously seen examples of different instances of same category in the desired pose, we learn a classifier that discriminates the desired pose from rest of the poses. Given a new instance from the same object category, we use the classification error to control the robot such that a resultant pose similar to the desired pose is attained.

In this paper, we estimate the desired pose for a new object instance of the specified category using a discriminative learning framework trained on previously seen instances of that category. Specifically, we use Support Vector Machine (SVM) [3] that learns a hyperplane separating the desired pose from rest of the poses with maximal margin. The SVM classification error which is computed as the distance from the hyperplane is used to predict the deviation between current pose and the desired pose of the object. The goal is to control the end-effector of the robot by iteratively minimizing the classification error such that the desired pose is attained starting from the initial pose of any given instance from the same object category. As the desired pose is attained the classification error becomes non-positive. Geometrically the trajectory could be interpreted in the classification subspace as a path of data sample crossing the decision boundary as shown in figure 1.

To discriminate pose variations robustly under varied appearance changes, we propose visual features similar to Histogram of Oriented Gradients (HOG) [4]. Since, HOG features are difficult to represent analytically, we use a modified version of HOG by retaining the principal orientation

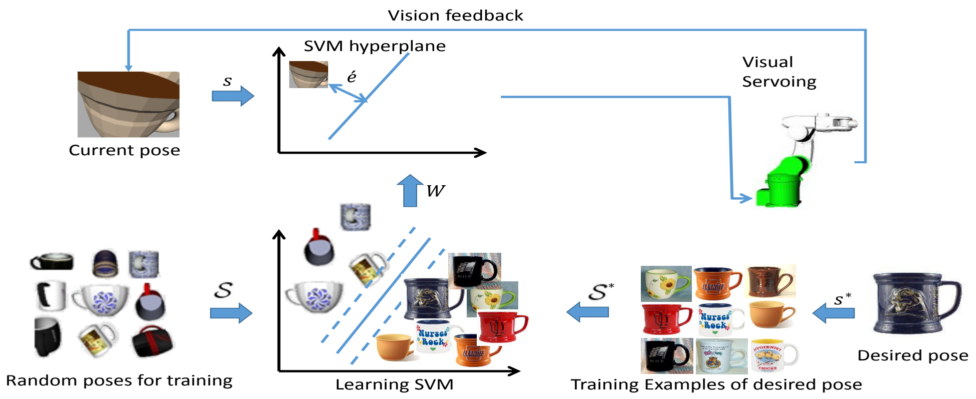

Fig. 2. **Overview of the proposed approach.** Given examples of desired pose of different instance of same category (bottom-right) $\mathcal{S}^*$ and some random poses $\mathcal{S}$, we learn a classifier (SVM) that separates the desired pose from other poses by a maximal margin. Once the classifier is learned, the decision boundary $\mathbf{w}$ or the SVM parameters (support vectors, their coefficients and kernel function) in case of kernel SVM, are used to compute the servoing error $\acute{e}$ for the current pose of a new object instance from the same object category. Levenberg-Marquardt gradient descent on $\acute{e}(\mathbf{s},\mathbf{w})$ is used to servo to the desired pose of the given object instance. Note that, we assume the object is segmented and its category is known.

bin followed by glyph image construction. By using SVM classification error as the task function and HOG type visual features our novel controller is able to attain the desired pose. It is known that for classification task, variant of SVM like exemplar SVM [5] and kernel SVM perform better that linear SVM. However, in the context of visual servoing the performance also depends on the feasibility and smoothness of the end-effector trajectory. In this paper, we empirically discuss the performance of three types of SVM namely, linear SVM, kernel SVM and exemplar SVM for classification and control tasks. Furthermore, We report the average residual error in camera pose under 2.5 cm and 10^o for servoing across different object instances.

Contributions: Our main contribution is a learning based framework capable of servoing across object instances without the requirement of manual correspondences unlike our previous work [2]. This is achieved through a novel task function based on SVM classification error, which has not been proposed before for a servoing task. The paper also reveals visual features that efficiently encode the shape of the object to handle the large intra-category variations. Also, we represent the interaction matrix relating them to camera control analytically. Finally, the discussion at the end of this paper regarding the choice of SVM error function from the point of view of precision in the final pose, the convergence basin and existence of multiple local minima articulates the reasoning behind this effort.

II. Problem Formulation

Assuming eye-in-hand configuration and world origin coinciding with given object's center, we denote camera's pose in SE(3) at given time as P. Given an object O and a desired camera pose P^*, the goal of a visual servoing scheme is to find a camera transformation $\mathcal{T}$, such that $P^* = \mathcal{T}P$. For image based visual servoing (IBVS), current pose and the desired pose are represented in form of a set of features extracted from images $\mathbf{s} = \phi(KPO)$ and $\mathbf{s}^* = \phi(KP^*O)$. Where K is the camera's intrinsic matrix and $\phi(\cdot)$ is the feature selection criterion. For IBVS, the goal is modified as finding the transformation $\mathcal{T}$ such that the error in features $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$ is regulated to zero at desired pose. The task is achieved by minimizing $\mathbf{e}$ iteratively and controlling camera velocity, $\mathbf{v} = -\lambda \mathbf{L_s}^+\mathbf{e}$. Where $\mathbf{L_s}$ is the interaction matrix that maps the rate of change of features to velocity and $(\cdot)^+$ represents pseudoinverse operation as defined in [1].

