Visual Navigation and Path Tracking Using Street Geometry Information for Image Alignment and Servoing

Ayham Shahoud *, Dmitriy Shashev and Stanislav Shidlovskiy

Faculty of Innovative Technology, Tomsk State University, 36 Lenin Ave, 634050 Tomsk, Russia; dshashev@mail.tsu.ru (D.S.); ssv@mail.tsu.ru (S.S.)
* Correspondence: ayham18@stud.tsu.ru

Abstract: Single camera-based navigation systems need information from other sensors or from the work environment to produce reliable and accurate position measurements. Providing such trustworthy, accurate, and available information in the environment is very important. The work highlights that the availability of well-described streets in urban environments can be exploited by drones for navigation and path tracking purposes, thus benefitting from such structures is not limited to only automated driving cars. While the drone position is continuously computed using visual odometry, scene matching is used to correct the position drift depending on some landmarks. The drone path is defined by several waypoints, and landmarks centralized by those waypoints are carefully chosen in the street intersections. The known streets’ geometry and dimensions are used to estimate the image scale and orientation which are necessary for images alignment, to compensate for the visual odometry drift, and to pass closer to the landmark center by the visual servoing process. Probabilistic Hough transform is used to detect and extract the street borders. The system is realized in a simulation environment consisting of the Robot Operating System ROS, 3D dynamic simulator Gazebo, and IRIS drone model. The results prove the suggested system efficiency with a 1.4 m position RMS error.

Keywords: visual servoing; path tracking; scene matching; visual odometry; landmark; Hough transform

1. Introduction

Using imaging devices to navigate aerial vehicles is a current topic of interest. In addition to the location measurements, visual systems offer a lot of information about the surrounding environment. There is much information in the images that could be used to calculate useful parameters for navigation. High-level scene information such as objects and pixel-level information such as corners could be used. Adopting a certain method for navigation depends on the application, carrier dynamic, and the work environment [1,2].

Image-based navigation (sometimes referred to as visual servoing) has been used a lot for military applications, especially in ground-air seekers with thermal cameras. Recently, it has been used a lot in industrial applications such as packaging and obstacle avoidance. The camera could be attached in a form called “eye-in-hand”, i.e., attached to the robot arm, or “eye-to-hand”, i.e., fixed in the workspace with a known transformation between the robot arm and the camera. The existence of distinguishable objects in the image could be beneficial for the navigation process in predicting the metric scale or bearing with a target, which is also effective in waypoint tracking and auto-landing [3,4].

Position-based visual navigation systems are used in many applications, especially to replace the Global Positioning System (GPS) in indoor systems or in applications where more independence is required. Although computer vision systems depend on the texture and light from the external environment, they are considered more independent than GPS navigation systems. The GPS systems are controlled by a provider and suffer from outages, multipath, spoofing, and jamming. If highly accurate measurements are required,
then a more expensive GPS device must be used, which is the same situation for Inertial Navigation Systems (INS) that also suffer from drift over time.

In outdoor navigation that depends on reference objects, sometimes it is difficult to collect a fully accurate description or metric information concerning the geometry of the object. The previous discussion does not apply to the streets which are nowadays of the most well-defined structures in cities and even in small villages. Many reasons stand behind the intensive efforts that ended up with accurate street descriptions such as the support of automatic driving systems, their needs in transport application software, and for city planning goals. Generally, the location and width of most streets are available for free somewhere on the Internet. On the other side, street intersections are less affected by weather, seasons, and human activities, therefore, relying on them as distinguished semi-natural landmarks is a brilliant idea to enhance the drone navigation system.

Visual navigation systems that use only a single camera need information from other sources to produce robust and accurate measurements. Examples of these sources are compasses, altimeters, other navigation system such as GPS, or information from known objects in the image. Providing information that is suitable, trustable, accurate, and available in the environment to the visual systems is a matter of interest to sustain the continuity of the measurements. In this work, the main purpose is to get beneficial information from streets since they are well-described, easy to detect, and available in urban environments, then employ that information in navigation and path tracking tasks for a drone. Two algorithms (operating modes) are implemented, the first algorithm calculates the drone position in NED coordinates using Visual Odometry (VO) based on local features tracking. The second algorithm uses image information represented by the street geometry and dimensions to predict the captured image scale, and the rotation angle relative to a predetermined landmark template. Probabilistic Hough transform is used to extract the street borders. While scene matching with the landmark template based on correlation techniques is done to compensate for VO drift, the alignment is done using the estimated scale and angle. After that, the drone continues a servoing process toward the landmark center. The drone control is done on the level of the guidance commands, i.e., the heading angle of the Unmanned Aerial Vehicle (UAV) and the speed are controlled using ROS framework, while the stabilization is preserved by the internal autopilot of the IRIS drone model. The urban flight simulation environment in the 3D dynamic simulator Gazebo is shown in Figure 1.

