Detecting, localizing, and recognizing trees with a monocular MAV: Towards preventing deforestation

by

Shah Utsav Dipakkumar, rishabh.khawad, Madhava Krishna

in


Singapore

Report No: IIIT/TR/2017/-1

Centre for Robotics
International Institute of Information Technology
Hyderabad - 500 032, INDIA
May 2017
Detecting, Localizing, and Recognizing Trees with a Monocular MAV: Towards Preventing Deforestation

Utsav Shah, Rishabh Khawad, and K Madhava Krishna

Abstract—We propose a novel pipeline for detecting, localizing, and recognizing trees with a quadcopter equipped with monocular camera. The quadcopter flies in an area of semi-dense plantation filled with many trees of more than 5 meter in height. Trees are detected on a per frame basis using state of the art Convolutional Neural Networks inspired by recent rapid advancements showcased in Deep Learning literature. Once detected, the trees are tagged with a GPS coordinate through our global localizing and positioning framework. Further the localized trees are segmented, characterized by feature descriptors, and stored in a database by their GPS coordinates. In a subsequent run in the same area, the trees that get detected are queried to the database and get associated with the trees in the database. The association problem is posed as a dynamic programming problem and the optimal association is inferred. The algorithm has been verified in various zones in our campus infested with trees with varying density on the Bebop 2 drone equipped with omnidirectional vision. High percentage of successful recognition and association of the trees between two or more runs is the cornerstone of this effort. The proposed method is also able to identify if trees are missing from their expected GPS tagged locations thereby making it possible to immediately alert concerned authorities about possible unlawful felling of trees. We also propose a novel way of obtaining dense disparity map for quadcopter with monocular camera.

I. INTRODUCTION

Trees are often found amidst agricultural farmland as a means of retaining subsoil moisture, improving ground water levels, and to prevent soil erosion during times of heavy rains. Indeed agro forestry [15], [7] where trees and shrubs are grown amongst crops as a means of sustainable agriculture is becoming widely popular. Also experts have argued for tree crops wherein trees are grown as crops in sloped terrains where normal farming can prove to be untenable [14]. Nonetheless, in many developing nations trees are myopically viewed as a source of immediate income, felled, and sold at marketplace as firewood. In case of more profitable and expensive trees such as sandalwood, there exists organized network of people involved in rapid destruction and smuggling of such species of trees.

In this paper we propose a novel framework capable of detecting, localizing, and recognizing trees. The monocular quadcopter flies in a plantation area teleoperated by the host gathering images along its route. The quad performs a translational maneuver gathering images that mimic a stereo pair. Through such translational maneuver, dense disparity and depth map of the scene is generated. Each such pair of images is further annotated by its GPS locations, the bearing with respect to a global reference, and feature descriptors of the segmented trees in those pair of images. The trees are themselves detected by state of the art Convolutional Neural Network [12] adapted to the current context. The segmented trees are given GPS tags or locations based on the disparity map generated and ego GPS coordinates of the quadcopter and are thus localized with respect to a global reference frame. These are then stored in the database with their GPS locations and feature descriptors.

In the query run, the quadcopter is made to move in the same plantation area though not necessarily along the same route. The quadcopter can enter and exit the site from different locations when compared with the database run. A similar process of tagging the trees with GPS coordinates ensues. Through GPS locations and feature descriptors of segmented trees obtained across the two runs, putative matches are initialized. The optimal association algorithm between the query and database runs makes use of these putative matching score and initializes an Interpretation Tree [2] in a Joint Compatibility Branch and Bound [10] framework.

1In this work, all references to Geotags and GPS Coordinates are in the Geographic Coordinate System of Latitude, Longitude, and Elevation.
The solution to the association problem is then obtained by dynamic programming over the Interpretation Tree. Figures describing the result and the pipeline are depicted in Figure 1 and Figure 2.