For the problem of servoing across object instances, the given object instance O is different from the desired instance O^*. Thus, there is no pose P for which $\mathbf{s} = \mathbf{s}^*$. Hence, the transformation $\mathcal{T}$ between current pose and desired pose could not be computed by IBVS. The modified objective for visual servoing across object instances is to define an error $\acute{e}(\mathbf{s},\mathbf{s}^*)$ such that:

$$\acute{e}(\mathbf{s},\mathbf{s}^*) = 0 \Rightarrow P^* = \mathcal{T}P. \tag{1}$$

III. Overview of the Solution

We make following assumptions: Firstly, the given object is segmented and labeled. This assumption could be relaxed by using sophisticated computer vision techniques like poselets [6], part based models [7] etc. Secondly, we are provided with a pose-bank, which contains instances of the given category under varied viewpoints. Thirdly, the desired pose is seen previously atleast for a few instances.

The central idea of the proposed work is to learn an error function defined in (1) and move the end-effector in the direction that minimizes the error. From the given labeled dataset, all the images similar to the desired pose are treated as negative samples and rest of the images are treated as positive samples (Note the sign convention for labels). We propose principle orientation glyph (POG) which are modified from HOG as visual features and train an SVM on the above dataset. Now, the desired pose could be represented by SVM parameters (support vectors, their coefficients, kernel function and classification threshold) that would discriminate the desired pose from the rest of poses by a large margin. Note that training a SVM is one time procedure and until the desired pose remains unchanged, the procedure need not be repeated. Moreover, the SVMs could be pre-trained for numerous poses and stored. Once SVM is trained, the desired pose could be attained by iteratively controlling the manipulator until classification error becomes non-positive. Figure 2 summarizes the proposed approach.

IV. HOG AS A VISUAL FEATURE

Classical IBVS proposes visual features based on object's geometry [1] such as points, lines, area, etc. These features require a robust extraction and an efficient tracking procedure, which is a non-trivial task and is considered as one of the bottlenecks in the expansion of visual servoing [8]. To overcome this issue, direct visual servoing was proposed in [8] where pixel intensities were considered as visual features. But, pixel intensities are very susceptible to illumination variations. Amaury et al. [9] used intensity histogram as visual features to improve robustness to illumination changes. It was reported in [10] that by using intensity histogram as visual features, all 6 degrees-of-freedom (DOF) could not be controlled. Thus, multiple histogram were proposed in [10], where the image was divided into multiple non-overlapping regions and the feature vector was composed by concatenation of the histogram of individual regions. Another approach [11] used the magnitude of intensity gradient as visual feature to improve robustness to illumination variations. However, all the mentioned approaches do not capture the object's shape efficiently.

HOG features were introduced in [4] for pedestrian detection and has been extensively used by computer vision literature for shape based object recognition. The central idea behind HOG is that, by using local distribution of intensity gradient local shape of the object could be captured. The image is subdivided into small non-overlapping regions (cells) and for each cell, histogram of gradient orientation is computed. The resultant feature vector is further normalized over a larger area (blocks) to ensure robustness to illumination variations. A global feature vector is then constructed by combining the normalized histograms of individual cells, which represents the shape of the object. Eventually, SVM is used with HOG to learn the shape. Note that, the features presented in [4] combined the advantage of [10] and [11], since the histogram was computed for image gradient similar to [11] and concatenation of local histogram was used similar to [10].

A. *Adapting HOG features for visual servoing*

HOG features have complicated block normalization step which makes it difficult to represent them analytically. Such closed form representation is required for computing the interaction matrix. Furthermore, using a small cell size leads to smaller convergence basin. Since for large variations in camera pose, features frequently cross the cell boundaries that makes it difficult to track them.

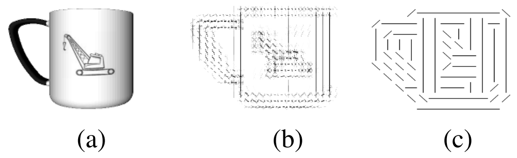

Fig. 3. **POG feature.** (a) The given image. (b) HOG glyph of the image (c) POG of the image. Notice how POG is able to preserve shape of the object.

In this paper we propose features that inherit the advantages of HOG features and could be used for visual servoing. Similar to HOG, we divide the image into small cells (typically 16×16 pixels), compute histogram of gradient orientations. However, we only retain the bin with maximum votes (principal orientation). We skip the block normalization step and construct the glyph image with only the principal orientation which we call principle orientation glyph (POG) image. Figure 3 shows the extracted POG image from the given image.

B. *Computation of interaction matrix*

The POG image $\mathbf{G}$ could be represented in the form of a feature vector by concatenating the individual rows

$$\mathbf{s}(\mathbf{r}) = (G_{1\bullet}, G_{2\bullet}, ..., G_{N\bullet}) \tag{2}$$

where $G_{k\bullet}$ denotes the k^{th} line of POG image. Size of the vector $\mathbf{s}$ is $NM \times 1$ where N and M are the number of rows and columns in $\mathbf{G}$. We assume that the given object is Lambertian i.e. the reflection of the given object is isotropic. Note that the mapping $\mathbf{I} \rightarrow \mathbf{G}$ is independent of the object therefore we can safely assume POG to be Lambertian. For a static object with Lambertian surface, temporal luminance consistency equation similar to [8] could be written as

$$\mathbf{G}(x + dx, y + dy, t + dt) = \mathbf{G}(x, y, t) \tag{3}$$

where (x, y) are the normalized coordinates of a point in the scene. Linearizing (3) around the point (x, y) gives

$$\frac{\partial \mathbf{G}}{\partial x}\dot{x} + \frac{\partial \mathbf{G}}{\partial y}\dot{y} + \dot{\mathbf{G}} = 0. \tag{4}$$

Substituting $\dot{x} = \mathbf{L}_x\mathbf{v}$ and $\dot{y} = \mathbf{L}_y\mathbf{v}$ in the above (4) gives

$$\dot{\mathbf{G}} = -(\nabla_x\mathbf{G}^T\mathbf{L}_x + \nabla_y\mathbf{G}^T\mathbf{L}_y)\mathbf{v}. \tag{5}$$

