2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Vision-based Control of a Quadrotor for Perching on Lines

Kartik Mohta, Vijay Kumar and Kostas Daniilidis

Abstract— **We formulate the position-based visual servoing problem for a quadrotor equipped with a monocular camera and an IMU relying only on features on planes and lines in order to fly above and perch on arbitrarily oriented lines. We show that we are able to compute the orientation of an arbitrarily oriented line, the speed of the robot and its position with respect to the target line using two points at a known distance on the line. The direction of the velocity is derived from optical flow induced by features on a plane in the background Finally, we demonstrate fully autonomous flight and perching using a small 230 gram quadrotor with all the computations running on the robot.**

I. Introduction

Birds can not only cover great distances by flight, but they are also very dexterous in their interactions with the environment. They are able to descend on targets (for example, trees) and exploit perception-action loops to swoop down to grasp fast moving prey. Such capacities would be advantageous for aerial robots. The ability to land on charging stations, to perch on branches to rest and save energy, or to land on skillion or butterfly roofs in surveillance tasks would greatly enhance their capability. To maintain agility, aerial robots must also be able to accomplish these tasks with a minimal sensor payload.

In this paper, we propose new algorithms and their onboard real-time implementation for perching of Micro Autonomous Vehicles (MAVs) on arbitrarily oriented lines in space. Traditionally such problems have been solved by obtaining the position and orientation of the vehicle in an absolute reference system via GPS or a Motion Capture System [1], [2] and prior knowledge of the target in the same system. A relaxation of these assumptions using vision only for the detection of the target and its localization, but GPS for navigation in an absolute reference frame has been demonstrated for autonomous landing of helicopters outdoors [3]. Advanced processing power and progress in visual odometry algorithms facilitated the GPS-free vision based navigation of MAVs [4]. However, because visual odometry is required in addition to the localization of targets, the computational burden is significant and prohibitive for a small MAV. Also, systems that use stereo measurements are constrained by limits on the length of the baseline induced by the small form factor of a MAV. It has thus far been an open challenge to create a system that can:

- overcome computational constraints associated with onboard processing
- avoid completely the external infrastructure for measurement of absolute pose
- rely only on vision to localize the target and estimate its orientation with respect to the camera: the line orientation in 3D
- operate robustly without a stereo/RGB-D camera
- land on arbitrarily oriented lines

Further, we are interested in solving this problem with minimal information about the environment. Remarkably, nature provides several examples of species (birds, insects) that are constrained by their computational power and physical dimensions and are yet able to operate successfully in the presence of all these challenges.

Our proposed system can perch on a line in space in the presence of a background plane. Both the line and the plane can be in any orientation with respect to gravity. We make use of the optical flow induced by the planar background and the angular velocity measurements from the IMU to isolate the purely translational flow that depends only on the normal to the plane and the linear velocity up to a scale. To determine this scale we use the optical flow of two points on the line with known distance to each other. Moreover, these two points enable the computation of the position and direction of the line which serve as the target for perching.

Unlike position-based control approaches which use the starting point as the origin, our position-based servoing operates on a frame whose origin is in the middle between the two estimated features on the line. The fact that these estimates are relative to the vehicle frame enables us to land on arbitrarily skewed lines. This is in contrast to other vision-based approaches in the literature that make use of the desired appearance of a *horizontal* plane, an assumption that greatly simplifies trajectory generation for vehicles landing from a hovering pose. To our knowledge, we are the first to be able to land on a line using two fiducials at a known distance on a line and without any assumption about the orientation of the line.

II. Related Work

Related approaches differ on the assumptions about the environment (horizontal target, distinctive features on the target, features with known position), and the sensors used on-board (monocular, binocular, IMU, GPS).

We will not refer to approaches that observe a UAV from ground cameras or approaches that do not do computations exclusively onboard.

An early approach similar to ours was by Shakernia et al. [5]. They deal only with the planar case and their plane pose estimation differs from ours in our use of an IMU to isolate

K. Mohta, V. Kumar, and K. Daniilidis are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA. {kmohta, kumar, kostas}@seas.upenn.edu. The authors acknowledge the support of the ARL grant W911NF-08-2-0004 and ONR grant N00014-09-1-1051.

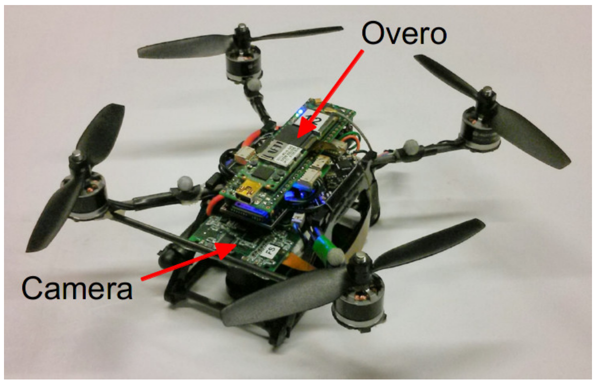

Fig. 1. The 230 gram KMEL Nano+ quadrotor used for the experiments equipped with a Gumstix Overo computer and Caspa camera.

the translational optical flow. The main difference is that they estimate the correspondence between the desired final position of image points and the current position. However, this is feasible only if the target has at least four distinctive features. Similarly, Martinez et al. [6] detect the frame to frame homography and try to align this homography with the one corresponding to the desired normal.