Figure 1. The 3D simulation flight environment in Gazebo with the IRIS drone.
The rest of this paper is organized as follows: Section 2 for related studies, Section 3 for position-based and image-based navigation systems, Section 4 for the drone control system, Section 5 for implementation and realization, Section 6 for simulation and results, Section 7 for results analysis, and Section 8 for the conclusion.

2. Related Studies

Integrating a computer vision navigation system with GPS or inertial systems to do a certain task such as path following or auto-landing for UAVs has been investigated a lot in recent years. With the current massive development in computing and imaging devices, a tendency appeared to use the available imaging devices to implement a high-performance visual navigation system that works autonomously and independently without integration with other systems. That tendency was obvious in many studies and inventions used developed algorithms and advanced simulation environments to obtain good results [5,6]. In [1], a survey was presented on computer vision for UAVs. Current developments, trends, and the basics of visual servoing in UAVs for navigation are explained. In [2], an autonomous vision-based recovery system for small fixed-wing unmanned aerial vehicles was presented. Detection of the recovery net and providing the bearing angle to the guidance algorithm were done. In [3], path detection and following by a drone were studied and implemented by calculating guidance commands depending on the path location in the images, no metric information was used, a low-cost computer was used, and finally, a comparison between neural networks and traditional image processing techniques for path detection was done. In [4], a vision-based guidance system for a drone was presented. The visual information obtained from a single camera fixed on the vehicle with the navigation and communication hardware were capable of surveying, identifying, and tracking ground, sea, and aerial targets. In [7], landmark selection for scene matching with the knowledge of color histogram was presented. In [8], an uncalibrated downward-looking camera on the UAV was used to calculate the UAV orientation based on clustered points. A vehicle heading calculation for a ground vehicle using road segmentation was presented in [9]. In [10], the classification of good and bad matching areas was done using a convolutional neural network. In [11], they depended on a neural network for road intersection matching. In [12], a method was presented for enhancing navigation performance through visual odometry in a GPS-degraded environment, a GPS/INS/Vision integrated solution was proposed to provide a robust and continuous navigation output, especially for the GPS-degraded environment.

In some works, such as [5,9,12], the vision navigation systems are used as an aiding system in combination with other systems or just to replace the GPS outages. In this paper, the visual navigation system was used as a standalone system without relying on other systems. It is worth mentioning that obtaining a visual navigation system that depends only on an imaging device is still one of the top interests for researchers. While some works, such as [3], depend on neural networks or other methods to detect and calculate the object location in the image, this work goes deeper by using the specification and equations satisfied by the object itself. Furthermore, using street geometry to calculate necessary information for drone visual navigation is a new contribution of this paper. The street is treated as an object, which has been used for image alignment using street width for scale calculation and using street border slope for orientation calculation on another side.

3. Position-Based vs. Image-Based Navigation Systems

In position-based positioning, two main methods could be distinguished, absolute and incremental. In absolute methods, the position is calculated by matching every captured image with a reference image (map, digital elevation model, or digital surface model), absolute positioning is drift-free. The matching could be established using correlation techniques such as cross-correlation, or by matching local features detected in both images. Correlation-based methods suit well-structured areas but need an alignment process before the matching, i.e., both images must have the same scale and orientation [10,11]. In incre-
mental methods, the position is calculated by matching successively captured images such as in the visual odometry systems, these systems suffer from drift over time. Additional information from other sensors, such as compasses and altimeters, is usually used in the calculation process in both VO and scene matching.

Scene matching and VO were used in this research for positioning. VO was implemented by matching successively captured images using local features in the images detected by Speed Up Robust Features (SURF). Perspective n Point (PnP) was used to calculate the camera transformation between every two successive images by doing 2D-3D matching [13]. Assumptions of flattened ground and a rigid attachment between a down-looking camera and the UAV were proposed. Moreover, Random Sample Consensus (RANSAC) was applied to determine the optimal transformation model and to exclude outlier correspondences. Concatenating the successive translations in the external coordinates NED yields the camera position in the NED frame. Cross-correlation was also used in scene matching-based positioning, but only near the landmarks’ areas since they were chosen to contain street intersections where correlation gives trusted results.

