Autonomous Navigation of Generic Monocular Quadcopter in Natural Environment

Kumar Bipin, Vishakh Duggal and K.Madhava Krishna

***Abstract*— Autonomous navigation of generic monocular quadcopter in the natural environment requires sophisticated mechanism for perception, planning and control. In this work, we have described a framework which performs perception using monocular camera and generates minimum time collision free trajectory and control for any commercial quadcopter flying through cluttered unknown environment. The proposed framework first utilizes supervised learning approach to estimate the dense depth map for video stream obtained from frontal monocular camera. This depth map is initially transformed into Ego Dynamic Space and subsequently, is used for computing locally traversable way-points utilizing binary integer programming methodology. Finally, trajectory planning and control module employs a convex programming technique to generate collision-free trajectory which follows these way-points and produces appropriate control inputs for the quadcopter. These control inputs are computed from the generated trajectory in each update. Hence, they are applicable to achieve closed-loop control similar to model predictive controller. We have demonstrated the applicability of our system in controlled indoors and in unstructured natural outdoors environment.**

I. Introduction

Miniature quadcopters have enormously gained in popularity over past years. Many commercial platforms like Parrot Ardrone and Bitcraze Crazyflie, which only support a monocular camera in terms of payload have entered the market. These platforms are used in applications such as visual surveillance, package delivery, filming and remote farming.

These applications require quadcopter to navigate at low altitude and avoid obstacle autonomously. This is generally achieved using distance sensors such as ultrasound sensors, laser scanners, stereo cameras, Microsoft Kinect or combinations of multiple sensors. However, such sensors lead to increased power consumption and reduced flight time making them unsuitable for miniature quadcopters.

Our work is primarily concerned with developing a navigation framework for such commercial miniature quadcopters, empowering them to navigate autonomously through unstructured environment at low altitude with a monocular camera as the primary sensing modality. The functionality of the proposed approach could be described as a threefold navigation framework; perception, planning and control.

First, the framework is primarily concerned with real-time perception of depth map, using frontal monocular vision as elementary extroperceptive sensor, based on supervised learning. The perception module of the framework appropriately selects the suitable image descriptors-color, texture and orientation as the feature vector and employs SVM (support vector machine) based framework for depth estimation, instead of commonly used graphical model like MRF (Markov Random Field) [17]. Inferencing depth in a multi-label MRF framework is often not suitable for real-time applications such as quadcopter navigation, but dense depth map obtained at 5Hz with the SVM based framework is appropriate for obstacle avoidance and navigation task, described in Section III-A. Similarly, using single view formulation circumvents the need for VSLAM based systems [19], [20] that require robust tracking of features inevitably. Sparse reconstruction of environment obtained from VSLAM system is often not suitable for navigation application.

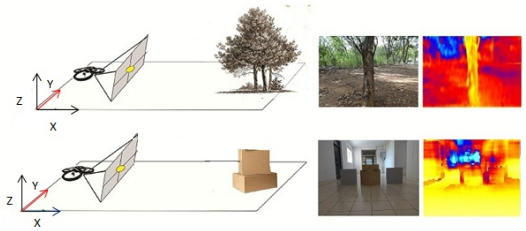

Fig. 1: A novel system level framework which performs perception, planning and control and allows autonomous navigation of generic monocular quadcopter through indoor and outdoor natural environment and avoids impact with obstacles.

Second, after constructing the depth map of the environment, we transform it to Ego-Dynamic Space [21] where robot's dynamic constraints are directly embedded into the spatial representation. The motions computed over transformed space comply with the motion constraints of the robot; improving effectiveness of collision avoidance methods those do not explicitly address these constraints. The quintessential contribution of this effort lies in dovetailing single view reconstruction to perform real time collision avoidance in realistic scenes. This is achieved by efficient characterization of and reasoning about the free areas based on the depth map. Ego Dynamic transformed depth map is clustered based on depth values to obtain 3D contiguous clusters. These clusters are reasoned about their appropriateness for navigation through a *binary integer programming* formulation where each cluster takes a binary values. The centroid of the cluster that minimizes the cost function subject to a set of affine equality and inequality constraints, is chosen as the next best waypoint for the quadcopter to reach its goal. The affine inequalities capture the kinematic and geometric constraints of the quadcopter system. The advantage of this novel waypoint selection method lies in reduced entailments to replan trajectories as shown through

The authors are with Robotics Research Lab, IIIT Hyderabad-500032,Telengana,India. mkrishna@iiit.ac.in, { kumar.bipin, vishakh.duggal }@research.iiit.ac.in.

comparisons tabulated in Section III-C.

Third, the trajectory generation and control module uses *convex programming* technique to optimize polynomial splines. Moreover, exploiting the differential flatness of the system, these polynomial trajectories encode the dynamics and constraints of the vehicle and decouple them from trajectory planning. The framework is fast enough for real time applications and results in a solution which is close to *time optimal*. The main contribution in this component is the relaxation on the need for a time traversal estimate between the starting point and the goal point as prior. The optimality of the proposed framework is aptly verified by the saturation of at-least one of the acceleration or velocity components in Section III-E. It is also very pertinent to point out that there is a vast body of work on sampling based planners-RRT that provide as a robust alternative paradigm for path and trajectory planning. Optimization based approaches on the other hand are capable of giving one shot solution to the goal that includes minimum time trajectories. The experiments and results discussed in Section IV demonstrate its practicality.