Templeton et al. [7] detect a planar region appropriate for landing using flow induced planar parallax. They use a GPS/IMU estimate as initial pose which is refined using visual bundle adjustment, in our case, prohibitive for the processing power of an MAV.

Saripalli et al. [3] use GPS for pose and vision for detecting the helipad via image moments. They estimate the relative pose of the target via shape moments of inertia only in 2D-space (x, y, θ) and they make use of an IMU for attitude control.

Johnson et al. [8] use tracking of features on the landing surface as input to structure from motion and triangulate those features in order to extract an elevation map and search for an appropriate landing spot. In [9], position of the plane is estimated by fitting a plane to the disparity map of an on-board stereo camera and attitude is estimated from an inertial system.

Several approaches use features on the target only to perform absolute pose estimation in the target coordinate system. For example, Gui et al. [10] use the known positions of 4 infrared lamps on the ground to estimate the pose of a UAV. This is clearly different from our approach in which we obtain relative pose estimates.

Approaches in [11], [12] were the first to introduce pure image based visual servoing for quadrotors by exploiting the structural passivity in the dynamics of image features [13]. Although their approach does not require an accurate depth estimate it necessitates an estimate of the absolute translational velocity which the authors obtain by fusing IMU and visual measurements.

Approaches that use visual SLAM first to estimate pose in a reference frame and then follow a trajectory in this space include [4], [14]. Such approaches are computationally much more expensive than our approach because they have to establish a frame of reference by running a full visual SLAM framework and then compute the desired target's pose in that frame.

Lastly, there is a body of literature in which bio-inspired approaches have been pursued and these are closest in spirit to our work. For example, in [15] only two 1D optical flow detectors are used to compute an approximation of the flow divergence and thus detect obstacles while Green et al. [16] use the Centeye optical flow sensor and build a control loop that tries to achieve the desired optical flow of a horizontal plane.

The work that is most similar to ours is the plane estimation work by Grabe et al. [17], [18]. Our scaled velocity estimation method is based on their work with a small optimization made possible through the assumption of a perspective projection model of the camera. In this paper, in addition to estimating the direction of the velocity using the features on a plane, we use two points at a known distance on a line in order to get the robot pose with respect to the line and the line orientation in 3D. Using these estimates, we are able to do position based visual servoing of a quadrotor in order to fly above an arbitrarily oriented line, as well as land on it.

III. Line Estimation

In this section we formulate the problem of estimating a line with an unknown orientation in space with the goal of obtaining the position and orientation of the line with respect to the robot. The ultimate goal is to plan trajectories that allow a robot to perch on the line.

We make two significant assumptions:

- We can detect two points on the line which are at a known distance from each other. This known distance allows us to recover the otherwise unknown scale.
- There is a plane having features which we can use for computing optical flow such that both the line and the plane are in the field of view of the camera. The line can be skewed with respect to the plane and the plane need not be horizontal.

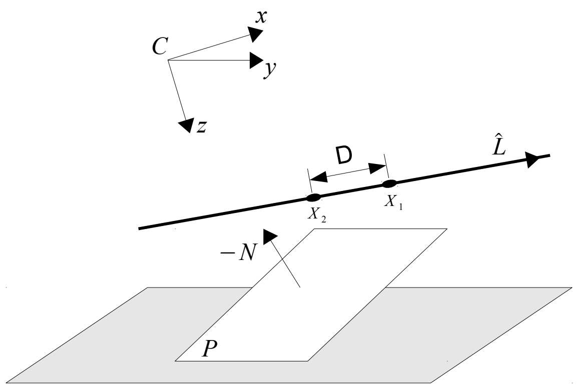

Fig. 2. Camera C moving above the plane P with a line in the front of the plane

The optical flow in the image is related to the camera velocities according to,

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \frac{1}{Z}\begin{bmatrix}-1 & 0 & x\\ 0 & -1 & y\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ v_z\end{bmatrix} + \begin{bmatrix}x\,y & -\left(1+x^2\right) & y\\ \left(1+y^2\right) & -x\,y & -x\end{bmatrix}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}$$

where x and y are the normalized image co-ordinates of the point, Z is the distance of the point from the camera along the principal axis of the camera, and $\boldsymbol{v} = \begin{bmatrix}v_x & v_y & v_z\end{bmatrix}^T$ and $\boldsymbol{\omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T$ are the linear and angular velocities of the camera expressed in the camera frame.

