Monocular Vision-aided Depth Measurement from RGB Images for Autonomous UAV Navigation

RAM PRASAD PADHY, Indian Institute of Information Technology, Design and Manufacturing (IIITDM), Kancheepuram, India
PANKAJ KUMAR SA, National Institute of Technology Rourkela, India
FABIO NARDUCCI and CARMEN BISOGNI, University of Salerno, Italy
SAMBIT BAKSHI, National Institute of Technology Rourkela, India

Monocular vision-based 3D scene understanding has been an integral part of many machine vision applications. Always, the objective is to measure the depth using a single RGB camera, which is at par with the depth cameras. In this regard, monocular vision-guided autonomous navigation of robots is rapidly gaining popularity among the research community. We propose an effective monocular vision-assisted method to measure the depth of an Unmanned Aerial Vehicle (UAV) from an impending frontal obstacle. This is followed by collision-free navigation in unknown GPS-denied environments. Our approach deals upon the fundamental principle of perspective vision that the size of an object relative to its field of view (FoV) increases as the center of projection moves closer towards the object. Our contribution involves modeling the depth followed by its realization through scale-invariant SURF features. Noisy depth measurements arising due to external wind, or the turbulence in the UAV, are rectified by employing a constant velocity-based Kalman filter model. Necessary control commands are then designed based on the rectified depth value to avoid the obstacle before collision. Rigorous experiments with SURF scale-invariant features reveal an overall accuracy of 88.6% with varying obstacles, in both indoor and outdoor environments.

CCS Concepts: • **Computing methodologies** → **Vision for robotics**; *Scene understanding*; Matching;

Additional Key Words and Phrases: Monocular vision, RGB-D vision, scene understanding, UAV, depth measurement, obstacle avoidance, autonomous navigation, SURF

ACM Reference format:
Ram Prasad Padhy, Pankaj Kumar Sa, Fabio Narducci, Carmen Bisogni, and Sambit Bakshi. 2023. Monocular Vision-aided Depth Measurement from RGB Images for Autonomous UAV Navigation. *ACM Trans. Multimedia Comput. Commun. Appl.* 20, 2, Article 37 (September 2023), 22 pages.
https://doi.org/10.1145/3550485

This research work has been partially supported by: NITROAA with support of Lenovo P920 and Dell Inception 7820 workstations; NVIDIA Corporation with support of NVIDIA Titan V and Quadro RTX 8000 GPUs; Grant Number SRG/2021/002399, Start-up Research Grant (SRG), funded by Science & Engineering Research Board (SERB), Department of Science and Technology (DST), Government of India.
Authors' addresses: R. P. Padhy, Indian Institute of Information Technology, Design and Manufacturing (IIITDM), Kancheepuram, India; email: ramprasad.nitr@gmail.com; P. K. Sa and and S. Bakshi, National Institute of Technology Rourkela, India, 769008; emails: pankajksa@nitrkl.ac.in, sambitbaksi@gmail.com; F. Narducci and C. Bisogni, University of Salerno, Italy; emails: {fnarducci, cbisogni}@unisa.it.

1551-6857/2023/09-ART37 $15.00
https://doi.org/10.1145/3550485

1 INTRODUCTION

Vision-based depth measurement is a challenging, yet renowned research area in the computer vision domain [17, 42]. Different solutions, ranging from the inputs from a single camera, or a pair of cameras to RGB-D cameras, are utilized to derive the 3D primitives of an image. Depth is known to be the most significant 3D characteristic of a scene, and an integral part of many machine vision applications, such as, scene reconstruction [15, 27], event detection [5], pose estimation [20], video surveillance [14], path planning [18, 31], tracking [9, 46], identification tasks [19], and so on.

In intelligent autonomous systems with a single camera (monocular vision), 3D scene understanding plays an important role in deriving the next course of action. Most of the animals have their eyes positioned in such a way that their **Field of View (FoV)** does not interfere, thus leading to monocular vision. The way the animals perceive the depth of an object is different than that of humans, who possess binocular vision (two eyes having an intersecting FoV). This unique ability of the animals has always fascinated researchers. In recent years, many robotic systems ranging from ground vehicles to aerial vehicles have implemented monocular vision-based navigation methods to safely maneuver the robots in unknown environments.

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have gained a lot of popularity in recent years due to their practical applications in both indoor and outdoor environments. Advancement in technology has led to the development of UAVs with high battery life and solid body structure. They provide a cutting edge research in the area of military applications, precision agriculture, aerial surveillance, disaster management, search and rescue operations, inventory management [12], and much more. Nowadays, drones are playing a crucial role in intelligent transport systems, too. Researchers as well as industrialists around the world are actively researching on how to effectively use these devices for transfer of goods or delivery of products. A good example is Amazon Prime Air, a drone delivery service currently in development by Amazon. However, central to all these research areas lies the existence of a robust navigation algorithm, that will enable the UAV to avoid obstacles of any kind in unknown environments and make it more intelligent and autonomous [45].

Nowadays, most of the available UAVs in market are equipped with inbuilt GPS sensor. GPS helps in obtaining the pose of the UAV precisely, and hence assists a great deal in safe autonomous navigation [8]. In other words, the task of autonomous navigation is relatively easier in GPS-aided outdoor environments as compared to that in indoor environments and remote places, where GPS is either inactive or works with very low precision. Hence, in GPS-denied environments, proximity sensors, such as LIDARs, can be employed for autonomous navigation task [21, 23]. However, UAVs are generally deficient in carrying high-power-consuming and heavy-weight sensors. Accordingly, most of them employ lightweight sensors; usually a single camera or a pair of cameras are used to perceive the distance of an object in unknown environments. In few scenarios, low-cost RGB-D cameras, such as, Microsoft Kinect, Intel Realsense, Orbbec Astra, and so on, can be utilized to measure the depth of the obstacles in the surrounding environments. However, heavy-weight and high power consumption of these sensors make them more reliable for ground-based robots as compared to the UAVs.

In this article, we propose a monocular vision-assisted method for UAVs that can navigate autonomously avoiding head-on collision with the frontal obstacles (Figure 1). Although many works have already been proposed in this field [33, 40, 44], measuring the depth from a monocular camera and thereby developing an effective vision system for autonomous navigation of UAVs is still a challenging problem. Our method takes into account the relative change in the size of an obstacle with respect to the motion of the UAV as an important factor while estimating the depth.

Fig. 1. UAV navigating towards a tree (obstacle). As the UAV moves further, the depth value keeps on decreasing.

Contributions: The contributions of our proposed work can be summarized as follows:

(1) We propose a monocular vision-assisted method based on scale-invariant features to measure the depth of a UAV with an impending frontal obstacle.
(2) We propose a method based on 1D-Kalman filter to rectify the noisy depth measurements after each iteration.
(3) The rectified depth is then utilized to design essential control command that can safely navigate the UAV while avoiding the frontal obstacles. We demonstrate the efficacy of our proposed framework with experiments in GPS-denied outdoor and indoor environments with varying obstacles.

The rest of the article is organized as follows: Section 2 presents an overview of the prior research. Section 3 delineates the proposed methodology followed by a detail elaboration on the proposed collision avoidance method in Section 4. The experimental results are discussed in Section 5. Section 6 provides the concluding remarks.

2 RELATED WORK

There has been a decent amount of research work reported in recent years on obstacle detection and avoidance for UAVs, both in indoor and outdoor environments. Depending upon the types of sensors used, prior research in this field may broadly be classified into three categories: (1) proximity sensors, (2) vision sensors, (3) sensor and **IMU (inertial measurement unit)** fusion.

Proximity sensors: Proximity sensors often detect the presence of obstacles in the vicinity of the UAV with the help of electromagnetic radiations (e.g., infrared rays). Bachraach et al. and Gageik et al. in their respective works utilized a laser range-finder (LIDAR) and ultra-sonic sensors to estimate the distance of any obstacle from a moving UAV [2, 16]. A potential field obstacle detection algorithm based on fluid mechanics is proposed by Cruz and Encarnação to safely avoid the obstacles on the path of the UAV [7]. In their work, the on-board proximity sensors are utilized to generate the local map of the environment and the presence of local minima is avoided by modeling the obstacles and the UAV goal position by Harmonic functions. Chen et al. developed

an Android application, which uses several ultrasonic sensor readings to create a SLAM system to map the surrounding unknown environment [6]. However, due to the limited payload of the UAVs, it is not possible to install a laser range finder and heavy-weight sensors in most of the cases. Also, the power consumption will be more.

Vision sensors: These are lightweight sensors with low power consumption. UAVs with stereo vision have two cameras mounted at a specific distance or a single camera with integrated rotation [25]. Parallax of stereo vision is exploited to obtain cues such as stereopsis, eye convergence, disparity that in turn help in finding the depth. However, monocular system measures the depth information by processing cues such as relative size, texture gradient, motion parallax, perspective, and so on [3, 33, 40]. The approaches are predominantly based on constructing the 3D structure of the environment using methods, such as **Simultaneous Localization and Mapping (SLAM)** [30], **Parallel Tracking and Mapping (PTAM)**, [39] and so on. Engel et al. employed a monocular SLAM approach to estimate the UAV pose and then followed this by an extended Kalman filter to compensate for the delay arising due to communication and computation requirements [11]. Lioulemes et al. and Padhy et al. in their respective works employed the Hough transformation-based vanishing point algorithms to move UAVs in unknown corridor environments [26, 37]. Wang et al. employed a learning method based on optical flow to calculate the time-to-collision of the UAV with the obstacle [44]. Carranza et al. introduced a probabilistic visual odometry approach to localize and control a low-cost **Micro Aerial Vehicle (MAV)** in GPS-denied indoor environments [29]. The literature also includes some work based on detecting visual features on the obstacles and then using their properties along with optical flow parameters to estimate the position of the obstacles [33, 40].