To the best of our knowledge there has been no such prior art on localizing trees with globally verifiable positions and further recognizing the presence of such trees in the future through a combination of state of the art vision, machine learning, and data association methods. That this pipeline proposes an initial solution to a very relevant problem in the domains of agriculture and environmental conservation is the quintessential contribution of this effort. Apart from this, we have provided creative solutions to the problem of finding depth to trees and data association. Since monocular reconstruction and depth is often erroneous, the drone is maneuvered to perform a purely translational motion to mimic a stereo pair. The dense disparity obtained from such a maneuver often provides more accurate depth and mapping of trees than a routine motion with a monocular camera.

II. RELATED WORK

The task of detecting the trees have been approached using various techniques, most notably using airborne laser and lidar [6]. Lidar has also been used with MAV for similar purposes [18]. However, sensors like laser scanner and stereo camera are power consuming. This makes them less ideal if the area we need to cover is large. Laser scanners are also heavy in weight, making them unsuitable for many publicly available MAVs. Moreover, laser scanning algorithms are dependent on the density and clustering of trees [17]. We use a single color image in our work obtained through a monocular camera to detect multiple trees using Convolutional Neural Networks.

Estimating the 3D structure of the scene in front of MAVs is a fascinating and vibrant field. The bouquet of methods for it include using Range Sensors, Stereo Camera, Monocular Camera, Structure from Motion, and other feature based techniques. Using active range sensors [4] is not very inspiring option in our task for reasons we mentioned earlier. Monocular SLAM based approach [9] would need to see the scene repeatedly for a lot more than two views in order to get accurate depth of the scene. Also we observe that VSLAM breaks quite often in complex natural environment that we work in. Dense methods such as DTAM [11], [1] are also GPU intensive. Moreover, DTAM is susceptible to breakage due to insufficient feature tracks if MAV pitches or rolls significantly. Whereas our method can provide dense disparity in just two views by creatively moving the monocular camera such that it mimics a stereo rig [16].

Data association in the context of SLAM has been studied extensively [8], [3]. Developing map matching algorithm using delaunay features [5] is not ideal for our work since there are cases when there are only one or two trees in front, or detected, in the scene. Moreover, the detected trees could be collinear or we detect more than three trees. We propose a dynamic programming based approach that exploits the geometry and appearance similarities along with previous trajectory information for tree matching and association.

Object detection has been rich domain in computer vision. Convolutional Neural Networks have dominated the field of detection in recent past [13], [12]. However, there is absence of networks that can detect the trees reliably. In this work, we train a convolutional neural network inspired by [12] using our dataset of trees that can detect multiple trees in given image.

III. SYSTEM: OVERVIEW AND ARCHITECTURE

Our system can be decomposed and explained functionally and behaviorally. Functionally, it is composed of modules, each module achieving particular goal assigned as we describe in next section. Behaviorally, our system can be thought of executing two different types of runs for each location. We define first run as the Database Run and second run (or any run after first run) as Query Run. Each run can be thought of comprising steps, the modules performing some task in each step.

Figure 2 depicts our system pipeline. We begin each step by performing a horizontal translation (moving from right to left) and obtaining image pair at ends of translation. This pair is fed to SPS Stereo [19] which outputs a dense disparity map with respect to left image. Along side, the left image is also passed to our Convolutional Neural Network which is designed to detect trees. As column 3 of our framework shows, the distance between quadcopter and tree
along with bearing between tree and North with quadcopter as center is calculated using depth information and geometry. Using this information along with the current coordinates of quadcopter, we tag each detected trees with their respective GPS coordinates as depicted in last column of figure.

Once the trees are tagged, the next task to perform depends on the run. In database run, we simply store the trees with their coordinates and feature descriptors. Associating some tree with itself in same run is trivial task as it boils down to finding the closest tree in the map of current run. Problems arise in association when it is to perform for different runs. We explain our solution to data association in detail in section 4.4. We manually remove some detected trees in the query run and show efficiency of our system to detect anomalies.

