Servoing Across Object Instances: Visual Servoing for Object Category

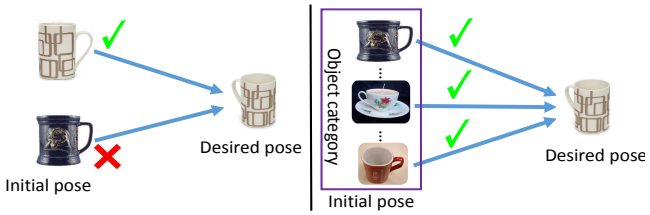
Harit Pandya[1], K. Madhava Krishna[1] and C. V. Jawahar[1]

***Abstract*— Traditional visual servoing is able to navigate a robotic system between two views of the same object. However, it is not designed to servo between views of different objects. In this paper, we consider a novel problem of servoing any instance (exemplar) of an object category to a desired pose (view) and propose a strategy to accomplish the task. We use features that semantically encode the locations of object parts and define the servoing error as the difference between positions of corresponding parts in the image space. Our controller is based on the linear combination of 3D models, such that the resulting model interpolates between the given and desired instances. We conducted our experiments on five different object categories in simulation framework and show that our approach achieves the desired pose with smooth trajectory. Furthermore, we show the performance gain achieved by using a linear combination of models (instances) vis a vis a controller that switches across models during servoing in terms of trajectory's length, smoothness and error in camera pose and image features.**

I. INTRODUCTION

Visual servoing (VS) framework guides a robot to acquire a desired pose with respect to a target object using vision based feedback [1]. In visual servoing, the camera pose is represented through a set of visual features and the objective is to minimize the servoing error, which is defined as the difference between the current values and the desired values of the visual features [2]. It is achieved by controlling manipulator velocity such that the servoing error is eventually reduced to zero. Traditional visual servoing assumes that object instance is known apriori. This limits the scope of visual servoing to situations where the robot servos to the same object. However, practical scenarios require the robot to interact with a variety of objects. Thus, a different strategy is required which could be employed to servo between views (poses) of different objects.

Analogous to the discussion on comparison between traditional visual servoing and servoing across instances, is that on comparison between object (image) instance retrieval and category retrieval. These are considered one of the classical problems in computer vision and both of them pose well defined problems. Instance based retrieval seeks a solution to the problem where a particular object is required, for example landmark identification [3], image or video search engine [4]. As the instance is unique, geometrical properties could be used for its retrieval. On the other hand, category retrieval considers the problem of retrieving the group of objects or identifying the group-label for objects with the same semantics, such as scene parsing [5], which tries to understand the given environment by classifying it into pre-trained categories. Instance based retrieval is considered to be faster and more precise due to the availability of fixed geometry, whereas category retrieval is scalable i.e. it could be applied to a wide range of objects. Scalability makes the retrieval problem even harder due to variations in appearances and shapes of the objects in a category. In a robotics setting, this is indeed very pertinent, as robots are entailed to interact across widely varying object instances. In such settings, unlike the traditional servoing where the feature error becomes zero for the desired pose, it is difficult to ascertain if indeed the correct pose has been reached for category based servoing. The problem complicates further as in category based servoing, all visual features might not be present in a given instance.



(a) Traditional VS (b) Ours

Fig. 1. **Comparing Traditional visual servoing with proposed approach.** Traditional visual servoing could only servo between views of the same object instance (same *cup*). On the other hand, the proposed approach is able to servo any instance from an object category category (any *cup*) to a desired pose.

In this paper, we introduce a novel problem of visual servoing any instance of an object category to a desired pose which could accomplish the task and as a result finds more diverse applications. Our approach uses a linear combination of 3D models to solve the task. In figure 1 we compare our approach with traditional visual servoing. It shows that the traditional visual servoing is capable of servoing between views of the same cup only. On the other hand, the proposed approach could servo between views of different cups. To remove the dependency on a specific instance and make the approach adaptable to variations in a category, we employ visual features which semantically encode locations of object parts. We refer them as part-based semantics. Such features also provide more tolerance to image noise under the duress of illumination variations and enhance alignment of visual features. Defining a suitable distance between locations of corresponding parts in images makes the comparison among instances over variation in poses feasible. We exploit a large number of 3D models to construct temporary models which are semantically

[1] International Institute of Information Technology, Hyderabad - 500032, India

(a) Initial pose (b) Desired pose (c) After servoing

Fig. 2. **Example of servoing using proposed approach.** Our approach is able to servo the given object from its initial pose (a) to a resulting pose (c) which is similar to the desired pose (b) given by another instance of same category .

most similar to both the given object instance and the desired instance at a given point of time. We propose two approaches to synthesize such temporary models. Our first approach selects the most suitable model at a point of time from the available 3D models, whereas the second approach constructs the temporary model through a linear combination of available 3D models. In either case, the resulting models are used to servo till pose of the given object matches the desired pose. Both of the approaches are capable of servoing across instances, while the latter one produces a smooth trajectory. Figure 2 shows an example of result achieved by our second approach. The approach is successfully able to attain the desired pose although the instances are different.