On the quadrotor, the gyroscopes provide an estimate of the angular velocity. Thus we can assume we know the rotational velocity and using this information we can get the purely translational flow, denoted by $\left[\tilde{\dot{x}},\, \tilde{\dot{y}}\right]^T$:

$$\begin{aligned}\begin{bmatrix}\tilde{\dot{x}}\\ \tilde{\dot{y}}\end{bmatrix} &= \begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} - \begin{bmatrix}x\,y & -\left(1+x^2\right) & y\\ \left(1+y^2\right) & -x\,y & -x\end{bmatrix}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}\\ &= \frac{1}{Z}\begin{bmatrix}-1 & 0 & x\\ 0 & -1 & y\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ v_z\end{bmatrix} \qquad (1)\end{aligned}$$

As described in the next section, we can determine the linear velocity of the camera up to a scale factor using the optical flow of features on the plane in the background. Denoting this estimated linear velocity by $\tilde{\boldsymbol{v}}$, we have,

$$\tilde{\boldsymbol{v}} = \boldsymbol{v}/d \qquad (2)$$

where d is the unknown scale factor.

From (1) and (2), we get

$$\begin{aligned}\begin{bmatrix}\tilde{\dot{x}}\\ \tilde{\dot{y}}\end{bmatrix} &= \frac{1}{Z}\begin{bmatrix}-1 & 0 & x\\ 0 & -1 & y\end{bmatrix} d \begin{bmatrix}\tilde{v_x}\\ \tilde{v_y}\\ \tilde{v_z}\end{bmatrix}\\ &= \frac{d}{Z}\begin{bmatrix}x\tilde{v_z} - \tilde{v_x}\\ y\tilde{v_z} - \tilde{v_y}\end{bmatrix}\end{aligned}$$

Let $\mu = \frac{Z}{d}$, so we get

$$\mu\begin{bmatrix}\tilde{\dot{x}}\\ \tilde{\dot{y}}\end{bmatrix} = \begin{bmatrix}x\tilde{v_z} - \tilde{v_x}\\ y\tilde{v_z} - \tilde{v_y}\end{bmatrix} \qquad (3)$$

Here we can determine μ by just using the optical flow of the feature and the scaled velocity determined from optical flow of features on the plane.

Thus, we can find μ_1 and μ_2 for both the tracked points on the line using (3). Let the camera frame positions of these two points on the line be $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. Since the distance between the two points is known, we have $\boldsymbol{X}_1 - \boldsymbol{X}_2 = D\hat{L}$ where D is the distance between the two points and $\hat{L}$ is the direction of the line in the camera frame.

$$\begin{aligned}D\hat{L} = \boldsymbol{X}_1 - \boldsymbol{X}_2 &= \begin{bmatrix}X_1\\ Y_1\\ Z_1\end{bmatrix} - \begin{bmatrix}X_2\\ Y_2\\ Z_2\end{bmatrix}\\ &= Z_1\begin{bmatrix}X_1/Z_1\\ Y_1/Z_1\\ 1\end{bmatrix} - Z_2\begin{bmatrix}X_2/Z_2\\ Y_2/Z_2\\ 1\end{bmatrix}\\ &= Z_1\begin{bmatrix}x_1\\ y_1\\ 1\end{bmatrix} - Z_2\begin{bmatrix}x_2\\ y_2\\ 1\end{bmatrix}\end{aligned}$$

So we have,

$$\begin{aligned}\frac{D}{d}\hat{L} &= \frac{Z_1}{d}\begin{bmatrix}x_1\\ y_1\\ 1\end{bmatrix} - \frac{Z_2}{d}\begin{bmatrix}x_2\\ y_2\\ 1\end{bmatrix}\\ &= \mu_1\begin{bmatrix}x_1\\ y_1\\ 1\end{bmatrix} - \mu_2\begin{bmatrix}x_2\\ y_2\\ 1\end{bmatrix} \qquad (4)\end{aligned}$$

In (4), we know all the terms on the right hand side, hence we can get $\frac{D}{d}\hat{L}$. From this, we can get $\frac{D}{d}$ as the norm and $\hat{L}$ by normalizing. Since we know the distance between the two points, D, we can determine d, and from (2) we are able to calculate the actual linear velocity of the camera. In addition, knowing μ_1, μ_2 and d, we can calculate Z_1 and Z_2 allowing us to find the camera frame co-ordinates, $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, of the two points on the line. We choose the world frame origin to be at the midpoint of these two points oriented such that the Z-axis is pointing straight up opposing the direction of gravity.

IV. Computation of Scaled Velocity

In this section, we introduce the problem of estimating the direction of the camera's body frame velocity using continuous homography. We have a plane P at an arbitrary orientation in the inertial frame and a camera moving in space such that the plane is in the field of view of the camera. Let $\boldsymbol{N}$ be the normal vector of the plane, $\boldsymbol{\omega}$ be the angular velocity, and $\boldsymbol{v}$ be the linear velocity of the camera all in the camera frame. The current position of a point on P (in the camera frame) is represented by $\boldsymbol{X}$ as shown in Fig. 3.