Comparing (5) with the definition of interaction matrix $\dot{\mathbf{G}} = \mathbf{L}_{\mathbf{G}}\mathbf{v}$ gives

$$\mathbf{L}_{\mathbf{G}} = -(\nabla_x\mathbf{G}^T\mathbf{L}_x + \nabla_y\mathbf{G}^T\mathbf{L}_y) \tag{6}$$

with $\mathbf{L}_x$ and $\mathbf{L}_y$, relating the change in point (x, y) and it's depth Z to the camera velocity as $\dot{x} = \mathbf{L}_x\mathbf{v}$ and $\dot{y} = \mathbf{L}_y\mathbf{v}$, are given by (refer [1] for details)

$$\begin{aligned} \mathbf{L}_x &= \quad [-1/Z \;\; 0 \;\; x/Z \;\; xy \;\; -(1+x^2) \;\; y] \\ \mathbf{L}_y &= \quad [0 \;\; -1/Z \;\; y/Z \;\; 1+y^2 \;\; -xy \;\; -x] \end{aligned} \tag{7}$$

Formulating POG as image allows numerical computation of gradients $\nabla_x\mathbf{G}$ and $\nabla_y\mathbf{G}$, without the need of their analytic computation. We compute $\nabla_x\mathbf{G}$ and $\nabla_y\mathbf{G}$ from POG image as

$$\begin{aligned} \nabla_x\mathbf{G}(x, y) &= \mathbf{G}(x + 1, y) - \mathbf{G}(x - 1, y) \\ \nabla_y\mathbf{G}(x, y) &= \mathbf{G}(x, y + 1) - \mathbf{G}(x, y - 1). \end{aligned} \tag{8}$$

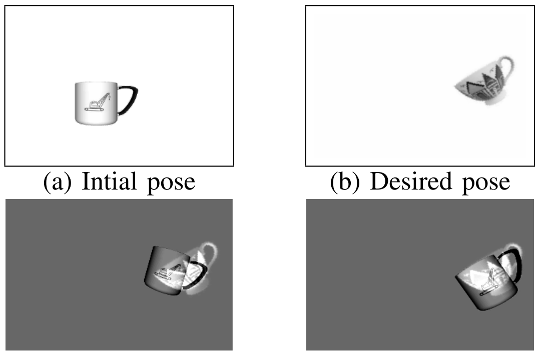

Fig. 4. **SVM based visual servoing vs direct visual servoing like [8].** The proposed task function of using SVM classification error results in precise alignment of the desired pose while the traditional feature error based positioning results in higher error when the instances are different as reflected in the final error images (c-d). The error image represents the difference in intensities between the desired and the current image with an offset of 128 such that gray color encodes zero error.

Classical visual servoing defines the positioning task as regularization of the error function $\|\mathbf{e}\| = \|\mathbf{s} - \mathbf{s}^*\|$. However, for servoing across instances there does not exist any $\mathbf{s}$ for which the error is regulated to zero, since the given object instance is different from the desired instance. Thus, for sevoing across object instances, $\|\mathbf{s} - \mathbf{s}^*\|$ does not attain accurate positioning. In this paper, we use SVM classification score $\acute{e}$ as the error function and the positioning task requires $\acute{e} \leq 0$. Positioning based on SVM based error $\acute{e}$ leads to more precise alignment as shown in figure 4.

C. Positioning task with linear SVM

Given a labeled dataset with data samples X_i's and their class labels y_i's where $X_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$, for all i in the dataset of size $\mathcal{N}$, a Support Vector Machine finds the optimal hyperplane separating the samples of one class from another. This is achieved by solving the following primal constraint convex optimization problem

$$(\mathbf{w}, b) = \underset{\mathbf{w}, b, \xi_i}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\| + C\sum_{i=1}^{\mathcal{N}} \xi_i$$
$$s.t. \quad y_i(\mathbf{w}^T X_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i \in 1,..,\mathcal{N}. \quad (9)$$

Where $\mathbf{w}$ is the required hyperplane, ξ_i's are the slack variables representing the violations made by the training samples and b is the threshold for classification. For testing phase, the class of the given data sample is predicted by finding the sign of $\mathbf{w}^T X + b$.

We initially train a linear SVM over POG features by assigning previously seen desired pose of various instances as negative samples (class label -1) i.e. $y_i = -1 \ \forall i \in \mathcal{S}^*$ and the remaining poses for various instances as positive samples (class label +1) i.e. $y_i = +1 \ \forall i \in \mathcal{S}$. Note that, to avoid confusion we stick to the above mentioned notation that desired poses belong to negative class $\mathcal{S}^*$ and the random poses belong to positive class $\mathcal{S}$ and the size of training dataset $\mathcal{S}^* \bigcup \mathcal{S}$ is given by $\mathcal{N}$. Now, the positioning task could be simply defined by the following error function

$$\acute{e} = \mathbf{w}^T\mathbf{s} + b \quad (10)$$

where $\mathbf{s}$ is the POG representing the current pose and $\mathbf{w}$, b are the parameters of the resultant hyperplane. Note that, $\acute{e} \leq 0$ for the desired pose and $\acute{e} > 0$ for all other poses. Also, note that (10) is independent of any specific object instance O^*.