Sensor and IMU fusion: Some of the prior works are based on fusing different sensor readings along with the inertial measurements. Efraim et al. fused the input from the UAV front camera with angular velocity measurements to safely navigate the UAV in corridor environments [10]. He et al. used the motion field information of the camera and other sensor readings of the UAV to accurately estimate the ego-motion parameters and then refined the motion measurements [22]. Achtelik et al. utilized the IMU measurements of the UAV in combination with the monocular vision information to overcome the issue of low-frequent vision updates [1]. Stubblebine et al. fused the laser data with monocular camera inputs to detect large obstacles in the path of the UAV [43]. Sasongko et al. generated way-points by constructing a restricted ellipsoid zone that was modeled using the already identified obstacle geometry. Further, the ellipsoid acts as the basis for finding new way-points [41]. In recent years, vision-based deep learning models also have shown good accuracy for depth measurement and subsequent safe UAV navigation [24, 34–36].

Our current work falls under the second category, i.e., the use of a vision sensor to avoid frontal obstacles. Unlike existing methods that are very much environment-dependent, our proposed method is competent enough to detect a wide range of obstacles in both indoor as well as outdoor environments. We utilize the video feed of the UAV front camera (monocular vision) and devise an algorithm to avoid head-on collision with frontal obstacles. The algorithm mainly relies on the scale-invariant features to measure the depth. Mori et al. also proposed a similar method that uses scale-invariant features to navigate the UAV [33]. Their method uses the scale-invariant features in combination with template image matching to compare relative obstacle sizes with different image spacing. As template image matching in real-time scenarios like navigation can be a complex task and increase the processing time, we refrained from this kind of high-end image-processing task. Rather, we compared the relative scale change of the scale-invariant key points from their mean position on the image to measure depth in consecutive frames. The proposed method is explained in detail in the next section.

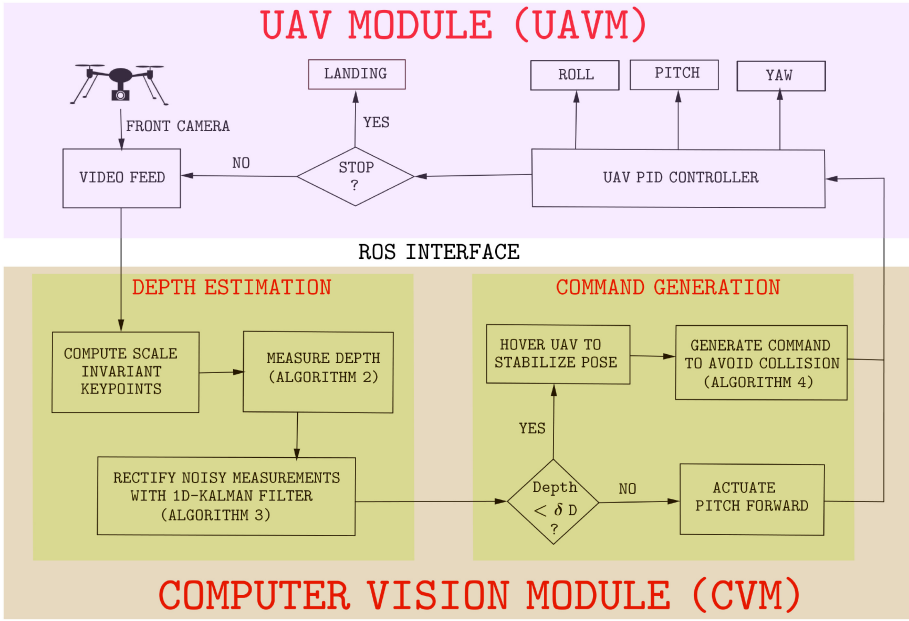


Fig. 2. The proposed System Architecture with two major components: UAV Module and Computer Vision Module (CVM). The video feed from the UAV front camera is transmitted on a frame-by-frame basis to the CVM through Robot Operating System (ROS) interface for further processing. CVM first extracts the scale-invariant keypoints and formulates a measure based on Euclidean distance scale (Section 4.2) to estimate the depth. 1D-Kalman filter is then employed to rectify the noisy depth measurements (Section 4.3). Depending on the rectified depth value, CVM then designs the necessary control command and communicates it to the UAV through the ROS interface (Section 4.4). UAV PID controller modifies the roll, pitch, and yaw values of the UAV to safely navigate it through the obstacles.

3 PROPOSED METHODOLOGY

The proposed methodology (Figure 2) involves the use of a monocular camera for estimating the depth of a frontal obstacle. We focus on the relative size expansion of an object on the image plane, when a camera moves towards it. However, detection and localization of objects inside an image is a computationally tedious task and hence cannot be employed for real-time navigation problems. Moreover, in most of the practical scenarios, there will be no well-defined objects situated in front of the UAV. We overcome this issue with the help of scale-invariant keypoints extracted from the image. Our approach to collision avoidance can be divided into two parts; first, we present a mathematical model and subsequently realize the same with scale-invariant features.

Modeling: As depicted in Figure 3, when the UAV moves from frame-j to frame-k, height of the object AB expands from h to sh ($\mathsf{s} > 1$) on the camera plane (pixel-wise expansion). The UAV carries a front-facing static pinhole camera, having focal length f. The points O_2 and O_1 are the positions of the camera aperture, when the UAV is at the frames j and k, respectively. Assume that the UAV is moving with a constant velocity of v.

It may be observed from Figure 3 that $\Delta\mathsf{ABO}_1 \sim \Delta\mathsf{A}_1\mathsf{B}_1\mathsf{O}_1$, and $\Delta\mathsf{ABO}_2 \sim \Delta\mathsf{A}_2\mathsf{B}_2\mathsf{O}_2$. Therefore,

$$\frac{\mathsf{f}}{\mathsf{sh}} = \frac{\mathsf{O}_1\mathsf{O}}{\mathsf{AB}} \quad \& \quad \frac{\mathsf{f}}{\mathsf{h}} = \frac{\mathsf{O}_2\mathsf{O}}{\mathsf{AB}}. \tag{1}$$

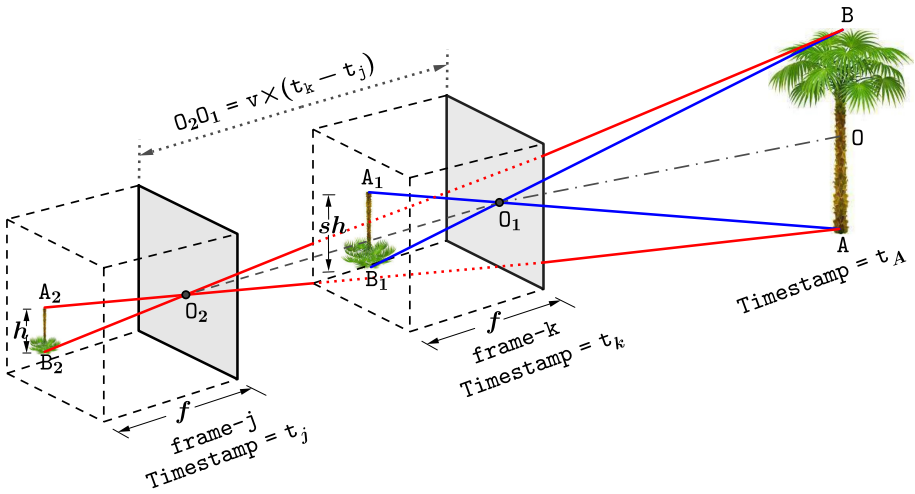


Fig. 3. Pinhole camera model showing the relative change in scale of an object on the image plane, as the UAV navigates from frame-j to frame-k with a constant velocity.

From Equation (1), it can be deduced that

$$s = \frac{O_2O}{O_1O} \Rightarrow s \times O_1O = O_2O \Rightarrow s \times O_1O = O_2O_1 + O_1O.$$

Therefore,

$$O_1O = \frac{O_2O_1}{s-1}, \text{ where } s > 1. \tag{2}$$

Equation (2) expresses the depth of the frontal obstacle AB from current frame-k. As the UAV is moving at a constant velocity of v, the equation can also be rewritten as:

$$\text{Depth}_k = \frac{v \times (t_k - t_j)}{s-1}, \text{ where } s > 1. \tag{3}$$

From Equation (3), it can be inferred that the depth at any position is inversely proportional to the rate of change of size of an object measured between two image frames.

Realization: The proposed mathematical model holds true in ideal scenarios, where the UAV navigates on a straight path with a constant velocity. However, to realize the Equation (3) in real-time scenarios, there exist two major issues:

(1) In most of the practical scenarios, there will be no well-defined/well-shaped object (like AB in Figure 3) located on the path of the UAV. If at all an object is present, then it would be computationally very expensive to detect and localize its size/shape on the image plane in real-time.
(2) There is every possibility that turbulence in the UAV due to its rotors and other external forces (like wind) can cause some deviation in the direction of the UAV from a straight path and may lead to noisy depth measurement.