Using the standard equations of motion, we get,

$$\dot{\boldsymbol{X}} = -\hat{\boldsymbol{\omega}}\boldsymbol{X} - \boldsymbol{v} \qquad (5)$$

Let d be the distance of the camera from the plane, so we have

$$\boldsymbol{N}^T\boldsymbol{X} = d \iff \frac{1}{d}\boldsymbol{N}^T\boldsymbol{X} = 1 \quad \forall\, \boldsymbol{X} \in P$$

Putting this in (5),

$$\begin{aligned}\dot{\boldsymbol{X}} &= -\hat{\boldsymbol{\omega}}\boldsymbol{X} - \boldsymbol{v}\frac{1}{d}\boldsymbol{N}^T\boldsymbol{X}\\ &= \left(-\hat{\boldsymbol{\omega}} - \frac{1}{d}\boldsymbol{v}\boldsymbol{N}^T\right)\boldsymbol{X} \qquad (6)\end{aligned}$$

Here the matrix,

$$\boldsymbol{H} \doteq -\hat{\boldsymbol{\omega}} - \frac{1}{d}\boldsymbol{v}\boldsymbol{N}^T \qquad (7)$$

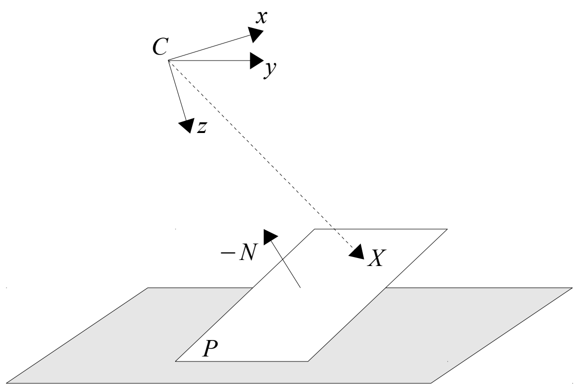

Fig. 3. Camera C is moving above the plane P, with normal N, tracking a point X on the plane

is known as the *continuous homography matrix.*

Now if we look at the image formation equations, we get

$$\lambda \boldsymbol{x} = \boldsymbol{X} \implies \dot{\lambda}\boldsymbol{x} + \lambda\dot{\boldsymbol{x}} = \dot{\boldsymbol{X}} \tag{8}$$

The feature velocity in the image, which is equal to the optical flow using the brightness constancy assumption, is $\boldsymbol{u} = \dot{\boldsymbol{x}}$ and from (6), (7) and (8), we get,

$$\dot{\lambda}\boldsymbol{x} + \lambda\boldsymbol{u} = \boldsymbol{H}\lambda\boldsymbol{x}$$

For a perspective projection camera, we have

$$\lambda = Z = e_3^T\boldsymbol{X} \implies \dot{\lambda} = e_3^T\dot{\boldsymbol{X}} = e_3^T\boldsymbol{H}\boldsymbol{X} = e_3^T\boldsymbol{H}\lambda\boldsymbol{x}$$

where $e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. From this we can easily show,

$$\boldsymbol{u} = \left(I - \boldsymbol{x}e_3^T\right)\boldsymbol{H}\boldsymbol{x} \tag{9}$$

Setting $x_3 = 1$ for the perspective projection camera and stacking the elements of $\boldsymbol{H}$ to form a vector,

$$\boldsymbol{H}^s = \begin{bmatrix} h_{11} & h_{21} & h_{31} & h_{12} & h_{22} & h_{32} & h_{13} & h_{23} & h_{33} \end{bmatrix}^T$$

we can rewrite (9) as

$$\boldsymbol{u} = \begin{bmatrix} x_1\boldsymbol{A} & x_2\boldsymbol{A} & \boldsymbol{A} \end{bmatrix}\boldsymbol{H}^s$$
$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -x_2 \\ 0 & 0 & 0 \end{bmatrix}$$

If we ignore the third component, we have,

$$\boldsymbol{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} x_1\boldsymbol{A} & x_2\boldsymbol{A} & \boldsymbol{A} \end{bmatrix}\boldsymbol{H}^s = \boldsymbol{X}\boldsymbol{H}^s$$
$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -x_2 \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} x_1\boldsymbol{A} & x_2\boldsymbol{A} & \boldsymbol{A} \end{bmatrix}$$

So if we have n points, we can stack these equations of the form

$$\boldsymbol{X}_i\boldsymbol{H}^s = \boldsymbol{u}_i \qquad i = 1, \ldots, n$$

to form a single equation

$$\boldsymbol{\mathcal{X}}\boldsymbol{H}^s = \boldsymbol{B} \tag{10}$$

where the matrices $\boldsymbol{\mathcal{X}} = \left[\boldsymbol{X}_1^T, \cdots, \boldsymbol{X}_n^T\right]^T \in \mathbb{R}^{2n\times 9}$ and $\boldsymbol{B} = \left[\boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_n^T\right]^T \in \mathbb{R}^{2n}$. This equation is slightly different from the one in [17], [18] in that we only have $2n$ rows for $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{B}$ instead of $3n$ which slightly reduces the computational requirements.

From the optical flow of one point in the image, we get two constraints on the elements of $\boldsymbol{H}$. In order to solve for the angular velocity, the normal of the plane, and the linear velocity up to scale, we need $\mathbf{rank}(\boldsymbol{\mathcal{X}}) = 8$, so at least four points in a general configuration are needed [19].