Known objects in the images could be used to calculate the navigation parameters. Usually, “known object” refers to a physical object or shape in the image such as buildings, trees, fences, and craters. It might also be a combination of many objects according to the environment. The main constraint for an object to be beneficent in navigation is the “rigidity”, which conserves the internal distances in the object. The object could be real or virtual; virtual shapes are constructed by stable features such as corners or other local features. An example of virtual objects might be a rigid circle or square constructed by some stable points with a fixed radius. Generally, any geometrical combination of points that satisfy the rigidity constraints could be useful to give information about the navigation parameters. Useful image information for visual servoing could be the object size, position, and orientation, which help in the prediction of the camera position and orientation [1]. If the real metric dimensions of the object are available, the image scale is calculated which solves the depth estimation problem in single-camera applications, or in applications where the scale is needed such as cross-correlation [14]. The position and the metric information are not always important in visual servoing. For example, sometimes it is enough to keep the object in the center of the image, then pixel-space measurements are sufficient [15]. That is opposite to position-based navigation, which recommends calculating the vehicle’s position and moving it based on that information.

Matching and detecting an object in an image could be done by many methods such as template matching between the object and the captured image, histogram matching, depending on artificial intelligence such as a neural network for object detection, or evaluation of some constraints (equations) satisfied by a set of object’s points. Some constraints were used in this work such as parallel lines which involve two minor conditions “straight line” and “parallelism”. Street borders and width were used to estimate the necessary information for landmark template accurate matching. The street borders are assumed to be straight lines and were detected using Hough transform.

4. Drone Control System

A general diagram of a drone control system is shown in Figure 2. Aside from the rotors, motors, and their drivers, the drone carries the necessary sensors to preserve its stability during all work phases such as launching, loitering, and landing. Usually, the inertial sensors or Angular Heading Reference Systems (AHRS) are employed in the stabilization task. All the drone’s actions are executed in a stabilized manner, such as stabilized hovering, launching, landing, climbing, and descending. The aerodynamic modeling of the drone is out of the scope of this research and the dynamic equations can be found in [16].
The commands generated by the Motor Mixer Algorithm (MMA) are used by the drone motors’ drivers to generate suitable actions such as changing the speed, altitude, and attitude in certain amounts determined by the guidance algorithm, which in turn depends on the drone mission [17]. MMA transforms the thrust and the signals related to the angles (yaw, pitch, and roll) into suitable control signals. The commands which represent the mission might be prepared offline and stored in an onboard computer to be used in an open-loop form, which needs a well-determined overall system model. Since obtaining a well-determined model is practically hard, mostly the closed-loop form is used based on the available measurements and the estimated state. In a simple path tracking mission, a few waypoints (or landmarks) are chosen in suitable locations to define the path, then they are stored in the onboard computer. During the mission, the drone seeks these waypoints.

The navigation system uses reading from sensors and visual systems to supply the drone with the needed parameters to generate suitable command signals. The quality of the mission execution depends on the individual involved subsystems’ characteristics such as navigation, control, motor drivers, sensors, and other subsystems. The more accurate the position supplied into the control system, the more accurate the path tracking mission. Using imaging systems is effective in such situations because they give much information about the surrounding structure in the environment. If the used information from the image is points in image coordinates \((u, v)\), then using the interaction matrix can give the required drone (camera) velocity to fly to the desired location [18]. The interaction matrix represents the relationship between the points motion in the real world and their projections on the image plane. The related relationship is presented in Equation (1).

\[
\begin{pmatrix}
\dot{v}_x \\
\dot{v}_y \\
\dot{v}_z \\
\end{pmatrix} = -\lambda (J^+)
\begin{pmatrix}
u_1 - u_1^* \\
u_2 - u_2^* \\
\vdots \\
u_n - u_n^* \\
\end{pmatrix}
\]

where: \(n\) is the number of features, \(\lambda\) is an optional constant, and \(J\) is the interaction matrix (or Jacobian) which has a pseudo-inverse \(J^+\). The left side vector is the required velocity for the drone to reach the position defined by the feature vector \((u_1^*, v_1^*, \ldots, u_n^*, v_n^*)\).

