Instance Invariant Visual Servoing Framework for Part-Aware Autonomous Vehicle Inspection using MAVs

by

Harith Pandya, Ayush Gaud, Gourav Kumar, Madhava Krishna

Report No: IIIT/TR/2019/-1

Centre for Robotics
International Institute of Information Technology
Hyderabad - 500 032, INDIA
August 2019
Abstract

Visual servoing approaches navigate the robot to a desired pose with respect to the given object using image measurements. As a result these approaches have several applications in manipulation, navigation and inspection. However, existing visual servoing approaches are instance specific i.e. they control camera motion between two views of the same object. In this paper, we present a framework for visual servoing to a novel object instance. We further employ our framework for autonomous inspection of vehicles using Micro Aerial Vehicles (MAVs), which is vital for day-to-day maintenance, damage assessment and merchandising a vehicle. This visual inspection task comprises of the MAV visiting the essential parts of the vehicle, for example wheels, lights, etc., to get a closer look at the damages incurred. Existing methods for autonomous inspection could not be extended for vehicles due to following reasons: Firstly, several existing methods require a 3D-model of the structure, which is not available for every vehicle. Secondly, existing methods require expensive depth-sensor for localization and path planning. Thirdly, current approaches do not account for semantic understanding of the vehicle, which is essential for identifying parts. Our instance invariant visual servoing framework is capable of autonomously navigating to every essential part of a vehicle for inspection and can be initialized from any random pose. To the best our knowledge this is the first approach demonstrating fully autonomous visual inspection of vehicles using MAVs. We have validated the efficacy of our approach through a series of experiments in simulation and outdoor scenarios.

Project website: http://robotics.iiit.ac.in/people/harit.pandya/vi_inspection

1 Introduction

Visual servoing utilizes image sensory information to move a robotic system towards a goal position with respect to the given object or scene. A visual servoing approach is composed of extracting a set of visual features from image measurements and controlling the robot such that these features match their desired configuration (Chaumette and Hutchinson, 2006). Based on whether the control objective is defined in the Cartesian space or in image space, the traditional visual servoing approaches are classified into Position Based Visual Servoing (PBVS) and Image Based Visual Servoing (IBVS) (Hutchinson et al., 1996). PBVS utilizes visual features to compute the object’s pose in the robot’s Cartesian space. Estimating the robot’s pose requires additional information regarding the geometry of the object. This can be obtained completely by explicit knowledge of the object’s 3D model or partially by 3D reconstruction of the object while servoing. IBVS on the other hand directly controls the robot in image space by
Aim: (a) The MAV is initially observing the current instance (yellow Beetle hatchback) from a side view. The instance invariant visual servoing task requires the MAV to move to the front of the Beetle so that observed view matches the desired view of a template instance (white Range Rover). Such navigation maneuver is difficult to attain using existing visual servoing approaches, due to large variation in shape and appearance between the two instances. (b) Using our instance invariant visual servoing framework, we propose a novel autonomous navigation framework for visual inspection of a vehicle using a low-cost off-the-shelf MAV. Starting from a random initial pose, the MAV servoes to different parts of the vehicle like wheels, lights, mirrors etc., to capture finer images of these parts. Which plays a crucial role in day-to-day maintenance and damage assessment of the vehicle.

Previous visual servoing approaches used low-level geometric primitives (for example corners, lines, contour) as visual features (Chaumette and Hutchinson, 2006) for servoing between two views of the same object as shown in figure 2(a). However, these features do not generalize well for matching between two different object instances due to large appearance and shape variations as illustrated in figure 2(b). In this paper, we propose part-aware keypoints as visual features that encode the global perspective of the object and retain only the meaningful information in contrast to local appearance based descriptors like SIFT (Lowe, 1999) and ORB (Rublee et al., 2011) as shown in figure 2(c). To learn these part-aware keypoints we employ a convolutional neural network (CNN) based on stacked hour-glass architecture (Newell et al., 2016).

Existing visual servoing controllers minimize an error function between the current and the desired configuration of visual features. Since these visual features are geometrically related, a possible solution exists in $SE(3)$ space such that the error converges to zero at the desired pose. However, when servoing across instances in the same object category, the geometry of the objects may not permit the error to diminish at the desired pose. This limits the scope of traditional visual servoing to views of the same object. Such problems arise frequently in practical scenarios where servoing to a new instance is required, especially in the case of manipulation and navigation. Hence, these approaches are not well suited for performing autonomous vehicle inspection.

Visual inspection has been widely adopted in industries to assess the quality of the product as well as to identify defects resulted from a manufacturing process. In industries almost every product goes through the process of visual inspection by trained persons. This procedure of visual inspection is highly monotonous and laborious, therefore autonomy in visual inspection is rapidly progressing its way into industries. Equipping the autonomous robots with mobile actuation and integrating sensory feedback helps them to perform inspection task outside factory settings. Micro Aerial Vehicles (MAV) further offer a unique opportunity to achieve sensory measurements from places which are generally beyond the reach of ground vehicles. Also, they can provide images from a different perspective with finer details. Hence, recently these MAVs are being used for numerous autonomous inspection tasks such as for large structures (Morgenthal and Hallermann, 2014 [Bircher et al., 2016b]), tunnels (Ozaslan et al., 2017), shipboards (Fang et al., 2017), agriculture (Das et al., 2015) etc.
Figure 2: (a) IBVS approaches (Crombez et al., 2015) are designed to servo between two views of the same object. (b) Challenges faced by existing IBVS approaches: variations in appearances, shapes of car and viewpoints especially non-overlapping views. (c) Local descriptors like SIFT based keypoints might result in an incorrect matching when the instances are different, whereas our part-aware semantics are more suitable for computing correspondences across object instances.

These autonomous visual inspection applications commonly require a pipeline comprising of exploration and mapping. The common objective is to achieve a dense reconstruction of the object of interest or structure to capture fine details and analyze these details offline. Various aspects of the autonomous inspection problem has been previously studied such as efficient coverage planning (Bircher et al., 2016a), obstacle avoidance (Bircher et al., 2016b), map representation (Teixeira and Chli, 2017), vision in degraded environment (Fang et al., 2017) etc. Visual inspection of automobiles on the other hand is slightly different as it is required to visit essential components of the vehicles and decide whether the component or part is in healthy condition or not. To achieve complete autonomy in visual inspection of a vehicle, the MAV should understand the notion of essential components/parts and should navigate to these parts for obtaining a finer view. Visual inspection has several benefits: Firstly, frequent inspection helps to monitor the health of the vehicle. Secondly, this part aware visual inspection of the vehicle could be helpful in estimation of damage to the vehicle, especially for an insurance claim. Thirdly, the inspection results could be also used for merchandising the vehicle. As a result the vehicular inspection is luring numerous ventures, for example QuickFoto Claim, Express Auto Inspection etc. However, such ventures require the manual effort for collecting a video of the vehicle, which is then processed offline. Car360 inc., employ either a turn-table or a large manipulator for automating the inspection process. However, their setup has scalability issues due to the requirement of a huge infrastructure.

In this work, we address the problem of visual servoing across instances of an object category. Specifically, provided the desired view of an object, the aim is to attain the same desired view for any other instance from that category. Figure 1(a) describes a servoing scenario in which a MAV starts from a random initial pose and is required to attain a pose with respect to the current car (yellow Beetle) such that the resultant view matches the desired view of a template instance (white Range Rover). We formulate this problem of servoing across instances as Pose induction followed by position based visual servoing (PBVS). Where, the Pose induction step infers the desired pose with respect to the current instance from the desired pose of the template. The PBVS step estimates the current pose of the robot with respect to the current instance and moves the robot towards the inferred desired pose. One of the key research challenges in the proposed inspection task is to address the large variations in shape and texture among vehicles. This poses an immense challenge for vision based classifiers to discriminate them from background. Not only there are variations in the overall design of vehicles but also the shape and texture of essential parts vary significantly. Detecting the location of these parts is one of the problems which is gaining interest among computer vision researchers. We propose to employ a stacked hourglass based CNN for identifying the locations of these essential parts.

After laying the foundation for servoing across instances, we propose a visual navigation pipeline for achieving autonomous part aware visual inspection of a vehicle using an inexpensive off-the-shelf multi-rotor MAV equipped with a monocular camera. For the inspection purpose, mere identification of these parts is not sufficient, the MAV is also required to visit every part for capturing the fine details. Thus, we present a novel dilated convolution neural network (DCNN) based framework that is able to detect the car and assign a dense pixel-wise segmentation to these parts.
Another key challenge in the proposed work is to navigate the MAV to these parts using a monocular camera and erroneous odometry. We therefore consider this navigation task for autonomous visual inspection as a visual servoing problem, where the objective of MAV is to attain a desired pose with respect to a selected part in the image space (for example, move the MAV in Cartesian space such that the selected part should be in center of image). Hence, here we propose a hierarchical instance invariant visual servoing pipeline that guides the MAV to acquire the desired pose with respect to every part sequentially. Figure[1](b) describes a use-case of visual inspection, where MAV is moving around a novel instance of a car and is inspecting headlights.

Contributions

Our contributions could be summarized as follows: Firstly, we have introduced a novel problem of instance invariant visual servoing through our previous works (Pandya et al., 2015), (Pandya et al., 2016) and (Kumar et al., 2017) that is more suitable in practical scenarios as compared to existing visual servoing approaches for manipulation and exploration tasks. In this paper, we propose a Pose induction framework for visual servoing to a novel object instance. Our framework accommodates the changes in appearance and illumination through part-aware keypoints learned using a CNN. Secondly, to our knowledge, this framework is the first attempt in addressing autonomous inspection of vehicles incorporating semantics. Our approach is able to generalize well despite of high intra-category variation in texture and shapes of vehicles. The presented framework is designed to work with a low-cost off-the-shelf MAV. Thirdly, we present a dilated-CNN based system that is able to achieve state-of-the-art part segmentation performance with 96.7% pixel-wise accuracy and 84.6% IOU on PASCAL parts dataset for person category. Finally, we present multi-view data augmentation and optical flow based Bayesian fusion refinements for improving the segmentation performance, especially for oblique views, which is common problem when using deep networks due to lack of sufficient training data. Our approach is able to perform visual servoing across large camera transformations and non-overlapping scenes, which is a non-trivial task for existing visual servoing approaches. We validated our approach through series of simulation and field experiments. Out of total 176 parts to be inspected, our approach was able to converge for 159 times to the given part for inspection with the mean error in part-area less than 0.5 % of the total image area.

2 Related Work

Instance invariant visual servoing: The problem of visual servoing across object instances was first introduced by (Pandya et al., 2015). They used part-aware keypoints for making the approach robust to textural variation across instances in an object category. They further proposed a linear combination of available 3D models for a servoing iteration. However, the semantic features were computed manually that makes the approach laborious for a 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 expensive. A discriminative learning based framework was also proposed for visual servoing across instances (Pandya et al., 2016). Where, authors proposed principal orientation glyph (POG) as visual features and a classification error based controller was used for achieving geometry invariance. However, their interaction matrix was numerically computed, which resulted in a relatively smaller convergence domain.