The paper contributes in the following ways. Firstly, it introduces a novel problem of servoing any instance of an object category to a given desired pose. Secondly, it proposes a servoing strategy for servoing across instances of a given category through, a controller design that effectively interpolates across semantic correspondences of various instances, thereby providing continuous servoing trajectories, even as the current and target instances widely vary. To the best of our knowledge, there has not been such a reporting previously in literature.

II. Preliminaries

Hutchinson et al. [1] define the positioning task as representative of feature error $e = s - s^*$ such that it reduces exponentially to $e = 0$ at the desired pose. Where s is the representation of current end-effector pose along with camera parameters and s^* represents the desired pose. For point features, proportionate control law could be used to produce desired end-effector velocity $V_c = -\lambda L^+ e$. Where, λ is the servoing gain, operator $(.)^+$ stands for pseudo-inverse and L is the interaction matrix or image Jacobian, which maps feature space to camera velocity in $SE(3)$. Interaction matrix L_s for image based visual servoing (IBVS) using point features [2] could be given as follows:

$$L_s = \begin{bmatrix} \frac{-1}{z} & 0 & \frac{x}{z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{z} & \frac{y}{z} & 1+y^2 & -xy & -x \end{bmatrix} \quad (1)$$

Where, x, y are the image coordinates of a point feature and z is its depth in camera's frame. Overall interaction matrix for multiple features could be made by vertically stacking the individual interaction matrices for each feature point.

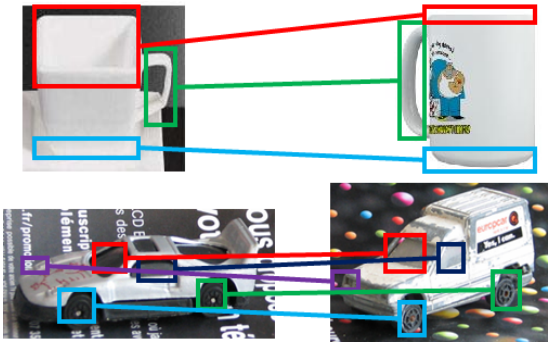


Fig. 3. **Part based semantic correspondences.** Examples of correspondences for categories *cup* (top) and *toycar* (bottom). These correspondences match the respective parts (for example handles, wheels etc.) among object instances.

Variation in different instances could be quantified after evaluating one-to-one correspondences of shared part based semantic features, followed by finding distance between them in Euclidean space. For example, figure 3 shows a valid comparison between two cups after associating visible semantic features such as handle, base etc. to their counterparts. In the proposed work, we define distance between two instances O_i and O_j, as the sum of individual Euclidean distances of semantic correspondences similar to [6], $\mathcal{D}(O_i, O_j) = \sum_k \|(O_{ik} - O_{jk})\| \psi(O_{ik}, O_{jk}) \quad \forall k \in \mathcal{K}$. Where, $\mathcal{K}$ is the set of all part based semantic features and the visibility function ψ is defined as

$$\psi(x_i, x_j) = \begin{cases} 0 & \text{if keypoint } x \text{ is visible in either} \\ & \text{image i or j but not in both} \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

Similarly, for visual features i.e. part based semantic features in image space written as a column vector s, semantic distance is indicated by, $\mathcal{D}(s_i, s_j) = \|s_i - s_j\| \psi(s_i, s_j)$. For simplicity, we do not show ψ in calculations, however we compute semantic distance only for shared features.

III. Problem Formulation and Solution

We consider a world frame of reference with origin $\mathcal{F}_o$. We also define a keypoint in Euclidean coordinate that semantically characterizes a part of an object instance. Then, any object instance O could be considered as a set of 3D keypoints P in world frame such that $O = [P_1 \;\; P_2 \;\; ... \;\; P_K]^T$ and $P_i = [X_i \;\; Y_i \;\; Z_i], \quad \forall i \in \mathcal{K}$. Where, $\mathcal{K}$ denotes set of all keypoints for a category. Visual feature s could then be produced by concatenating the image coordinates resulting from projection of the object instance into a pin-hole camera, $s = [p_1 \;\; p_2 \;\; ... \;\; p_K]^T$. Where, $p_i = [x_i \;\; y_i]^T = \mathcal{P} P_i$ and $\mathcal{P}$ is the camera projection matrix. The servoing task requires moving current camera frame attached to the manipulator, $\mathcal{F}$ to a frame $\mathcal{F}^*$ such that, eventually $s = s^*$. Since target object and desired object are different instances, there might not exist a transformation T such that $\mathcal{F}^* = T\mathcal{F}$. The