Solution: To overcome the first issue, instead of considering well-shaped objects on the FoV, some scale-invariant keypoints are extracted from the image, and the Euclidean distance between each keypoint and their centroid (centroid of all the keypoints) is used as a potential feature to detect the scale change. In other words, the line joining the centroid and a keypoint is treated as a likely

obstacle in front of the UAV. The second issue is countered with the help of the well-established Kalman filter to rectify the noisy measurements.

Limitations: If the UAV moves to a completely different direction altogether because of gusty wind or other external factors, then our algorithm will not be able to handle that case. Also, The proposed algorithm will only work to detect static frontal obstacles. It will not be able to detect side-wise as well as moving obstacles.

4 COLLISION AVOIDANCE

As the UAV moves forward, any obstacle that comes in front of it is to be detected in advance so the UAV can navigate safely avoiding any collision. In our work, we rely on keypoints that bear the property of scale-invariance. In this section, we first outline the scale-invariant feature that has been used in our approach of frontal obstacle avoidance followed by the formulation of depth measurement. Subsequently, we suggest a method for the rectification of above depth, where a Kalman filter is employed to remove noisy measurements. Finally, an algorithm is presented for designing the necessary control commands based on the rectified depth value.

4.1 Scale-invariant Features

In recent years, scale-invariant features such as SIFT [28], SURF [4], and ORB [38] have gained huge popularity in computer vision due to their scale and rotation invariance characteristics. In our work, we have used SURF to detect frontal obstacles for the reason that it is fast to compute in comparison to SIFT [32] and more robust to large-scale changes as compared to ORB. The speeded-up feature of SURF makes it a suitable candidate for real-time navigation tasks.

Our approach looks for matching of keypoints extracted from the current frame and that of the previous frames. As the method is designed to detect only those obstacles, which are on the head-on path of the UAV, the keypoints that lie far away from the image center do not contribute towards the detection of obstacles. Hence, to reduce the processing time, the algorithm considers only the central one-fourth of the image while extracting the keypoints. Since the keypoints are scale-invariant, they tend to match even if they are subjected to any affine transformation caused by the forward motion of the UAV. Hence, the SURF features would be detected in consecutive frames as the UAV moves towards an obstacle. The **Brute-Force Matcher (**`BFMatch`**)** is then employed to compute the best keypoint matches between a pair of frames. Each SURF keypoint is represented using 128-bit feature descriptors. The `BFMatch` takes the keypoint feature descriptors of one frame and matches with that of another frame using some distance calculation. The closest descriptor pairs are added to the list of matches. Each match contains the keypoint indices (queryIdx and trainIdx) of both the frames along with the distance (l^2-norm) between their descriptors. If jth frame (query frame) occurs before kth frame (train frame) and M_{jk} denotes the list of matched keypoints between the above pair of frames, then

$$M_{jk} = \texttt{BFMatch}(P_j.desc, P_k.desc), \tag{4}$$

where the symbols P_j and P_k denote the SURF keypoint lists of the jth and kth frames, respectively, and $P_j.desc$, $P_k.desc$ represent the corresponding list of feature descriptors. Out of these matched keypoints, some of the outlier matches (wrongly detected matches) are discarded based on the following conditions:

Condition 1: The matches, whose feature distance is small are recognized as good matches. Hence, the entries for which the descriptor distance is greater than 0.25 [28] are considered as false matches and removed from the final list.

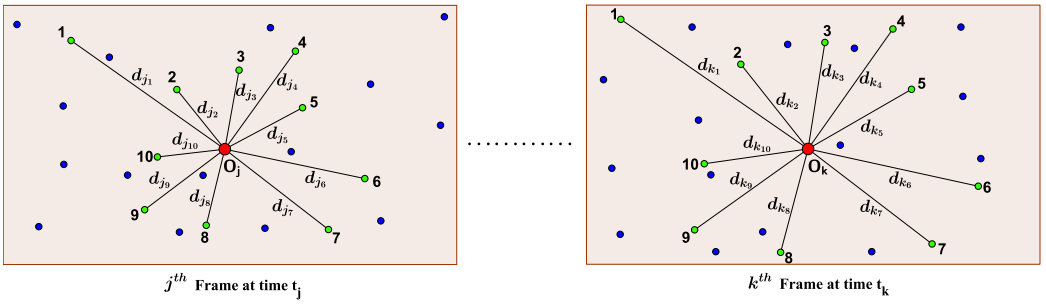


Fig. 4. Illustration of depth estimation using SURF keypoints for a pair of frames. In the figure, jth and kth frames are not necessarily consecutive. t_j and t_k are the timestamps of the jth and kth frames, respectively. Scale-invariant keypoints are represented by blue and green dots. There are total of 10 matched keypoints ($|M_{jk}| = 10$) between the pair of frames and are represented by green dots. The red dots (O_j and O_k) stand for the centroid of the matched keypoints (only green dots) in their respective frames. It may be noted that position-wise $O_j \approx O_k$ and $d_{k_m} > d_{j_m}$, where $m = 1, \ldots, 10$.

Condition 2: As the UAV is moving forward, hence by principle the Euclidean distance of a matched keypoint from the centroid position (centroid of the matched keypoints in respective frames) should increase in subsequent frames (Figure 4). The keypoints for which the above defined distance becomes smaller or remains the same are removed from the final list of matched keypoints. Here, the expansion of the Euclidean distance of the matched keypoints from one frame to another is termed as the Euclidean scale. The Euclidean scale for the mth matched keypoint can be mathematically formulated as:

$$s_m = \frac{d_{k_m}}{d_{j_m}}, \; m = 1, \ldots, |M_{jk}|, \tag{5}$$

where d_{j_m} and d_{k_m} represent the Euclidean distances of the mth matched keypoint from the centroids of the matched keypoints in jth and kth frames, respectively. Based on the proposed rule, if $s_m \leq 1$, then the mth entry is discarded from the final match list M_{jk}.

Condition 3: Let μ be the mean and σ be the standard deviation of all the calculated Euclidean scale values. Depth can be better estimated, if the final scale values lie in the confidence interval of $(\mu - 2\sigma, \mu + 2\sigma)$. Based on the proposed rule, if $s_m \leq (\mu - 2\sigma)$ or $s_m \geq (\mu + 2\sigma)$, then the mth entry is removed from the final match list.

Condition 4: Our algorithm requires only those keypoints, which correspond to the obstacle on the image plane. All other keypoints are outliers. By the principle of a pin-hole camera model, when a camera moves in the forward direction, the relative increase in the height on the image-plane for the objects situated at the same depth in front of the camera should be equal. This can be verified in Figure 3. Hence, the Euclidean scale of the matched keypoints present at the same depth should be approximately same. If an obstacle is situated on the head-on path of the UAV during navigation, then it can be inferred that the center of the image must correspond to a pixel that contributes towards the creation of the obstacle shape on the image plane. Hence, the matched keypoint that is at the closest distance to the center as compared to other matched keypoints most likely corresponds to a point on the obstacle. Therefore, the Euclidean scale (5) for every matched keypoint present on the obstacle should be approximately equal to that of a matched keypoint situated at the closest distance to the center. All other keypoints that do not satisfy the above condition are discarded from the final list of matched keypoints. If the xth matched keypoint is

the closest one to the center and p is the list containing the indices of only those matched keypoints that correspond to the obstacle, then, mathematically:

$$\frac{d_{k_x} - d_{j_x}}{d_{j_x}} \approx \frac{d_{k_{p[i]}} - d_{j_{p[i]}}}{d_{j_{p[i]}}}, \tag{6}$$

where $i = 1, \ldots, |p|$. From Equation (6), it can be inferred that the xth matched keypoint is also an entry of the list p. All the matched keypoints in M_{jk} that do not satisfy the above condition are discarded from the final list of matched keypoints.

Once the required keypoint matches are filtered out, RANSAC [13] homography matrix measurement method is used to find the perspective transformation of both the frames. It is useful in showing the keypoint matches between the frames. The conditions, defined for discarding the outlier matches, are depicted in Algorithm 1.