In this work, we address the problem of visual servoing across instances of an object category. Specifically, provided the desired view of an object, the aim is to attain the same desired view for any other instance from that category. We formulate this problem of servoing to a novel instance as Pose induction and alignment problem. The Pose induction step infers the desired pose with respect to the current instance from the desired pose of the template. The pose alignment step estimates the current pose with respect to the current instance and moves the robot towards the desired pose using PBVS.

Part-localization and segmentation: Computing descriptors that provide unique and accurate correspondences among multiple views of the same instance or across different instances have been one of the classical problems in computer vision. Previous approaches from computer vision literature report superior performance in keypoint correspondences when the keypoints are conditioned on object category, especially when the keypoints were semantically
related to object’s parts (Maji and Shakhnarovich, 2012; Felzenszwalb et al., 2010). Motivated from recent breakthroughs in CNNs, (Tulsiani and Malik, 2015) presented a CNN that was able to learn part-aware keypoints through supervision. They reported a significant improvement in keypoint prediction accuracy by conditioning keypoints inference on viewpoint estimations. Recently, (Newell et al., 2016) proposed a stacked hour-glass architecture for CNN that showcased state-of-the-art results for keypoint prediction for human category. In this paper, we use a similar stacked hour-glass CNN from our previous work (Kumar et al., 2017) trained on PASCAL 3D dataset (Xiang et al., 2014) for cars. We obtained superior results for keypoints detection over (Tulsiani and Malik, 2015) for ‘car’ object category.

As compared to object localization where only the center of the object is estimated, the semantic segmentation problem consists of assigning a class label to every image pixel. Thus, the segmentation helps in better understanding about the object and the scene compared to mere object localization. Classical segmentation approaches based on Markov Random Fields (Blake et al., 2004; Fang et al., 2013) considered this as graph labelling problem where the objective was to minimize the network’s energy by assigning correct labels to every pixel. Another branch of approaches based on Conditional Random Fields (CRF) (Gonfaus et al., 2010; Maire et al., 2011; Plath et al., 2009) used multiple cues from different modalities to improve segmentation performances. Recent advances in convolutional neural networks have revolutionized even this problem of segmentation. In the past couple of years, several CNN based models were proposed for category level segmentation (Badrinarayanan et al., 2015; Dai et al., 2016; Chen et al., 2016; Yu and Koltun, 2015). For the purpose of visual inspection of a vehicle we are more interested in part level segmentation. Recently, a few approaches proposed CNNs to cater part level segmentation (Zhou et al., 2017; Oliveira et al., 2017).

In this paper, we present a dilated CNN based framework trained on PASCAL-parts dataset (Chen et al., 2014) for segmenting essential parts of the vehicle. Since, our control framework employs this part-segmentation feedback for visual servoing, we are required to see cars from several viewpoints. However, the training data available from PASCAL-parts dataset does not provide an extensive set of viewpoints. Therefore, we present a simulation framework for synthetic data augmentation. (Oliveira et al., 2017) have also reported that data augmentation (spatial and color) results in significant performance improvement. Furthermore, in our case the camera attached to robot is also moving therefore we use this active vision to our advantage by fusing segmentations from multiple views. We propose optical flow based warping along with Bayesian fusion (Ma et al., 2017), for improving the segmentation performance.

**Autonomous visual inspection:** Early works on autonomous visual inspection using a fixed camera were tailored to the requirement of determining the quality of product during a manufacturing process (Chin and Harlow, 1982). With the addition of mobility to camera, the applications of autonomous visual inspection increased by many-fold. Recently, the MAVs are being extensively used for inspection of large structures (Morgenthal and Hallermann, 2014; Bircher et al., 2016b), tunnels (Ozaslan et al., 2017), shipboards (Fang et al., 2017), agriculture (Das et al., 2015) etc., due to the ability of MAVs flying stably at lower altitudes and reaching places that are difficult to approach for humans. To achieve complete autonomy in inspection problems, challenges related to navigation need to be addressed such as accurate state estimation. Several navigation approaches (Bircher et al., 2016b; Usenko et al., 2017; Dryanovski et al., 2013; Shen et al., 2011) rely on accurate depth sensors like LIDARs for accurate state estimation and mapping which makes the MAV heavy and expensive. (Fraundorfer et al., 2012; Schauwecker and Zell, 2014) on the other hand use a stereo camera pair for depth estimation, which requires matching features between frames and increases time delay in the system. Although the stereo pair is not heavy compared to the LIDAR sensor, however it is still expensive and is not present in majority of off-the-shelf MAVs such as Parrot’s Bebop, DJI Phantom etc. Recent approaches from (Wu et al., 2013; Weiss et al., 2011; Lin et al., 2018) propose a visual inertial navigation framework that fuses the IMU measurements with image measurements from monocular camera, but for accurate localization their framework requires a global shutter camera and hardware synchronization between the clocks of camera and IMU, which is again not present in off-the-shelf MAVs. Hence, in this paper we use visual servoing in image space to navigate the MAV to the desired pose of essential parts circumventing the requirement for accurate pose estimation.
3 Problem description

The problem of vehicular inspection, as it is considered in this paper, consists of a low-cost off-the-shelf MAV equipped with a monocular camera for inspecting a given vehicle (current instance). The given vehicle could be any novel instance of a car, previously unseen by our approach. We therefore formulate this navigation problem as an instance invariant visual servoing. Where a template model is employed to estimate the pose of the MAV with respect to the given vehicle and navigate the MAV to sides (left $F^*_l$, right $F^*_r$ and front $F^*_f$) of the vehicle from a random initial pose $F_0$. Here, the images of sides of the template model act as desired views in which the MAV is required to servo sequentially.

The objective of the inspection problem is to visit the pose $F^*_i = [X^*_i, Y^*_i, Z^*_i, \theta^*_i], \forall i \in N_P$ corresponding to every essential part $i$ among the set of all parts $N_P$ of the vehicle (wheels, headlights and mirrors) starting from a random initial pose $F_0$. We define $F^*_i$ such that the part $i$ occupies a predefined area $A^*_i$ and lies in the center of the image captured by MAV at $F^*_i$ i.e. $[u, v] = [N_i/2, M_i/2]$, for the image with size $M_i \times N_i$. We assume the navigable workspace is obstacle free and the initial pose of the MAV denoted by $F_0$, has the vehicle in field-of-view of the camera. All the computations need to be done near real time on-board or using a base station.
4 Overall pipeline

The overall pipeline comprises of a perception and a control module. The perception module processes the image sequence and returns (i) a bounding box capturing the car, (ii) keypoint locations of the parts in the image, and (iii) pixel-wise segmentation of essential parts. The control pipeline is responsible for navigating the MAV to the desired pose \( F_i^* \) for every part in a sequential order. The MAV starts at a random pose \( F_0 \) and visits a vantage pose for every side \( F_i^* \) \( \forall s \in \{l,f,r\} \), (front \( F_l^* \), left \( F_f^* \) and right \( F_r^* \)). After reaching to the vantage point, the MAV visits all the parts of that side sequentially. The key challenge here is that the poses \( F_i^* \), \( F_j^* \) are not known in Cartesian space, rather these are to be inferred from images using the vision pipeline. As shown in figure 3(a), we divide the control pipeline into three modules: Pose induction, Face-servoing and Part-servoing. Where, the Pose induction and Face-servoing modules are components of the instance invariant visual servoing framework. The outputs of every module is visualized in figure 3(b). We now briefly summarize the functionality of every module.

The Pose induction module is responsible for reconstructing the given vehicle (current instance) and estimating the vantage poses for every side \( F_i^* \). Provided a random starting pose \( F_0 \), the MAV initially follows a straight line maneuver orthogonal to the optical axis for reaching an arbitrary pose \( F_{0,i}^0 \), such that the corresponding images \( I_0 \) and \( I_0' \) captured from \( F_0 \) and \( F_0' \) form a multi-view stereo pair. We refer to this step as multi-view stereo initialization. We further use our keypoint prediction network to extract part locations from these images. These part locations from \( I_0 \) and \( I_0' \) are then used for triangulation and semantic reconstruction of parts as 3D keypoints \( P_i = [X_i,Y_i,Z_i] \) in the Cartesian space. A standard model (template) of a car is then aligned with the reconstructed 3D model so that the vantage poses \( F_i^* \) could be transferred from the template model to the current reconstruction, as for the template model these vantage poses can be easily computed using 2D-to-3D correspondences from desired images. As shown in figure 3(b), the Pose induction module produces four outputs: (i) Semantic reconstruction of the current instance, (ii) Initial pose of MAV with respect to the current instance \( F_0 \), (iii) Desired poses of all sides \( F_i^* \), \( \forall s \in \{l,f,r\} \) and (iv) The order in which these sides should be visited.

After \( F_i^* \) is estimated in Cartesian space, the Side navigation module guides the MAV to \( F_i^* \) using position based visual servoing. Here, we use odometry provided by the MAV for localization. However, the odometry measurements obtained by MAV are inaccurate and tend to drift over time, therefore we additionally use an image moment based visual servoing for pose refinement. The resulting of images captured by the MAV after the refinements are also shown in figure 3(b).

Eventually, our part servoing module guides the MAV to a desired pose \( F_i^* \) for all parts belonging to that side in a sequential order. We employ the semantic part-segmentation parameterized by image moments \( p_i = [u_i,v_i,A_i] \) as visual features for navigating to parts using IBVS. After attaining the desired pose we use the reverse-PBVS module to return back to \( F_i^* \) and continue this procedure until all the parts visible from that side are servoed. Then, to switch side we again use PBVS followed by IBVS refinement to attain the vantage point for the next side (\( F_f^* \) or \( F_r^* \) or \( F_l^* \)). This process is repeated for all parts and is summarized in algorithm 1. Images along with overlaid segmentations captured by the MAV from a few \( F_i^* \) are shown in figure 3(b) for a test case. We also encourage the reader to refer the video at the project website for our workflow in action.

It should be noted that the current instance of the car is different from the standard model (template) of the car thus classical IBVS and PBVS methods could not be directly used. Also, we need to navigate to all sides of the car, that requires servoing between non-overlapping scenes which could not be achieved only by using IBVS, thus we employ this switching based hierarchical servoing approach.