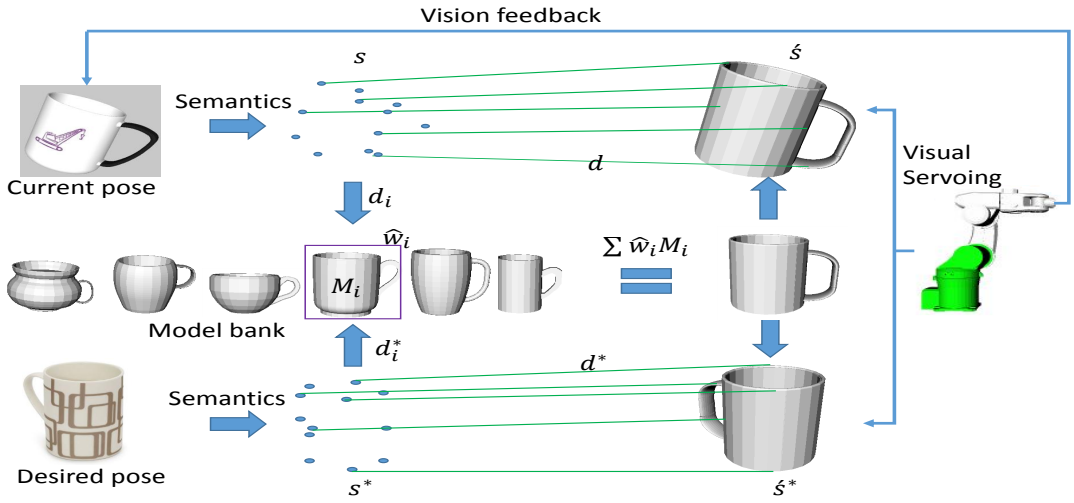


Fig. 4. **Overview of the proposed approach.** Current pose of the given object viewed by manipulator's camera (top-left) and desired pose provided by the user (bottom-left). Both current pose and desired pose are transformed into their respective semantic representations s and s^*. A weight $w_i = d_i + d_i^*$ is given to each model M_i, such that the model most similar to both current and desired instance receives maximum weight. Here, d_i is semantic distance between M_i and s, similarly d_i^* is semantic distance between M_i and s^*. A model is generated from linear combination $M = \sum w_i M_i$, which is employed for servoing between pose $\acute{s}$ and $\acute{s}^*$ such that $d = min(\|s - \acute{s}\|)$ for all $\acute{s}$ and $d^* = min(\|s^* - \acute{s}^*\|)$ for all $\acute{s}^*$. Entire process is repeated till convergence.

primary challenge is to approximate T through a series of transformations $\prod_i \widehat{T}_i$, i.e. $\mathcal{F}^* \approx \prod_i \widehat{T}_i \mathcal{F}$. Where, every transformation $\widehat{T}_i$ is a result of a visual servoing step.

A. Overview of the solution

We assume that both the current[1] and the desired pose of the target object are given in the form of images. However, both current and desired pose could represent different instances as shown in figure 4. Since the instances are not same, therefore geometrical features could not be used for comparisons. Thus, we employ part based semantics. We further assume that given an image of the object, both its category as well as semantic features could be retrieved in the form of keypoints. For which, we rely on state-of-the-art computer vision algorithms like poselets [6], part based models [7] etc. We also assume that the object category has been previously seen by the robot and large number of models $M_i \quad \forall i \in N$ are available in form of 3D semantic keypoints.

As there might not exist a T, such that $s^* = T\mathcal{P}O$. To address this issue, at every servoing step, we approximate $\widehat{T}_i \approx \Delta T$ by constructing temporary models and proceeding a servoing step with such models. The temporary models approximate both, the given instance as well as the desired instance. We present two approaches to construct such temporary models. Our first approach selects the best suitable model from the available models, while the second approach constructs them through a linear combination of available models $M = \sum_i w_i M_i$ as shown in figure 4. The primary advantages of using the latter approach are: (i) a linear combination of 3D models helps generate large dataset of models from small number of given instances; (ii) a smooth transition could be achieved when servoing across instances.

The resultant model M constructed using either of the approaches interpolates between the given instance O and the desired instance O^*. The current pose of the object $s = \mathcal{P}O$ and the desired pose s^* could now be approximated by our temporary model as $\acute{s} = \mathcal{P}M$ and $\acute{s}^* = \mathcal{P}^*M$ respectively. Where, the camera matrix $\mathcal{P}$ minimizes the distance $d = \min(\|\mathcal{P}M - s\|)$ and $\mathcal{P}^*$ minimizes $d^* = \min(\|\mathcal{P}^*M - s^*\|)$ as shown in figure 4. Since, s and s^* represent same instance M, the problem reduces to visual servoing over same instance, which could be managed by classical IBVS iteration. The entire process is repeated with the updated image after servoing step, till convergence. However, computing $\mathcal{P}$ and $\mathcal{P}^*$ at every servoing step could be very inefficient. Hence, to improve upon the efficiency we compute the linear combination in image space, i.e. $\acute{s} = \sum_i w_i s_i$ and $\acute{s}^* = \sum_i w_i s_i^*$. Where, s_i and s_i^* minimize $d_i = \min(\|s - s_i\|)$ and $d_i^* = \min(\|s^* - s_i^*\|)$. However, the image produced by a simple linear combination of two images needs not be an interpolated image, as it usually contains ghost contours. But, if the linear combination is taken after aligning relevant semantic features with their corresponding counterparts, then satisfactory results could be achieved [8]. In our case, keypoints are nothing but the semantic features. Hence, they could be used safely for computing the linear combination.