ALGORITHM 1: Discarding the outlier matches

Input: Initial match list M_{jk}
Result: Final match list M_{jk}
for *each match m in* M_{jk} **do**
  Discard the match m from M_{jk} if either of the below four conditions satisfies:
  Condition 1: m.descriptor_distance > 0.25;
  Condition 2: Euclidean scale (5), $s_m \leq 1$;
  Condition 3: $s_m \leq \mu - 2\sigma$ or $s_m \geq \mu + 2\sigma$, where μ and σ are, respectively, the mean and standard deviation of the list containing the Euclidean scales of the matches in M_{jk};
  Condition 4: $s_m < s_x - \delta$ or $s_m > s_x + \delta$, where x is the closest matched keypoint from the image center as compared to other matches in M_{jk};
return M_{jk}

4.2 Depth Measurement

Our approach will now estimate an effective depth value using the Euclidean scale defined by Equation (5). If the time taken by the UAV while navigating from jth to kth frame is $(t_k - t_j)$ units and the velocity is v units, then the depth of the UAV from kth frame to the obstacle can be formulated as (applying Equation (5) in Equation (3)):

$$D_{kjm} = d_{j_m} \left[\frac{v \times (t_k - t_j)}{d_{k_m} - d_{j_m}} \right], \; m = 1, \ldots, |M_{jk}|. \tag{7}$$

Equation (7) measures the depth considering only the mth matched keypoint between the current frame k and one of the previous frames j. As there are $|M_{jk}|$ number of matched keypoints present in the final match list M_{jk}, the average depth can be given as:

$$D_{kj} = \frac{1}{|M_{jk}|} \sum_{m=1}^{|M_{jk}|} D_{kjm}, \; |M_{jk}| \neq 0. \tag{8}$$

It may be noted that Equation (8) is based on all the matched keypoints of the current frame k and one of the previous frames j. A pictorial illustration is provided in Figure 4. However, in practice, the subsets of keypoints extracted from the current frame can match with the subsets of keypoints extracted from one or more previous frames. It is explained with the help of Figure 5. Hence, after processing of each frame, instead of a single depth, there may be a number of depth

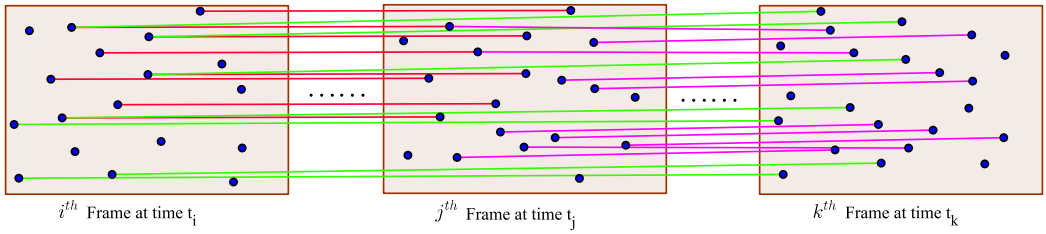


Fig. 5. Keypoint matching shown using three frames. Blue dots represent the scale-invariant keypoints. Red, green, and magenta line segments correspond to the matching between ith and jth, ith, and kth, jth, and kth frames, respectively. Here, a subset of keypoints from one frame matches with a subset of keypoints from another frame. A single keypoint can be a part of one or more matches. Also, some of the keypoints do not contribute at all towards matching between any of the frames. It may be noted that ith, jth, and kth frames are not necessarily consecutive.

values in accordance with the number of matched frames. As it would be time-consuming to find matches with all the previous frames, the proposed algorithm considers the matching for previous 10 frames only (approximate number of frames that were processed in last one second). After processing the current frame, all the depth values can be averaged out to obtain a single value. Averaging out helps in minimizing the inconsistencies in depth prediction. Hence, the final depth at frame k can be formulated as:

$$D_k = \frac{1}{N} \sum_{j \in L} D_{kj}, \tag{9}$$

where L is the list that contains the previous 10 frames and $N(\neq 0)$ denotes the number of frames from the list L, which has a non-empty keypoint matching with the current frame k. N can be mathematically given as:

$$N = \sum_{j \in L} [1 \times \{if(|M_{jk}| \neq 0)\}]. \tag{10}$$

The algorithm for depth measurement is outlined in Algorithm (2).

ALGORITHM 2: Depth Measurement at frame k

Input: List of previous 10 frames: L
Result: Final Depth at frame k: T_k
for *each frame entry j in L* **do**
  $M_{jk} \leftarrow$ Brute-force keypoint matching for frame j and current frame k;
  Discard the outlier matches from M_{jk} using Algorithm 1;
  for *each match m in M_{jk}* **do**
    Evaluate D_{kjm} by (7);
  Compute the representative depth of all $|M_{jk}|$ keypoints as in (8):
  $D_{kj} \leftarrow \frac{1}{|M_{jk}|} \sum_{m=1}^{|M_{jk}|} D_{kjm}, |M_{jk}| \neq 0$;
Calculate N by (10);
Measure final depth as in (9):
$D_k \leftarrow \frac{\sum_{j \in L} D_{kj}}{N}$;
return D_k

4.3 Depth Rectification

In our work, we expect the UAV to navigate on a straight line with a constant velocity. However, the turbulence in the UAV, caused by gutsy wind or its rotors, may slightly deviate the path of the UAV. This deviation may induce some noise in the measurement of the depth (9).

We have modeled the problem of rectifying the depth with a constant velocity-based Kalman filter model. As there is only a single state variable to predict, the basis of 1D Kalman filter is followed. Let the initial depth be D_0 when our algorithm detects the keypoints on the field of view for the very first time. As the velocity of the UAV is kept constant (v units), the depth after each timestep Δt, can be predicted as:

$$D_i = D_{i-1} - (v \times \Delta t), \tag{11}$$

where i denotes the iteration number. However, the deviation in path, as discussed above, creates uncertainties in the estimation. The generative model for the estimate can be written as the summation of the true state and a white Gaussian noise (w_1) caused due to uncertainties:

$$D_i = D_{i-1} - (v \times \Delta t) + w_1. \tag{12}$$

Taking the expectation and the variance of T_i, we have:

$$\begin{cases} \hat{D}_i = \hat{D}_{i-1} - (v \times \Delta t) \\ \sigma_i^2 = \sigma_{i-1}^2 + \sigma_{w_1}^2. \end{cases} \tag{13}$$

Equation (13) represents the Kalman filter prediction step. If $\sigma_{w_1} = 0$, i.e., there is no uncertainty, then depth can be determined precisely in future. However, if $\sigma_{w_1} > 0$, then as the process goes on, the predicted depth would become increasingly uncertain. Therefore, we need some measurement, such as the depth estimated using our proposed algorithm (Algorithm 2) for a better prediction. As the measurement is not accurate, it gives another additive Gaussian noise assumption. The measurement can therefore be written as:

$$Z_i = D_i + w_2, \tag{14}$$

where w_2 denotes the noise due to measurement. Now, there are two pieces of information available; first, a mean generated due to the prior model, and second, based on the UAV vision data. Hence, these two information can be put into the Kalman Filter update equations:

$$\begin{cases} \kappa_i = \frac{\sigma_i^2}{\sigma_i^2 + \sigma_{w_2}^2} \\ \hat{D}_i' = \hat{D}_i + \kappa_i (Z_i - \hat{D}_i) \\ \sigma_i'^2 = (1 - \kappa_i)\sigma_i^2, \end{cases} \tag{15}$$

where κ_i is the Kalman gain for ith iteration. Equation (15) represents the Kalman filter correction step. When $\kappa_i = 1$, there is complete trust on predicted data, however if $\kappa_i = 0$, then there is no trust. The steps required for Kalman filter are delineated through Algorithm (3).

4.4 Designing Control Commands

In this article, we have mainly focused on detecting and avoiding the frontal obstacles. It may be noted that there is no predefined route to follow or goal position to reach for the UAV. Once a stop command is sent to the UAV from the ground-based system, it will land there.

Initially, after takeoff, the UAV is allowed to navigate on a straight path through the pitch-forward command. Now, to move the UAV autonomously avoiding the frontal obstacle, the next set of commands are designed based on the rectified depth obtained after applying the Kalman filter. After every iteration of Kalman filter, the depth value keeps on decreasing. The pitch-forward

ALGORITHM 3: Depth rectification using 1D Kalman filter

Input: $\hat{D}_{i-1}, \sigma^2_{i-1}, Z_i, \sigma^2_{w_1}, \sigma^2_{w_2}, \Delta t$
Result: $\hat{D}_i^{'}, \sigma_i^{'^2}$
$\hat{D}_i = \hat{D}_{i-1} - (v \times \Delta t)$; } Prediction step
$\sigma_i^2 = \sigma^2_{i-1} + \sigma^2_{w_1}$; }
$\kappa_i = \frac{\sigma_i^2}{\sigma_i^2 + \sigma^2_{w_2}}$; } Correction step ▷ Kalman gain
$\hat{D}_i^{'} = \hat{D}_i + \kappa_i(Z_i - \hat{D}_i)$; }
$\sigma_i^{'^2} = (1 - \kappa_i)\sigma_i^2$; }
return $\hat{D}_i^{'}, \sigma_i^{'^2}$ ▷ updated prediction

command is sent continuously to keep the UAV on a straight path, until the depth attains a certain threshold value δD or lower. Once the threshold value is attained, a hover command is sent from the CVM to stop the UAV from colliding with the obstacle. Then, as a rule of thumb, the UAV moves in yaw-left (anti-clockwise) or yaw-right (clockwise) direction for 90°. After the completion of the yaw command, the UAV hovers for some time to stabilize the pose before again navigating in pitch forward direction. The above process keeps on repeating, until a stop command is issued from the CVM. The complete set of rules for obstacle-free navigation is depicted in Algorithm (4).