Similar to the case with the line estimation, we make use of information available from gyroscopes on the robot to compensate for the feature motion due to rotation and use the rotation compensated optical flow as given by (1). Since the effect of rotation has been removed, the continuous homography matrix now has the form,

$$\boldsymbol{H} = -\frac{1}{d}\boldsymbol{v}\boldsymbol{N}^T$$

In this case, $\mathbf{rank}(\boldsymbol{H}) = 1$ and $\boldsymbol{N}$ is a unit vector, so we can get $\boldsymbol{N}$ as the first column of the $\boldsymbol{V}$ matrix in the singular value decomposition of $\boldsymbol{H}$ as in [17]:

$$\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

Enforcing the visibility constraint, $N_z > 0$, allows us to avoid any sign ambiguities. Once we get $\boldsymbol{N}$, we can calculate $\frac{1}{d}\boldsymbol{v} = -\boldsymbol{H}\boldsymbol{N}$. Also, when using the rotation compensated optical flow, the matrix $\boldsymbol{H}$ has five degrees of freedom. Hence in (10), we only need three points in a general configuration to compute $\boldsymbol{H}^s$.

So theoretically, only three points are sufficient to compute the plane's normal and the camera's scaled velocity, however due to noise in the sensors, in practice, the model using just three points is not accurate. In order to get better estimates, it is recommended to have a large number of tracked features on the plane and then use RANSAC in order to fit a model to the data. We use RANSAC with the re-projection error defined in [18] as the error metric to decide how well a point fits the model.

V. Experimental setup

A. Hardware

The quadrotor used for our experiments is a Nano+ from KMel Robotics as shown in Fig. 1. The quadrotor has an arm length of 0.085 m and weighs 0.23 kg. It is equipped with a Gumstix Overo FireSTORM computer which is used for all the high-level estimation and control tasks. This is based on a DM3730 digital media processor from Texas Instrument which has a ARM Cortex-A8 micro-processor running at up to 1 GHz along with hardware modules for video input and output. The camera used on the robot is a Gumstix Caspa camera board which outputs a 752×480 image at up to 60 fps. This board plugs into the dedicated camera port on the Overo and outputs a bayer pattern image which is converted to a UYVY image using the hardware image processing pipeline present in the DM3730 allowing us to do image debayering at no CPU cost. The image processing pipeline also has a hardware resizer which can be used for resizing the image without any CPU usage. The Overo also has on-board wifi which is used for sending high-level commands and receiving state estimates for logging purposes. The communication between the Overo and the micro-controller of the robot is done via a UART port.

B. Software

ROS is used as the framework for developing all the software mainly because it provides transparent relocation of the processes across the machines allowing us to run a particular set of nodes on the robot for testing and running others on a separate computer during the development phase.

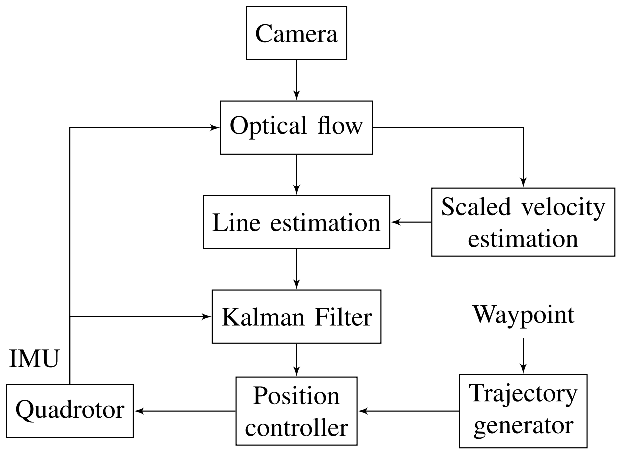

Fig. 4. A block diagram of the various software components in the system. Each block represents a separate ROS node and the arrows represent message flow between the nodes.

We used the OpenCV library [20] for all the image processing related functions. The optical flow calculation is done using a set of AGAST [21] features which are tracked using the pyramidal Lukas-Kanade tracker present in OpenCV. When the system is started, it detects a set of approximately 200 features which are then tracked. As the quadrotor moves, some features go out of the field of view while for some we lose tracking, leading to a gradual decrease in the number of tracked features. If the number of tracked features goes below a certain threshold we reinitialize the whole set of features again. The optical flow from this is then used in the estimation of the line and scaled velocity as described in the previous sections. This complete system is able to run at a rate of $10\,\mathrm{Hz}$ on the Gumstix Overo computer.

C. Control

Even though we get position and velocity estimates from the vision system at the rate of $10\,\mathrm{Hz}$, the quadrotor requires control feedback at a higher rate. We combine the vision estimates with a $50\,\mathrm{Hz}$ IMU output from the quadrotor using a Kalman filter in order to get the estimates at a higher rate in order to control the quadrotor. The Kalman filter state consists of the position, velocity and the accelerometer biases, while we trust the orientation from the IMU.

Once we get the full pose of the camera from the vision system, we can transform it to the robot frame, by using the known camera-robot transform, and then use the traditional controllers for the quadrotor. For our experiments, we used a simple PID controller for controlling the position of the robot which generates the thrust, roll, pitch and yaw commands for the low-level controller running on the quadrotor.