[1]Note that, throughout this paper we use current to denote the given point of time and instance to denote an exemplar of a category.

B. Algorithm

As described in previous section, servoing is feasible from s to s^* by using semantics, even though they represent different instances. However, visual servoing in its traditional form could not be applied directly when instances are different. We propose two approaches to cater for this issue. The first approach (Algorithm 1) switches between models by selecting the most suitable model for a servoing step. While, the second approach (Algorithm 2) generates a new model by a linear combination of available models for each servoing step. At every servoing step, Algorithm 1 finds the best matching poses s_i and s_i^* for every model, to current and desired pose s and s^* respectively. Most suitable model is then found which is most similar to both, the current pose and the desired pose i.e. with minimum $d_i + d_i^*$. This model is then used for a servoing step from s_i towards s_i^*. Note that, visual servoing becomes feasible as the instance remains same for the servoing step. The algorithm converges when the residual error for the best model $\|s_j - s_j^*\|$ is within a certain threshold (δ). Also, s_i and s_i^* are found by searching for every pose given by model M_i, i.e. $q_i \in \mathcal{P}M_i$. To decrease the computations, for each model M_i various poses q_i are precomputed offline by rotating and scaling the model. These poses are stored in a pose bank for future use, hence finding the most suitable pose s_i is reduced to an efficient searching algorithm.

Algorithm 1 Visual Servoing for object Category with switching.

initialize $e = \infty$, δ
while $e \geq \delta$ **do**
  for all $M_i \in Models$ **do**
    $s_i = \text{argmin}_{q_i}(\|s - q_i\|) \quad \forall q_i \in \mathcal{P}M_i$
    $s_i^* = \text{argmin}_{q_i}(\|s^* - q_i\|) \quad \forall q_i \in \mathcal{P}M_i$
    $d_i = \|s - s_i\|$
    $d_i^* = \|s^* - s_i^*\|$
  end for
  $j = \text{argmin}_i(d_i + d_i^*)$
  $e = \|s_j^* - s_j\|$
  $IBVSstep(s_j, s_j^*, M_j)$
end while

Algorithm 2 assigns a weight w_i to every model at a servoing step. The weight is chosen such that, the model nearest to both current pose and desired pose i.e. with minimum $\|s - s_i\| + \|s^* - s_i^*\|$, receives maximum weight. The weights are further normalized to maintain the scale. A model M is generated through a linear combination of the available models with the assigned weights, which is employed for a servoing step. Algorithm 2 converges when residual error e for the resultant model is within a certain threshold δ. In Algorithm 2, similar to Algorithm 1 poses q_i are precomputed and stored in a pose bank.

Algorithm 2 Visual Servoing for object Category with linear combination of models.

initialize $e = \infty$, δ
while $e \geq \delta$ **do**
  for all $M_i \in Models$ **do**
    $s_i = \text{argmin}_{q_i}(\|s - q_i\|) \quad \forall q_i \in \mathcal{P}M_i$
    $s_i^* = \text{argmin}_{q_i}(\|s^* - q_i\|) \quad \forall q_i \in \mathcal{P}M_i$
    $w_i = 1/(\|s - s_i\| + \|s^* - s_i^*\|)$
  end for
  for all $M_i \in Models$ **do**
    $w_i = w_i / \sum w_i$
  end for
  $M = \sum_i w_i M_i$
  $\acute{s} = \sum_i w_i s_i$
  $\acute{s}^* = \sum_i w_i s_i^*$
  $e = \|\acute{s}^* - \acute{s}\|$
  $IBVSstep(\acute{s}, \acute{s}_t^*, M)$
end while

C. Control Scheme

In the following sub-section, we derive the control law for the the proposed approaches. Note that, the major difference between the proposed approaches is how the weight w_i is selected. Selecting the best model M_i implies assigning a weight $w_i = 1$ to M_i and zero weights to others. Thus, Algorithm 1 is a special case of Algorithm 2, where weights are sparse and discrete. Discrete weights makes the design and analysis of control law difficult. Thus, here we limit our scope to design of control law for Algorithm 2 only.