ALGORITHM 4: UAV autonomous navigation algorithm

Input: rectified depth obtained from Kalman filter
Result: autonomous command generation
UAV take-off;
Actuate UAV with pitch forward;
while *command* ≠ *stop* **do**
  $D_c \leftarrow depth$; ▷ Rectified depth value keeps on updating
  while $D_c > \delta D$ **do**
    Actuate UAV with pitch forward;
    $D_c \leftarrow depth$;
  Hover UAV to avoid the collision;
  Actuate UAV in yaw-left or yaw-right direction for 90°;
  Hover UAV to stabilize the pose;
  Actuate UAV with pitch forward;

5 EXPERIMENTS AND RESULTS

We carried out a series of real-world experiments to demonstrate the efficacy of our system. The experiments are conducted in different environments, such as indoor rooms of varying dimensions and visual appearance, outdoor places having static obstacles, and so on. The accuracy of the proposed algorithm is evaluated with scale-invariant SURF keypoints. We also analyzed the effects of the Kalman filter-based model to rectify the depth estimates for better navigation of the UAV.

In this section, we first describe the system setup and the experiments on depth estimation. This is followed by the results on frontal obstacle avoidance with scale-invariant SURF keypoints.

5.1 System Setup

(a) Parrot AR Drone quadcopter is used to validate our proposed algorithm in unknown indoor and outdoor environments. The primary advantage of the AR Drone over other UAVs available

(a) (b) (c)

Fig. 6. Different set of test obstacles. It can be visualized that the test set contains diverse obstacles, with entries from both indoor and outdoor environments.

in market lies in its low cost, robustness to crashes, and can safely be used in close proximity of humans because of its indoor hull. However, the underlying hardware and the software on-board cannot be modified. The UAV has its own internal PID controller mechanism and cannot be altered. Hence, only the modified commands (change in roll, pitch, and yaw values) can be sent from the CVM.

(b) The **Robot Operating System (ROS)** is employed for control and communication between the UAV and the ground-based system.

(c) The only sensor that is required for running our proposed algorithm is the forward-facing pin-hole camera. Although it is capable of producing HD resolution (1,280 × 720) videos, nHD resolution (640 × 360) video is sufficient enough to run our proposed algorithm. It has a 90° FoV (fish-eye lens) and produces video feed at 30 fps. While implementing the proposed algorithm, the fish-eye video is first converted to its rectilinear equivalent. Once it is converted to the rectilinear form, then only the pinhole camera model-based depth estimation algorithm is applied to measure the depth. If direct fish-eye image is taken as input to the model, then it will not be able to measure the depth accurately.

(d) All the experiments are carried out using C++ with OpenCV libraries for accomplishing the computer vision tasks. For faster computation, multi-threading is used whenever necessary. We are planning to publicly release our code soon to facilitate future research.

(e) Although we have used the Parrot AR Drone for our experiments, our vision algorithm will work for any UAV having a fixed front camera.

5.2 Depth Measurement Using SURF Features

We let the UAV fly autonomously from a start point in the forward direction. To validate our algorithm, in some of the experiments, obstacles have already been placed on the path of the UAV,

(a) (b)

Fig. 7. Right-hand side image (b) corresponds to the current image frame, whereas the left-hand size image (a) corresponds to an image frame, which occurred 0.5 second before the occurrence of the current frame.

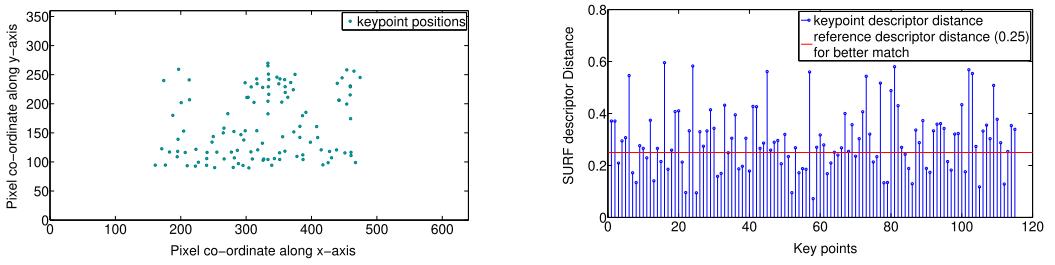
Fig. 8. Initial SURF keypoint matches (total 115). The keypoints are extracted from the central one-fourth of the image plane (represented by a green-border rectangle).

while in some of the cases the static objects (such as trees, walls, pillars), which are already there in the environment, are treated as potential obstacles. A set of those obstacles is delineated in Figure 6. It may be noted that any object that is capable of producing scale-invariant keypoints can be detected by our proposed method. The assumptions to our algorithm are:

(a) Obstacles are sufficiently textured to generate reliable SURF features;
(b) The UAV navigates on a straight-line path with a constant velocity and during motion, yaw/roll velocities are negligible;
(c) Obstacles are assumed to be static, if not, then the camera motion predominates the obstacle motion.

Figure 7(b) represents the current image frame (frame k), which occurred after about 0.5 second of the occurrence of the frame shown in Figure 7(a) (frame j). Scale-invariant SURF keypoints are first extracted from the central one-fourth portions of both the frames and Brute-Force matching is employed to obtain a total of 115 matches. The matches are shown in Figure 8. Not all these matched keypoints correspond to the obstacle. It can be noticed that some of the matches are wrongly detected. The positions of the corresponding matched keypoints in the current frame are plotted in Figure 9(a). Figure 9(b) represents the plot for the feature descriptor distances of all the matched points; the red horizontal line depicts the reference line for classifying the good matches.

All the matches where the descriptor distance is less than or equal to 0.25 are considered as good matches, and others are discarded from the final list of matches. Total of 49 matches remained after applying the above rule, and the corresponding matches are shown in Figure 10. The positions



(a) (b)

Fig. 9. (a) Initial matched SURF keypoints (total 115), (b) Matched keypoint descriptor distances.

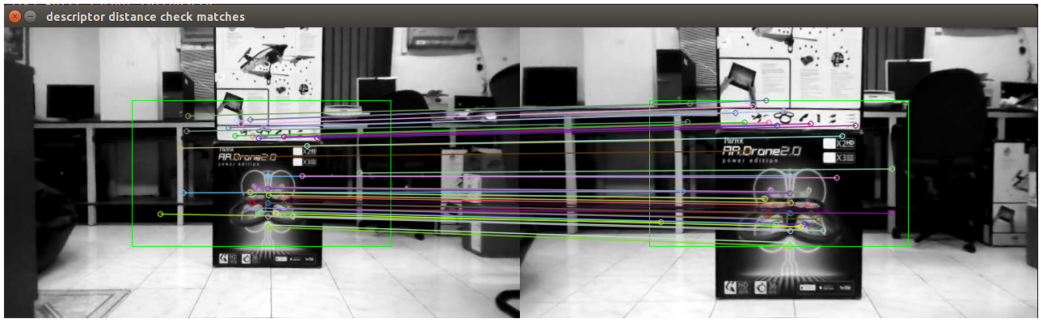
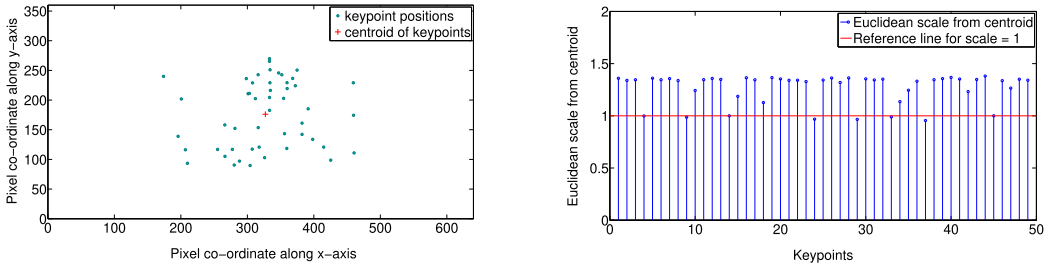


Fig. 10. Keypoint matches (total 49) after discarding the entries where the SURF descriptor distance >0.25.



(a) (b)

Fig. 11. (a) Keypoints (total 49) where descriptor distance ≤0.25, (b) Euclidean scale from centroid of matched keypoints.

of the matched keypoints in the current frame are shown in Figure 11(a), and the centroid of the keypoints is represented by a red + symbol. For each of the matched keypoints, the Euclidean scale (5) is estimated by evaluating the ratio of the Euclidean distances from their centroid position in the current frame to that of the previous frame. These scale values are plotted in Figure 11(b).

As the UAV is moving forward, the Euclidean scale (5) should always be greater than 1. Hence, the matches that do not satisfy the above condition are discarded from the final match list. The reference red line in Figure 11(b) stands for Euclidean scale equal to 1 and the matches for which the scale is below this reference line are discarded from the final list. After applying the above rule, total of 41 matches remained and are shown in Figure 12. The corresponding keypoints in the current frame and their Euclidean scales are plotted in Figures 13(a) and 13(b), respectively.