II. Related Work

The perception of depth map is crucial to obstacle avoidance and autonomous navigation. The principal existing techniques for depth map estimation include monocular cues, structure-from-motion and motion parallax. Where structure-from-motion and motion parallax approaches need stable tracking and the depth maps often are not dense enough for obstacle avoidance. Moreover, studies done both on human and on animals show that monocular cues like texture, texture gradient, color and haze provide information relevant to estimate depth.

A significant research by Michels *et al.* [13] and Lenz *et al.* [14] presents reactive obstacle avoidance approaches for aerial and ground vehicle which uses above mentioned monocular cues for depth map estimation. The action space was populated by simple reactive actions of moving along the diagonal left, right, forward or upward. Inspired by the above work, our proposed method uses Support Vector Machine (SVM) based model instead of MRF to estimate an accurate depth map in real time with relatively low complexity. Specifically, our estimation problem is to be formulated as a multi-class classification problem.

Most closely related to our approach in the direction of autonomous navigation, a similar work by Alvarez *et al.* [15] presents obstacle avoidance for quadcopter having monocular vision which estimates regulated depth map from set of consecutive images and generates way-point along a horizontal line defined in world co-ordinate in front of current position of quadcopter. This way-point is expected to lead to the largest forward movement without collision. In contrast we propose a binary integer programing based optimal framework for choosing waypoints that operates in the frontal volumetric space of the vehicle. Furthermore, presented work uses DTAM [20], which is susceptible to breakage (due to insufficient feature tracks) when quadcopter

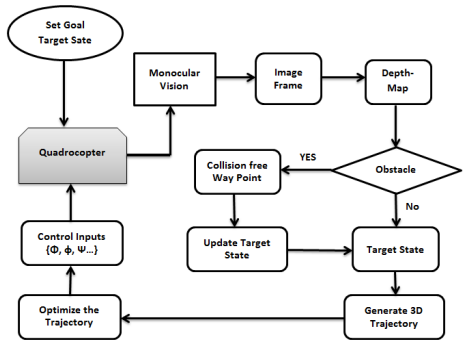

Fig. 2: The process flow chart of proposed navigation framework.

pitches or rolls significantly. Additionally, it requires GPU for calculating dense depth of the scene in real time and does not describe any control strategies to drive the vehicle to output way-point. Our proposed perception module is robust against sudden motion with moderate computation power (*no GPU*) required to achieve real time dense depth of the scene. Similarly, Ross *et al.* [16] presents a state-of-the-art imitation learning techniques to train a linear controller based on visual features. Whereas we present optimization based planning and control mechanism that could drive the vehicle to target state in minimum time.

Considerable number of contributions exists on modeling, design, control [5], [6] and trajectory generation [4] for quadcopter. Similarly, [3] [4] trajectory generation is tightly coupled with complex vehicle dynamics and requires an *a prior selection of the time* to traverse between one way-point and the next. The control module proposed in our current work addresses these issues of "*a prior knowledge of the time*" with help of an objective function formulated as a *convex optimization problem* and provides a solution which is close to time optimal.

Conclusively, we have achieved monocular obstacle avoidance with dense depth maps and limited computational budget ideally suited for payload constrained UAV systems. To the best of our knowledge, such a scheme of way-point reasoning based on an optimization formulation that seamlessly integrates with single view dense depth maps has not been presented in literature before.

III. Navigation Framework

This section describes an overview of the proposed framework consisting of perception, planning and control modules. First, the perception module utilizes supervised learning approach to estimate depth map from a sequence of frames coming from video stream of frontal monocular camera. The learning algorithm has been trained on real camera images labelled with ground truth distances to the closest obstacle. The resulting algorithm learns monocular vision cues that accurately estimate the actual depth of obstacles in the scene. The depth map is then transformed into Ego Dynamic Space and afterwards is processed by local planning module which computes the optimal traversable way-point, which must satisfy the quadcopter motion primitive and physical geometry constraints. Finally, the trajectory planning and control

module generates trajectories and control commands using class of motion primitives-B-Spline [8], [9] by maximizing its velocity profile subject to dynamic feasibility constraints-velocity, acceleration and jerk are met. Fig. 2, delineates the process flow chart of closed-loop navigation framework. At every controller update the initial state of the control problem is updated whereas update of target state depends upon arrival of new way-point from local planning module.

A. Perception

The perception module of the proposed navigation framework follows an approach derived from impressive research by Saxena *et al.* [17] which takes a supervised learning method for depth estimation from a single monocular image. For the training set, Standford University's 3D images (Make3d) database is used. This image database contains 400 images (which includes forest, trees, building etc.) with a resolution of 1704×2272, and 400 corresponding ground truth depth images (resolution of 55×305) by a laser distance scanner having gray values ranging from 1 to 81 meters. In addition, we collected the dataset of 200 color images and their corresponding depth information using Microsoft Kinect sensor which uses the infra-red light to compute the distance from the target objects. Since the Microsoft Kinect is originally designed for the interactive gaming purpose, its best operating range is for objects within depth from 0.7 to 8 meters. Therefore, we have focused on indoor scenes and shaded outdoor scenes for experiments.