The standard control law for visual servoing as stated in the previous section is $\dot{e} = -\lambda L_s(s - s^*)$, where s and s^* are features in current and desired pose of same instances. In contrast to traditional visual servoing, for our case, current object and the desired object are two different instances. As the features are semantic keypoints so linear combination could also be extended to s and s^*.

$$s \approx \acute{s} = \sum_i w_i s_i$$
$$s^* \approx \acute{s}^* = \sum_i w_i s_i^* \tag{3}$$

Then, the servoing error function could be written as:

$$e = \sum_i w_i s_i - \sum_i w_i s_i^* \tag{4}$$

Where, the weights are given by:

$$w_i = \frac{1}{\|s - s_i\| + \|s^* - s_i^*\|} \tag{5}$$

These weights are chosen such that the model which is most similar to current pose and desired pose i.e. with minimum $\|s - s_i\| - \|s^* - s_i^*\|$ gets maximum weight. Note that, the denominator in (5) never becomes zero as, we are servoing across instances i.e. $s \neq s_i$. Since, the camera velocity would remain same for all models M_i and neither s^*, nor s_i^* vary, hence without loss of generality, it can be assumed that $\dot{s} = L_s V_c$, $\dot{s^*} = 0$, $\dot{s_i} = L_{s_i} V_c$, $\dot{s_i^*} = 0$. The derivative of the above error function would then be:

$$\dot{e} = \sum_i \dot{w_i}(s_i - s_i^*) + \sum_i w_i L_{s_i} V_c \tag{6}$$

Where, the derivatives of the weights are given by,

$$\begin{aligned} \dot{w}_i &= -\tfrac{1}{z_i^2}\dot{z}_i \\ \dot{z}_i &= -2(s-s_i)^T(\dot{s}-\dot{s}_i) \\ &= -2(s-s_i)^T(L_s-L_{s_i})V_c \end{aligned} \tag{7}$$

Furthermore, ensuring exponential decoupled decrease in error by putting $\dot{e} = -\lambda e$ gives,

$$V_c = -\lambda L_e^+ e \tag{8}$$

Where,

$$L_e = \sum_i w_i L_{s_i} - 2\sum_i w_i^2(s_i - s_i^*)(s-s_i)^T(L_s - L_{s_i}) \tag{9}$$

The interaction matrices L_{s_i} are similar to 10.

$$L_{s_i} = \begin{bmatrix} \frac{-1}{z_i} & 0 & \frac{x_i}{z_i} & x_i y_i & -(1+x_i^2) & y_i \\ 0 & \frac{-1}{z_i} & \frac{y_i}{z_i} & 1+y_i^2 & -x_i y_i & -x_i \end{bmatrix} \tag{10}$$

It could be seen that at $e = 0$, velocity also becomes zero, thus the approach converges at equilibrium point $e^* = 0$. As each 3D model is known apriori, depth term in the interaction matrix could be taken as it is from the geometry. Hence, it is possible to compute the interaction matrix at every step of visual servoing. However, the depth term in (10) is required in camera's frame of reference, while the coordinates of Models' keypoints are known in world frame of reference. Thus, to convert the keypoints into camera's frame of reference, camera's pose is required, which could be stored while precomputing poses for the pose bank. Then, we have a strong assumption that the camera's pose must lie in pose bank. However, visual servoing is fairly tolerant to depth errors. Thus, the pose bank should be made sufficiently large to cover various poses in the workspace. We further assume that target remains static throughout the servoing process and workspace is defined such that there exists a trajectory between current pose and desired pose without violating joint limit constraints and singularity constraints. Significant work has been done in visual servoing literature on relaxing these constraints [9], [10].

IV. Experiments and Results

A. ServoKt Dataset

Our approach requires 3D models and images represented as part based semantics. As to our knowledge, there does not exist any standardized dataset with semantic annotations. Thus, we created ServoKt dataset which expresses part based semantics for 2D images as well as for 3D models, so that, accurate correspondences could be established among images and models of a category. Furthermore, dataset is able to express large variations in appearances and geometry as shown in figure 5. The dataset comprises of 5 object categories namely *Teapot*, *Cup*, *Bottle*, *Flashlight* and *Toycar*. Each category contains 100 images and 20 3D models per category. Each image has been manually annotated with 2D keypoints, which represent pixels coordinates. Also, every model has been manually annotated with 3D keypoints, which are mean subtracted coordinates in 3D space. Figure 5 shows a few examples of annotated images and 3D models from ServoKt dataset. Note that, 3D coordinates are fixed with respect to a given origin. Thus, annotations represent mean subtracted coordinates to make the model independent of origin and scale. However, images represent a camera pose, thus they are not mean subtracted. We have used images from freely available datasets, such as Caltech 256 dataset [11], LHI dataset [12], Cambridge toy cars Dataset [13], Inria toy cars dataset [13]. We have collected 3D models by downloading freely available CAD models from Google 3D Warehouse [14] and retaining the good quality models. These models had different scales to transform them into a uniform scale before beginning the annotation procedure. An instance may not contain all keypoints of the object category as intraclass variance is high, also in an image some parts might not be visible. Figure 5(c) summarizes ServoKt dataset.