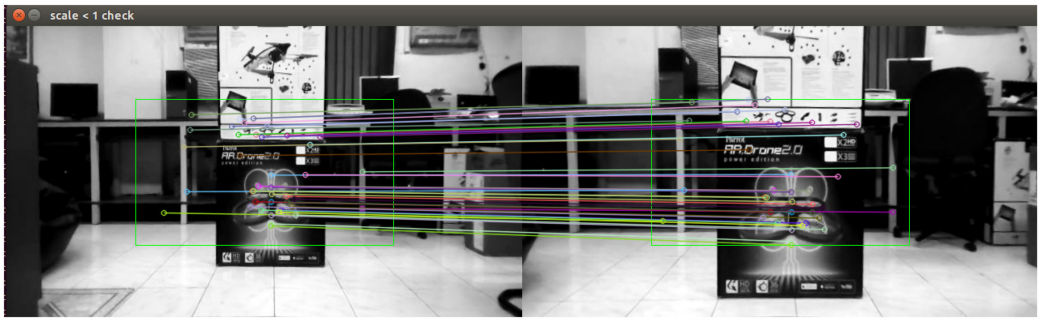
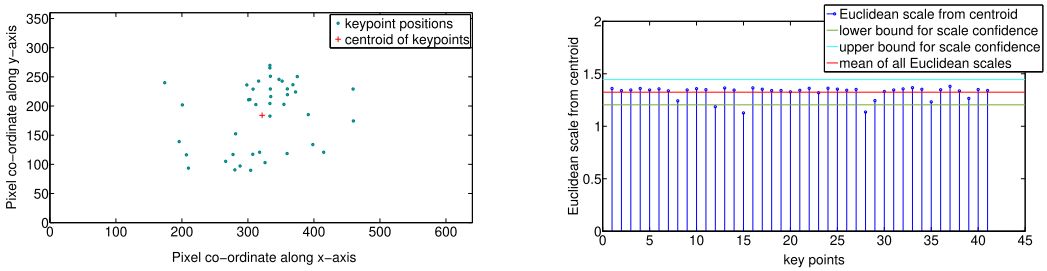


Fig. 12. Keypoint matches (total 41) after discarding the entries where Euclidean scale ≤ 1.



(a) (b)

Fig. 13. (a) Keypoints (total 41) where the Euclidean scale >1, (b) Euclidean scale from centroid of matched keypoints. Mean μ = 1.32451, standard deviation σ = 0.0594153.

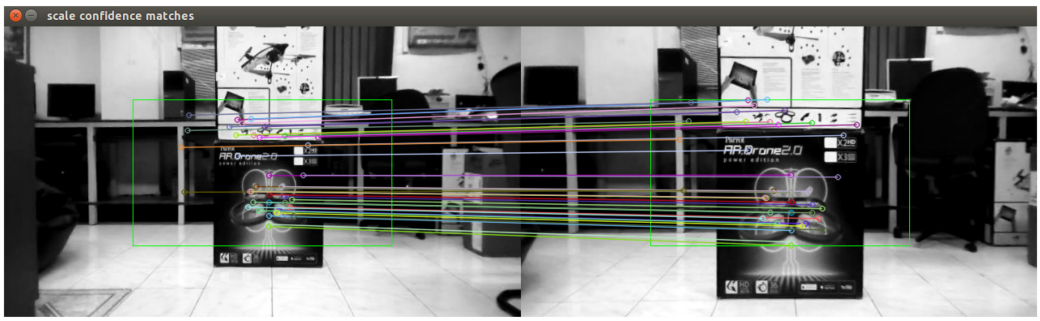
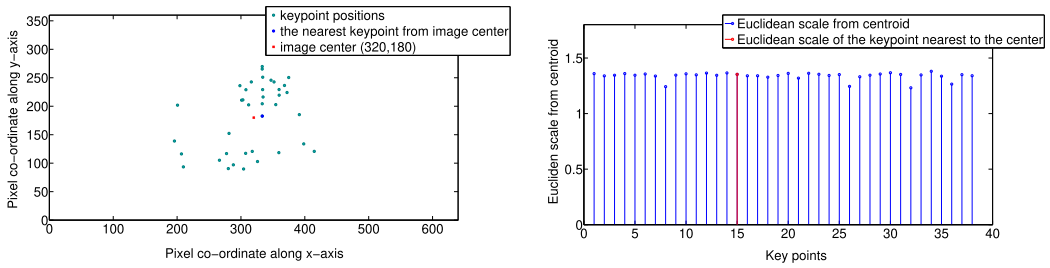


Fig. 14. Keypoint matches (total 38) after discarding the non-confident entries, where the Euclidean scale from the centroid does not lie in the confidence interval $(\mu - 2\sigma, \mu + 2\sigma)$.

Mean (μ) and standard deviation (σ) of all these Euclidean scale values are calculated and the matches for which the scale values do not lie in the interval $(\mu - 2\sigma, \mu + 2\sigma)$ are considered as non-confident matches and are discarded. The remaining 38 matches are shown in Figure 14, and the corresponding keypoints in the current frame are plotted in Figure 15(a). The Euclidean scale values for the revised match list are plotted in Figure 15(b). The red bar line in the figure represents the Euclidean scale of the matched keypoint, which is at the closest distance to the center in comparison to the other matched keypoints.

All the Euclidean scale values are rounded up to two decimal places and the matches for which the Euclidean scale values are equal to that of the closest matched keypoint to the center are



(a) (b)

Fig. 15. (a) Keypoints (total 38) where the Euclidean scale lies in the confidence range, (b) Euclidean scale from centroid of matched keypoints after removal of non-confident matches.

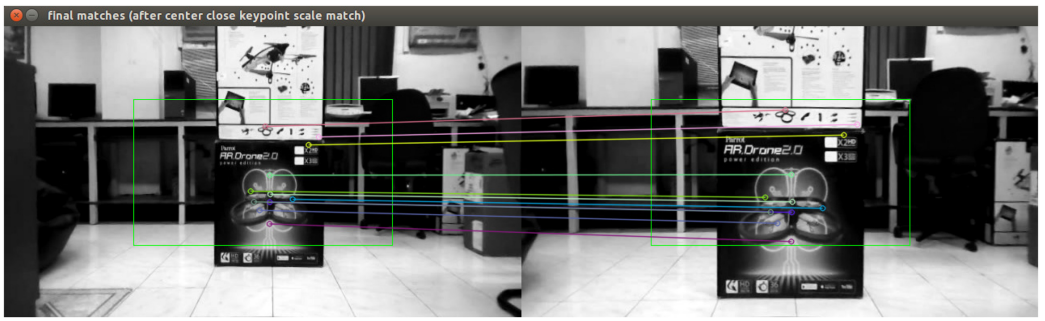
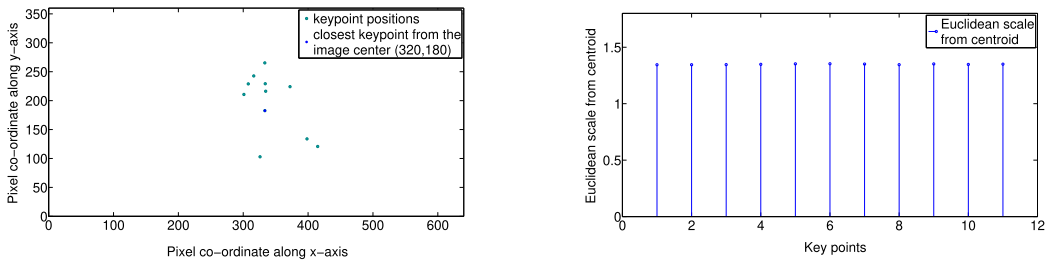


Fig. 16. Final list of keypoint matches (total 11), where the Euclidean scale of all the entries is approximately equal to that of the closest keypoint to the center. It can be seen that all the keypoints are positioned on the impending obstacle.



(a) (b)

Fig. 17. (a) Final matched keypoints (total 11) having Euclidean scale approximately equal to that of the closest matched keypoint from the center, (b) Euclidean scale of final matched keypoints (scales are approximately equal).

considered as good matches for depth estimation and others are discarded. After applying the above rule, total of 11 matches remained and are shown in Figure 16, and the corresponding matched keypoints in the current frame are plotted in Figure 17(a). The resultant matches are considered as the final match list for depth estimation, and the corresponding Euclidean scale values are plotted in Figure 17(b). From the figure, it can be inferred that the Euclidean scale values of the final matched keypoints are almost same.

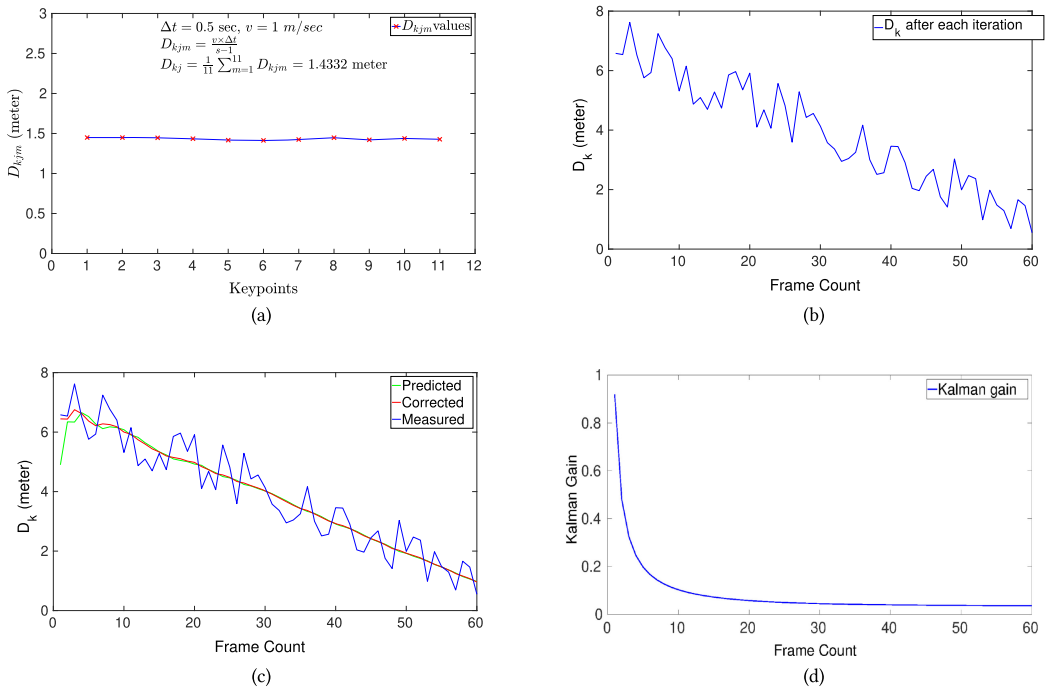


Fig. 18. (a) Measured depth values considering each of the final matched keypoints, (b) D_k values as measured by our vision algorithm, (c) Rectified depth values, (d) Kalman gain data over 60 frames.