The controller is given desired position, velocity and acceleration commands by a trajectory generator which generates minimum-jerk trajectories for the straight line segments between waypoints. For landing, the trajectory generator switches to a different mode where the procedure consists of generating a trajectory to a point $0.2\,\mathrm{m}$ above the mid-point of the two detected points on the line, stabilizing at that point and then descending slowly with constant velocity for a fixed amount of time before shutting off the motors.

VI. Experimental results

We conducted extensive experiments with different configurations in order to test the robustness of the system. A representative configuration is shown in Fig. 5. The experiments performed to test the performance of the line estimation algorithm include (a) hovering above the line, (b) controlling along some trajectories while keeping the line in the field of view and finally (c) landing on the line. In order to have the robot land on the line and not fall off once it reaches the goal position, we added a $6\,\mathrm{cm} \times 6\,\mathrm{cm}$ patch of Velcro on the line and the bottom of the robot so that once the robot makes contact with the line, it perches in place.

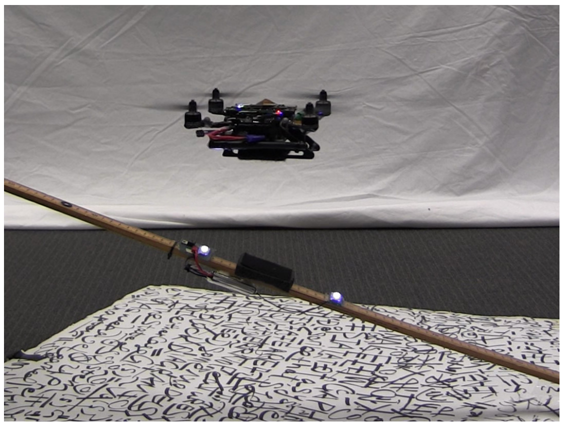

Fig. 5. The quadrotor preparing to land on an inclined line. The video accompanying this paper shows some of our experiments. A higher resolution version can be found at `http://youtu.be/ff5jLO99S9E` or downloaded from `http://mrsl.grasp.upenn.edu/kartikmohta/videos/ICRA2014.mp4`

Fig. 6 shows a comparison of the position and velocity estimates using Vicon, Vision, and Vision + IMU fusion using the Kalman filter when flying above a horizontal line. During the experiment, the quadrotor was commanded to follow trajectories to different positions above the line, as can be seen in the figure. The mean errors in the position and velocity estimates are $0.076\,\mathrm{m}$ and $0.063\,\mathrm{m\,s^{-1}}$ and the standard deviations of these errors are $0.046\,\mathrm{m}$ and $0.064\,\mathrm{m\,s^{-1}}$ respectively.

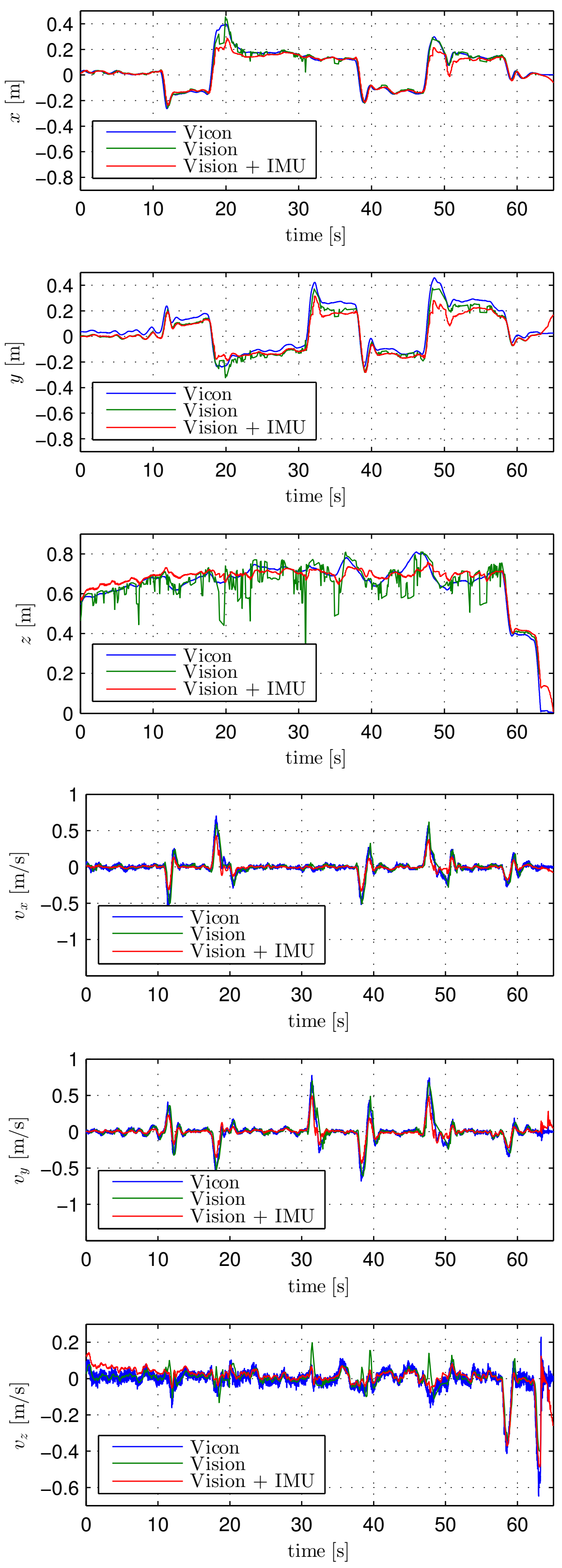

Fig. 6. Estimated position and velocity of the robot from Vicon (blue), Vision (green) and Vision + IMU fusion (red) when flying above a horizontal line

