ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

RSL Robotic Systems Lab
Prof. Dr. Marco Hutter

Master Thesis

Autonomous Vision-based Safe Proximity Operation of a Future Mars Rotorcraft

Autumn Term 2024

Supervised by:
Roland Brockers
Robert Hewitt
Marco Hutter

Author:
Lasse Fierz

Contents

List of Acronyms

- **UAV:** Unmanned Aerial Vehicle
- **SFM:** Structure From Motion
- **LSD:** Landing Site Detection
- **LS:** Landing Site
- **BA:** Bundle Adjustment
- **DEM:** Dense Elevation Map
- **OMG:** Optimal Mixture of Gaussian
- **LOD:** Level Of Detail
- **HiRISE:** High Resolution Imaging Science Experiment (High Resolution Satellite Imagery)
- **LRF:** Laser Range Finder
- **GT:** Ground Truth
- **LSM:** Landing Site Manager

Chapter 1

Related Work

Autonomous safe landing is perhaps the most important part of a rotorcraft's mission. It comes therefore as no surprise, that tremendous amounts of work have been accomplished in the pursuit of achieving this crucial feat.
Visual sensors are highly advantageous for navigation due to their lightweight nature and the extensive research dedicated to their development over the years. The minimal weight of these sensors makes them particularly suitable for applications where payload capacity is a critical concern, such as in the case of a rotorcraft Mars missions. Decades of intensive research have culminated in highly sophisticated algorithms and methodologies that leverage the rich data captured by visual sensors and enable the daunting task of autonomous landing.
[1, 2] and [3] use artificial landing markers as indications of valid landing sites. While [1] and [3] use stationary markings, [2] enabled a rotorcraft to land on a moving target. These approaches, though useful in urban environments, are not applicable on uncharted terrain as found on Mars.
[4, 5, 6] and [7] pursue implementations based on homography assumptions. This is not possible in our setup as we cannot assume homographic conditions on Mars' rough terrain.
A very handy tool for the creation of depth maps to segment landing sites on are range sensors like Lidar as [8, 9, 10] and [11] show. As for our purposes a rotorcraft has to fly on Mars' 1% air density however, weight is a limiting constraint rendering Lidar sensor a suboptimal choice.
For the Mars Mission's lander NASA has used a vision based strategy using a predefined map of Mars' surface and a downwards facing monocular camera to orient the lander in the predefined map[12]. When compared to a lander however, rotorcrafts need to consider much smaller hazards. The available HiRISE satellite images are not sufficient in resolution to supply such prior information to the landing process of a UAV. Rover images could be used as well as Ingenuity's footage however the usage of this data would limit possible flight areas significantly.
[13] use a similar approach as the one used by LORNA. Compared to LORNA's Landing Site Detection [14, 15] however, a non-robot-centric DEM is used. The advantage of LORNA's approach is the implicit drift handling by considering the robot-centered terrain map.
Modern approaches like [16, 17] and [18] use a 2.5D terrain representation similar to the setup used in this project.
Other novel approaches use learning based methods as did [19, 20] and [21]. Though certainly promising regarding accuracy and in the long run definitely a pathway to consider, learning based methods come with significant costs in the context of the task at hand. First of all considering the limitations present in Mars missions, the probable additional computational overhead from learning based methods can not

be neglected. Furthermore, neural network based solutions give up simplicity and interpretablity for the benefit of precision. This is not to be underestimated in a tricky environment such as Mars terrain. Additionally, learning based methods require substantial training data which, in the context of autonomous UAV landing, is not available in large quantities. Lastly complex and specific missions flown on Mars pose their unique challenges. Niche information about these problems can be fused into conventional methods in the form of prior assumptions and constraints.

Chapter 2

System Overview

Hereafter is an image depicting the high level structure of the LORNA project.