5 Pose induction

The objective of the Pose induction module is to estimate the desired pose \( F_i^* \) using a template model \( Y \) along with its desired image \( I_i^* \) for the current side and transfer it to the current instance \( X \). We employ a CNN to compute the part-correspondences \( x \) from current \( I_X \) and desired image \( I_i^* = I_i^* \) of the current side. Once we obtain the predictions \( y^* \) from \( I_i^* \), we use its 3D model (template) to compute the desired camera pose \( F_i^* \) by solving the perspective-n-point
Algorithm 1: Instance invariant visual servoing for visual inspection

1: multi-view stereo-initialization maneuver
2: semantic 3D Reconstruction
3: alignment with standard model and estimate $F^*_s$
4: for all sides do
5: identify side and plan side-visit ordering
6: PBVS to $F^*_s$
7: IBVS refinement
8: for all parts of current side do
9: Part-IBVS
10: PBVS back to $F^*_s$
11: end for
12: end for

$(PnP)$ problem as described in figure 4. Kindly note that here we are dealing with two reference frames, one frame is attached to the center of current object instance which is being servoed $F_X$ and other is attached to the template $F_Y$. Hence, we align both $X$ and $Y$ in a single virtual canonical frame $F_v$, so that $F^*_s = F^*_X$ could be transferred from the template $F^*_Y$. Also, note that the $F^*_Y$ computation is required only once per side and hence could be performed offline. Our framework performs real-time reconstruction and servoing of the current instance.

5.1 Keypoint Prediction

In this work, we leveraged the recent hourglass architecture deep convolutional neural network for our task of part-aware keypoint prediction. This network architecture was initially proposed by (Newell et al., 2016) for human pose estimation. The design of hourglass network captures information at multiple scales similar to (Tulsiani and Malik, 2015). However, this model is faster and more accurate compared to (Tulsiani and Malik, 2015) since it uses a single network instead of an ensemble of three different deep networks. Use of stacked hourglass provides an end-to-end solution for estimating part-aware keypoints. We trained a deep convolutional network composed of eight hourglass module stacked one after the other. The highest resolution of the hourglass is 64x64. The full network starts with a 7x7 convolutional layer with a stride 2 followed by residual module and a max pooling which brings down the resolution from 256x256 to 64x64. The stacking of hourglass modules assures both repeated bottom-up and top-down re-evaluation of initial feature estimates. We trained this network on annotated images of cars from PASCAL 3D dataset (Xiang et al., 2014) using Stochastic Gradient Descent (SGD) with a Euclidean loss. The prediction accuracy of our network was approximately 93% with a tolerance of two pixels, which is better compared to 81.3% claimed by (Tulsiani and Malik, 2015) on PASCAL 3D dataset for car category, which was annotated for fourteen different keypoints. Our network gives the confidence scores along with the image coordinates corresponding to each keypoint for the given image. The low scores corresponding to occluded parts or less guessable parts helped us to filter out the less confident predictions by the network. These predictions were further used as features for reconstruction and pose estimation for visual servoing.

5.2 Desired Pose Estimation

The problem of determining pose of a calibrated camera from $n$ correspondences between 3D reference points and their 2D projections, is known as $Perspective-n$-Point problem ($PnP$). The solution to this $PnP$ problem is used for estimating the camera extrinsics $(R, t)$. Provided a set of $n$ points $X_i$ in 3D and their 2D correspondences $x_i$ estimating the camera pose can be posed as a problem of minimizing re-projection error as:

$$\min_{R} \sum_{i=1}^{n} ||K(RX_i + t) - x_i||^2$$

subject to: $R^TR = I$ (1)
Figure 4: **Pose induction:** The objective of the Pose induction module is to estimate a vantage pose for the current side $F_Y$. This is achieved by using a standard 3D model of a car (template) and the provided desired view in the form of an image. Our deep network predicts the keypoints which, along with template instance model information, is used to predict the desired transformations ($F_Y$) in its canonical frame using perspective-n-point (PnP) solver. The keypoints predicted on the current image along with the previous views are used for semantic reconstruction of the current object $X$ and the origin ($F_X$) which is its centroid. This reconstruction is aligned with $F_Y$ using the axis normalization and alignment module in a canonical frame $F_v$ to obtain the desired pose specific to the current vehicle $F_s = F_X$. This pose is then fed to the Side navigation module that employs pose based visual servoing (PBVS) controller for generating control commands for the MAV.

PnP is a well established problem in 3D geometry and there are multiple solutions to the problem. We have used ASPnP (Zheng et al., 2013) since it solves the 3D-to-2D correspondences using a Gröbner basis solver, which guarantees a globally optimal camera pose. The desired pose of the camera $F_Y$ is determined from the annotated 3D model $Y$ and the desired 2D part correspondences $y^*$ using PnP.

### 5.3 Semantic Reconstruction

In our approach, the features used for reconstruction are detected keypoints which uniquely corresponds to parts of a vehicle, therefore we use the term semantic reconstruction. The process of semantic reconstruction starts with a multi-view stereo initialization i.e. giving a translation orthogonal to the optical axis of the camera in a horizontal plane to form a stereo image pair. Assuming the knowledge of camera parameters, the stereo pair obtained is used in triangulation for estimating 3D coordinates of the visible keypoints. The triangulated points are determined in the initial camera frame $F_0$, which is then transferred to a frame $F_X$ that is attached to the centroid of the reconstructed model $X$. The odometry readings are used for determining the actual scale of the current instance which is further utilized in the pose alignment step to scale up the normalized reconstruction of the current instance. Note that only a few keypoints are visible from the initial pose, hence only a partial 3D model is reconstructed.
Figure 5: **Axis-alignment:** The axis alignment is one of the crucial steps for the Pose induction module. This procedure transfers the desired pose $F_Y$ from the template instance $Y$ to the current instance $X$. (a) The partial reconstruction of the current instance $X$ as well as the template instance $Y$ are shown with blue and red wireframes respectively. (b) All the required frames are shown. $F_Y$ is the desired camera pose obtained by using PnP between desired image and template model (red wireframe). The frame $F_X$ is attached to the centroid of the current instance. We then select a suitable keypoint (say left front wheel) and attach the canonical frame $F_v$ capturing a semantic relation of keypoints (such as x-axis of the frame is parallel to line joining the front wheels). Next we select the same keypoint capturing same semantic relation in template instance and attach the template frame $F_Y$. (c) Since $F_Y$ and $F_v$ are semantically the same frame, therefore they could be directly aligned and using equation (2), we can compute the desired pose in current frame $F_{X^*}$. Note that axis alignment is only possible since our part-based keypoints have semantic meaning unlike SIFT or ORB keypoints.

### 5.4 Normalization and alignment to the canonical frame

The reconstruction of the current instance $X$ from semantic reconstruction pipeline is in the frame $F_X$. While the desired camera pose given by PnP is in the frame of template instance $F_Y$. Therefore, to compute the desired camera pose with respect to $X$, we need the transformation between $F_X$ and $F_Y$, which unfortunately is not always available. We tackle this issue by defining a canonical frame $F_v$ and transforming (aligning) both $F_X$ and $F_Y$ to $F_v$ using an alignment protocol. A valid alignment protocol requires exactly three rules, one rule to define an origin and two rules to define the alignment of any two coordinate axes. The procedure of aligning a 3D model to our canonical frame is shown in figure 5. Our transformation protocol is feasible because the part-based keypoints have semantic meaning associated with them. For example, consider the transformation between a frame attached to the current instance $F_X$ and the canonical frame $F_v$ attached to the right front wheel of $X$. Further consider a set of four keypoints corresponding to “left front wheel”, “right front wheel”, “left rear wheel” and “right rear wheel”. The origin of the frame $F_X$ is selected to be at the centroid of the reconstructed points, while that of the template and the canonical frames ($F_Y$ and $F_v$) to be at the right front wheel. The x-axes in all the frames are parallel to the ray from the right front wheel to the left front wheel and the y-axes are parallel to the ray from the right front wheel to the right rear wheel starting at their corresponding origin. By aligning the frame attached to the template of the canonical frame, we get the transformations $^vT_Y$, while $^vT_X$ is known since both $F_X$ and $F_v$ belong to same instance $X$. To maintain the homogeneity in the scale, the models are further normalized. The Pose induction process can then be explained with the following equation:

$$F_X^* = (^vT_X)^{-1} vT_Y F_Y^*$$

(2)

Where $T$'s are 4x4 homogeneous transformation matrices and could be represented as:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}.$$

After alignment the estimated desired pose for the current side $F_{s^*}^* = F_{X^*}$ is used by the PBVS controller for navigating the MAV from the initial pose $F_0$ to the current side $F_{s^*}$, since now they are defined in same frame $F_X$. This alignment is only possible because our keypoints have a semantic meaning associated with them.
6 Position based visual servoing towards side

PBVS approaches control the robot directly in the Cartesian space, thus the resulting camera trajectory is a straight line. Furthermore, for PBVS approaches, the interaction matrix is full rank, therefore these approaches can tackle large camera variations without getting stuck in local minima. These properties of PBVS approaches make them ideal candidate for visual navigation tasks especially between non-overlapping scenes. In our problem we need to navigate the MAV from \( F_0 \) to all the sides of the vehicle sequentially, starting with the current side \( (F^*_r, F^*_l, F^*_f) \). However, we only know \( F^*_r \) or \( F^*_l \) or \( F^*_f \) with respect to a template \( Y \). Thus, the standard PBVS techniques could not be directly applied here.

Therefore, the Pose induction step as explained in previous section employs a template model \( Y \) and the semantic reconstruction \( X \) to estimate the required camera pose \( F^* \) such that the camera projection at \( F^* \) matches \( I^* \). As a result, now both \( F_0 \) as well as \( F^*_s \) are in same frame. Therefore we can apply classical PBVS to navigate the MAV to \( F^*_s \). Similar to the classical PBVS, we consider \( F_c \) as the current pose and \( F^*_s \) as the desired pose. Where, the current pose of the MAV is obtained from the odometry provided by MAV. Let \( \hat{s}^* R_c \) and \( \hat{s}^* t_c \) denote the rotation and translation of the current pose of MAV \( F_c \) with respect to the desired pose \( F^*_s \). Then PBVS control law could be stated as follows (Chaumette and Hutchinson, 2006):

\[
\begin{align*}
v_c &= -\lambda_l \hat{s}^* R_c^T \hat{s}^* t_c \\
o_c &= -\lambda_a \theta \hat{s}^* t_c
\end{align*}
\]

Where, \( v_c \) and \( \omega_c \) are linear and angular commanded velocities, \( \lambda_l \) and \( \lambda_a \) are step sizes corresponding to linear and angular velocities respectively. \( \theta \) is angle-axis representation of the \( \hat{s}^* R_c \).

6.1 IBVS based refinement

The PBVS controller explained above generates the required velocity command to move the MAV from \( F_0 \) to \( F^*_s \). However, the PBVS controller requires the estimation of current pose of the camera at every time instance which was furnished by noisy odometry reading of the MAV. Moreover, there could also be errors on account of the large difference in shape between the current instance and the template. Hence, MAV could attain an incorrect desired pose \( F^*_s = F^*_s + \hat{s}^* t_c \). We therefore use an IBVS refinement to minimize the error \( \hat{s}^* t_c \) in the desired pose.