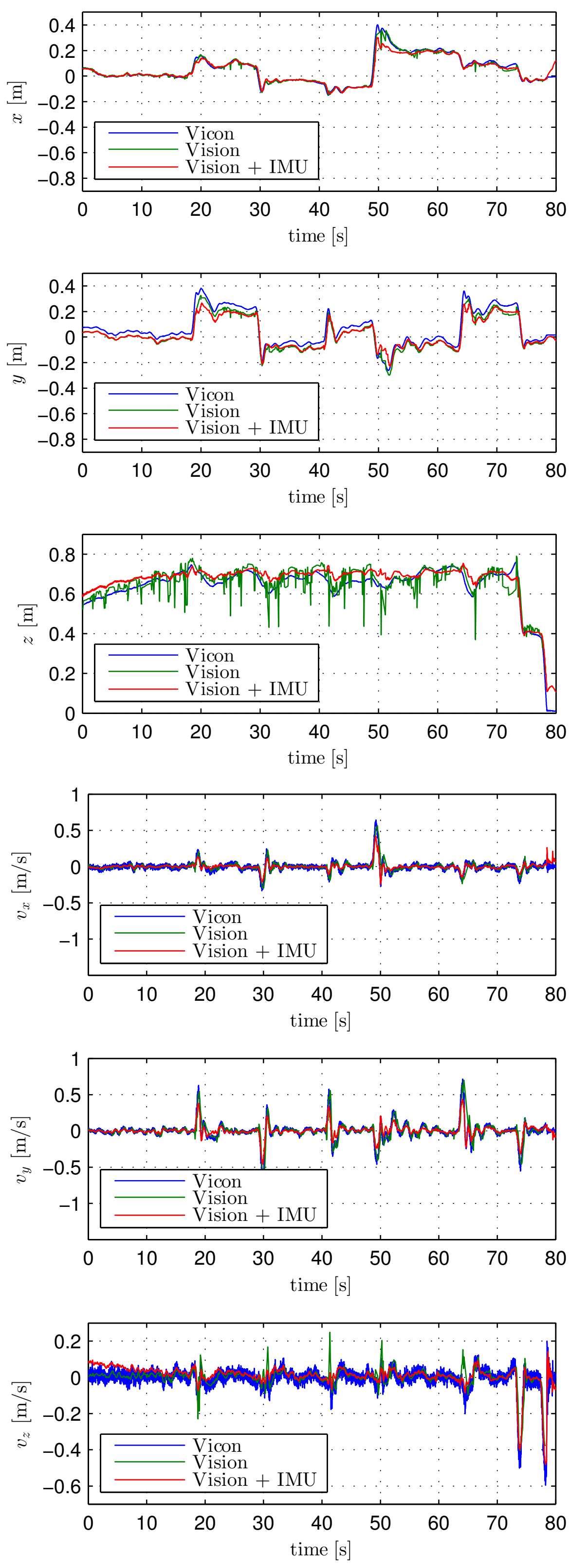

Fig. 7. Estimated position and velocity of the robot from Vicon (blue), Vision (green) and Vision + IMU fusion (red) when flying above a line inclined at approximately 20°

Similarly, Fig. 7 shows the estimates when the robot is flying above a line inclined at approximately $20°$. The performance of the system is very close to the case with a horizontal line. The mean errors in the position and velocity estimates for this case are $0.063\,\mathrm{m}$ and $0.053\,\mathrm{m\,s^{-1}}$ and standard deviations of these errors are $0.033\,\mathrm{m}$ and $0.047\,\mathrm{m\,s^{-1}}$ respectively.

The actual and the estimated line inclination estimate for three different configurations are shown in Table I.

TABLE I

ACTUAL (FROM VICON) AND ESTIMATED LINE INCLINATION

Actual inclination (degrees)	**Estimated inclination (degrees)**
0.079	0.061
19.502	15.336
26.632	20.808

In Fig. 8, we show a scatter plot of the x, y positions of the starting points from where we were able to land successfully on the line. Note that this is only from a subset of all the experiments performed and is intended to show how far we are able to start in the X-Y plane for a successful landing. The starting z position was between $0.5\,\mathrm{m}$ to $0.8\,\mathrm{m}$ for the different points. The x, y positions were limited between $-0.25\,\mathrm{m}$ to $0.25\,\mathrm{m}$ due to the field of view of the camera.