In the proposed approach the images are divided into small patches, and a single depth value is estimated for each patch. The features used should capture three types of monocular cues: texture energy, texture gradients, and haze. The texture energy is computed by applying the Laws' mask to the image intensity channel. Similarly, haze is captured by applying a local averaging filter to the color channels. Lastly, to compute the estimate of the texture gradient robust to noise six oriented edge filters are convoluted with the intensity channel, more detail in [17]. The computation of the feature vector for any given patch i in the image $I(x,y)$ could be summarised as follows. The output of each of the 17 (9 Laws' mask, 2 color channels and 6 texture gradients) filter $F_n(x,y)$, $n = 1,\ldots,17$ as : $E_i(n) = \sum_{(x,y)\in patch(i)} |I(x,y) * F_n(x,y)|^k$, where $k = 1,2$ gives the sum absolute energy and sum squared energy receptively, and provides an initial feature vector of dimension 34. The local image features centered on the patch are insufficient to estimate the absolute depth at a patch. Therefore, global information of the image is captured by extracting features at three scale-image resolutions. For each patch, after including features from itself and another two resolution scales, the features vector for estimating depth at a particular patch has $3*34 = 102$ dimensions. Using the labelled image's patches

TABLE I: Quantitative analysis of the depth-map estimation

Method	Average E_{ln}	Average $E_{relative}$	Time$[sec]$
SVM-Linear	0.985	0.815	0.203
SVM-RBF	0.714	0.626	480
MRF	0.605	0.593	90

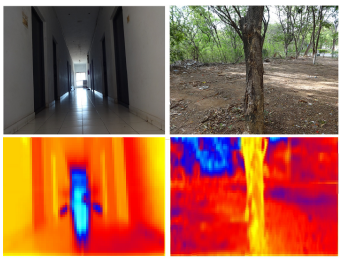

Fig. 3: The image sequence shows frontal camera view of quadcopter and corresponding predicted depth map of operating environment (yellow, red and blue colors represents near, farther and farthest distance from quadcopter)(best view in color).

((number of patches in each image) $\times$ (number of images in training set)$= (55\times305\times(400+200) = 10,065,000)$ and corresponding 102 dimensions features vector as describe above. We address the single image depth estimation as a multi-class classification problem, obtained results shown in Fig. 3.

For reducing the complexity of classification, we quantize the real-valued continuous depth values into labels L_i, $i = 1,\ldots,N$, where the covering range of depth values for label L_i from "near" to "far" is gradually increasing. This corresponds to the fact that the depth discrimination in human vision decreases as the object distance increases. To achieve real-time performance the fast *liblinear* package based SVM-Linear model is used. Additionally, we have experimented with *radial basic function* kernel for SVM model as multi-class classification problem. Although, results obtained produced significant improvement over SVM-Linear; the computation time was too high to service the real-time problem. The estimated depth D_E is transformed into *Ego Dynamic Space* which is described in subsequent section. As shown in Table I, the quality of the estimated depth D_E is measured by the following metrics, [13] (where D is ground truth depth value.):

Depth Error $E_{ln} = |ln(D) - ln(D_E)|$
Relative Depth Error $E_{relative} = \frac{D-D_E}{D}$

B. Ego Dynamic Space Transformation

The Ego Dynamic Space builds a spatial representation where the distances to the obstacles are transformed into effective distances that depend on the robot dynamic constraints: maximum deceleration and response time of the system. Trajectory planning without considering the dynamic constraints of the robot is susceptible to collision. Whereas, our proposed planning module calculations are directly rooted to *Ego-dynamic Space* [21]; it implicitly takes the quadcopter dynamics into account, assuring feasible motion execution.

$$d_{obs} = D_E - avg(e^{E_{ln}}); \quad d_{obs} = d_{eff} + d_{brake} \tag{1}$$

$$d_{eff} = v.\Delta t \quad \text{and} \quad d_{brake} = v^2/2.a_b \tag{2}$$

$$d_{eff}^2/(2.\Delta t^2.a_b) + d_{eff} - d_{obs} = 0 \tag{3}$$

where D_E is the distance of the obstacle from quadcopter derived from depth map, $avg(e^{E_{ln}})$ represents the average

error of the estimated depth map and d_{obs} incorporates error in estimated depth. d_{eff} is the *effective distance*: the maximum distance that the quadcopter can travel at a constant velocity v during the period Δt and d_{brake} minimum distance (applying the maximum deceleration a_b) for stopping the quadcopter safely before hitting the obstacle. We obtain d_{eff}, where $d_{eff} \leq 0$ represents imminent collision thus the *Ego-Dynamic Space Transformation* of d_{obs}:

$$EDT\{d_{obs}\} \quad \longrightarrow d_{eff} = a_b.\Delta t^2\left(\sqrt{1+\frac{2.d_{obs}}{a_b.\Delta t^2}}-1\right) \tag{4}$$

C. Planning

Consequently, the planning module of the proposed navigation framework computes the traversable way-point from $EDT\{d_{obs}\}$. Each entry in $EDT\{d_{obs}\}$ represents depth of specific patch of the current frame and it is clustered into M segments. Any of these segments $S_i, \forall i = 1,\ldots,M$ represents probable next way-point for the quadcopter. Each segment contains inherent properties : radius (r_i), centroid (o_i) and distance of centroid along line of sight X-axis (d_i) which would be used further for determining the optimum next way-point (reference axis shown in Fig. 4)

In the following section, selecting optimum segment as next way-point from $S_i, \forall i = 1,\ldots,M$ for collision free motion is formulated as a *binary integer programming problem*. Quadcopter motion kinematics and shape define the consideration and applicability of various constraints which would be discussed at various points in the text.