The depth D_{kjm} is estimated by considering each of the final matched keypoint m using Equation (7), and all these values are averaged out to obtain the representative depth D_{kj} (8). All the D_{kjm} values are plotted in Figure 18(a), and it can be inferred that these values are almost equal. Figures 17(a) and (b) show the final match list of 11 keypoints.

In the above experiment, depth is evaluated considering only the matched keypoints of the current frame and a previous frame, which occurred 0.5 second before the occurrence of the current frame. It may so happen that the current frame may possess scale-invariant matches with more than one of the previous frames. As it is computationally tough in real-time to consider all the matches for depth estimation, we have considered only the previous 10 frames to estimate an effective depth. The UAV speed is kept constant ($\approx$ 1 m/sec) through out all the experiments. The proposed keypoint matching algorithm might not work if the UAV speed is very high. In those scenarios, the algorithm will not be able to find robust SURF features to measure the depth. As we are mostly considering the indoor scenarios, we assume that the UAV speed is controlled ($\approx$1 m/sec) and all the experiments are conducted with this speed. Although the UAV is capable of sending the video feed at a rate of 30 fps, our algorithm processes approximately 10 fps. In other words, while estimating the final depth, our algorithm considers the keypoint matching with the frames, which occurred in last 1 second. All the D_{kj} values, generated by taking into account the matches with previous 10 frames (frame j), are averaged out to estimate the final depth D_k (9), from the current position (frame k) to the obstacle. Figure 18(b) shows the D_k values obtained for consecutive 60 frames. From the figure, it can be observed that the resultant plot is not smooth. As it is assumed that the velocity of the UAV is kept constant throughout the experiments, ideally the depth values should decrease linearly giving rise to a smooth graph. However, it can be deduced that the depth measurements are erroneous and hence need to be rectified.

Table 1. Results for Obstacle Avoidance with Varying Outdoor and Indoor Obstacles

OUTDOOR OBSTACLES			INDOOR OBSTACLES		
Obstacle Type	Number of Trials	Obstacles Avoided	Obstacle Type	Number of Trials	Obstacles Avoided
Tree trunk	50	49	Wall	50	45
Tree leaves	50	47	Card box	50	45
Pillar	50	38	Chair	50	43
Car	50	42	Human	50	44
Wall	50	43	Door	50	47
TOTAL	250	**219**	**TOTAL**	250	**224**

Depth rectification: External factors, such as wind or the rotor turbulence of the UAV, are the prime reasons for obtaining the noisy measurements. These measurements are rectified by employing a 1D Kalman filter model, which assumes the velocity of the UAV to be constant throughout the experiment. As described in Section 4.3, D_i and σ_i^2 are updated after each iteration (Algorithm 3), hence these can be initialized randomly. The initial estimation (D_0) for depth is assumed to be 5 meters and initial estimated error (σ_0^2) is taken as 1100. However, tweaking the values of process noise variance ($\sigma_{w_1}^2$) and measurement noise variance ($\sigma_{w_2}^2$) is crucial for obtaining clear readouts. After experimenting with various sets of measurement data, $\sigma_{w_1}^2$ and $\sigma_{w_2}^2$ are experimentally chosen as 0.125 and 97, respectively, to obtain a smooth transition for depth values. As our algorithm roughly processes 10 frames per second, Δt in Algorithm 3 is taken as 0.1 second. Figure 18(c) shows the rectified output of the Kalman filter. It can be visualized that the corrected depth values are shaping to form a smooth curve. Corresponding Kalman gain data are plotted in Figure 18(d).

Designing control commands: After each iteration of Kalman filter, the depth value keeps on decreasing. When the depth attains a threshold δD or lower, a hover command is sent to the UAV from the CVM. For our experiments, we have experimentally chosen δD = 0.5 meter. In other words, 0.5 meter prior to the actual collision happens, hover command is actuated to avoid the collision. The UAV is allowed to hover for 1 second. Hover command is executed to stabilize the pose of the UAV. Then, as depicted in Algorithm 4, the UAV actuates in yaw-left or yaw-right direction for 90°. After the successful completion of yaw command, the UAV again hovers for 1 second before navigating in pitch forward direction. The above process keeps on repeating until a stop command is sent from the CVM. Once the stop command is successfully executed, the UAV lands at its current position.

5.3 Results and Discussion

The previous methods in this field generally concentrated on environment/obstacle-specific or goal-oriented applications. However, the proposed method is designed for both indoor and outdoor environments with a wide range of obstacles. To the best of our knowledge, there does not exist any publicly available standard dataset in this field. We tested our proposed method with varying outdoor and indoor obstacles as depicted in Table 1. The experimentation is done in such a way that each test object consisted of variety of sample instances. Our proposed method is able to avoid the obstacles with an overall accuracy of 88.6%.

The low rate of obstacle avoidance in some of the cases (like Pillar in Table 1) can be attributed to the fact that the test instances do not have sufficient texture to form reliable SURF keypoints and hence, an effective depth could not be generated.

6 CONCLUSION

Depth estimation with monocular vision is an integral part for understanding the 3D structure of an unknown environment. In this work, we present a mathematical model and an algorithm for measuring the depth of a UAV with any static frontal obstacle. Noisy depth measurements are then rectified by using a 1D-Kalman filter model based on constant velocity. Collision with the frontal obstacle is avoided by designing the necessary control commands. The only sensor that we use is a static monocular camera. The experiments are carried out in unknown GPS-denied indoor and outdoor environments with varying obstacles. Our algorithm is able to detect and avoid the frontal obstacles with good accuracy. However, our algorithm may fail in two scenarios: (a) increase in velocity of the UAV may delay the wireless communication with the ground system, (b) scenarios where the algorithm could not find reliable scale-invariant SURF keypoints. Our algorithm can be used in disaster-struck buildings, where it is not possible for humans to venture and carry out rescue operations. Also, it can be used to avoid trees, pillars, walls, and other static obstacles in outdoor environments.

It must be noted that the proposed algorithm works only for static frontal obstacles. However, the UAV might encounter dynamic obstacles on its path as well as side-wise obstacles, e.g., other drones in the sky (fleet of drones). Hence, as a future improvement to this work, the proposed algorithm can be effectively modified to detect and avoid the dynamic obstacles in the environment. However, updated sensor setup might be required for detecting side-wise obstacles.

REFERENCES

[1] Markus Achtelik, Michael Achtelik, Stephan Weiss, and Roland Siegwart. 2011. Onboard IMU and monocular vision based control for MAVs in unknown in-and outdoor environments. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3056–3063.

[2] Abraham Bachrach, Samuel Prentice, Ruijie He, and Nicholas Roy. 2011. RANGE–robust autonomous navigation in GPS-denied environments. *J. Field Robot.* 28, 5 (2011), 644–666.

[3] Xicheng Ban, Hongjian Wang, Tao Chen, Ying Wang, and Yao Xiao. 2021. Monocular visual odometry based on depth and optical flow using deep learning. *IEEE Trans. Instrum. Meas.* 70 (2021), 1–19. DOI: https://doi.org/10.1109/TIM.2020.3024011

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. 2008. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.* 110, 3 (2008), 346–359.

[5] Xiaojun Chang, Yao-Liang Yu, Yi Yang, and Eric P. Xing. 2017. Semantic pooling for complex event analysis in untrimmed videos. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 8 (2017), 1617–1632.

[6] Michael Y. Chen, Derrick H. Edwards, Erin L. Boehmer, Nathan M. Eller, James T. Slack, Christian R. Speck, Sean M. Brown, Hunter G. Williams, Samuel H. Wilson, Christopher S. Gillum, et al. 2013. Designing a spatially aware and autonomous quadcopter. In *Systems and Information Engineering Design Symposium*. IEEE, 213–218.

[7] Gonçalo Charters Santos Cruz and Pedro Miguel Martins Encarnação. 2012. Obstacle avoidance for unmanned aerial vehicles. *J. Intell. Robot. Syst.* 65, 1–4 (2012), 203–217.

[8] Youjing Cui and Shuzhi Sam Ge. 2003. Autonomous vehicle positioning with GPS in urban canyon environments. *IEEE Trans. Robot. Autom.* 19, 1 (2003), 15–25.