For this IBVS refinement step we use the segmentation mask generated by our part segmentation module (refer section 7.1) and extract image moments (center and area of segmentation). These image moments \( [x_g, y_g, A] \) are used as visual features to control the MAV using IBVS. The approach is similar to Part-IBVS and the details are described in section 7.2. In contrary to Part-IBVS, here we use the segmentation of the entire vehicle for extracting the image moments. Note that the desired image \( (I^*_s = I^*_s) \) remains same as that for the PBVS module.

7 Image based visual servoing towards parts

The instance invariant PBVS presented in previous section navigates the MAV to a side of the vehicle \( F_s \) using part based keypoints. Approaching further to a part requires zooming in to a single keypoint, which is degenerate scenario for IBVS as the interaction matrix becomes singular and hence non-invertible. Also, PBVS could not be used due to erroneous odometry. Therefore, we propose to employ image moments of the part segmentation mask as visual features for zooming into the part, which we refer to as Part-IBVS. The moments used are location of the centroid of the part segmentation mask \( (x_p, y_p) \) and its area \( A \). We propose a cascaded dilated convolution neural network architecture for computing the semantic segmentation of a part. This is further used in extracting the visual features and controlling the MAV using IBVS.
Figure 6: **The architecture of our cascaded DCNN.** The proposed network consists of two DCNN modules stacked together. The part-segmentation output from the first module of the cascade is used to compute an overall mask of the vehicle. The bounding box extracted from this mask is fed to the second module of the cascade along with the input image, which significantly improves the performance of pixel-wise segmentation. The procedure can also be seen as zooming-in to individual instances for better segmentation. Both the modules of the cascade are trained separately so that the error vehicle segmentation from the first module is not propagated to the second module. The detailed architecture of the individual module is presented in table 1.

7.1 Part segmentation

Recent CNN based approaches (Badrinarayanan et al., 2015; Dai et al., 2016; Zhou et al., 2017) define the problem of semantic segmentation as assigning every pixel $i$ of input image $I_x$ a class label $y_i$ from a list of predefined categories $N_{cl}$. Which is obtained by training the network under supervised settings on a dataset of $N_{tr}$ samples, each consisting of an image $I_{x_{tr}}$ and corresponding ground truth label provided in form of pixel-wise annotation image $I_{y_{tr}}$. The network weights $\theta$ for the architecture $f$ are learned by minimizing the following cost function:

$$\theta = \arg\min_{\theta} \sum_{i=1}^{N_{tr}} \sum_{k=1}^{N_{cl}} \log(y_i) \text{smax}(f_k(I_{x_{i}}; \theta))$$

(5)

where smax denotes the soft-max function,

$$\text{smax}(m_k) = \frac{\exp(m_k)}{\sum_{l=1}^{N_{cl}} \exp(m_l)}$$

(6)

over all classes. The inference step consists of assigning a label $k$ with the maximum score to each pixel $i$ of the input image $I_x$:

$$y(i) = \max_k (f_k(x; \theta)).$$

(7)

In this paper, we propose a cascaded dilated convolution neural network based framework for part segmentation of a given vehicle. Previously, DCNN was utilized by (Yu and Koltun, 2015), for semantic segmentation to improve the performance over fully convolutional networks (Long et al., 2015). (Yu and Koltun, 2015) dropped pool4 and pool5 from fully convolutional VGG-16 network, and replaced the following convolutions with dilated convolutions, which improved the segmentation accuracy. (Zhou et al., 2017) proposed a branched cascaded architecture for semantic
<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Kernel size</th>
<th>Pad</th>
<th>Stride</th>
<th>Dilation</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1x3x376x376</td>
</tr>
<tr>
<td>conv1_1</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x64x376x376</td>
</tr>
<tr>
<td>conv1_2</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x64x376x376</td>
</tr>
<tr>
<td>pool1</td>
<td>2x2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1x64x188x188</td>
</tr>
<tr>
<td>conv2_1</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x128x188x188</td>
</tr>
<tr>
<td>conv2_2</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x128x188x188</td>
</tr>
<tr>
<td>pool2</td>
<td>2x2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1x128x94x94</td>
</tr>
<tr>
<td>conv3_1</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x256x94x94</td>
</tr>
<tr>
<td>conv3_2</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x256x94x94</td>
</tr>
<tr>
<td>conv3_3</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x256x94x94</td>
</tr>
<tr>
<td>pool3</td>
<td>2x2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x256x47x47</td>
</tr>
<tr>
<td>conv4_1</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x512x47x47</td>
</tr>
<tr>
<td>conv4_2</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x512x47x47</td>
</tr>
<tr>
<td>conv4_3</td>
<td>3x3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1x512x47x47</td>
</tr>
<tr>
<td>conv5_1</td>
<td>3x3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1x512x47x47</td>
</tr>
<tr>
<td>conv5_2</td>
<td>3x3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1x512x47x47</td>
</tr>
<tr>
<td>conv5_3</td>
<td>3x3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1x512x47x47</td>
</tr>
<tr>
<td>fc6</td>
<td>7x7</td>
<td>12</td>
<td>1</td>
<td>4</td>
<td>1x4096x47x47</td>
</tr>
<tr>
<td>fc7</td>
<td>1x1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1x4096x47x47</td>
</tr>
<tr>
<td>fc-final14_p</td>
<td>1x1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1x15x47x47</td>
</tr>
<tr>
<td>upsample_p</td>
<td>16x16</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>1x15x376x376</td>
</tr>
</tbody>
</table>

Table 1: Network architecture describing individual module of the stacked DCNN employed for pixel-wise semantic segmentation.

They divide the image into three branches, (i) objects, (ii) stuff and (iii) parts, and use three different streams for each of them. The loss function is constructed by combining the individual loss of each of the branch. Similarly, the network proposed by (Oliveira et al., 2017) consists of two streams, one for object detection and other for the parts. The part-network operates at twice the scale of the object similar to (Felzenszwalb et al., 2010), to capture finer details of the parts. (Dai et al., 2016) also indicated that capturing parts conditioned over the object masks improves the performance of parts segmentation.

### 7.1.1 Two-stage stage cascading

In this paper, we present a two-stage cascaded DCNN network architecture for object part segmentation. In contrast to (Zhou et al., 2017) Oliveira et al., 2017), we propose to sequentially stack the two cascaded modules, the motivation behind this is to condition the part segmentation on object masks. The first DCNN module in the cascade is responsible for object segmentation at instance level while the second module of the cascade processes every instance separately for estimating the part segmentations. This is analogous to zooming-in every object instance so that the effect of scale is normalized. The issue with training parts conditioned to object segmentation is that the error resulted by the object segmentation module propagates to the parts segmentation module. This issue could be easily seen in the segmentation results presented by (Dai et al., 2016). To tackle this issue we feed initial image along with segmentation bounding box for the part-module. The architecture of our cascaded network is shown in figure 6. Each layer is initialized by the weights of the baseline network provided by (Zhou et al., 2017). We use the pixel-wise softmax loss function given by equation (5), (6) to train both the networks of the cascade. With the proposed refinements in the architecture we are able to obtain a significant improvement in the performance for part-segmentations. For the task of visual navigation we are relying on our networks for accurate part localization and segmentation. However, such tasks require the vehicles to be seen from even oblique poses, which are generally not present in training dataset. Thus, to enhance the segmentation performance, we further propose two refinements as described in the following sub-sections.
7.1.2 Synthetic data augmentation

The PASCAL part dataset consists of images of vehicles taken by persons not explicitly for visual inspection tasks. Thus, there exist numerous viewpoints from which the vehicle has not been previously seen. To cater to the deep network’s need for large dataset with sufficient viewpoints, we augment the training data with images generated by rendering the 3D models of car from several variations of viewpoints and distance. We have manually annotated ten 3D models of car, by assigning the labels for the desired parts to vertices of CAD models. These models are rendered using Gazebo platform (Koenig and Howard, 2004) from 32x6x3 viewpoint variations comprising of 32 yaw variations and 6 height changes and 3 discrete levels in depths to generate our dataset of 5760 synthetic images. It is known that data augmentation improves the performance of a network. For example, it was reported by (Oliveira et al., 2017) that color and spatial augmentation has significantly improved the performance. We further achieve a noticeable improvement in the accuracy pertaining to synthetic multi-view data augmentation. We evaluate the performance upgrade due to the data augmentation in section 8.4.

7.1.3 Multi-view segmentation fusion

In the previous subsection we use data augmentation to prepare our network to foresee various viewpoints, however it is exhaustive to cover all the possibilities in SE(3) space. Any error in segmentation could lead to a collision with the vehicle, since our navigation pipeline relies on segmentation for localization. Therefore, we use optical flow based image warping to transform the part segmentation label for a given pixel in previous image to its corresponding location in current image. Provided the coordinates of \(i\)th pixel \(x_i^j\) in previous camera frame \(F_j\), the warped image coordinates

\[
x_k^j = \phi(x_i^j, \eta) = \pi(T(\eta)\pi^{-1}(x_i^j, Z_j(x_i^j)))
\]

are computed by the warping function \(\phi(\cdot)\), which transforms pixel \(x_i^j\) to the current camera frame \(F_k\) based on the depth \(Z_j(x)\) at \(x_i^j\) in image \(I_j\) and pose \(\eta\) (Ma et al., 2017). Where, \(T\) denotes the camera Euclidean camera transformation matrix and the function \(\pi(\cdot)\) refers to the projective transform induced by the camera at the current pose \(\eta\). However, in our scenario the depth is not known, we tackle this issue by assuming that the motion between two iterations is small. Thus by estimating the optical flow between two subsequent images \(I_j\) and \(I_k\), we can forward warp the segmentation mask from \(I_j\) to \(I_k\).