1) Objective Function: Objective of defining optimum criteria for way-point selection is to ensure collision free and smooth motion for the quadcopter. The motion of quadcopter along the line of sight is most disciplined, hence movement in the field of view plane (in $Y-Z$ plane) needs to be minimized for smoother trajectories. Inspired by human intuition, which prefers moving towards the nearest and maximum size free space to avoid frontal obstacle, corresponding cost function is defined as:

$$C_i = \frac{\rho_i}{r_i}; \forall i = 1\ldots M \tag{5}$$

$$\min_{\kappa} \quad C^T\kappa \tag{6}$$

where ρ_i is the distance of corresponding segment centroid from the line of sight (in $Y-Z$ plane), r_i is its radius, and C is the coefficient vector of the cost function meeting the objectives of optimum way-point selection and κ is a binary integer vector of length M with exactly one index allowed value 1 at any given time. In order to achieve way-point resulting in collision free and smooth trajectories, respective objective function needs to be minimized subject to kinematics and physical geometry constraints of quadcopter.

2) Constraints: Quadcopter dynamics allow it to fly at varying speeds and along three axis simultaneously. Thus constraints based on these dynamic parameters need to be taken into consideration for ensuring collision free trajectory. These constraints mentioned above can be expressed by affine functions of κ in the form of equality and inequality constraints:

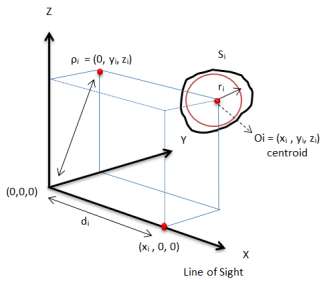

Fig. 4: The quadcopter (0,0,0) has local co-ordinate system (X,Y,Z), where X points towards line of sight of frontal camera and $Y-Z$ plane corresponds to Field of View which is perpendicular to line of sight.

$$A_{eq}\kappa = b_{eq} \tag{7}$$

$$A_{in}\kappa \leq b_{in} \tag{8}$$

We introduce these constraints in the following paragraph based on threshold distance to collision, quadcopter kinematics and shape.

a) Collision threshold: Current velocity of the quadcopter is taken into consideration for determining the minimum threshold distance for collision avoidance. Segment $S_i, \forall i = 1,\ldots,M$ whose distance from quadcopter current position is below the threshold is considered with high collision probability or as an obstacle. $\alpha = v_t.t_t$, where α is the minimum threshold distance for collision probability with v_t the current velocity, distance of centroid along line of sight d_i (along X-axis), and t_t is total update-cycle time of navigation framework, which includes computation time and response time of the system. Those segment centroids whose distances from the current location of the quadcopter are greater than α are preferred. These are posed as the following inequality constraints:

$$\frac{1}{d_i} \leq \frac{1}{\alpha}; \forall i = 1\ldots M \tag{9}$$

b) Expanse: Quadcopter also requires minimum free space along the field of view plane (in $Y-Z$ plane) depending upon its shape and motion towards next way-point. As mentioned earlier motion along the line of sight is most disciplined; in case motion along the field of view plane is desired then drift and inconsistent motion needs to be taken into consideration while determining minimum traversable space which is shown in Fig. 5. This inequality constraint adds M constraints to the system one for each segment.

$$\eta = \delta + \max(\sigma_y, \sigma_z) \tag{10}$$

$$\frac{1}{r_i} \leq \frac{1}{\eta}; \forall i = 1,\ldots,M \tag{11}$$

where η is the minimum desired threshold radius of segment S_i and δ constant parameter dependent on the shape of quadcopter. σ_y and σ_z are standard deviation of error in state estimation along the Y axis and Z axis respectively.

TABLE II: Optimum waypoint selection and Random waypoint selection method

Method	Re-palnning Required [*Average*]
Random	8 [*times*]
Optimum	3 [*times*]

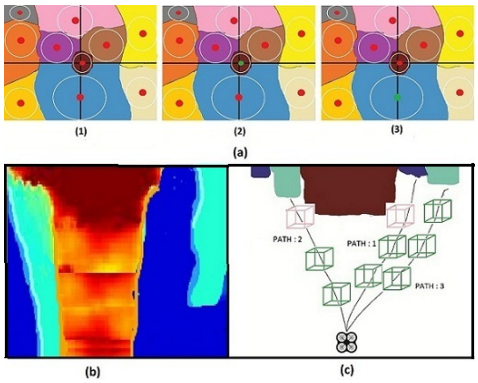

Fig. 5: Optimum way point selection. (a_1) Various candidate segments in scene represented by unique colors and their corresponding centroids (red dots). (a_2) Non-traversable way point selected (green dot) if expanse constraint is removed from formulation. (a_3) Optimum traversable selected segment as way point (green dot) for Spatial Windows analysis .(b) Simulated depth image. (c) Top-view of simulated depth image, paths are verified for collision using Spatial Windows method, path 1 and 2 have possibility of collision (shown in pink cube). Path 3 shows collision free navigation (shown in green cube).

c) *Binary index:* To ensure that only one segment is selected and all other binary variables are anchored to zero which entails the following equality constraints:

$$A_{eq}\kappa = b_{eq} \tag{12}$$

where A_{eq} is a vector of length M with all elements with value 1 and b_{eq} unit value 1. Finally, the overall optimization problem is represented as:

$$\min_{\kappa} \quad C^T\kappa; \quad subject \quad to \quad A_{eq}\kappa = b_{eq}; \quad A_{in}\kappa \leq b_{in} \tag{13}$$

D. Spatial Window

Once the traversable waypoint is selected, we generate path between the current position and waypoint using Bezier polynomial. The generation of trajectory begins with interpolation of $n+1$ points $(q_0,\ldots,q_n)$ from initial to target position while ensuring end conditions are met.

$$q_i(\tau) = \sum_{i=0}^{n} \binom{n}{i} \tau^i (1-\tau)^{n-i} p_i, \forall \tau = 0,\ldots,1 \tag{14}$$

where $\binom{n}{i}$ are binomial coefficient and $\binom{n}{i}\tau^i(1-\tau)^{n-i}$ are the *Bezier/Bernstein* basis polynomials, and p_i are scalar coefficient called *control points* calculated by solving the equation (14) with boundary conditions (i.e. current position and waypoint).