D. Positioning task with kernel SVM

Pose of an object could be seen as a low dimensional feature lying on a higher dimensional manifold of the image which results in highly non-linear data. Linear SVM is not able to discriminate between such variations of a complex manifold. Kernel SVM finds an optimal hyper plane in a higher dimension where the data samples are linearly separable. The training step finds such an optimal hyperplane by optimizing following dual of (9)

$$[\alpha, b] = \underset{\alpha_i}{\operatorname{argmin}} \sum_{i=1}^{\mathcal{N}} \alpha_i - \frac{1}{2}\sum_{i=1,j=1}^{\mathcal{N}} \alpha_i\alpha_j y_i y_j \mathcal{K}(X_i, X_j)$$
$$s.t. \quad \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall i \in 1,..,\mathcal{N}. \quad (11)$$

Where the kernel function $\mathcal{K}(X_i, X_j) = \langle\psi(X_i), \psi(X_j)\rangle, \ \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ computes the inner product without explicitly computing the feature in higher dimension $\psi(X)$. The class label of any given sample X could be predicted by finding the sign of $\sum_{i=1}^{\mathcal{M}} \alpha_i y_i \mathcal{K}(X_i, X) + b$. $\mathcal{M}$ is the number of support-vectors and α_i and y_i are the coefficients and labels of the support vectors and b is the classification threshold. Hence, for kernel SVM the positioning task could be written as:

$$\acute{e} = \sum_{i=1}^{\mathcal{M}} \alpha_i y_i \mathcal{K}(X_i, \mathbf{s}) + b \quad (12)$$

where $\mathbf{s}$ is the POG representing the current pose. Similar to linear SVM, both position as well as classification tasks require $\acute{e} \leq 0$.

E. Positioning task with exemplar SVM

The linear SVM is not able to capture the variations of pose and kernel SVM overfits the training data especially in our case where number of positive samples is very small. Exemplar SVM was introduced by [5] for object classification. Although being a simple technique, the performance is comparable to state-of-art classification methods [7],[6]. The central idea behind exemplar SVM is using the ensemble of linear SVM's, where a linear SVM is trained for every negative sample (as per our convention). Every exemplar SVM fires for the most similar object. Thus, given sufficient number of negative examples even complex variations in a category could be learned. Authors of [5] used HOG features at multiple scale for the classification task, however for visual servoing scale variation results in different classes. Thus, we use our POG feature at single scale with exemplar SVM. Moreover, by fixing the scale we can assign positive examples explicitly thereby, waving the requirement for hard mining positive examples. Again, since the scale is not changed the ensemble function could be reduced to finding the minimum SVM score instead of non-maxima separation. However, switching hyperplane while servoing leads to jerky trajectory and stability issues (oscillation between exemplars). A more appropriate positioning would be to use a combination of exemplars. We use logistic regression to get the combination. This meets two purposes. Firstly, it results in a stable and smoother trajectory. Secondly, learning the sigmoid parameters on validation set gives higher weights to better performing SVMs, which in turn boosts the performance of the system. Sigmoid fitting is similar to the calibration step used by the authors in [5]. The positioning task could be written analytically as

$$\acute{e} = 1/(1+\exp(\sum_i \alpha_i(\mathbf{w}_i^T\mathbf{s}+b_i)+b_0)),\ \ \forall i \in \mathcal{S}^* \tag{13}$$

where the constant b_0 and the weights α_i for each exemplar SVM are learned by fitting logistic regression to the raw SVM scores.

F. Control law

Traditional visual servoing uses following law for controlling camera velocity $\mathbf{v}$:

$$\mathbf{v} = -\lambda\mathbf{L}_\mathbf{s}^{+}(\mathbf{s}(\mathbf{r})-\mathbf{s}^*) \tag{14}$$

$\mathbf{L_s}$ is the interaction matrix, $\mathbf{s}$ is current feature vector at position $\mathbf{r}$ and $\mathbf{s}^*$ is the desired feature vector. As described in [11], this control law is equivalent to applying Gauss-Newton optimization on a cost function:

$$\mathcal{C} = (\mathbf{s}(\mathbf{r})-\mathbf{s}^*)^T(\mathbf{s}(\mathbf{r})-\mathbf{s}^*) \tag{15}$$

The above control law considers the direction of descent, $\mathbf{d}$ as:

$$\mathbf{d}(\mathbf{r}) = -(\mathbf{L}_\mathbf{s}^T\mathbf{L_s})^{-1}\nabla\mathcal{C} \tag{16}$$

Where $\mathbf{L_s}$ is the image Jacobian at $\mathbf{r}$. As the size of feature vector increases, the non-linearity of the error surface increases. Direct visual servoing and similar approaches consider entire image as a feature vector that makes the error surface highly non-linear. Thus, the convergence properties becomes very sensitive to the direction of descent. Therefore, such approaches consider Levenberg-Marquardt optimization method, which gives the flexibility to select an optimum direction of descent by adjusting a damping factor μ,

$$\mathbf{v} = -(\mathbf{H}+\mu\,\mathrm{diag}(\mathbf{H}))^{-1}\mathbf{L}_\mathbf{s}^T\nabla\mathcal{C} \tag{17}$$

where the Hessian is approximated as $\mathbf{H} = \mathbf{L}_\mathbf{s}^T\mathbf{L_s}$. As the cost function given by SVM based positioning task is again highly non-linear, we also consider the servoing law similar to (17) with interaction matrix for POG features is given as $\mathbf{L_s} = \mathbf{L}_G$ and the cost function $\mathcal{C}$ in our case is given by (10), (12) and (13). The gradient of the cost function required in (17) for linear, kernel and exemplar SVM's could be given as

$$\nabla\mathcal{C}_{\text{linear}} = \mathbf{w} \tag{18}$$

$$\nabla\mathcal{C}_{\text{kernel}} = \sum_{i=1}^{\mathcal{M}}\alpha_i y_i\nabla\mathcal{K}(X_i,s) \tag{19}$$

$$\nabla\mathcal{C}_{\text{exemplar}} = (\exp(f)/(1+\exp(f))^2)\sum_i\alpha_i\mathbf{w}_i^T \tag{20}$$

$$f = \sum_i\alpha_i(\mathbf{w}_i^T\mathbf{s}+b_i)+b_0 \tag{21}$$

where $\mathbf{w}$ is the SVM hyperplane. In the proposed work, we have used radial basis function (RBF) kernel due to its high expressive power. The kernel function and its gradient for RBF could be given as

$$\mathcal{K}(X_i,X_j) = \exp(-\gamma\|X_i-X_j\|) \tag{22}$$

$$\nabla\mathcal{K}(X_i,\mathbf{s}) = -\gamma\mathcal{K}(X_i,\mathbf{s})(\mathbf{s}-X_i) \tag{23}$$