In the case of depending on a predefined object, servoing to the object center might be a choice. The purpose will be to adjust both object and the current camera image centers, a suggested formula is presented in Equation (4). This technique was used because the waypoints are located at the landmark center [15,19].

The advanced simulation environments present good models for drones’ autopilots such as IRIS from Ardupilot and px4. Using these models in 3D environments such as the adopted one, which is presented by Gazebo and shown in Figure 1, gives more realistic
results with zero costs. In addition to the zero cost, they provide high flexibility in tuning and repeating tests which make realization in the real-world easier. ROS allows to subscribe to the sensors’ signals attached to the drone such as the camera image, GPS, compass, and to give commands to the drone such as the desired heading angle and velocity. ROS was adopted in this research. In brief, the designed subsystem takes the visual information from the camera and generates the required velocity and heading angle to control the drone path.

5. Implementation and Realization

The desired path, which is about 425 m in length, was constructed based on seven waypoints. Each waypoint was connected with a reference template of a size 100 × 100 pixels. The waypoints are located at the center of each reference template and their coordinates in the NED coordinate system were stored. The selection of the templates’ locations was done in street intersections satisfying that each one must contain a long part of the street that appears as two long parallel lines. Usually, the street dimensions are available as public free information on the Internet, they do not need real checks and measurements in the real locations. In the simulation environments, it is easy to do some measurements. Choosing the suitable templates and applying offline preprocessing (enhancement, cropping, and rotating) to them to meet the aforementioned requirements were not hard tasks using the OpenCV library.

Examples of the selected landmarks from the simulation environment are presented in Figure 3. As mentioned in the introduction, two operating modes or algorithms were suggested, the whole algorithm is illustrated in Figure 4. The two operating modes are presented in detail in the next sub-sections.

Figure 3. Examples of the landmarks.

5.1. Operating Mode Far from the Landmark (d > thr1)

In this mode, the drone flies at an average speed of 10 m/s with a heading calculated by Equation (2). While the position is continuously calculated using VO, the heading is updated. During the flight, the drone undergoes vibrations, and the roll and pitch angles reach high magnitudes up to 7 degrees. Moreover, the flattened ground assumption is not perfect because some features are located on the ground, trees, and buildings with different heights that vary up to 15 m, and that is not negligible for a drone flying at an altitude of 100 m. From the previous discussion, the VO measurements are corrupted with errors, an error integration over time is expected; therefore, it is necessary to compensate for VO drift. To solve that problem, a fine matching method was proposed to be applied near the landmarks using normalized cross-correlation. The heading angle $Y_{UAV}$ is calculated using Equation (2), where $p_1 (x_1, y_1), p_2 (x_2, y_2)$ are the coordinates of the current and next waypoints respectively in NED.

$$ Y_{UAV} = \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right) $$ (2)
5.1. Operating Mode Far from the Landmark (d > thr1)

In this mode, the drone flies at an average speed of 10 m/s with a heading calculated by Equation (2). While the position is continuously calculated using VO, the heading is updated. During the flight, the drone undergoes vibrations, and the roll and pitch angles reach high magnitudes up to 7 degrees. Moreover, the flattened ground assumption is not perfect because some features are located on the ground, trees, and buildings with different heights that vary up to 15 m, and that is not negligible for a drone flying at an altitude of 100 m. From the previous discussion, the VO measurements are corrupted with errors, an error integration over time is expected; therefore, it is necessary to compensate for VO drift. To solve that problem, a fine matching method was proposed to be applied near the landmarks using normalized cross-correlation. The heading angle $\gamma_{UAV}$ is calculated using Equation (2), where $p_1(x_1, y_1)$, $p_2(x_2, y_2)$ are the coordinates of the current and next waypoints respectively in NED.

5.2. Operating Mode near the Landmark (d < thr1)

In this mode, street borders are extracted and the fine scene matching starts. Firstly, the grayscale image is transformed to a binary image depending on Canny edge detector, then the probabilistic Hough transform is applied to detect the lines in the image [20]. After that, the long parallel lines in the landmark template which represent the street edges are extracted as explained in Algorithm 1. In OpenCV, line detection using Hough transform contains the accuracy parameters in addition to two other main parameters. The first parameter is the minimum length for a series of points to be considered as a line, and the second one is the maximum allowed distance that judges if two close lines belong to the same line or not. From the template construction, it was known that the street edge lines are parallel and long compared to other lines in the image. A classification procedure took

![Figure 4. Navigation and waypoints tracking algorithm.](image-url)
place to vote on those lines which are less likely to be the desired street border. In that procedure, the short lines and those which are far from parallelism (according to their equations) were excluded. The two lines that remained after the voting are the desired borders [21,22].