(a) (b)

Object category	# of images	# of 3D models	# of keypoints
Cup	100	21	19
Teapot	100	22	20
Bottle	100	20	8
Flashlight	100	22	12
Toycar	100	20	16

(c)

Fig. 5. ServoKt dataset: (a) Images of object categories with respective keypoint annotations (b) 3D models of object categories. (c) Summary of the dataset: number of images and 3D models per category. Number of keypoints denote unique parts employed for representing the object. Note the variation in appearance, pose and geometry for both (a) and (b).

B. Experiments

1) Experimental setup: All the experiments were performed in simulation on a 6 DOF manipulator and eye-in-hand camera configuration [2] was employed for visual servoing. We selected a 3D model from ServoKt dataset treated it as the given object, unknown to the robot. Other 3D models were assumed to be available along with their annotations for algorithm to find the best fit model to the given object. To compute s_i and s_i^* required by both algorithms, a pose bank was generated by viewing all models from 5832 views per object. Note that, the size of the pose bank along with the number of models is the tradeoff between the computational expense and accuracy of the approach. For the following set of experiments, we have restricted our

Fig. 6. **Qualitative Results.** (a) Initial pose captured by the robot with a random camera pose for provided 3D object from the dataset. (b) Servoing result of Algorithm 2 for desired pose image. (c) The desired pose in form of image provided to the robot. 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.

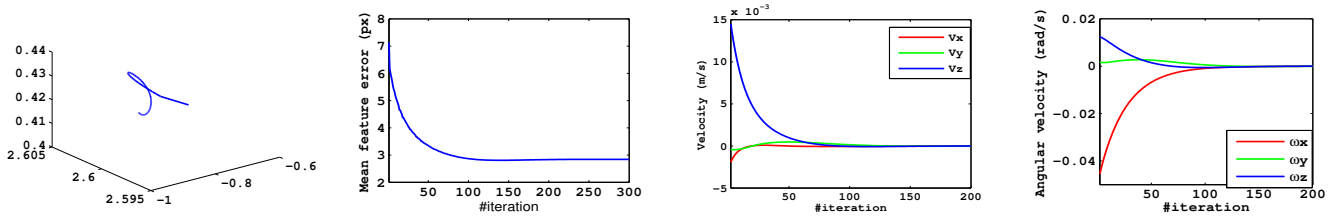


Fig. 7. **Servoing with algorithm 2.** (a) Camera trajectory is continuous and smooth. (b) Feature error decreases exponentially. Camera translation velocity (c) and rotational velocity (d) becomes zero at steady state. Note that, the desired pose and initial pose are different instances hence there is residual feature error on convergence.

pose bank to rotation only poses. The translation and scaling are handled by aligning the mean of initial pose (s) to that of each candidate pose (s_i) in the pose bank and maintaining a reasonable scale. Similarly, the mean of the desired pose (s^*) was also aligned to each of its respective counterpart (s_i^*) in the pose bank. Such pose bank is more biased to rotation and results in a precise rotation compared to translation. Also, for the following set of experiments the visual features s and s^* were computed by projecting of object O rather than applying vision preprocessing, since throughout this paper we have assumed that given an image 3D part based semantics could be extracted in form of keypoints.

2) Qualitative results: The objective of this section is to show the efficacy of the proposed algorithm to servo to a diverse set of target instances across various object categories. The desired image for the servoing algorithm is shown in figure 6(c). The current pose of the object as seen by the camera is shown in 6(a). It is to be noted that these objects (in the row of figure 6(a) are not part of the 3D models available to the algorithm. Based on the current object pose and the desired image a linear combination of keypoint descriptors, s_t and s_t^* are computed as described in Algorithm 2. The result of the servoing based on these descriptors is depicted in the row of figure 6(b). The similarity in the pose of figure 6(b) and figure 6(c) verifies qualitatively the efficacy of the proposed algorithm. Note that, the algorithm is able to servo to desired poses for diverse instances of the same object as well as across five categories of objects is the keynote theme of this paper.

3) Quantitative results: We have validated the performance of our approach (Algorithm 2) on various performance measures like trajectory smoothness, terminal velocity, camera pose error and feature error.

In the first experiment, we used the initial pose and desired pose configuration as shown in figure 4. The approach converged with a smooth trajectory and zero terminal velocity of the end-effector as shown in figure 7. It could also be seen that the feature error decreases exponentially.