The paths are verified for collision with obstacles using *Spatial Window methodology*. Which is defined as the set of all possible quadcopter locations that can be attained by motion command along the generated trajectory within the admissible dynamic interval. The corners of the *Spatial Windows* are given by:

$$(x,y,z)^i_{max} = (x,y,z)^i + \sigma_{(x,y,z)} \quad and \quad (x,y,z)^i_{min} = (x,y,z)^i - \sigma_{(x,y,z)}$$

where $(x,y,z)^i$ are possible quadcopter locations with corresponding state estimation error σ_x, σ_y and σ_z which is determined experimentally. This is necessary as binary optimization based way point selection only establishes collision free at the start and end of the trajectory. Hence Spatial Window is created along the path and in case obstacles are detected, corresponding path is treated with the possibility of collision and considered as non-traversable by quadcopter which is shown in Fig. 5. If path to selected waypoint is determined non-traversable, corresponding segment centroid is marked non-traversable and next best segment is selected. The advantages of optimum waypoint selection method over randomly selected waypoints is shown in Table II. Over several experiments done it is seen on an average the optimal waypoint selection requires lesser number of replanning than the random method. Once collision free path is found, we generate trajectory and control for the quadcopter. In continuation, a novel method for generating minimum time trajectory towards next way-point is described in the following section.

E. Trajectory Planning and Control

In this section, the trajectory planning problem has been formulated as a convex optimization problem where an objective function which has to be minimized is defined in parametric form, containing no time information. Polynomial splines which are numerically stable for higher order including large number of segments are used in our formulation. The proposed method could be used to generate trajectory and controls for any arbitrary type of vehicle but specifically modified for quadcopter motion.

1) Trajectory Planning: The selected optimum collision free path in the previous section is segmented into $q_{0,l},\ldots,q_{n,l},\ l \in \{x,y,z\}$. Trajectory conforming over points $q_{i,l}, \forall i = 0,\ldots,n$ is generated, while keeping into consideration the kinematics and dynamics constraints of quadcopter. The motion of quadcopter is modeled using polynomial based spline, B-Spline:

$$s(u) = \sum_{j=0}^{m} P_j B^p_j(u), \quad u_{min} \leq u \leq u_{max} \tag{15}$$

where P_j are the scalar de Boor control points, B^p_j are the B-spline basis functions of degree p. Bezier polynomial with strict relation between curve degree and number of control points made it inapt for our formulation. B-Spline polynomial a generalization of Bezier polynomials overcomes this limitation and with strong convex hull and inherent continuity (C^{p-1}) properties, made it an optimum choice for our formulation. To avoid high force and moments on quadcopter during motion, jerk continuous quintic B-spline($p = 5$) is used. To generalize the analysis, a non dimensional time variable, u, is defined as

$$t_i = \lambda_i \cdot u_i, \qquad \forall i = 0,\ldots,n \tag{16}$$

where $u = [u_0,\ldots,u_n]$ and λ_i is a scalar time scaling variable. The B-Spline function "s" is then defined as:

$$s(t_i) = s(\lambda_i \cdot u_i), \qquad \forall i = 0,\ldots,n \tag{17}$$

After, differentiating the equation (17), velocity, acceleration and jerk profile of motion are obtain:

$$v(t_i) \quad = \dot{s}(t_i) = \frac{\dot{s}(u_i)}{\lambda_i} = \sqrt{\dot{s}_x(t_i)^2 + \dot{s}_y(t_i)^2 + \dot{s}_z(t_i)^2} \tag{18}$$

$$a(t_i) \quad = \ddot{s}(t_i) = \frac{\ddot{s}(u_i)}{\lambda_i^2} = \sqrt{\ddot{s}_x(t_i)^2 + \ddot{s}_y(t_i)^2 + \ddot{s}_z(t_i)^2} \tag{19}$$

$$j(t_i) \quad = \dddot{s}(t_i) = \frac{\dddot{s}(u_i)}{\lambda_i^3} = \sqrt{\dddot{s}_x(t_i)^2 + \dddot{s}_y(t_i)^2 + \dddot{s}_z(t_i)^2} \tag{20}$$

where $v(t_i)$, $a(t_i)$ and $j(t_i)$ are the velocity, acceleration and jerk of quadcopter at time t_i. Similarly $\dot{s}_x(t_i)$, $\ddot{s}_x(t_i)$, $\dddot{s}_x(t_i)$ and others represent individual velocity, acceleration and jerk components along each axis.