G. Stability analysis

Previous approaches considered $\mathcal{L} = 1/2\|\mathbf{e}\|^2$ as a candidate Lyapunov function. Whose, derivative could be given as $\dot{\mathcal{L}} = \mathbf{e}^T\dot{\mathbf{e}} = -\lambda\mathbf{e}^T\mathbf{L_e}\hat{\mathbf{L}}_\mathbf{e}^{+}\mathbf{e}$. Where $\hat{\mathbf{L}}_\mathbf{e}$ is an approximation of the interaction matrix pertaining to modeling and calibration errors. For global stability, $\mathbf{L_e}\hat{\mathbf{L}}_\mathbf{e}^{+}$ must be positive definite. If the number of features is greater than 6, $\mathbf{L_e}\hat{\mathbf{L}}_\mathbf{e}^{+}$ has a non-trivial null space. Thus, only local stability near the equilibrium point $\mathbf{s}^*$ could be guaranteed (refer [1] for details).

In our case there is no unique equilibrium point. All the points satisfying $\acute{e} \leq 0$ are solutions. It is sufficient to analyze the stability of points lying on the boundary of the solution region instead of analyzing the stability for every solution since, the servoing stops as soon as the stability region is reached. In our case, boundary of the solution region is the SVM hyperplane. We consider a Lyapunov candidate function

$$\mathcal{L} = 1/2\acute{e}^2. \tag{24}$$

Note that $\acute{e}$ is scalar for linear, kernel and exmplar SVM. The derivative of the above mentioned Lyapunov candidate is given as

$$\dot{\mathcal{L}} = \acute{e}^T\dot{\acute{e}} \tag{25}$$

For linear SVM differentiating (10) gives

$$\begin{aligned}\dot{\acute{e}} &= \mathbf{w}^T\dot{s} = \mathbf{w}^T\mathbf{L_s}\mathbf{v}\\ \mathbf{v} &= -\lambda(\hat{\mathbf{H}}+\mu\,\mathrm{diag}(\hat{\mathbf{H}}))^{-1}\hat{\mathbf{L}}_\mathbf{s}^T\mathbf{w}\\ \Rightarrow \acute{e}^T\dot{\acute{e}} &= -\lambda\acute{e}\mathbf{w}^T\mathbf{L_s}(\hat{\mathbf{H}}+\mu\,\mathrm{diag}(\hat{\mathbf{H}})^{-1})\hat{\mathbf{L}}_\mathbf{s}^T\mathbf{w}.\end{aligned} \tag{26}$$

Similarly, for kernel SVM

$$\begin{aligned}\acute{e}^T\dot{\acute{e}} &= -\lambda\acute{e}\mathbf{g}\mathbf{L_s}(\hat{\mathbf{H}}+\mu\,\mathrm{diag}(\hat{\mathbf{H}}))^{-1}\hat{\mathbf{L}}_\mathbf{s}^T\mathbf{g}\\ \mathbf{g} &= (\sum_i\alpha_i y_i\nabla\mathcal{K}(\mathbf{s},X_i)).\end{aligned} \tag{27}$$

And for exemplar SVM

$$\acute{e}^T\dot{\acute{e}} = -\lambda\acute{e}^3 exp(f)\nabla f\mathbf{w}_i\mathbf{L_s}(\hat{\mathbf{H}}+\mu\mathrm{diag}(\hat{\mathbf{H}}))^{-1}\hat{\mathbf{L}}_\mathbf{s}^T\nabla f. \tag{28}$$

Where f is given by (21), $\nabla f = \sum\alpha_i\mathbf{w}_i$, $\mu > 0$ and $\hat{\mathbf{H}} = \hat{\mathbf{L}}_\mathbf{s}\hat{\mathbf{L}}_\mathbf{s}^T$. Note that we are analyzing the perturbations from SVM hyperplane boundary with $\acute{e} > 0$, since for $\acute{e} < 0$ we are already in the solution region hence, the servoing algorithm terminates. For the control to be locally stable $\acute{e}\dot{\acute{e}}$ must be negative definite (the number of features is greater than 6, thus only local stability could be achieved). As mentioned $\acute{e} > 0$, and by definition $f > 0$, $\mu > 0$. Thus, for $\acute{e}\dot{\acute{e}}$ to become negative definite equations (26 - 28) require $\mathbf{L_s}(\hat{\mathbf{H}}+\mu\,\mathrm{diag}(\hat{\mathbf{H}}))^{-1}\hat{\mathbf{L}}_\mathbf{s}^T$ should be positive definite. Since, $\mathbf{H}$ has a unique Cholesky decomposition $\mathbf{H} = \hat{\mathbf{L}}_\mathbf{s}\hat{\mathbf{L}}_\mathbf{s}^T$, it is by definition positive semi-definite. Further, adding the positive diagonal elements of $\mu\,\mathrm{diag}(\hat{\mathbf{H}})^{-1}$ ensures that $(\hat{\mathbf{H}} + \mu\,\mathrm{diag}(\hat{\mathbf{H}}))^{-1}$ is positive definite and could be written as $\mathbf{M}^T\mathbf{M}$ by Cholesky decomposition giving

$$\acute{e}^T\dot{\acute{e}} = (\mathbf{M}\mathbf{L}_\mathbf{s})^T(\mathbf{M}\hat{\mathbf{L}}_\mathbf{s}). \tag{29}$$

Thus, $\acute{e}^T\dot{\acute{e}}$ will be positive definite in the neighborhood of SVM hyperplane when $\hat{\mathbf{L}}_\mathbf{s}$ is a good approximation of $\mathbf{L}_\mathbf{s}$.