The street border extraction pseudo code is shown in Algorithm 1. All the used thresholds were determined taking into consideration the work environment, the aforementioned constraints, and the known template structure. Finally, the street width is available in pixels and is already known in meters ($l_m$), then the image scale is calculated by Equation (3).

Algorithm 1: The street border extraction algorithm.

**Inputs**: Lines detected using probabilistic Hough transform. A threshold “$l$” for line length with an initial value set to 50 pixels, and a threshold “$\alpha$” for line slope difference to be considered parallel and it is set to $\alpha = \tan(3^\circ)$.

**Outputs**: Street borders

**Pseudo code:**
1. For each line in Lines
   If line $\geq l$
   Longlines. Add (line)
   End for
2. If Longlines is empty
   $l = 25$
   repeat 1
   If Longlines is still empty
   Exit() //border detection failed, only VO is active
3. For each line “$i$” in Longlines
   $m_i =$ slope of “$i$”
   For each line “$n$” in Longlines
   $m_n =$ slope of “$n$”
   If ($|m_i - m_n| < \alpha$ and “$i$” is not a member of the candidates of “$n$”)
   i_candidates. Add (n)
   End for
   End for
4. For each k_candidates in candidates sets
   If (k_candidates.members_count()) < 2
   k_candidates.delete() //A single line with no candidate
   End for
5. If (candidates_sets.members_count()) = $\phi$
   Exit() //border detection failed (no parallel lines), only VO is active
6. For each candidates set
   select the longest two lines and drop the others
   In case of ambiguity between two lines, select the line with $\text{Min}(|m_i - m_n|)$
   End for
7. Keep the candidates set with the longest lines and drop the others
8. The remainder candidates set is the street borders (the output)

1 To avoid redundancy, if line A is a candidate to line B (i.e., to be its pair or the other street edge), the opposite is true too.

After extracting the street in the landmark, the angle between the captured image and the template can be found and used for image alignment, which is necessary before cross-correlation. Knowing each line equation from Hough transform made the task easier. In both the template and the captured image, the slopes of the two lines constituting the street border are theoretically the same because they are parallel. To align the two images, the slope of the street (a borderline) in one image must equal the slope of the corresponding street in the other image. Calculating the required angle $\psi$ was easy using some mathematical functions as in Equation (3), then rotating and scaling were done. After that, matching the captured image with the landmark template was done using normalized cross-correlation. If the two street edges (lines) in the captured image are defined by the
slope $m_1$ and constants $c_1$ and $c_2$, respectively, and in the template image by the slope $m_2$ and constants $c_{21}$, $c_{22}$ then:

$$\begin{align*}
\psi &= \tan^{-1}(m_2) - \tan^{-1}(m_1) \quad : \psi \text{ is the rotation angle} \\
lp &= \frac{|c_2 - c_1|}{\sqrt{1 + m_1^2}} \quad : lp \text{ is the street width in pixels} \\
s_{\text{est}} &= \frac{lm}{lp} \quad : s_{\text{est}} \text{ is the estimated scale}
\end{align*}$$

In the second operating mode, the drone starts to adjust the center of the captured image with the center of the template. That is done by bearing and flying toward the template center until the condition $d < \text{thr2}$ is satisfied. After bearing and adjusting the centers of the images to a distance less than $\text{thr1}$, the drone starts to fly ahead to the next waypoint (center of the next landmark) with a heading angle calculated depending on the location of the current and the next waypoints. The second threshold $\text{thr2}$ was chosen to be 1 m, this value represents the accepted error margin for this work, selecting very small values will lead to instability of the drone near the waypoint center. The second operating mode (when $d < \text{thr1}$) is time consuming, so beginning to search for the landmark template must be at an optimum point. The $\text{thr1}$ value was set to the landmark template dimension (about 20 m). When VO drift becomes large, matching the captured image with the landmark might fail, but in this environment, the VO free working (i.e., without compensation) did not exceed 15 m. In case of failure in the extraction of the street borders in the sector ($d < \text{thr1}$), the drone will continue the task but with only VO without drift compensation. Failure in street border extraction is supervised and detected using the street border characteristics (length and parallelism) as explained in Algorithm 1. An explanation of the drone behavior near the waypoint is illustrated in Figure 5. A more comprehensive study about failures in street borders extraction, matching, and switching between operating modes should be considered in other works in future.