In the second experiment, we have tested the performance of our approach for error in camera pose. We selected a 3D model from the ServoKt dataset which we used as the target object. It was servoed in 20 random poses using Algorithm 2. We further computed the rotational and translation error

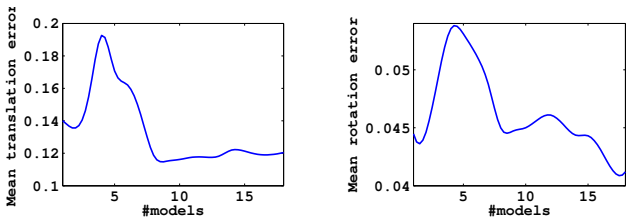
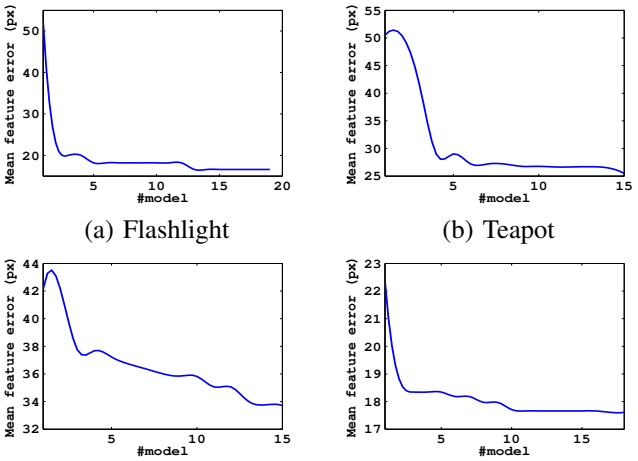


Fig. 8. (a)-(b) Effect of increasing 3D models on camera pose error post servoing. Note that, as the number of models increase, initially the pose error increases as a linear combination does not approximate the object. However, as the number of models become sufficient, the error starts decreasing. Note that, we have not considered kinematic constraints of the robot, hence there exist configurations where joint limit is reached, resulting higher error in camera pose.



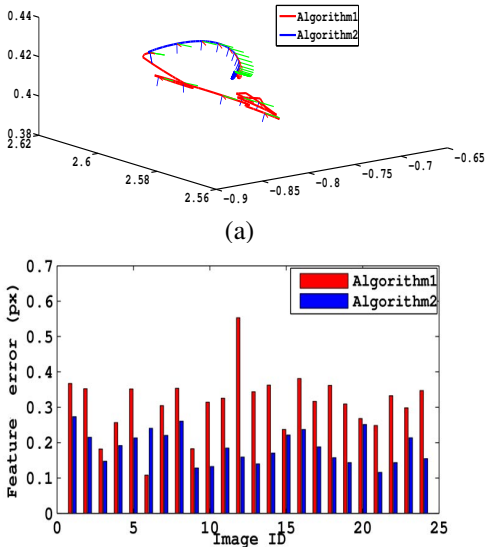
(a) Flashlight (b) Teapot

(c) Bottle (d) Toycar

Fig. 9. **Extending to other categories.** We also test our approach on four other categories: (a) *flashlight* (b) *teapot* (c) *bottle* (d) *toycar*. The feature error initially increases and gradually decreases thereafter as the number models are increased. Note that, for *flashlight*, performance is higher compared to other categories due to lower variation in geometry.

between final pose attained by the camera and the desired pose (both in Euclidean world). Six keypoints were used to match the pose, however all annotated keypoints were used for visual servoing. We then varied the numbers of models available to the algorithm for computing the best-fit model to the given object from one to eighteen. Whole process has been repeated five times with different model being selected as target object for statistical averaging. The translation and rotational errors between camera pose were defined as $e_{translational} = \|X_{desired} - X_{attained}\|/\|X_{desired}\|$ and $e_{rotational} = \|Q_{desired} - Q_{attained}\|/\|Q_{desired}\|$, where, X denoted the Euclidean coordinates and Q denoted the quaternions. Motivation for the experiment was to validate the premise that as the models available to the algorithm increase there is a higher probability to find the models similar to the given object. Hence, as the number of models increase the total error should decrease and given infinite models, there exist a model for which eventually error in camera pose becomes zero. The results are plotted in figure 8. As it could be seen in the figure 8, initially the error increases as irrelevant models are also given weights, but as the number of models increase more of the relevant models are selected with higher weights, the error decreases in accordance with our premise. Note that, there exists a higher residual error in translation compared to that in rotation. The reason for such observation is that the pose bank is biased for rotation variations compared to scale variations.

We also implemented our approach on four other categories from ServoKt dataset. We randomly selected 10 image pairs and servoed over them by varying the number of models available to the robot. We repeated the experiment five times for statistical averaging. The residual feature error has been plotted in figure 9 against number of models. It could be observed that the error initially increases till sufficient number of models have been found. Then, it gradually decreases. This trend shows that given a sufficient number of models, a new model could be linearly interpolated. The number of models required to produce good results depends upon both, the category itself as well as the variation among the 3D models. For the above experiment, it could be observed from figure 9, that 12 models are sufficient to achieve satisfactory results. Note that, we have servoed between different instances hence there exists a residual feature error.