V. Experiments and Results

We have performed various experiments to validate the efficacy of our approach. All the experiments reported here are performed in simulation on computer with Intel i5 processor and 4 GB RAM using ViSP library [12] for servoing and OpenRAVE library [13] for rendering 3D model of object in various viewpoints. The experimental setup consists of a 3D model that acts the given unseen object instance, a free-flying camera and a labeled pose bank containing various instances discriminating desired pose from other poses. We have assumed a constant depth of the scene as 1m for all experiments. We further assume that the object category as well as the image representing desired pose of different instance are known and the object is segmented. Since, the desired instance is different from the given instance, even at zero camera error feature will not overlap. Hence, feature error is not the best metric to analyze performance when servoing across different instances. The desired camera pose is one of the metric that is invariant to geometry of the object. However, knowledge of desired camera pose in real world is a non-trivial task. Therefore, we use the simulation framework that allows us to quantitatively measure the performance of our approach since the desired camera pose is known in the world frame.

1) Using POG as visual features: In this experiment we validate the proposed control law for POG features. We consider a 6 DoF positioning task for a non-planar object. Here, the given object instance is same as the desired instance, thus the positioning task is given by (1). The initial camera displacement is $\Delta r = (5cm, 2cm, 5cm, 12.5^o, -8.4^o, -15.5^o)$ (6 DOF). Figure 6 shows that the proposed control law converges successfully to the desired pose. It could also be seen that the feature error tends to zero at the desired pose. Also, it could be observed from figure 6(f) that the approach results in zero terminal velocity. However, the velocity profile is not smooth since POG features are discrete (POG features remain constant within a cell) thereby giving non-smooth decrease in velocity unlike continuous luminance features.

2) Comparison of SVM control laws: The aim of the experiment is to validate the control laws for linear, kernel and exemplar SVMs and to compare their performance for a positioning task. For this experiment, we consider 21 3D models as previously seen instances (training phase) and one as the given object (test phase). These models are pre-rendered into 5000 poses capturing variations in rotation and translation.Images similar to the desired pose (see figure7(b)) are used as negative samples and rest of the data is used as positive samples for training SVMs. Having such large

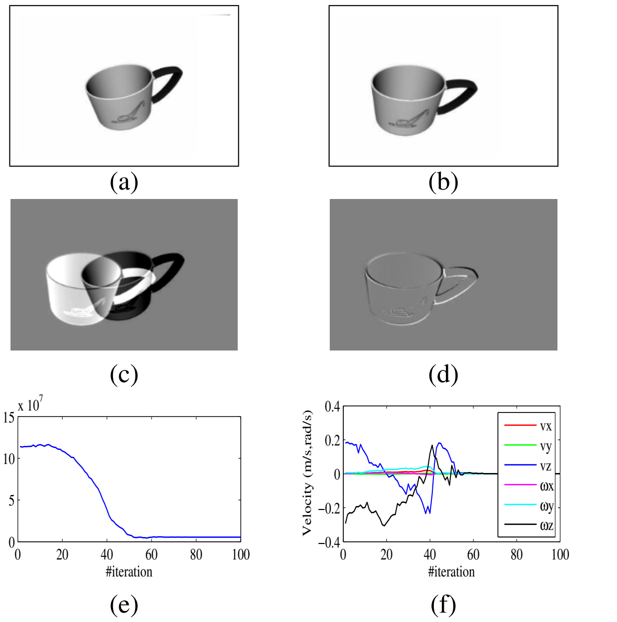

Fig. 6. **Visual servoing using POG features.** (a) A non-planar target in initial pose. (b) Desired pose of same object instance. (c) Error in POG features (pixel intensities) between images of initial pose and the desired pose. (d) Residual error in image between resultant and desired pose after visual servoing. (e) Feature error (f) Camera velocity.

imbalance between negative and positive data could easily overfit the resultant hyperplane. Thus, we perform a cross-validation step on three 3D models which were not used for either training or testing. After training the SVM, the classification error is used to control the camera motion. It could be seen from figure 7(c-e) that linear, kernel and exemplar SVM's are able to attain the desired pose. As the solution is not unique it is difficult to compare the resultant pose. It could be seen from figure 7(g), that the classification error becomes zero for all three SVMs at the desired pose, that validates the positioning task. Also, it could be seen that the decrease in the error is not strict. This is because sevoing happens in a 6 dimension embedding of 66000 dimensional POG feature manifold. Thus, a direct trajectory decreasing error monotonically could be infeasible. The convergence rate of the kernel SVM largely depends upon the kernel scaling factor γ. For a small γ, the slope is steep near a support vector and flat otherwise. Thus, for case of overfitting, the convergence basin for kernel SVM is very small. In terms of evolution of classification error and velocity profile, linear SVM is slower but less susceptible to initialization. The exemplar SVM inherits the advantages of both linear and kernel SVM, is thus performing better in terms of error and velocity profile.