Following the approach described in [2], for calculation of a minimum time motion profile, the proposed method must bring velocity profile $v(t)$ of the system as close possible to maximum velocity limit v_{max} subject to maximum acceleration a_{max} and maximum jerk j_{max} constraints provided by physical limit of vehicle dynamics. The resulting objective function needs to be minimized is given by :

$$\Omega = \sum_{i=0}^{n} \left[\frac{\dot{s}(u_i)}{\lambda_i} - v_{max} \right]^2 \tag{21}$$

Furthermore, the equation (21) can be expressed as a standard form of quadratic function:

$$\Omega = f_o(\chi) = \chi^T P \chi + q^T \chi + r \tag{22}$$

where $\chi = \left[\frac{1}{\lambda_0}, \frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_n}, \right]$ is vector of optimization variables.

2) Constraints: Given initial and target states introduce constraints derived from both initial and final position, velocity and acceleration. These constraints can be expressed by affine functions of χ in the form of equality and inequality constraints those would be discussed subsequently:

$$A_{eq}\chi = b_{eq} \tag{23}$$

$$A_{in}\chi \preceq b_{in} \tag{24}$$

a) Initial and Final States: As stated above, position, velocity and acceleration are known for the initial and final state provided by Navigation Framework, which introduces 14 equality constraints. Those are included in the matrix A_{eq} and b_{eq}.

b) Velocity: This constraint introduce $n-1$ constraints and are included in matrix A_{in} and b_{in}.This condition reads as :

$$v(t_i) = \frac{\dot{s}(u_i)}{\lambda_i} \leq v_{max}, \quad \forall i = 0, \ldots, n. \tag{25}$$

c) Acceleration: This constraint is decoupled by allocating a constant maximum allowable acceleration magnitude to each co-ordinate separately ($a_{max,x}$, $a_{max,y}$, $a_{max,z}$), [1]. These maximum accelerations are selected such that they fulfil the following constraints:

$$\sqrt{(a_{max,x})^2 + (a_{max,y})^2 + (a_{max,z})^2} \leq a_{max} \tag{26}$$

$$g + a_{max,z} \geq a_{min} \tag{27}$$

These are compliant to the affine formulation presented above in equation (24) and contributes $(3 \times n - 1)$ additional inequality constraint to matrix A_{in} and b_{in}. These inequalities are represented as:

$$\ddot{s}_l(t_i) = \frac{\ddot{s}_l(u_i)}{\lambda_i^2} \quad \leq a_{max,l} \quad l \in \{x,y,z\}, \forall i = 0, \ldots, n. \tag{28}$$

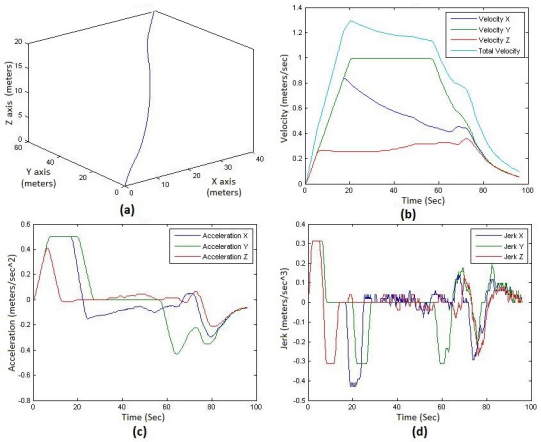

Fig. 6: The simulation result of minimum time trajectory with velocity and acceleration profile is obtained by solving (32). It can be seen that the velocity and acceleration level constrained ($v_{max} = 1.30m/s$ and $a_{max,l} = 0.5m/s^2, l \in \{x,y,z\}$) are perfectly satisfied whereas jerk profile remains bounded ($j_{max,l} = 0.5m/s^3, l \in \{x,y,z\}$). Similar to the usual time optimal planning with acceleration input, at least one of the acceleration component is near saturation, when the velocity profile is not the limit curve. The acceleration profile is zero when the velocity profile is at the limit curve. The simulation result is obtained on Gazebo simulator for the Parrot AR.Drone

d) Jerk: In addition, bounded jerk implies continuity in the acceleration. This contributes another $(3 \times n - 1)$ which are included in matrix A_{in} and b_{in}. These inequalities are represented as:

$$\dddot{s}_l(t_i) = \frac{\dddot{s}_l(u_i)}{\lambda_i^3} \leq j_{max,l} \quad l \in \{x,y,z\}, \forall i = 0, \ldots, n \tag{29}$$

where $j_{max,l} = \frac{1}{\sqrt{3}} f_{min} \omega_{max}, \quad l \in \{x,y,z\}$ which is upper bound on the allowable jerk per axis, f_{min} the minimum collective thrust of quadcopter and maximum angular rate of rotation ω_{max} as described in [6]. The overall optimization problem is represented as :

$$\min_{\chi} \quad \Omega; \quad subject \quad to \quad A_{eq}\chi = b_{eq}; \quad A_{in}\chi \preceq b_{in} \tag{30}$$

3) Minimum Time Trajectory: Optimization problem (21) with equality and inequality constraints (23), (24) is quadratic and convex hence it converges to global optima extremely fast, which results in a vector of *optimal* time scaling parameters $\chi = \left[\frac{1}{\lambda_0}, \frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_n}, \right]$. These are substituted in equation (16) to obtain the minimum scaled time $t_0, \ldots, t_n$ for the planned trajectory. Thus, total time of trajectory:

$$T = \sum_{i=0}^{n} \alpha_i \tag{31}$$

where $\alpha_i = t_{i+1} - t_i$ and T is feasible [12] *minimum trajectory time*, calculated from trajectory optimization whereas similar work [3], [4] require estimate of T as priori for trajectory generation. There is no inherent relation between $\frac{1}{\lambda_i}, \frac{1}{\lambda_j}; \forall i \neq j$, hence may cause uneven stretching of time interval α_i. Therefore dynamic time scaling may cause velocity, acceleration or jerk discontinuity at joints of B-Spline segments [10], [11]. To remove such incidentally introduced discontinuities, B-spline trajectory parameters have to be recomputed (i.e. de Boor control points with time parameters