We further enhance the segmentation by fusing the warped segmentation of previous view with the current segmentation using maximum a-posteriori probability (MAP) estimate thereby producing a multi-view consistent output. (Ma et al., 2017) used warping based multi-view segmentation association followed by Bayesian fusion, which improved the segmentation accuracy. For a pixel \(j\), provided the segmentation class labels for the sequence images up-till current frame \((z_1, z_2, ..., z_t)\), the predicted label for current frame at the current pixel \(j\) is estimated using Bayesian fusion:

\[
p(j|z_{1:t}) = \frac{p(z_t|j, z_{1:t-1})p(j|z_{1:t-1})}{p(z_t|z_{1:t-1})}
\]

Assuming the measurements satisfying independent and identically distributed and using log-odd notation, the update rule simplifies to summation of the likelihood term repeated over sequence of frames:

\[
\log(p(j|z_{1:t})) = \log(p(z_t|j, z_{1:t-1})) + \log(p(j|z_{1:t-1})) - \log(p(z_t|z_{1:t-1}))
\]

7.1.4 Datasets

CNNs often require a large amount of data for training, moreover training a segmentation network is even more exhaustive, since every pixel has to be labeled. Thus, in contrary to classification and object detection datasets, there are very few datasets for image segmentation. Due to increase in categories, part segmentation is highly laborious, as result there are only two principal datasets with focus on part level semantic segmentation having sufficient number of instances (i) Scene Parsing dataset (Zhou et al., 2017): this dataset consists over 15000 instances of cars and persons...
each. However, the part-level labels are not accurate as reported by (Zhou et al., 2017), which introduces unnecessary noise into the system. (ii) PASCAL parts dataset: This dataset is composed of 10,103 images taken for PASCAL VOC challenge and provides part annotations for each of 20 PASCAL classes. For the task of autonomous inspection, we are interested in cars as category. (Zhou et al., 2017) is the only paper that report part-segmentation results on Scene Parsing dataset for the car category, however the ground truth labels for the parts are noisy which refrains from obtaining a valid benchmark. Another paper that showcases results for part segmentation is from (Oliveira et al., 2017), however they have trained their network on the person category. Thus, for the comparison purposes we train our networks on persons as well. Following the dataset selection guidelines described by (Oliveira et al., 2017), we merged labels at two granularity levels: coarse and fine. For the coarse version we consider four labels (head, torso, arms, legs). In the finer version, we have 14 labels discriminating even between the left and right side of the person (head, torso, upper right arm, lower right arm, right hand, upper left arm, lower left arm, left hand, upper right leg, lower right leg, right foot, upper left leg, lower left leg and left foot). Furthermore, similar to (Oliveira et al., 2017), we divide the persons dataset into training and validation sets. For coarse granularity, we use 70% data for training and 30% for testing the network and for fine granularity, we use 80% data for training and 20% for testing the network as used by (Oliveira et al., 2017).

7.1.5 Network training

We train all the DCNN baseline network and its refinements and cascades separately. For all the networks we initialize the weights from the category level segmentation weights provided by (Zhou et al., 2017) and fine-tune the networks on PASCAL parts dataset. The solver is based on stochastic gradient descent since the batch-size selected is 1. The solver’s parameters are as follows: Learning rate is initialized with 0.00003, momentum was constant at 0.9. We use step learning rate policy and the learning rate is reduced to 1/10th after every 70K iterations. We have trained our networks for 400K iterations, which takes around 48 hours on a Titan-X GPU.

7.1.6 Evaluation metric

For dense pixel-wise segmentation, two metrics are reported as a standard practice: (i) pixel-wise accuracy and (ii) intersection over union (IOU). Let $n_{ij}$ be the number of pixels of class $i$ predicted to belong to class $j$, where $t_i = \sum_j n_{ij}$ be the total number of pixels of class $i$. The pixel accuracy is then given by $PA = \sum_i n_{ii} / \sum_i t_i$. The pixel accuracy takes into account also the prediction of background pixels and the background pixels cover of majority of the image. Therefore pixel-wise accuracy is not considered as optimal metric to evaluate the performance of segmentation. Although considering the background pixels, when computing the pixel-wise accuracy is not essentially futile, since the background prediction is important to avoid false positives (Oliveira et al., 2017). Therefore, we evaluate our segmentation network on intersection over union (IOU), along with pixel-wise accuracy. Where IOU is computed as $IOU = \frac{1}{N} \sum_i n_{ii} / (t_i + \sum_j n_{ji} - n_{ii})$.

7.2 Part-IBVS

Position based visual servoing requires accurate estimation of current camera pose in Cartesian space. However, such knowledge is generally not available or could be susceptible to noise as in our scenario. Image based visual servoing on the other hand controls the camera motion directly in the image space. This is achieved by considering some geometrical primitives as visual features such points, lines, regions followed by moving the camera such that the configuration of these features matches a desired configuration. In classical IBVS approaches keypoints is considered as visual features $s = [x_1, y_1, ..., x_N, y_N]$. The control law is then defined by assigning the camera a velocity $v_c$ that minimizes the following objective function:
Figure 7: **Part-servoing pipeline**: We propose dilation based convolutional neural network for extracting pixel-wise part-segmentation. Image moments (centroid and area) are extracted from these segmentation mask as visual feature for IBVS algorithm. The velocity control commands issued by IBVS controller are smoothly tracked by MAV’s local controller, finally the loop is closed by MAV’s vision sensor.

\[
L = \frac{1}{2} (s - s^*)^T (s - s^*)
\]

\[
\Rightarrow v_c = -\lambda \nabla L
\]

\[
\Rightarrow v_c = -\lambda L_s (s - s^*)
\]

Where, \( L_s \) is the interaction matrix that maps feature velocity in the image space to the camera velocity \( v_c \) in Cartesian space. For a 3D point \([X, Y, Z]\) as visual feature, the interaction matrix is given as (Chaumette and Hutchinson, 2006):

\[
L_s = L_x = \begin{bmatrix}
-\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1 + x^2) & y \\
0 & -\frac{1}{Z} & \frac{y}{Z} & 1 + y^2 & -xy & -x \\
\end{bmatrix}
\]

In this paper the objective for Part-IBVS is to attain a desired pose with respect to the given part. However, the part belongs to a novel instance of a car which has not been previously seen, i.e. the instance in the desired image is different from the current instance. Thus, classical IBVS could not be directly employed for achieving our task. Therefore, in this paper we use the segmentation masks provided by our part-segmentation CNN for visual servoing as shown in figure 7. The objective is modified as to place the camera attached to the MAV such that the center of the segmentation coincides with the camera center and the area of the segmentation equals a predetermined area.

To accomplish this modified task we consider image moments as visual features (Chaumette, 2004). Since, our MAV is under-actuated and only 4 degree of freedoms could be controlled. Therefore, we employ following image moments as visual features: (i) centroid of the segmentation and (ii) the area occupied the segmentation in the given image.

\[
s = [x_g \ y_g \ A]
\]

\[
s^* = [0 \ 0 \ A^*]
\]

The interaction matrix for these visual features is given by:
Finally, to control the MAV, we use the exponential decay controller used by classical image-based visual servoing approaches presented in equation (11).

### 7.3 Reverse-PBVS

The objective of this module is to navigate the MAV back to the desired pose of the current side $F_s^*$ from current pose resulted from part-IBVS module $F_i^*$. Since, both $F_s^*$ and $F_i^*$ are in same frame $F_X$, therefore classical PBVS could be directly applied to achieve this objective. Similar to the PBVS module, we rely on the noisy odometry provided by the MAV for state estimation. Thus, the state estimation could be incorrect and an IBVS refinement could be applied for rectifying the state estimation error. However, IBVS refinement will consume additional time for alignment, we therefore skip the IBVS refinement for this reverse-PBVS module, since we observed only small pose errors while inspecting three-to-four parts. When we switch sides, the side-navigation again employs IBVS refinement.

### 8 Experiments and Results

We evaluate our approach extensively through a series of experiments. Initially we assess the performances of individual components of the pipeline on both qualitative as well as quantitative measures. Section 8.1 presents results of the keypoint network from our previous work (Kumar et al., 2017), (Murthy et al., 2017) and in section 8.2 we showcase results of our segmentation network. Sections 8.3 and 8.4 are dedicated to the refinements proposed in this paper to improve the part-segmentation performance. In section 8.5 we evaluate our instance invariant visual servoing in Gazebo based simulation framework. In section 8.6 we extensively gauge the complete pipeline for visual part-inspection. Finally in section 8.7 we implement our approach on Parrot Bebop-2 drone and perform multiple field experiments for our completely autonomous inspection pipeline.

### 8.1 Evaluating Keypoint Network

![Qualitative results showing the 2D keypoint localization performance of the employed architecture. Top 7 keypoints per instance are shown (in accordance with the confidence scores output by the CNN). Discriminative features are extracted consistently across instances, pose variations, and occlusions. The last row shows some failure cases](image)
Herein, we evaluate the accuracy of our 2D keypoint localization framework. To evaluate the performance of our network based on hourglass architecture proposed by (Newell et al., 2016), we use the standard Percentage of Correct Metrics (PCK) used by (Tulsiani and Malik, 2015). In our analysis, we use a very tight threshold of 2 pixels to determine whether or not our keypoint estimate is correct. We compare the accuracy obtained for the car class with the approaches (Li et al., 2016), (Tulsiani and Malik, 2015). Table 2 shows the keypoint localization accuracy obtained by the hourglass network architecture. The results indicate a significant performance boost in the task of keypoint localization, which also improves the performance of the Pose induction module. A few keypoint predictions are shown in figure [8]

<table>
<thead>
<tr>
<th>Approach</th>
<th>PCK(%) (α=0.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Tulsiani and Malik, 2015)</td>
<td>81.3</td>
</tr>
<tr>
<td>(Li et al., 2016)</td>
<td>81.8</td>
</tr>
<tr>
<td>Hourglass architecture (Newell et al., 2016)</td>
<td>93.4</td>
</tr>
</tbody>
</table>

Table 2: Overall performance of keypoint network on the PASCAL 3D dataset with 14 body parts.

8.2 Evaluating Part Segmentation Network

The objective here is to thoroughly validate the segmentation network, since semantic segmentation of parts is the central component of our visual inspection pipeline. A wrong segmentation could lead the MAV in an incorrect direction due to lack of any manual supervision during the test time. Furthermore, our keypoint network requires bounding-box of the car as additional prior for keypoints predictions. This bounding box is computed by the first cascade of our network. Another option while computing the bounding box is YOLO (Redmon et al., 2016), faster-RCNN (Ren et al., 2015) which again requires an additional network that means more computation during test time. Also YOLO and faster-RCNN are not trained for zoomed in part predictions, as result for any zoomed in car they result in highly inaccurate bounding box predictions. These situations arise frequently in our scenario of part inspection which makes YOLO and faster-RCNN not well suitable for this task.