IV. SYSTEM: MODULE DETAILS

Our system consists of four modules: 1) Tree Detection, 2) Depth Estimation, 3) GPS Tag Calculations, and 4) Matching and Association.

A. Tree Detection Module

We use deep convolutional neural network for detecting multiple trees in given image. Our neural network is inspired by [12]. The network uses features from the entire image to predict each bounding box. It divides the input image into a $S \times S$ grid, here we choose $S$ to be 7. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. The network has 24 convolutional layers followed by 2 fully connected layers. The final output of our network is the $S \times S \times 31$ tensor of predictions, where third dimension is computed as (5 prediction for each bounding box) * (2 bounding box predictions for every cell) + (total classes). The tensor contains probabilities of 21 classes in each grid cell. We output the detection for each cell containing higher probabilities than threshold. (The details of the architecture can be found in [12]).

However, our network is different in one major way: we train the network with Pascal VOC 2007, 2010, 2012 and our own dataset of trees. We created a dataset of more than 1000 images, containing more than 4000 trees. We annotated the dataset manually and trained the network to detect original 20 class of Pascal VOC datasets along with trees (hence total 21 classes). We initialize the weights of the network with a model pretrained on imagenet and finetune it for our detection purposes. We start with learning rate 0.001 and SGD learning policy with batch size 64 and steps at 1000, 2000, 5000, 10000, and 16000 iterations. We set momentum 0.9 and decay rate at 0.0005. We train it for 25000 iterations. Leaky-ReLU is used as nonlinear activation.

Figure [3] shows the result of our detection module. The results show that we are able to detect multiple trees in different lighting conditions and at different heights of quadcopter. One important thing to note is that it is possible to detect all the trees in all the image by setting threshold very low. However, that results in large number of false positives and hence we instead work with the nearby trees which network is able to detect with high confidence. This is fine in our work since quadcopter will eventually fly around all trees in different runs for multiple times.

B. Depth Estimation

In this module, we propose a novel way for calculating dense disparity map using a quadcopter equipped with a monocular camera. The quadcopter captures the image at its current position, then performs a horizontal translation, and again obtain an image at the end of translation. This pair of images is fed to stereo correspondence algorithm. We use SPS Stereo [19] in our work. It outputs dense disparity map calculated using the image pair. Figure [4] shows output of our method.

In this work, we only require distance between quadcopter and detected trees. This distance is calculated using the relationship between disparity and depth as follows

$$Z = f \cdot \frac{b}{d}$$

(1)

Here, $f$ is focal length in pixels, $b$ is the baseline computed in meters, $d$ is the disparity of the point in pixels, and $Z$ is the perpendicular distance between camera plane and the plane containing tree. The baseline, $b$, is obtained using the odometry data from IMU of quadcopter. The final distance and bearing between quadcopter and tree are calculated using simple geometry and trigonometry.

C. GPS Tag Calculations

This module geotags all detected trees. We have obtained the distance and bearing of trees from the depth estimation module and we know the current GPS position of the quadcopter. Moreover, as we just performed a horizontal translation, we can calculate the bearing of quadcopter with the North. We use haversine formula to calculate the baseline using the GPS as follows:

$$a = \sin^2 \left(\frac{\Delta \phi}{2}\right) + \cos \phi_1 \cdot \cos \phi_2 \cdot \sin^2 \left(\frac{\Delta \lambda}{2}\right)$$

$$c = 2 \cdot \tan^{-2} \left(\frac{\sqrt{a}, \sqrt{1 - a}}{}\right)$$

$$d = R \cdot c$$

(2)
of observations. The task of association is considerably difficult in second run. Since the initial GPS position fix can be different than what it was in dataset run, the whole map created in query run could be off by some measure. Hence we can not directly use neighbourhoods between trees in different runs and associate the closest one as we did in first run. We solve the association in the second run using the relative geometry between trees and appearance similarity of the trees as we describe next.