3) Qualitative results on real images: The objective of this experiment is to show the efficacy of the proposed algorithm to servo to a diverse set of target instances across various object categories. We have trained an exemplar SVM for every desired pose given in figure 5(b) using synthetic data. The initial pose images (real images) are shown in figure 5(a) are servoed to the desired pose using the trained

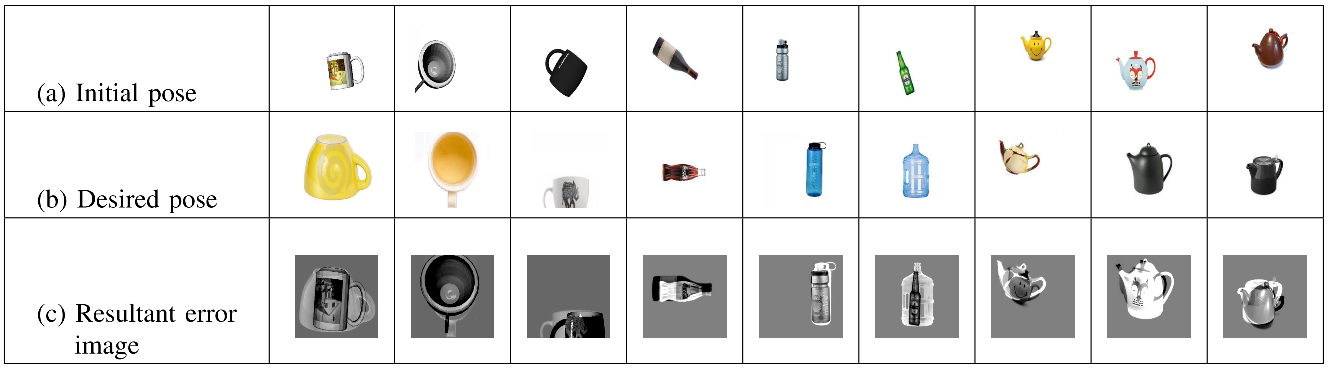

Fig. 5. **Qualitative results.** (a) Initial pose captured by the robot with a random camera pose for provided 3D object from the dataset. (b) Desired pose. (c) Resultant positioning compared to the desired pose by applying kernel SVM based visual servoing. Note the similarity in the resultant pose achieved by the proposed approach compared to the desired pose provided, over wide range of desired poses. Also note that, the selected 3D model in (a) acts as unseen target object.

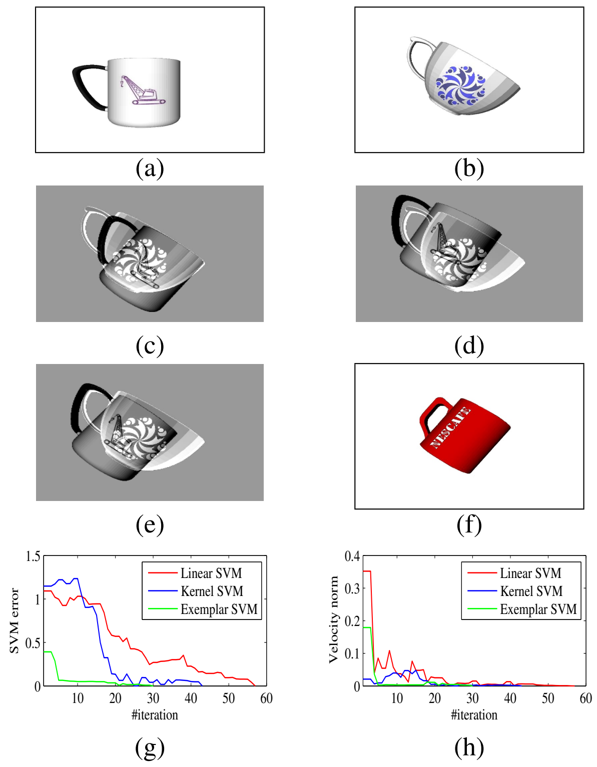

Fig. 7. **Comparison of SVM control laws.** (a) Initial non-planar scene. (b) Desired pose object instance different from the given object in initial pose. (c) Error image for resultant pose using linear SVM. (d) Error image for resultant pose using kernel SVM. (e) Error image for resultant pose using exemplar SVM. (f) The selected exemplar with minimum regression score. (g) SVM error comparing the SVMs. (h) Velocity profiles for SVMs. Note that, exemplar SVM outperforms the other SVMs since it inherits advantages of both linear as well as kernel SVM.

SVMs although neither of them is part of training data. The large overlapping between shapes in figure 5(c) shows the efficacy of the proposed algorithm for servoing over diverse set of data by efficiently encoding the desired pose in form of a classifier.

4) Quantitative results: The aim of the experiment is to ensure that, regulating the SVM classification error indeed results in precise positioning of camera. Since, have used

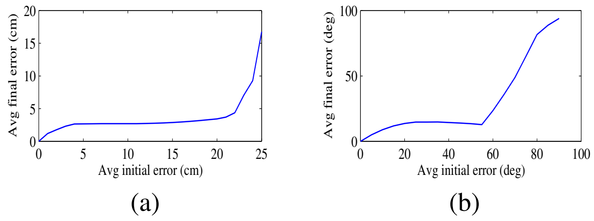

Fig. 8. **Quantitative results.** Initial displacement in translation and rotation of camera is plotted against residual camera error. It could be seen the approach converges for a high variation camera pose.

simulation environment, ground truth regarding the camera pose could be computed efficiently. Here, considering *teacup* as object category and starting from a desired pose, we varied the initial camera displacement (both translation and rotational) and servoed to the desired pose. The initial camera translation was varied till the object remained in field of view of camera. We have used the exemplar SVM trained from the previous section to compute the residual translation and rotation errors. It could be seen from figure 8(a) and (b) that for a fair amount of initial displacement (until more than half of the features remain in camera's field of view) the proposed approach is able to converge to the desired camera pose accurately. The experiment was performed on 10 different desired poses and 5 instances per view, thereby averaging the result before plotting. The average final error in camera pose is under 2.5 cm for translation and 10^o for rotation. The translation error is comparable with [2] but the rotational error is slightly higher. However, a direct comparison with [2] is not fair because we do not assume 3D keypoint annotations.