To compare our results with (Oliveira et al., 2017), we test our network for both coarse as well as fine part predictions. Similar to (Oliveira et al., 2017), for the coarse parts dataset we retain four labels namely: and for fine parts we predict all the 14 parts of person category from PASCAL parts dataset. (Oliveira et al., 2017) have trained their network on 80% and 70% images respectively for coarse and fine part predictions. However, (Oliveira et al., 2017) do not provide an extensive list of training and testing images, therefore we randomly select 70% images for training and the remaining 30% images for testing the coarse predictions. Similarly, 80:20 ratio of was used for training and testing the segmentation network on fine parts.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean of all parts</th>
<th>Head</th>
<th>Torso</th>
<th>L arm</th>
<th>LW arm</th>
<th>L hand</th>
<th>LW hand</th>
<th>R arm</th>
<th>LW arm</th>
<th>R hand</th>
<th>LW hand</th>
<th>R leg</th>
<th>LW leg</th>
<th>R foot</th>
<th>LW foot</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN</td>
<td>74.0</td>
<td>66.2</td>
<td>56.6</td>
<td>46.0</td>
<td>34.1</td>
<td>58.9</td>
<td>44.1</td>
<td>31.0</td>
<td>49.3</td>
<td>44.5</td>
<td>40.8</td>
<td>48.5</td>
<td>47.6</td>
<td>41.2</td>
<td>48.8</td>
</tr>
<tr>
<td>Part-Net (Spatial)</td>
<td>81.8</td>
<td>78.0</td>
<td>69.5</td>
<td>63.1</td>
<td>59.0</td>
<td>71.2</td>
<td>63.0</td>
<td>58.7</td>
<td>65.4</td>
<td>60.6</td>
<td>52.0</td>
<td>67.9</td>
<td>60.3</td>
<td>50.0</td>
<td>64.3</td>
</tr>
<tr>
<td>Part-Net (Spatial + Color)</td>
<td>84.0</td>
<td>81.5</td>
<td>74.1</td>
<td>68.0</td>
<td>64.0</td>
<td>75.4</td>
<td>67.4</td>
<td>61.9</td>
<td>72.4</td>
<td>67.1</td>
<td>56.9</td>
<td>73.0</td>
<td>66.1</td>
<td>57.7</td>
<td>69.2</td>
</tr>
<tr>
<td>Ours (Dilated Net baseline)</td>
<td><strong>92.9</strong></td>
<td><strong>84.1</strong></td>
<td><strong>76.2</strong></td>
<td><strong>69.9</strong></td>
<td><strong>67.0</strong></td>
<td><strong>77.2</strong></td>
<td><strong>71.0</strong></td>
<td><strong>68.6</strong></td>
<td><strong>72.2</strong></td>
<td><strong>70.1</strong></td>
<td><strong>67.0</strong></td>
<td><strong>71.3</strong></td>
<td><strong>68.9</strong></td>
<td><strong>72.2</strong></td>
<td><strong>73.5</strong></td>
</tr>
</tbody>
</table>

Table 3: Part-wise comparison results of our baseline dilated network with existing approaches for segmentation on the PASCAL dataset with 14 body parts. We have evaluated our network on human category to benchmark with (Oliveira et al., 2017).

Figure[9] shows the predictions by our network for a few images from PASCAL parts dataset. It can be seen that even for multiple humans and at different scales also our network is able to successfully capture the part information. We
Figure 9: Qualitative results for pixel-wise segmentation of 14 parts on person category from PASCAL dataset. Here we compare the performance of the baseline DCNN network with single DCNN module, which significantly improves the segmentation performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>IOU</th>
<th>PA</th>
<th>Precision</th>
<th>Recall</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN (Long et al., 2015)</td>
<td>57.35</td>
<td>71.79</td>
<td>77.28</td>
<td>67.92</td>
<td>150ms</td>
</tr>
<tr>
<td>SegNet (Badrinarayanan et al., 2015)</td>
<td>45.22</td>
<td>44.82</td>
<td>49.88</td>
<td>80.71</td>
<td>47.7ms</td>
</tr>
<tr>
<td>ParseNet (Liu et al., 2015)</td>
<td>64.25</td>
<td>70.02</td>
<td>74.66</td>
<td>78.95</td>
<td>88ms</td>
</tr>
<tr>
<td>Part-Net (Oliveira et al., 2017)</td>
<td>78.23</td>
<td>85.47</td>
<td>86.00</td>
<td>87.78</td>
<td>225ms</td>
</tr>
<tr>
<td>Fast-Net (Oliveira et al., 2017)</td>
<td>81.92</td>
<td>88.81</td>
<td>88.74</td>
<td>90.04</td>
<td>48.7ms</td>
</tr>
<tr>
<td>M-Net (Oliveira et al., 2017)</td>
<td>78.15</td>
<td>84.95</td>
<td>86.29</td>
<td>87.60</td>
<td>130ms</td>
</tr>
<tr>
<td>M-Net (Heavy)</td>
<td>84.62</td>
<td>91.51</td>
<td>91.47</td>
<td>90.57</td>
<td>345ms</td>
</tr>
<tr>
<td>Ours (Dilated Net baseline)</td>
<td>82.98</td>
<td>96.72</td>
<td>91.61</td>
<td>89.06</td>
<td>161.7ms</td>
</tr>
</tbody>
</table>

Table 4: Overall performance on segmentation on the PASCAL dataset with 4 body parts. Our network performs at par with state-of-the-art in IOU while giving a significant boost in Pixelwise accuracy (PA).

We now evaluate the cascaded architecture presented in this paper on PASCAL parts dataset for car category. Similar to previous sub-section we divide the dataset in 80:20, and use 80% data for training and rest for testing. We compare the performance of sequential cascading with the baseline DCNN for parts segmentation. Figure 10 shows the predictions by both the networks on PASCAL parts and the quantitative evaluation on the same is presented in Table 5. It can...
Figure 10: Qualitative results for pixel-wise segmentation of 11 parts on cars category from PASCAL dataset. Here we show the advantage of cascaded architecture over the baseline, which is a single DCNN module.

Table 5: Results for part-segmentation on the cars category of PASCAL dataset using our baseline Dilated network and the proposed cascaded architecture.

<table>
<thead>
<tr>
<th>Method</th>
<th>Front</th>
<th>Back</th>
<th>Left</th>
<th>Right</th>
<th>Roof</th>
<th>Door</th>
<th>Win-</th>
<th>Head</th>
<th>L</th>
<th>R</th>
<th>Mirror</th>
<th>Wheel</th>
<th>Mean</th>
<th>PA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dilated Net (Baseline)</td>
<td>50.2</td>
<td>23.5</td>
<td>24.1</td>
<td>30.4</td>
<td>17.8</td>
<td>32.7</td>
<td>58.0</td>
<td>34.6</td>
<td>14.9</td>
<td>7.8</td>
<td>58.2</td>
<td>32.02</td>
<td>49.01</td>
<td></td>
</tr>
<tr>
<td>Dilated Net (Cascaded)</td>
<td>86.0</td>
<td>78.0</td>
<td>75.3</td>
<td>68.5</td>
<td>70.5</td>
<td>73.8</td>
<td>80.9</td>
<td>74.9</td>
<td>48.0</td>
<td>56.1</td>
<td>83.9</td>
<td>61.23</td>
<td>86.20</td>
<td></td>
</tr>
</tbody>
</table>

be seen from the table that cascading improves the segmentation performance significantly both in terms of pixel accuracy and IOU. The reason accounting for this improvement is that segmentation is degraded on cars that are at farther distance as could be seen in figure 10. Furthermore, the output from the first module of the cascade is normalized for scale for each car which, when passed to the second module, helps in capturing the fine details of the parts, thus improving the segmentation performance for individual parts significantly on the cars that are farther, and a marginal improvement even on the cars that are closer due to uniformity in the scale. Note that only (Zhou et al., 2017) presents part-segmentation results on car category on ADE20K dataset. However, annotations provided by ADE20K are noisy as reported by (Zhou et al., 2017), therefore we do not directly compare with (Zhou et al., 2017).

8.4 Effect of proposed refinements

In this section, we evaluate the effect on segmentation performance due to the refinements proposed in this paper. Since these proposed refinements in part segmentation networks are based on synthetic data augmentation and multi-view Bayesian fusion, which are not available on public datasets therefore we report results only in simulation.
8.4.1 Synthetic data augmentation

Vehicle part inspection requires viewing cars from oblique angles which are generally not available in images provided by public datasets. We therefore create a dataset of ten 3D CAD models and manually annotate the meshes. These models are rendered using Gazebo platform from 32x6x3 viewpoint variations comprising of 32 yaw variations and 6 height changes and 3 discrete levels in depths to generate our dataset of 5760 synthetic images. This data is augmented to the PASCAL parts dataset for training the network. We have seen noticeable improvement in the network performance especially for oblique views. Figure 11 qualitatively compares the performance of our cascaded network trained only on PASCAL parts dataset with that trained over the multi-view augmented dataset. Although only the data from simulation is augmented still we obtain substantial improvements even in real images of vehicles while executing the complete visual inspection task.

8.4.2 Multi-view fusion

Our pipeline relies on part segmentation, for which the network has been extensively trained on both simulation as well as real data. However, we employ image based visual servoing for navigation which could result in camera viewpoints in SE(3) space which the network may be unable to generalize. Therefore we use warping based multi-view fusion approach discussed in section 7.1.3 for improving the segmentation. The comparative results are shown on simulation environment only, since there is no public dataset with such viewpoint variations for part-segmentation. We showcase the improvement in results both qualitatively (refer figure 16) and quantitatively (refer table 6) by evaluating it together with the visual servoing pipeline. It can be seen that by using the Bayesian fusion, not only the convergence rate increases but also the performance of part-IBVS improves.

8.5 Gazebo simulation experiments

After evaluating the components of our vision pipeline such as keypoint predictions, part segmentation and related refinements individually, we now evaluate our Pose induction and navigation module for the entire visual inspection pipeline in Gazebo simulation framework. The setup comprises of a MAV platform, for which we use RotorS Gazebo (Furrer et al., 2016) library, and twelve 3D CAD models of cars as shown in figure 12. We use one car among them as the template model Y and rest are used as previously unseen novel instances X for the inspection. The simulation environment mimics ideal condition with accurate odometry readings from the MAV and no disturbances from external factors like wind.
8.5.1 Results for instance-invariant PBVS

In this experiment we evaluate the instance-invariant PBVS comprising of Pose induction and Side navigation module in Gazebo simulation. The objective is that starting from a provided random pose $F_0$ and a desired image of a side $I_s^*$, MAV needs to estimate $F_s^*$ using the Pose induction and reach there using the Side navigation module. We select 12 models of the car from our simulation dataset and use them as the current instances $X$. We initially render the template model $Y$ to get the desired views $I_s^*$ corresponding to the sides. Furthermore, the MAV is initialized at a random pose.
Figure 14: **Quantitative results of instance-invariant PBVS pipeline.** (a) The initial translation error between the initial pose of MAV and desired pose for 12 different instances and 3 pose per instance is compared with the final error resulting from our instance invariant PBVS approach followed by IBVS refinement. (b) Initial and resultant error in yaw. Note our approach is able to perform servoing across 12 different instances using only the template model. Furthermore, our approach can tackle large camera transformation with initial error of over 10 m and even non-overlapping scenes where yaw error is greater than 90 degrees.