Figure 3 shows the typical scenario of query run. The detected trees are shown as green nodes according to their GPS coordinates (left). We want to assign the detected trees to the corresponding trees in the map generated in database run (right). We consider the definition of neighbourhood as area covered with some radius from given reference calculated using GPS information. In the map, red nodes fall outside the neighbourhood and hence are not utilised for further calculations. Among non-red nodes, blue nodes are the ones that lie in the neighbourhood but are not associated with detected trees. The green nodes in both side are the ones supposed to match. These non-red nodes are used to find appropriate association between them and detected trees.

At the core of the association process lies the fact of invariability of the geometries. Though the map created by two runs could be slightly off by some margin in some direction, the relative geometry between trees stays intact. The relative geometry is calculated in terms of bearing and distance among trees. Along with the relative geometry, we also rely on the appearance similarity that we measure through the uniqueness of SIFT descriptor matches.

The data association problem is solved by a modification of the Joint Probability Branch and Bound Algorithm [10]. The JCBB generates various hypotheses association and searches for the largest set that is jointly compatible. The search itself is done incrementally over an interpretation tree I (Figure 4). The interpretation tree effectively prunes away a number of invalid association very early in the hypotheses without having to traverse all branches of the tree for the optimal solution.

Node in I is represented as $n_{ij} = \{(a_i, b_j), P(S, v_{ij})\}$ consisting of two elements, $v_{ij} = (a_i, b_j)$, symbolizing the vertex of the node and $P(S, v_{ij})$ as the unique path from $S$ to

Here, $c$ is the angular distance, and $a$ is the square of half the chord length between the points. $d$ is the final distance between two points. $\phi$ is latitude, $\lambda$ is longitude, and $R$ is the Earth’s mean radius. We compare this baseline with the baseline obtained from odometry data. If they have more discrepancy than some threshold, we discard the current observations and begin again with detection and depth estimation module. Such discrepancy usually points towards low confidence in GPS position.

The bearing between the quadcopter and true north is calculated using

$$\theta = atan2(sin \Delta \lambda \cdot cos \phi_2, cos \phi_1 \cdot sin \phi_2 - sin \phi_1 \cdot cos \phi_2 \cdot cos \Delta \lambda)$$

However, this bearing is calculated considering only quadcopter’s movement and not its viewing angle. Since we are performing the horizontal translation, we add 90 degrees in the calculated bearing, which gives us the bearing between the viewing direction of quadcopter and the North. Now, we add the bearing of individual trees with bearing of quadcopter which will give us final bearing of each tree with the North with respect to quadcopter’s current viewing direction.

Now we have the absolute bearing between tree’s direction from quadcopter and the North. GPS coordinates of quadcopter, and the distance between quadcopter and tree. Using this information, we calculate the GPS tag for each tree using following formula

$$\phi_2 = asin(sin \phi_1 \cdot cos \delta + cos \phi_1 \cdot sin \delta \cdot cos \theta)$$
$$\lambda_2 = \lambda_1 + atan2(sin \theta \cdot sin \delta, cos \phi_1 \cdot cos \delta - sin \phi_1 \cdot sin \phi_2)$$

Here, $\theta$ is the bearing (clockwise from north), $\delta$ is the angular distance $(d/R)$. Hence, this module tags all the detected trees on map with global coordinates.

D. Matching and Association

The matching and association module is a very important module for both the runs and has varying degree of complexity in different runs. In the first run, we achieve the association fairly easily by considering geometric proximity of observations. The task of association is considerably