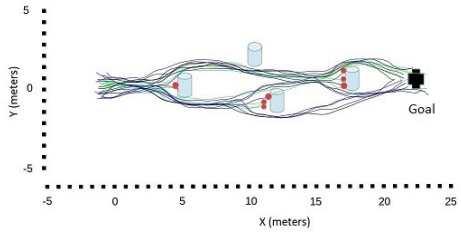

Fig. 7: A set of 25 experimental flights in outdoor natural environment. Trajectories in blue and green colors represent with and without *Ego Dynamic Space Transformation* of estimated depth respectively (red dot represents collision and cylinder corresponds to tree obstacle).

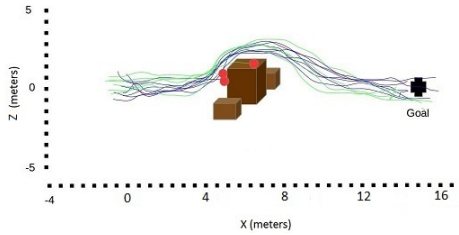

Fig. 8: A set of 15 experimental flights in indoor environment. Trajectories in blue and green colors represent with and without *Ego Dynamic Space Transformation* of estimated depth respectively (red dot represents collision and box corresponds to obstacle).

TABLE III: Error Analysis of Ardrone's State Estimation

Trajectory	$\sigma_x[cm]$	$\sigma_y[cm]$	$\sigma_z[cm]$
Short$[\leq 15m]$	11.71	10.32	5.72
Long $[>15m]$	41.60	26.7	10.38

$t_0,\ldots,t_n$). The proposed framework solved this by using *Squared Distance Minimization* approach where

- The end points are exactly interpolated i.e. $q_0 = s(t_0)$ and $q_n = s(t_n)$.
- The internal points q_i, $\forall i = 1,\ldots,n-1$ as calculated previously by (14), are approximate in the least square sense, by minimizing the optimization:

$$Q = \sum_{i=0}^{n} [q_i - s(t_i)]^2 \tag{32}$$

$$Q = f_0(\xi) = \xi^T P \xi + q^T \xi + r \tag{33}$$

The equation (33) is standard form of quadratic function representation of optimization function (32). Where $\xi = [p_0, p_1, \ldots, p_m,]$ is vector of optimization variables corresponding to de Boor control points. The constraints described in previous sections (v_{max}, a_{max} and j_{max}) can be expressed as affine function of ξ in form of equality and inequality constraints as:

$$A_{eq}\xi \quad = b_{eq} \quad and \quad A_{in}\xi \quad \preceq b_{in} \tag{34}$$

Time optimal motions are characterized by saturation of either the velocity or acceleration components at any given instant. As can be seen from the Fig. 6, that the acceleration components are saturated for significant amount of time. The performance can be improved by increasing the resolution of discretization of the optimization problem, although at the cost of increased computation time. Thus, a trade-off needs to be achieved between real time performance and time optimality.

F. Quadcopter Dynamics and Control

Quadcopter such as *Parrot*^{TM} Ardrone 2.0, [7] used in experiment is described by six degree of freedom of the rigid body as $q = [x,y,z,\theta,\phi,\psi]^T$, where the triple (x,y,z) represents the position of the center of mass of quadcopter and "roll-pitch-yaw" (θ,ϕ,ψ), set of Euler angles which represents the orientation in the same reference frame. The trajectory generation and planning is carried out using generic parameters-acceleration ($\ddot{x}$, $\ddot{y}$ and $\ddot{z}$) which may not be the actual control commands (ϕ and θ) for the quadcopter. Therefore, specific control inputs complaint with the quadcopter under consideration must be derived from the generic control parameters. Trajectory planning procedure described above is fast enough to be performed for every control update. Therefore a feedback loop is closed by re-planning the entire trajectory on each control update after dispatching the control commands of previous control interval to the quadcopter which is shown in Fig. 2. Since control commands to the underlying inner controller are generated in discrete time intervals, a numerical average of the ($\ddot{x}$, $\ddot{y}$ and $\ddot{z}$) over a typically small interval (determined experimentally) is computed and are tranformed into control commands using (35) and (36). After simplification of dynamic model [5] control commands for system describe above could be derived as follows:

$$\phi = \frac{\arctan(\ddot{x}\cos\psi + \ddot{y}sin\psi)}{\ddot{z}+g} \quad < 90^\circ \tag{35}$$

$$\theta = \frac{\arctan(\ddot{x}\sin\psi - \ddot{y}cos\psi)}{\ddot{z}+g}\cos\phi \quad < 90^\circ \tag{36}$$

where ϕ and θ are motion command along y and x axis respectively along with $\ddot{z}$ which is another control command for height. The proposed Navigation Framework works with the assumption of fixed heading angle $(\psi = 0)$. Moreover, singularities in this model only appear at $g = -\ddot{z}$, which is associated with the situation where the quadcopter is in free fall or executing extreme acrobatic manoeuver that can never be approached in normal flight conditions. Conclusively this method is similar to model predictive control where an optimum trajectory is generated at each time step.