VI. Discussion

A. Computational complexity

SVM training complexities for linear and kernel SVM's vary between $\mathcal{O}(\mathcal{N}^2)$ and $\mathcal{O}(\mathcal{N}^3)$depending on the complexity of dataset when trained using LIBSVM library [14] with the dataset of size $\mathcal{N}$. Our implementation of exemplar SVM learns a linear SVM for each negative example along with a logistic regression for their ensemble hence, the combined training complexity is $\mathcal{O}(\mathcal{N}^2\mathcal{N}^- + \mathcal{N}^2)$. Where $\mathcal{N}^-$ is the size of negative training data. Note that, in our case the negative samples similar to the desired pose are very small as compared to the size of entire dataset i.e. $\mathcal{N}^- \ll \mathcal{N}$. Further, the SVMs could be pre-trained and saved bypassing the requirement for retraining before servoing. The average training time for learning linear and kernel was around 56 to 60 sec.

At the time of servoing, the classification error is computed for every iteration. For a linear SVM this can be computed by simply a vector multiplication of constant order $\mathcal{O}(1)$. Similarly, for exemplar SVM the complexity is $\mathcal{O}(\mathcal{N}^-)$ and that for combining the score through trained regression curve is of order $\mathcal{O}(1)$. However, for computing the kernel SVM error, the kernel function needs to be evaluated over all support vectors resulting in the complexity $\mathcal{O}(\mathcal{M})$. Thus, learning based approach is much faster compared to [2] which requires a search over entire dataset for linear combination for every iteration resulting complexity of order $\mathcal{O}(\mathcal{N})$ per iteration. Average time for visual servoing with proposed approach is 40 sec compared to 300 sec. Note that, the specified computation times are benchmarked on system described in section V.

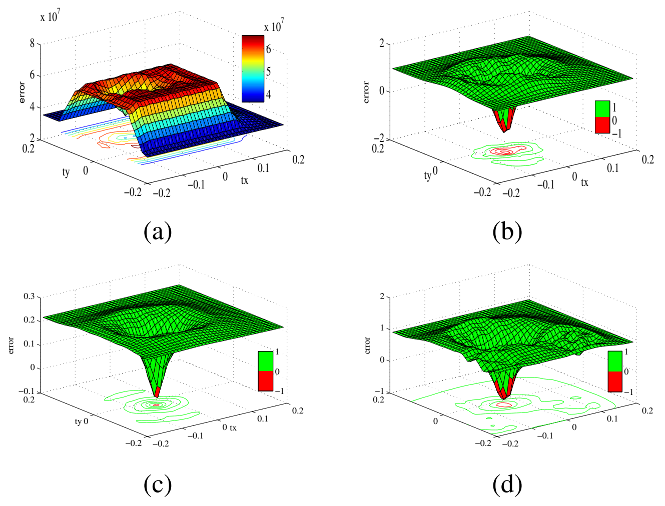

Fig. 9. **Analysis of cost function for different SVMs.** The desired pose for which cost function was analyzed is given in figure 7(a). (a) Using photometric visual servoing for servoing across instances results in irregular and unsuitable error. (b) Linear SVM (c) RBF kernel SVM. (d) Exemplar SVM with logistic regression. Note the innermost red contours, which determine the precision of an SVM (smaller contour means more precise). Also note the outermost contour which determines the convergence basin (Lager contour means better convergence). Finally, note the isolated counters that represent local-minimas.

B. Cost function analysis for SVMs

In this work, we have presented three different SVMs. In the current section we try to address the issue is which SVM is most suitable in visual servoing context. We have plotted the cost function (classification/feature error) by varying translation in x and y direction for desired pose given by figure 7(a). The small red contour represents the solution region (smaller means better positioning), the area of outermost region relates the convergence basin, steepness of curve represents rate of convergence and isolated contours show existence of local minima. Based on the mentioned parameters it could be observed from figure 9(a) that the feature error function e is very irregular suffering with lots of local minima, which makes it unsuitable for servoing across instances. For the linear SVM 9(b), the convergence basin is wide and however rate of convergence is slow and positioning error is high due to its linear nature. On the other hand, kernel SVM (using sigmoid kernel) is a very good classifier but due to its non-linear nature the convergence basin is steep and narrow(refer figure 9(c)). For exemplar SVM's, so that the control is linear and the selected SVM fires only for the nearest exemplar. The convergence basin is wide and the positioning is more precise are shown by the innermost red contour in figure 9(d).

VII. Conclusion

In this work, we have introduced a novel positioning task based on discriminative learning that could be used to servo across instances for the given object category. Also, we have introduced POG features that efficiently capture the shape of a given object under large appearance changes. Through simulation framework we have shown that the method can effectively servo across object instances even under high appearance and shape variations in the object category. The video material for the paper is available at the project webpage[1].

References

[1] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," in *RAM*, 2006.

[2] H. Pandya, K. M. Krishna, and C. V. Jawahar, "Servoing across object instances: Visual servoing for object category," in *ICRA*, 2015.

[3] H. T. S. Asa Ben-Hur, David Horn and V. Vapnik, "Support vector clustering," in *JMLR*, 2002.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[5] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV*, 2011.

[6] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *ICCV*, 2009.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," in *PAMI*, 2010.

[8] C. Collewet and É. Marchand, "Photometric visual servoing," in *TRO*, 2011.

[9] A. Dame and É. Marchand, "Mutual information-based visual servoing," in *TRO*, 2011.

[10] Q. Bateux and É. Marchand, "Direct visual servoing based on multiple intensity histograms," in *ICRA*, 2015.

[11] É. Marchand and C. Collewet, "Using image gradient as a visual feature for visual servoing," in *IROS*, 2010.

[12] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," in *RAM*, 2005.

[13] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Robotics Institute, Tech. Rep., 2008.

[14] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," in *TIST*, 2011.

[1]http://robotics.iiit.ac.in/people/harit.pandya/discriminative_learning_visual_servoing