Fig. 4: Results Showcasing Dense Disparity Generation. The Quadcopter performs a horizontal translation and captures pair of images at end of translation. This pair is used to calculate dense disparity map. Normalized for better visualization.
The path element $P(S, v_{ij})$ is nothing but a unique enumeration of the vertices from the source $S$ to the considered node $n_{ij,p}$ along that path. Please note that while the vertices $v_{ij}$ at a level $i$ are not unique and appear more than once, the node $n_{ij,p}$ associated with the vertex $v_{ij}$ becomes unique due to the path element $P$. The figure shows two such nodes associated with vertex $a_2 b_2$ circled in green and red with their corresponding pattern shown in green and red as well. Define $C_{ij,p}$ as the combination set of all vertices in the path element of the node $n_{ij,p}$ taken two at a time. For example the circled blue node would have its $C_{ij,p}$ as the elements $\{e_{12}, e_{23}, e_{31}\}$, where each element $e_{ij} = \|a_i - a_j\| - \|b_i - b_j\|$. Then define the potential associated with $n_{ij,p}$ as

$$\psi_{ij,p} = \psi_v(v_{ij}) * \psi_e(P_{ij})$$

where $P_{ij}$ is $P(S, v_{ij})$ for short. $\psi_v$ is the potential obtained by matching feature description of the tree $a_i$ in current scan with the tree $b_j$ in the database. $\psi_e(P_{ij}) = \sum_{ij \in C_{ij,p}} e_{ij}$ thus measures the geometric consistency of the current and database observation along the path $P$. The optimal association is thus the optimal path from $S$ to the leaf nodes obtained by solving standard Bellman equation in an incremental dynamic programming framework optimizing over the node potentials. Since I is constructed incrementally, many computations and path searches are successfully pruned. Further in all our experiment the tree grows at-most till 4 levels keeping the combination set associated with each node well within computational limits.

V. Experiments

We have evaluated our framework on a low cost commercial Bebop 2 quadcopter by Parrot. It is equipped with frontal monocular camera, ultrasound altimeter, Ublox Neo M8N GPS module, and onboard IMU. It transmits frames at 640 x 368 resolution at 30 fps to the host device through WiFi connection. The number of maximum visible satellites are 19. We use Robot Operating System (ROS) as middleware for communication with quadcopter. Because of the limited computational power onboard, we perform all calculations on host system.

We conducted experiments on three locations on our campus. Table 1 and Table 2 provides statistics of our experiments for database run and query run, respectively. In the database run, the false positives are the cases in which any tree like structures (such as poles) are detected as trees. The false negatives are the scenarios where the CNN cannot to detect the tree even once in all scenes containing the tree; typically the case of extremely bent shapes or unusual colors. The mean error in Table 1 represents the average difference between actual distance between trees and distance calculated using assigned GPS coordinates in meters.

In the query run, we use the map generated from database run and perform data association. The false positive and false negative in context of query run implies wrong associations. They typically occur when there are errors in GPS data received. Our system can cope up with some false positives in query run given multiple observations of trees (entry 1 of Table 2).

One remarkable capability of our system is to detect the missing trees. We artificially remove some trees from the detection of neural network to simulate the absence of trees. The last two rows of Table 2 represents these experiments.
Once the association run is over and none of the trees are matched with the missing trees from database run, we can raise the flags with the GPS coordinates of missing trees.

Figure 7 depicts the association between small set (for better visualization) of trees in different runs. As we mentioned earlier, the map created in different runs could be translated in some direction depending on the initial GPS location fix. However, our system uses robust features of relative geometry between trees and is able to associate with higher accuracies.

VI. CONCLUSION

This paper showcased a novel pipeline for detecting, localizing, and recognizing trees making use of state of the art machine learning methods along with a novel way to obtain dense disparity map using monocular quadcopter. In particular the trees are detected and segmented leveraging the recent explosive growth in the area of Convolutional and Deep Neural Networks. The segmented trees are localized with GPS tags and get recognized in subsequent query runs through an optimal data association algorithm implemented as a solution to a dynamic programming problem. The method reveals high fidelity detection and recognition over 3 datasets gathered by a monocular quadcopter. Most interestingly the algorithm is able to detect that trees are missing and thus serves as a potential solution for automated deforestation detection. To the best of our knowledge neither has the problem of localizing and recognizing trees with a monocular quadcopter been posed before nor has a solution been presented.

ACKNOWLEDGEMENT

This work was supported by grants received from DeitY, National Program on Perception Engineering, Phase-II.

REFERENCES