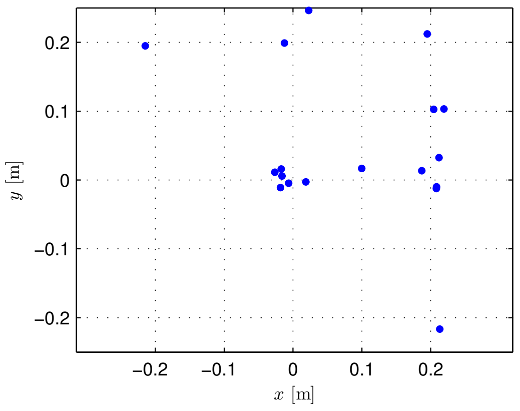

Fig. 8. The starting x,y positions for successful landings. The z starting position varies between $0.4\,\mathrm{m}$ to $0.8\,\mathrm{m}$

VII. CONCLUSION AND FUTURE WORK

In this paper, we presented a new algorithm for estimating the pose and linear velocity of a camera from the optical flow induced by a plane and two points defining a line in space. Recovering the scale factor has been facilitated by the knowledge of the distance between those two points. Using this vision algorithm we have demonstrated flights like hovering above the line, controlling along trajectories, and landing on the line on a small quadrotor with all the computations running on the robot. This work is directly relevant to autonomous flight and perception-action loops required for perching or landing on branches or roof tops. In our future work we aim to address the same problem using dynamics of the visual features directly in the image domain.

REFERENCES

[1] D. Mellinger, M. Shomin, and V. Kumar, "Control of Quadrotors for Robust Perching and Landing," in *Proceedings of the International Powered Lift Conference*, Oct 2010.

[2] R. Cory and R. Tedrake, "Experiments in fixed-wing UAV perching," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2008.

[3] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *Robotics and automation, 2002. Proceedings. ICRA'02. IEEE international conference on*, vol. 3. IEEE, 2002, pp. 2799–2804.

[4] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Robotics and automation (ICRA), 2010 IEEE international conference on*. IEEE, 2010, pp. 21–28.

[5] O. Shakernia, Y. Ma, T. J. Koo, and S. Sastry, "Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control," *Asian journal of control*, vol. 1, no. 3, pp. 128–145, 1999.

[6] C. Martínez, I. F. Mondragón, M. A. Olivares-Méndez, and P. Campoy, "On-board and ground visual pose estimation techniques for UAV control," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 301–320, 2011.

[7] T. Templeton, D. H. Shim, C. Geyer, and S. S. Sastry, "Autonomous vision-based landing and terrain mapping using an mpc-controlled unmanned rotorcraft," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1349–1356.

[8] A. Johnson, J. Montgomery, and L. Matthies, "Vision guided landing of an autonomous helicopter in hazardous terrain," in *Robotics and automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE international conference on*. IEEE, 2005, pp. 3966–3971.

[9] Z. Yu, K. Nonami, J. Shin, and D. Celestino, "3D vision based landing control of a small scale autonomous helicopter," *International Journal of Advanced Robotic Systems*, vol. 4, no. 1, pp. 51–56, 2007.

[10] Y. Gui, P. Guo, H. Zhang, Z. Lei, X. Zhou, J. Du, and Q. Yu, "Airborne Vision-Based Navigation Method for UAV Accuracy Landing Using Infrared Lamps," *Journal of Intelligent & Robotic Systems*, pp. 1–22, 2013.

[11] N. Guenard, T. Hamel, and R. Mahony, "A practical visual servo control for an unmanned aerial vehicle," *Robotics, IEEE Transactions on*, vol. 24, no. 2, pp. 331–340, 2008.

[12] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck, "Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle," *Robotics, IEEE Transactions on*, vol. 25, no. 3, pp. 743–749, 2009.

[13] T. Hamel and R. Mahony, "Visual Servoing of an Under-Actuated Dynamic Rigid-Body System: An Image-Based Approach," *IEEE Transactions on Robotics & Automation*, vol. 18, no. 2, pp. 187–198, 2002.

[14] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 3, pp. 46–56, 2012.

[15] J.-C. Zufferey and D. Floreano, "Fly-inspired visual steering of an ultralight indoor aircraft," *Robotics, IEEE Transactions on*, vol. 22, no. 1, pp. 137–146, 2006.

[16] W. E. Green, P. Y. Oh, and G. Barrows, "Flying insect inspired vision for autonomous aerial robot maneuvers in near-earth environments," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 3. IEEE, 2004, pp. 2347–2352.

[17] V. Grabe, H. H. Bulthoff, and P. R. Giordano, "On-board velocity estimation and closed-loop control of a quadrotor UAV based on optical flow," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, May 2012, pp. 491–497.

[18] ——, "Robust optical-flow based self-motion estimation for a quadrotor UAV," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012, pp. 2153–2159.

[19] Y. Ma, S. Soatta, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision*. Springer-Verlag, 2004.

[20] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, vol. 25, no. 11, Nov. 2000.

[21] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and Generic Corner Detection Based on the Accelerated Segment Test," in *Proceedings of the European Conference on Computer Vision (ECCV'10)*, September 2010.