Figure 5. Drone behavior near the center of a landmark, when $d < \text{thr2}$, changing the heading to the next waypoint starts.

5.3. Drone Guidance Commands

The drone flies at an average speed $V_{av}$ of 10 m/s, but when it reaches “thr1”, it reduces the speed proportional to the distance to the next waypoint. As it flies away from the same landmark center, it recovers the average speed at the same ratio to reach 10 m/s. Whenever the condition $d < \text{thr2}$ is satisfied, the heading $Y_{\text{UAV}}$ to the next waypoint (new landmark) is entered into the drone guidance program. At the same time, the speed starts to increase and recover its average value. The final guidance and navigation system is illustrated in Figure 6.
Equation (3) can be used in both modes, where in the second mode, \( p_1 \) and \( p_2 \) are captured image and template centers, respectively, in pixels after the alignment, while the velocity is given using Equation (4).

\[
V = \begin{cases} 
V_{av} & \text{if } d > \text{thr1} \\
\max(\lambda \times V_{av}, 4) & \text{if } \text{thr2} < d < \text{thr1} \\
\max(\lambda \times V_{av}, 4) & \text{if } d < \text{thr2}, \text{with a new heading } (Y_{UAV})
\end{cases}
\]

where: \( \lambda = \frac{d}{\text{thr1}} \)

(4)

The lower speed bound was set to 4 m/s to avoid zero division which might lead to stuck at the waypoint. Furthermore, it was necessary to avoid increasing the execution time of the overall task.

6. Simulation and Results

The overall system was implemented in a simulation environment consisting of ROS and Gazebo with IRIS drone model from Ardupilot. Controlling the drone velocity and heading was done using suitable commands in ROS. All the programs for image processing and controlling the drone were written using Python under Linux on i5i0300-2.5 Ghz CPU. The selected flight environment was urban, where a lot of distributed street intersections are available. Planning a drone mission under such considerations does not reduce the problem generality for many reasons. From one side, the drone applications in urban environments are developing. On the other side, the “street” and “street intersection” terms do not mean highways or main streets only, they could be extended to involve small roads or even other geometrical intersections which are common in urban environments. Of course, all of that is according to the equipment and application.

The drone model was equipped with a GPS and a camera with 300 \( \times \) 300 image pixels, the GPS position was used as a reference. The calculated position on each axis is shown in Figures 7 and 8. The horizontal path is shown in Figure 9, and the position error is shown in Figure 10. The high position accuracy was obtained close to the waypoints because of the landmark-image fine matching and the fine drone behavior toward the landmark center. A position RMS error of 1.4 m was obtained near landmarks with an average execution time of 60 ms. The mentioned behavior of the drone enhanced the path tracking task. The speed reduction resulted in a reduction of the drone angles vibration around the horizontal axis. As a result, a more accurate matching was obtained from aside, and a fine rotating to the new heading angle. The maximum position error equaled 2.8 m, compared to 14 m with only the VO system without drift compensation, which is shown in the same mentioned figures. The VO will drift until the next compensation; if illogical scale or angle values are obtained, then no matching with the landmark will take place, and the VO will continue.
drifting until the next waypoint. A deeper solution for fault or failure detection must be derived in the future.

![Graph of position vs. time](image1.png)

**Figure 7.** The calculated position on the x-axis with the GPS reference position and the VO-only calculated position.

![Graph of position vs. time](image2.png)

**Figure 8.** The calculated position on the y-axis with the GPS reference position and the VO-only calculated position.
7. Results and Analysis

This work presented an implementation and simulation of a computer vision navigation system that supports path tracking depending only on an imaging device and some constraints. Incremental positioning using visual odometry and servoing to a predefined landmark were done. Geometrical information from the landmarks represented by street edges and width was used to predict the image scale and the relative angle between the captured image and the template, and to compensate for VO drift. The system was implemented in a simulation environment based on ROS and tested on a predefined path.
constructed by several waypoints centralized with landmarks. The average path height was 100 m, the position RMS error was 1.4 m, and the maximum position error was 2.8 m.