(a)

(b)

Fig. 10. **Comparing Algorithm 1 and 2.** (a) Algorithm 2 converges to a stable solution with smooth trajectory while Algorithm 1 oscillates between two models. (b) The feature error at the time of convergence for the linear combination approach of Algorithm 2 is significantly less than the switching models approach of Algorithm 1.

4) Comparison between switching and linear combination based approaches: As mentioned in the previous section, both Algorithm 1 and 2 minimize semantic distance on desired pose by employing suitable model at every servoing step. Algorithm 2 is a more generalized version of Algorithm 1, finding a single suitable model could be considered solving the optimized weight assignment problem constraints that weights being non-negative integers and summation of weights being one. As there are no integral constraints for Algorithm 2, it is supposed to find a better optimized solution i.e. a more suitable model. Another advantage of using Algorithm 2 is a smoother transition across models which in turn results in a smooth trajectory. Since the models are not being switched at every instance, it gives a continuity in the trajectory, which is desirable for the robotic manipulator. An extreme case could be when the desired instance O is exactly in middle of two 3D models $O = \frac{1}{2}M_1 + \frac{1}{2}M_2$. Here Algorithm 1 will not able to decide between the two models and would keep oscillating between them. To validate these claims we compared the trajectory of Algorithm 1 with Algorithm 2 as shown in figure 10(a). It could be seen

that Algorithm 2 converges to a stable solution following a smooth path while Algorithm 1 keeps on oscillating between two models. We also compare these approaches on the residual feature error after large number (1000) iterations, assuming that the algorithm is not able to converge. As shown in figure 10(b), that Algorithm 2 results in a lower residual error.

5) Failure cases: The primary assumption made for proposed approach is that instances similar to the given object are previously seen. For example, if the robot has seen only cups with handles but the given instance does not have a handle, proposed approach won't be able to find a suitable cup. Another failure situation arises in the case of extrapolation, where linear combination does not give optimal solution. For example the given object is smaller than all seen models then ideally the smallest model would perform better.

V. Conclusion and discussions

In this work, we have introduced a novel problem of visual servoing across instances of an object category and we have also provided an approach to accomplish the task employing a dataset of 3D models. We have also designed the closed form control law for the approach and empirically shown the convergence of the proposed approach. With this strategy visual servoing could be applied to variety of objects without modeling every instance. We have demonstrated our work on five different object categories with moderate intraclass variance and intend to make the dataset used publicly available for further research.

The primary motive as discussed in section I has been to empower the robots to interact with a wide variety of objects, including the unseen one. A critical component that enables this possibility is the framework that matches semantic parts across object instances. In this paper we have used state of the art algorithm in computer vision that compute semantic part based correspondences between images. The reliability of the system to an extent hinges on the ability of the current state of the art approaches to compute such correspondences. In our early stages of work we also identified the need for preventing exaggerated model switching to guarantee a pleasing and acceptable servoing performance. To this end, we identified a method that interpolates between semantic correspondences over a diverse set of instances. As mentioned earlier while a routine indiscriminate linear combination between images would produce ghost images and degrade performance, a linear combination across aligned semantic correspondences could still be suitable. This was exploited to obtain stable, converging and pleasing camera trajectories for a variety of instances and objects. Furthermore, the premise that indeed the desired pose has been reached at the convergence ($\acute{s} = \acute{s}^*$) employing linear combination has been shown through empirical results.

In future we plan to expand the proposed approach to a wider variety of objects and also contribute to the state-of-the-art in finding pose of the object based on part based semantics.

VI. Acknowledgement

We thank TCS research fellowship for financial support.

References

[1] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," in *ICRA*, 1996.
[2] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," in *MRA*, 2006.
[3] R. Arandjelovic and A. Zisserman, "Name that sculpture," in *ICMR*, 2012.
[4] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *ICCV*, 2003.
[5] C. Liu, J. Yuen, and A. Torralba, "Nonparametric scene parsing via label transfer," in *PAMI*, 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] B. Schölkopf, F. Steinke, and V. Blanz, "Object correspondence as a machine learning problem," in *ICML*, 2005.
[9] É. Marchand and F. Chaumette, "Feature tracking for visual servoing purposes," in *RAS*, 2005.
[10] E. Marchand, F. Chaumette, and A. Rizzo, "Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing," in *IROS*, 1996.
[11] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007.
[12] B. Yao, X. Yang, and S.-C. Zhu, "Introduction to a large-scale general purpose ground truth database: methodology, annotation tool and benchmarks," in *EMMCVPR*, 2007.
[13] P. Bendale, W. Triggs, N. Kingsbury, *et al.*, "Multiscale keypoint analysis based on complex wavelets," in *BMVC*, 2010.
[14] Google, "3d warehouse." http://sketchup.google.com/3dwarehouse/.