For quantitatively validating the instance-invariant PBVS, we again consider the 12 models from our synthetic dataset. The MAV is initialized from a random starting point and the objective is to attain the desired pose for one of the sides $F_s^*$. The qualitative results of the simulation are shown in figure [15]. The first column shows the desired view $I_s^*$ of the template instance $Y$. While the second column shows the image captured by MAV from the initial pose $F_0$ of the current instance $X$. The third column represents the final view captured by the MAV using only instance invariant PBVS. Finally, the last column showcases the improvement in the performance of final pose attained by the MAV owing to the IBVS refinement. It can be observed that the resultant image after IBVS refinement is close to the provided desired view. Note that although the odometry of the MAV is accurate, there are still errors in the final pose attained by the MAV, this is due to the large variation in shape between $X$ and $Y$. It can be seen that our approach can easily tackle large variations in shape, texture and camera pose and it even works for non-overlapping scenes, which are challenging scenarios for existing visual servoing approaches.

For quantitatively validating the instance-invariant PBVS, we again consider the 12 models from our synthetic dataset. The MAV is initialized from a random starting point and the objective is to attain the desired pose. Here we place all the models at same location and orientation in the Gazebo world. We further consider 3 desired pose per model which remain constant for all the models so that, we have 12 x 3 experiments in total. For every experiment, we record the initial error which is the between initial pose and the desired pose in translation and yaw. We then use our instance-invariant PBVS framework to navigate the MAV to the desired pose and record the final error which is error between the final pose achieved by our approach and desired pose. It can be seen from figure [14] that our approach is able to converge near the desired pose despite of large initial pose error. Note that since we are servoing across instances there will be a residual error in the final pose.

### 8.6 Evaluating the autonomous inspection pipeline

After exhaustive testing of individual components, we finally validate the performance of the complete pipeline in Gazebo simulation framework. Here we use 13 CAD models of cars in addition to the template model. The MAV starts with random initial pose and servo to the 8 essential parts (4 wheels, 2 headlights, 2 mirrors) for every car. We do not test on taillights because taillights are not part of PASCAL part dataset. With sufficient training sample our approach could easily be extended to other parts or object categories. Here we assume that the workspace is obstacle free and car is visible in the initial random pose. The qualitative results for excluding and including Bayesian refinement are shown in figures [15] and [16] respectively. These results display the final images captured after servoing to a individual parts overlaid with their segmentation masks computed by our network. It could be seen that with our approach the MAV is able to reach the correct parts despite different shapes of essential parts and their configurations.
Table 6: **Quantitative results for complete autonomous part inspection pipeline with and without Bayesian fusion in Gazebo simulation.** Here we compare the performance of our approach on 12 cars by measuring the final area ($A$) of a part captured by MAV in percent of the image area as compared with the desired area ($A^*$) for a part. The $A^*$ (10% for wheels, 2% lights and mirrors) is a tuning parameter that decides zoom level for images. It can be seen that by using the Bayesian fusion both the failure cases reduce as well as the resultant segmentation error $|A - A^*|$ decrease i.e. the final area is closer to the desired area. The failure cases where the MAV is not correctly aligned with the vehicle are reported as '-'. The areas reported in the above table are computed using the ground truth labels of the corresponding parts in the resultant image.

The quantitative results are designed to measure the servoing performance on resultant values of area. Table 6 shows the area in percentage of image size (856x480 px) achieved after servoing to a specific part. The chosen desired area for wheels is 10 % and for mirrors and lights is 2 % of the total area of the image. The table also highlights the improvement in performance due to the multi-view Bayesian fusion. It can be seen that with the fusion the mean of resultant areas are nearer to their desired values. Furthermore, the number of instances where the visual servoing diverges (marked by '-' in the table) are also less after Bayesian fusion.

Once we have evaluated our approach on visual inspection parameters, we now show the performance of our approach on visual servoing metrics, specifically on error in visual features (image area and image centroid), camera pose error, velocity profile and camera trajectory for both PBVS as well as part-IBVS. We consider a single test case and plot
Final area ($A$) of parts achieved with Bayesian on real cars

<table>
<thead>
<tr>
<th>Car Name</th>
<th>RB-wheel</th>
<th>RF-wheel</th>
<th>R-mirror</th>
<th>R-head</th>
<th>L-head</th>
<th>L-mirror</th>
<th>LF-wheel</th>
<th>LB-wheel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accent 1</td>
<td>10.6</td>
<td>7.6</td>
<td>0.8</td>
<td>3.9</td>
<td>1.4</td>
<td>0.4</td>
<td>10.8</td>
<td>12.0</td>
</tr>
<tr>
<td>Accent 2</td>
<td>9.5</td>
<td>7.8</td>
<td>-</td>
<td>-</td>
<td>1.9</td>
<td>0.5</td>
<td>6.1</td>
<td>8.7</td>
</tr>
<tr>
<td>Indigo 1</td>
<td>6.9</td>
<td>12.5</td>
<td>0.9</td>
<td>-</td>
<td>4.6</td>
<td>2.0</td>
<td>9.6</td>
<td>10.0</td>
</tr>
<tr>
<td>Indigo 2</td>
<td>7.9</td>
<td>4.6</td>
<td>1.0</td>
<td>3.3</td>
<td>1.6</td>
<td>6.7</td>
<td>9.8</td>
<td></td>
</tr>
<tr>
<td>Alto 1</td>
<td>11.0</td>
<td>7.7</td>
<td>0.4</td>
<td>2.4</td>
<td>1.4</td>
<td>1.3</td>
<td>7.9</td>
<td>8.3</td>
</tr>
<tr>
<td>Alto 2</td>
<td>10.3</td>
<td>6.4</td>
<td>1.3</td>
<td>2.7</td>
<td>1.1</td>
<td>1.3</td>
<td>13.8</td>
<td>11.5</td>
</tr>
<tr>
<td>Alto 3</td>
<td>11.8</td>
<td>6.8</td>
<td>0.8</td>
<td>2.5</td>
<td>1.4</td>
<td>-</td>
<td>7.5</td>
<td>12.4</td>
</tr>
<tr>
<td>E2O 1</td>
<td>9.8</td>
<td>11.8</td>
<td>0.4</td>
<td>1.5</td>
<td>4.3</td>
<td>0.4</td>
<td>7.3</td>
<td>7.8</td>
</tr>
<tr>
<td>E2O 2</td>
<td>6.2</td>
<td>18.6</td>
<td>1.2</td>
<td>2.7</td>
<td>1.5</td>
<td>1.0</td>
<td>6.4</td>
<td>5.8</td>
</tr>
<tr>
<td>E2O 3</td>
<td>12.0</td>
<td>12.0</td>
<td>0.6</td>
<td>3.4</td>
<td>1.9</td>
<td>-</td>
<td>6.1</td>
<td>8.7</td>
</tr>
<tr>
<td>Mean area</td>
<td>9.8</td>
<td>9.9</td>
<td>1.2</td>
<td>2.5</td>
<td>2.3</td>
<td>1.1</td>
<td>8.2</td>
<td>9.5</td>
</tr>
<tr>
<td>Desired area ($A^*$)</td>
<td>10</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 7: Quantitative results for complete autonomous part inspection pipeline with Bayesian fusion in outdoor experiments. Here we present the resultant measured area after visual servoing for every part on outdoor experiments for 10 runs on 4 cars. Note that $A^*$ for mirrors is modified to 1% due to safety considerations.

8.7 Field experiments on drone and real data

Finally, as a final proof of concept and viability of our approach, we implemented our pipeline on the Parrot Bebop drone with a monocular camera and used a PID controller for tracking the velocity commands generated by our visual servoing controller. In this experiment, we evaluate our approach on four cars parked outdoors using the Parrot Bebop drone. Since MAVs are under-actuated, only 4 DOF control tasks were selected for visual servoing. In the real world, it is difficult to accurately predict the position of a drone. Hence, we report the quantitative results using an approximate odometry provided by the MAV. The CNN forward pass for keypoint prediction as well as part detection was performed using a laptop computer with Core i7 CPU, Nvidia GTX 960M GPU and 16 GB RAM. It takes 800 ms for one iteration to complete on the machine which includes the forward pass of the cascade network, keypoint network and optical flow computation. The image captured by the drone and corresponding control commands generated by the visual servoing controller are exchanged between the system and MAV over Wi-Fi. Note that for inspection tasks the MAV moves to three different sides of the vehicle requiring to fly through large camera transformations between non-overlapping scene with a total autonomous flight time of around 12 minutes without any manual intervention during the flight.

Similar to simulation environment we present both the qualitative (refer figure 17) and quantitative (refer table 7) results for 10 experiments on 4 different cars (Suzuki Alto, Hyundai Accent, Tata Indigo and Mahindra e2o), each
with a random initial pose such that car is visible. It can be seen from the table that our approach is able to converge for more than 90% times and the average error in the area is 0.4%, which is under acceptable limits. A few failure cases are also shown in the figure [17] which occur due to incorrect Pose induction and wrong segmentation produced by the network. In this section, again we present results for a single test case and plot evolution of visual servoing metrics (refer figure [19] over the entire run. The images captured by the MAV for this run are presented in figure [17] column 1. The velocity profile is smooth for the PBVS and reverse-PBVS steps, while it is jittery for the part-IBVS step due to the forward pass delay resulted by our networks. The trajectory visualized from bird eye view highlights the motion of the drone towards all the parts and to-and-fro vantage point. Note that the trajectory of the plotted in figure [19]g) is using the inaccurate odometry of the MAV, which is the reason of discontinuity in the trajectory. It could be seen that our approach is able to attain the correct areas for all the parts despite of such inaccurate odometry and illumination variations.

9 Conclusion

In this work, we have introduced a novel Pose induction framework for visual servoing across instances of an object category. We have further proposed network architectures, which are able to achieve state-of-the-art results for key-point detection and part-segmentation on vehicles. We evaluated our approach through various experiments in Gazebo simulation environment as well as on a MAV. Our approach is able to achieve complete autonomy for visual servoing across instances and is able to cater high intra-category variation in appearance and shape among vehicle category. In contrast to previous visual servoing approaches, our approach is capable of servoing between large camera transformations especially between non-overlapping scenes. Although we have presented the results of our approach only for vehicles, this approach can be easily extended to other object categories as well. In this paper we have also addressed the problem of autonomous inspection of vehicles using a low-cost off-the-shelf MAV. The MAV starts with zero prior knowledge about the scene and still it is able to inspect every part of the vehicle using only a monocular camera and noisy odometry.