The execution time of the algorithm was 60 ms, and the major part of the time was consumed in the matching process and lines detection. The execution time might be larger than the execution time in some GPS-based methods, but it is more independent and even more accurate when a cheap GPS is adopted. Selecting the street intersections as robust matchings in urban areas, and as well-known structures, proved its efficiency firstly by benefiting from them to estimate the image scale and orientation, and secondly by matching with them which produced a fine and accurate estimate of the drone absolute location. The VO position error grows with time and reaches its maximum value before entering the second fine matching mode. Then, the error starts to decrease near the landmark and falls under 1 m. As more landmarks are used, the maximum position error will decrease, but the task execution time grows because of the velocity control profile near the landmarks and the required image processing time. The suggested behavior of the drone near the waypoint succeeded in reducing the vibration of the drone when it changes its direction. Relying on landmark templates for precise matching and scale estimation was effective since the drone undergoes vibrations in large angles during the flight and direction changing, which affects the ground distance, and hence the captured image scale. Depending on the scale calculation, it is possible to control the drone altitude near the landmark precisely, which would be amazing if it is considered during the path planning before the mission in the future.

In the second operating mode in this work (when \( d < \text{thr} \)), a similar servoing technique to that presented in [15] was used, but it depends on a thresholding technique to detect the desired object, then the drone locks on the object. The lock is done by adjusting the object center with the captured image center without any error margins, which might lead to vibration or instability. Moreover, the thresholding technique resulted in the loss of lock because of the ambiguity of the similar objects and other effects such as illumination change. Compared to [15], depending on street geometry and accepting an error margin up to 1 m resulted in stable and accurate results. In brief, both systems are considered vision-based only, but in our work, a deeper investigation of the target or object characteristics (its equations) was more effective.

It is important to mention that deriving a comprehensive model error for the whole system is not a simple task. It may require independent research and it is out of the scope of this work. A lot of errors’ sources must be considered such as the position of the detected features errors, successive transformation model errors, the estimated scale and orientation, line detection accuracy, each component used in the environment, camera, sensors, and the control system. After that, a deeper solution for fault detection can be derived which is necessary to implement a safe and accurate visual-only navigation system.

8. Conclusions

The paper showed that benefiting from the street’s geometry and dimensions in drone navigation is effective and not limited to ground applications. Furthermore, the expected large increment in the active drone numbers in city skies on one hand, and having no-fly areas of private property or special restricted zones on another hand, will raise the need for not only for anti-collision systems but also for considering the available and allowable flight areas. Of course, the streets could be the logical free suitable options, then benefiting from them to improve the vision-based navigation system will be a brilliant idea. The streets’ planning might take more attention from the governments and engineers. The previous discussion does not mean that the drones will fly only over the streets, but that they could be highly considered during the drone mission planning stage. Benefiting from any part of the street might also be possible, not only the street intersections.

Relying on both image-based and position-based methods ended up with accurate and stable results without the need for any other sensors, i.e., only visual sensors were used which is a good contribution of this paper. On another note, in urban environments, street
intersections or well-structured areas are very common and choosing them as semi-natural landmarks is a good option because of their robustness against volatile changes. Employing the extracted information from the street as explained in the paper is another contribution of this paper. The street was treated as an object, which has been used for image alignment using street width for scale calculation and using street border slope (edges) for orientation calculation on another side. Servoing to the landmark center benefitting from the extracted information and compensating the visual odometry drift succeeded due to the previous employment of information.

The proposed method could be used in a wide range of applications such as delivery, precise payload deployment near the waypoints in catastrophic areas, and auto landing. Our future work will focus on realizing this work in the real world using the DJI Matrice 600 pro drone. The realization appears to be a not-so-difficult task now due to the implementation and testing in such a high advanced 3D environment based on ROS and Gazebo. That environment allowed us to do many tests and tune a lot of parameters as in real work but with zero cost. We will also focus on path tracking techniques using probabilistic methods.

Author Contributions: Conceptualization, A.S.; Funding acquisition, D.S. and S.S.; Investigation, A.S. and D.S.; Methodology, A.S., D.S. and S.S.; Software, A.S.; Writing—original draft, A.S.; Writing—review & editing, A.S., D.S. and S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This study was supported by the Tomsk State University Development Program (Priority-2030).

Conflicts of Interest: The authors declare no conflict of interest.

References