IV. Experiments

We have evaluated the proposed framework on a low cost commercial quadcopter *Parrot*^{TM} Ardrone 2.0, [7] which is equipped with frontal monocular camera (93° Field of View, 640x360 pixels), ultrasound altimeter and onboard IMU. Moreover, we followed the approach suggested by Grabe *et al.* [18] for reliable state estimation. Table III presents error analysis of Ardrone 2.0's state estimation. In particular, our

TABLE IV: Performance analysis of Navigation Framework for 40 experiments

Method	Indoor		Outdoor		Success Rate
	Success	Failure	Success	Failure	
Without EDT	4	2	6	4	62.5%
With EDT	7	2	12	3	79.16%

Fig. 9: Quadcopter avoiding obstacles based on obstacle detection using depth map

solution is based on the fusion of the scaled visual velocity obtained from optical flow (downward facing camera of Ardrone 2.0) with high frequency reading of an onboard IMU, using classical EKF method. For all experiments, state estimation and purposed framework computations were carried out on a conventional desktop computer running ROS (*Robot Operating System*) as middle-ware with Ubuntu 12.04 LTS which is equipped with an Intel Core i3 processor @ 3.2 GHz and 8GB of RAM. Real time sequence of control commands are dispatched at 1.5Hz, with average duration of control updates at 448ms (computation times for perception, planning and control modules- 0.203s, 0.224s and 0.021s) respectively. The computation cycle time was approximately equal to system response time ($\simeq$ 400ms) of Ardrone 2.0 justifying its practicality. We conducted several successful flights with a set of feasible velocity v_{max}, acceleration a_{max} and jerk j_{max} in indoor and outdoor natural environment to validate the applicability of our navigation framework. As shown in the video, a experiment starts with quadcopter at hover state and starts moving towards goal position, Fig. 9. Once the obstacles are detected (using depth map) along the path of quadcopter, intermediate traversable waypoint which avoids obstacles is computed. Thereafter, collision free trajectory is generated over this waypoint towards the goal as shown in Fig. 7,8. Table IV shows our experimental results of 40 runs with 74 avoided obstacles and overall success ratio of 79.16%. Experiments showed 16.66% improved obstacle avoidance efficiency using EDT transformed approach. Where, the narrow *Field of View* was the largest contributor to failure. Furthermore, highly inaccurate depth estimation has been alleviated by conducting the experiments in environment similar to training data set. In addition, dynamics, stability and robustness of flight against external disturbances, rely upon internal controller of commercial quadcopter which is abstracted from trajectory planning.

V. Conclusion and Future Works

We have presented in this paper a novel navigation framework consisting of perception, planning and control modules applicable for obstacle avoidance in an natural environment which can be easily adapted for various commercial miniature quadcopters. Our navigation framework being a local planner, impending integration with global path planner is actively persuaded.

References

[1] Federico Augugliaro, Angela P Schoellig, and Raffaello D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1917–1922. IEEE, 2012.

[2] Arthur Richards and Jonathan P How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 1936–1941. IEEE, 2002.

[3] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.

[4] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for quadrotor flight. In *International Conference on Robotics and Automation*, 2013.

[5] Simone Formentin and Marco Lovera. Flatness-based control of a quadrotor helicopter via feedforward linearization. In *CDC-ECE*, pages 6171–6176, 2011.

[6] Mark W Mueller and Raffaello D'Andrea. A model predictive controller for quadrocopter state interception. In *Control Conference (ECC), 2013 European*, pages 1383–1389. IEEE, 2013.

[7] Pierre-Jean Bristeau, François Callou, David Vissière, Nicolas Petit, et al. The navigation and control technology inside the ar. drone micro uav. In *18th IFAC World Congress*, volume 18, pages 1477–1484, 2011.

[8] Walter Hoffmann and T Sauer. A spline optimization problem from robotics. *Rediconti di Mathematica*, 26:221–230, 2006.

[9] Hyungjun Park. Choosing nodes and knots in closed b-spline curve interpolation to point data. *Computer-Aided Design*, 33(13):967–974, 2001.

[10] Huaiping Yang, Wenping Wang, and Jiaguang Sun. Control point adjustment for b-spline curve approximation. *Computer-Aided Design*, 36(7):639–652, 2004.

[11] Wolfgang Rackl, Roberto Lampariello, and Gerd Hirzinger. Robot excitation trajectories for dynamic parameter estimation using optimized b-splines. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2042–2047. IEEE, 2012.

[12] Luigi Biagiotti and Claudio Melchiorri. Trajectory planning for automatic machines and robots. pages 188–194, 2008.

[13] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 593–600. ACM, 2005.

[14] Ian Lenz, Mevlana Gemici, and Ashutosh Saxena. Low-power parallel algorithms for single image based obstacle avoidance in aerial robots. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 772–779. IEEE, 2012.

[15] H. Alvarez, L.M. Paz, J. Sturm, and D. Cremers. Collision avoidance for quadrotors with a monocular camera. In *Proc. of The 12th International Symposium on Experimental Robotics (ISER)*, 2014.

[16] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1765–1772. IEEE, 2013.

[17] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, pages 1161–1168, 2005.

[18] Volker Grabe, Heinrich H Bulthoff, and Paolo Robuffo Giordano. A comparison of scale estimation schemes for a quadrotor uav based on optical flow and imu measurements. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5193–5200. IEEE, 2013.

[19] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[20] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[21] Javier Minguez, Luis Montano, and Oussama Khatib. Reactive collision avoidance for navigation with dynamic constraints. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 588–594. IEEE, 2002.