10 Acknowledgement

We thank TCS research fellowship for financially supporting this work. We also thank Ravi, Laxit and Saket for allowing us to validate our approach on their car.
<table>
<thead>
<tr>
<th>Car 1</th>
<th>Car 2</th>
<th>Car 3</th>
<th>Car 4</th>
<th>Car 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Start" /></td>
<td><img src="image2" alt="Start" /></td>
<td><img src="image3" alt="Start" /></td>
<td><img src="image4" alt="Start" /></td>
<td><img src="image5" alt="Start" /></td>
</tr>
<tr>
<td><img src="image6" alt="Left" /></td>
<td><img src="image7" alt="Left" /></td>
<td><img src="image8" alt="Left" /></td>
<td><img src="image9" alt="Left" /></td>
<td><img src="image10" alt="Left" /></td>
</tr>
<tr>
<td><img src="image11" alt="Front" /></td>
<td><img src="image12" alt="Front" /></td>
<td><img src="image13" alt="Front" /></td>
<td><img src="image14" alt="Front" /></td>
<td><img src="image15" alt="Front" /></td>
</tr>
<tr>
<td><img src="image16" alt="Right" /></td>
<td><img src="image17" alt="Right" /></td>
<td><img src="image18" alt="Right" /></td>
<td><img src="image19" alt="Right" /></td>
<td><img src="image20" alt="Right" /></td>
</tr>
<tr>
<td><img src="image21" alt="LB-Wheel" /></td>
<td><img src="image22" alt="LB-Wheel" /></td>
<td><img src="image23" alt="LB-Wheel" /></td>
<td><img src="image24" alt="LB-Wheel" /></td>
<td><img src="image25" alt="LB-Wheel" /></td>
</tr>
<tr>
<td><img src="image26" alt="LF-Wheel" /></td>
<td><img src="image27" alt="LF-Wheel" /></td>
<td><img src="image28" alt="LF-Wheel" /></td>
<td><img src="image29" alt="LF-Wheel" /></td>
<td><img src="image30" alt="LF-Wheel" /></td>
</tr>
<tr>
<td><img src="image31" alt="L-Mirror" /></td>
<td><img src="image32" alt="L-Mirror" /></td>
<td><img src="image33" alt="L-Mirror" /></td>
<td><img src="image34" alt="L-Mirror" /></td>
<td><img src="image35" alt="L-Mirror" /></td>
</tr>
<tr>
<td><img src="image36" alt="L-Head" /></td>
<td><img src="image37" alt="L-Head" /></td>
<td><img src="image38" alt="L-Head" /></td>
<td><img src="image39" alt="L-Head" /></td>
<td><img src="image40" alt="L-Head" /></td>
</tr>
<tr>
<td><img src="image41" alt="R-Head" /></td>
<td><img src="image42" alt="R-Head" /></td>
<td><img src="image43" alt="R-Head" /></td>
<td><img src="image44" alt="R-Head" /></td>
<td><img src="image45" alt="R-Head" /></td>
</tr>
<tr>
<td><img src="image46" alt="R-Mirror" /></td>
<td><img src="image47" alt="R-Mirror" /></td>
<td><img src="image48" alt="R-Mirror" /></td>
<td><img src="image49" alt="R-Mirror" /></td>
<td><img src="image50" alt="R-Mirror" /></td>
</tr>
<tr>
<td><img src="image51" alt="RF-Wheel" /></td>
<td><img src="image52" alt="RF-Wheel" /></td>
<td><img src="image53" alt="RF-Wheel" /></td>
<td><img src="image54" alt="RF-Wheel" /></td>
<td><img src="image55" alt="RF-Wheel" /></td>
</tr>
<tr>
<td><img src="image56" alt="RB-Wheel" /></td>
<td><img src="image57" alt="RB-Wheel" /></td>
<td><img src="image58" alt="RB-Wheel" /></td>
<td><img src="image59" alt="RB-Wheel" /></td>
<td><img src="image60" alt="RB-Wheel" /></td>
</tr>
</tbody>
</table>

**Figure 15:** **Qualitative Results for entire part inspection pipeline.** This figure showcases the results of entire pipeline for five cars. The first row shows the images captured from a random initial pose. The images captured by the MAV from $F_i^*$, for all sides are shown in second, third and fourth row. The subsequent rows show the images captured for every part and the corresponding part segmentation masks predicted by our network. Note that despite of the different shapes of cars and starting MAV at random poses, our approach aligns the MAV for visual inspection. The figure also shows some failure cases of our approach where the parts are out of MAV’s field of view (for example, right mirror of car 2 and 4) or segmentation mask is not correct (left mirror of car 1 and 4).
<table>
<thead>
<tr>
<th></th>
<th>Car 1</th>
<th>Car 2</th>
<th>Car 3</th>
<th>Car 4</th>
<th>Car 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td><img src="image" alt="Car 1 Start" /></td>
<td><img src="image" alt="Car 2 Start" /></td>
<td><img src="image" alt="Car 3 Start" /></td>
<td><img src="image" alt="Car 4 Start" /></td>
<td><img src="image" alt="Car 5 Start" /></td>
</tr>
<tr>
<td>Left</td>
<td><img src="image" alt="Car 1 Left" /></td>
<td><img src="image" alt="Car 2 Left" /></td>
<td><img src="image" alt="Car 3 Left" /></td>
<td><img src="image" alt="Car 4 Left" /></td>
<td><img src="image" alt="Car 5 Left" /></td>
</tr>
<tr>
<td>Front</td>
<td><img src="image" alt="Car 1 Front" /></td>
<td><img src="image" alt="Car 2 Front" /></td>
<td><img src="image" alt="Car 3 Front" /></td>
<td><img src="image" alt="Car 4 Front" /></td>
<td><img src="image" alt="Car 5 Front" /></td>
</tr>
<tr>
<td>Right</td>
<td><img src="image" alt="Car 1 Right" /></td>
<td><img src="image" alt="Car 2 Right" /></td>
<td><img src="image" alt="Car 3 Right" /></td>
<td><img src="image" alt="Car 4 Right" /></td>
<td><img src="image" alt="Car 5 Right" /></td>
</tr>
<tr>
<td>LB-Wheel</td>
<td><img src="image" alt="Car 1 LB-Wheel" /></td>
<td><img src="image" alt="Car 2 LB-Wheel" /></td>
<td><img src="image" alt="Car 3 LB-Wheel" /></td>
<td><img src="image" alt="Car 4 LB-Wheel" /></td>
<td><img src="image" alt="Car 5 LB-Wheel" /></td>
</tr>
<tr>
<td>LF-Wheel</td>
<td><img src="image" alt="Car 1 LF-Wheel" /></td>
<td><img src="image" alt="Car 2 LF-Wheel" /></td>
<td><img src="image" alt="Car 3 LF-Wheel" /></td>
<td><img src="image" alt="Car 4 LF-Wheel" /></td>
<td><img src="image" alt="Car 5 LF-Wheel" /></td>
</tr>
<tr>
<td>L-Mirror</td>
<td><img src="image" alt="Car 1 L-Mirror" /></td>
<td><img src="image" alt="Car 2 L-Mirror" /></td>
<td><img src="image" alt="Car 3 L-Mirror" /></td>
<td><img src="image" alt="Car 4 L-Mirror" /></td>
<td><img src="image" alt="Car 5 L-Mirror" /></td>
</tr>
<tr>
<td>L-Head</td>
<td><img src="image" alt="Car 1 L-Head" /></td>
<td><img src="image" alt="Car 2 L-Head" /></td>
<td><img src="image" alt="Car 3 L-Head" /></td>
<td><img src="image" alt="Car 4 L-Head" /></td>
<td><img src="image" alt="Car 5 L-Head" /></td>
</tr>
<tr>
<td>R-Head</td>
<td><img src="image" alt="Car 1 R-Head" /></td>
<td><img src="image" alt="Car 2 R-Head" /></td>
<td><img src="image" alt="Car 3 R-Head" /></td>
<td><img src="image" alt="Car 4 R-Head" /></td>
<td><img src="image" alt="Car 5 R-Head" /></td>
</tr>
<tr>
<td>R-Mirror</td>
<td><img src="image" alt="Car 1 R-Mirror" /></td>
<td><img src="image" alt="Car 2 R-Mirror" /></td>
<td><img src="image" alt="Car 3 R-Mirror" /></td>
<td><img src="image" alt="Car 4 R-Mirror" /></td>
<td><img src="image" alt="Car 5 R-Mirror" /></td>
</tr>
<tr>
<td>RF-Wheel</td>
<td><img src="image" alt="Car 1 RF-Wheel" /></td>
<td><img src="image" alt="Car 2 RF-Wheel" /></td>
<td><img src="image" alt="Car 3 RF-Wheel" /></td>
<td><img src="image" alt="Car 4 RF-Wheel" /></td>
<td><img src="image" alt="Car 5 RF-Wheel" /></td>
</tr>
<tr>
<td>RB-Wheel</td>
<td><img src="image" alt="Car 1 RB-Wheel" /></td>
<td><img src="image" alt="Car 2 RB-Wheel" /></td>
<td><img src="image" alt="Car 3 RB-Wheel" /></td>
<td><img src="image" alt="Car 4 RB-Wheel" /></td>
<td><img src="image" alt="Car 5 RB-Wheel" /></td>
</tr>
</tbody>
</table>

Figure 16: **Qualitative results for entire part inspection pipeline with Bayesian fusion in Gazebo simulation.** The failure cases are reduced to 3 because of the multi-view Bayesian fusion of segmentation masks.
Figure 17: **Qualitative results for complete autonomous part inspection pipeline with Bayesian fusion in outdoor experiment.** Here we present the inspection results for 5 runs on 4 cars. We evaluated our approach in challenging illumination environments such as shadows and heavy sunlight.
Figure 18: **Performance of visual servoing for a single car.** The performance of both PBVS and Part-IBVS on visual servoing measures are presented here. (a) Shows the error in camera pose ($\|F_c - F_s^*\|$) representing translation and yaw for the PBVS. (b) Shows the camera transformation error ($\|F_c - F_s^i\|$) for Part-IBVS to every part. (c,d) Show the error in visual features (image moments for the part-segmentation mask) over time for every part using Part-IBVS. (c) The error between the current area and the desired area of part-segmentation $\text{abs}(A - A^*)$ in percentage of area of image $(M \times N)$ is plotted. (d) The error between centroids current and desired segmentation mask $\| [x_g - x_g^*, y_g - y_g^*] \|$ in pixel is plotted. (e,f) Show the norm of camera velocities for PBVS and Part-IBVS respectively. (g) Shows the camera trajectory for inspection to every part combining our hierarchical PBVS and IBVS approach. The improvement due to IBVS refinement could be clearly seen in (g). Note that the feature error and velocity profiles are smooth and exponentially decay as desired.
Figure 19: Performance of visual servoing for an outdoor run for inspection of actual car using a Bebop drone. (a,b) Show the error in camera pose for the PBVS and Part-IBVS respectively. (c,d) Show the error in visual features (image moments for the part-segmentation mask) over time for every part using Part-IBVS. (e,f) Show the norm of camera velocities for PBVS and Part-IBVS. (g) Shows the camera trajectory for inspection to every part combining our hierarchical PBVS and IBVS approach. Note that the visual feature errors are not very smooth this is due to the optical flow errors in challenging illumination conditions, however they exponentially decay to their desired values. Also, note that the trajectory is plotted using the visual odometry provided by the MAV, which is highly inaccurate. Despite of such noisy odometry our approach is able to accomplish the task of visual inspection with complete autonomy.
References