[9] Heng Deng, Qiang Fu, Quan Quan, Kun Yang, and Kai-Yuan Cai. 2020. Indoor multi-camera-based testbed for 3-D tracking and control of UAVs. *IEEE Trans. Instrum. Meas.* 69, 6 (2020), 3139–3156. DOI: https://doi.org/10.1109/TIM.2019.2928615

[10] Hanoch Efraim, Shai Arogeti, Amir Shapiro, and Gera Weiss. 2017. Vision based output feedback control of micro aerial vehicles in indoor environments. *J. Intell. Robot. Syst.* 87, 1 (2017), 169–186.

[11] Jakob Engel, Jürgen Sturm, and Daniel Cremers. 2014. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robot. Auton. Syst.* 62, 11 (2014), 1646–1656.

[12] Antonio De Falco, Fabio Narducci, and Alfredo Petrosino. 2019. An UAV autonomous warehouse inventorying by deep learning. In *International Conference on Image Analysis and Processing*. Springer, 443–453.

[13] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

[14] Pasquale Foggia, Alessia Saggese, and Mario Vento. 2015. Real-time fire detection for video-surveillance applications using a combination of experts based on color, shape, and motion. *IEEE Trans. Circ. Syst. Vid. Technol.* 25, 9 (2015), 1545–1556.

[15] Daniele Fontanelli, Federico Moro, Tizar Rizano, and Luigi Palopoli. 2014. Vision-based robust path reconstruction for robot control. *IEEE Trans. Instrum. Meas.* 63, 4 (2014), 826–837. DOI: https://doi.org/10.1109/TIM.2013.2289091

[16] Nils Gageik, Paul Benz, and Sergio Montenegro. 2015. Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors. *IEEE Access* 3 (2015), 599–609.

[17] Zan Gao, Yinming Li, and Shaohua Wan. 2020. Exploring deep learning for view-based 3D model retrieval. *ACM Trans. Multim. Comput., Commun. Applic.* 16, 1 (2020), 1–21.

[18] Sean Garvey and Scott Koziol. 2018. Robot path planning using analog circuit phase delay. *IEEE Trans. Syst., Man, Cyber. Syst.* 50, 10 (2018).

[19] Aleksei Grigorev, Shaohui Liu, Zhihong Tian, Jianxin Xiong, Seungmin Rho, and Jiang Feng. 2020. Delving deeper in drone-based person re-id by employing deep decision forest and attributes fusion. *ACM Trans. Multim. Comput., Commun. Applic.* 16, 1s (2020), 1–15.

[20] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 2015. An integrated framework for 3-D modeling, object detection, and pose estimation from point-clouds. *IEEE Trans. Instrum. Meas.* 64, 3 (2015), 683–693.

[21] Guojian He, Xingda Yuan, Yan Zhuang, and Huosheng Hu. 2021. An integrated GNSS/LiDAR-SLAM pose estimation framework for large-scale map building in partially GNSS-denied environments. *IEEE Trans. Instrum. Meas.* 70 (2021), 1–9. DOI: https://doi.org/10.1109/TIM.2020.3024405

[22] Zhihai He, Ram Venkataraman Iyer, and Phillip R. Chandler. 2006. Vision-based UAV flight control and obstacle avoidance. In *American Control Conference*. IEEE, 2166–2170.

[23] Sebastian Hening, Corey A. Ippolito, Kalmanje S. Krishnakumar, Vahram Stepanyan, and Mircea Teodorescu. 2017. 3D LiDAR SLAM integration with GPS/INS for UAVs in urban GPS-degraded environments. In *AIAA Information Systems-AIAA Infotech @ Aerospace*. AIAA, 448–457.

[24] Thomas Lee, Susan McKeever, and Jane Courtney. 2021. Flying free: A research overview of deep learning in drone navigation autonomy. *Drones* 5, 2 (2021), 52.

[25] Huei-Yung Lin, Chun-Lung Tsai, and Van Luan Tran. 2021. Depth measurement based on stereo vision with integrated camera rotation. *IEEE Trans. Instrum. Meas.* 70 (2021), 1–10. DOI: https://doi.org/10.1109/TIM.2021.3073687

[26] Alexandros Lioulemes, Georgios Galatas, Vangelis Metsis, Gian Luca Mariottini, and Fillia Makedon. 2014. Safety challenges in using AR.drone to collaborate with humans in indoor environments. In *International Conference on Pervasive Technologies Related to Assistive Environments*. ACM, 33–36.

[27] Xiaobai Liu, Yibiao Zhao, and Song-Chun Zhu. 2018. Single-view 3D scene reconstruction and parsing by attribute grammar. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 3 (2018), 710–725.

[28] David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* 60, 2 (2004), 91–110.

[29] José Martínez-Carranza, Esteban Omar Garcia, Hugo Jair Escalante, and Walterio Mayol-Cuevas. 2015. Towards autonomous flight of low-cost MAVs by using a probabilistic visual odometry approach. In *Advances in Artificial Intelligence and its Applications*. Springer, 560–573.

[30] Ivan Maurović, Marija Seder, Kruno Lenac, and Ivan Petrović. 2018. Path planning for active SLAM based on the D* algorithm with negative edge weights. *IEEE Trans. Syst., Man Cyber. Syst.* 48, 8 (2018), 1321–1331.

[31] Bassel Abou Merhy, Pierre Payeur, and Emil M. Petriu. 2008. Application of segmented 2-D probabilistic occupancy maps for robot sensing and navigation. *IEEE Trans. Instrum. Meas.* 57, 12 (2008), 2827–2837. DOI: https://doi.org/10.1109/TIM.2008.926048

[32] Darshana Mistry and Asim Banerjee. 2017. Comparison of feature detection and matching approaches: SIFT and SURF. *GRD J.-Glob. Res. Devel. J. Eng.* 2, 4 (2017), 7–13.

[33] Takayoshi Mori and Stefan Scherer. 2013. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *IEEE International Conference on Robotics and Automation*. IEEE, 1750–1757.

[34] Ram Prasad Padhy, Shahzad Ahmad, Sachin Verma, Sambit Bakshi, and Pankaj Kumar Sa. 2021. Localization of unmanned aerial vehicles in corridor environments using deep learning. In *25th International Conference on Pattern Recognition (ICPR)*. IEEE, 9423–9428.

[35] Ram Prasad Padhy, Xiaojun Chang, Suman Kumar Choudhury, Pankaj Kumar Sa, and Sambit Bakshi. 2019. Multi-stage cascaded deconvolution for depth map and surface normal prediction from single image. *Pattern Recog. Lett.* 127 (2019), 165–173.

[36] Ram Prasad Padhy, Sachin Verma, Shahzad Ahmad, Suman Kumar Choudhury, and Pankaj Kumar Sa. 2018. Deep neural network for autonomous UAV navigation in indoor corridor environments. *Procedia Comput. Sci.* 133 (2018), 643–650.

[37] Ram Prasad Padhy, Feng Xia, Suman Kumar Choudhury, Pankaj Kumar Sa, and Sambit Bakshi. 2018. Monocular vision aided autonomous UAV navigation in indoor corridor environments. *IEEE Trans. Sustain. Comput.* 4, 1 (2018).

[38] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision*. IEEE, 2564–2571.

[39] Inkyu Sa, Hu He, Van Huynh, and Peter Corke. 2013. Monocular vision based autonomous navigation for a cost-effective MAV in GPS-denied environments. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 1355–1360.

[40] Simanto Saha, Ashutosh Natraj, and Sonia Waharte. 2014. A real-time monocular vision-based frontal obstacle detection and avoidance for low cost UAVs in GPS denied environment. In *IEEE International Conference on Aerospace Electronics and Remote Sensing Technology*. IEEE, 189–195.

[41] Rianto Adhy Sasongko, S. S. Rawikara, and Hansel J. Tampubolon. 2017. UAV obstacle avoidance algorithm based on ellipsoid geometry. *J. Intell. Robot. Syst.* 88, 2–4 (2017), 567–581.

[42] Shervin Shirmohammadi and Alessandro Ferrero. 2014. Camera as the instrument: The rising trend of vision based measurement. *IEEE Instrum. Meas. Mag.* 17, 3 (2014), 41–47. DOI: https://doi.org/10.1109/MIM.2014.6825388

[43] Andrew Stubblebine, Brennan Redmond, Brian Feie, and Elad Kivelevitch. 2015. Laser-guided quadrotor obstacle avoidance. In *AIAA Infotech @ Aerospace*. AIAA, 2026–2037.

[44] Chaoqun Wang, Wei Liu, and Max Q.-H. Meng. 2015. Obstacle avoidance for quadrotor using improved method based on optical flow. In *IEEE International Conference on Information and Automation*. IEEE, 1674–1679.

[45] Chaoqun Wang, Han Ma, Weinan Chen, Li Liu, and Max Q.-H. Meng. 2020. Efficient autonomous exploration with incrementally built topological map in 3-D environments. *IEEE Trans. Instrum. Meas.* 69, 12 (2020), 9853–9865. DOI: https://doi.org/10.1109/TIM.2020.3001816

[46] Di Yuan, Xiaojun Chang, Zhihui Li, and Zhenyu He. 2022. Learning adaptive spatial-temporal context-aware correlation filters for UAV tracking. *ACM Trans. Multim. Comput., Commun. Applic.* 18, 3 (2022), 1–18.

Received 15 February 2022; revised 9 June 2022; accepted 8 July 2022