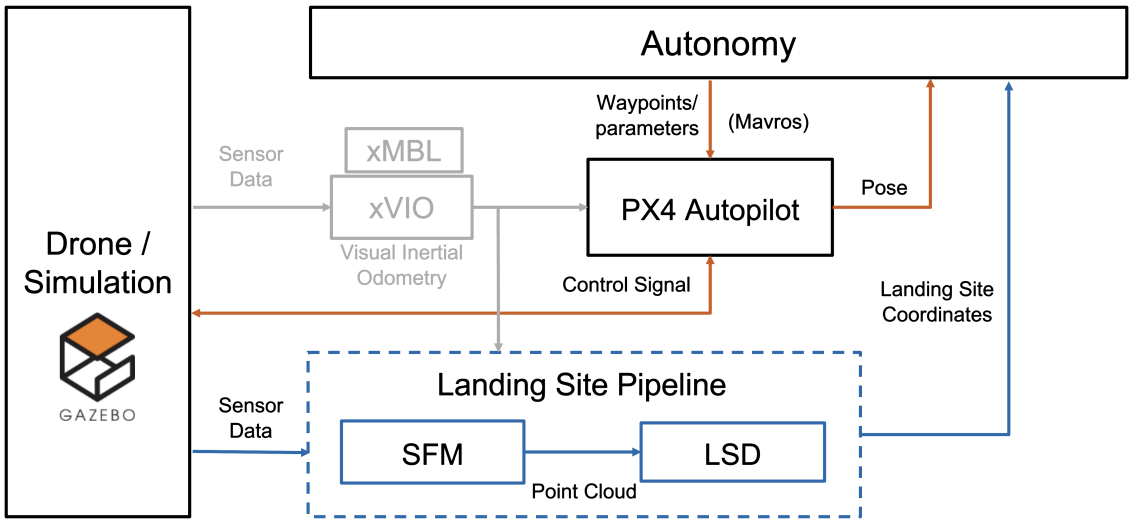


Figure 2.1: LORNA Project Setup

As this thesis revolved around the combination of existing software instances, it is essential to display the individual parts comprising the LORNA project in more detail.

2.1 Simulation

Despite being able to deploy the landing site detection pipeline onto the voxl2 processor the majority of this thesis was done using a Gazebo Garden simulation of the drone.

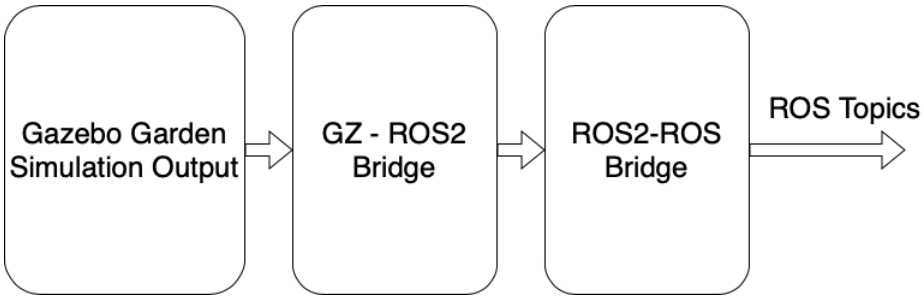


Figure 2.2: Gazebo ROS Bridge

As the entire software stack of the LORNA project is dependent on ROS instead of ROS2 a bridge was used to convert the sensor information from Gazebo to ROS2 and from ROS2 to ROS.

2.2 Landing Site Acquisition Pipeline

The landing site acquisition pipeline consists of two nodes. A structure from motion node [22] which creates a point cloud using a keyframe based stereo approach on monocular images and a landing site detector node [14, 15] which aggregates the depth measurements into a rolling buffer based multi-resolution depth map and segments landing sites on the created DEM. The found landing sites are then supplied to the autonomy.

2.2.1 Structure From Motion (SFM)

SFM - Bundle Adjustment

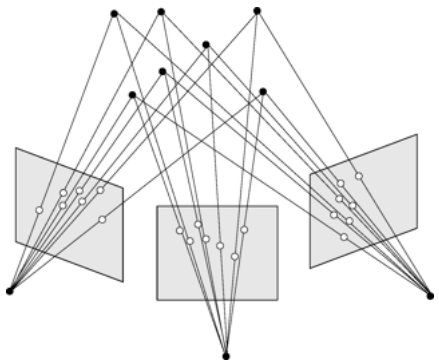


Figure 2.3: Bundle Adjustment Procedure

In a first step the keyframes are filled with the incoming images and their respective camera pose information. Once the keyframes are filled, each iteration the input as well as all the keyframe poses are refined using a Bundle Adjustment algorithm.

SFM - Stereo Depth

The keyframes and their refined poses are then compared to the new incoming image with regards to image overlap and feature retention. Chosing the most adequate keyframe and the incoming frame, one can create a depth image. The depth image is then converted into a point cloud and packaged together with the respective poses of the images. This allows not only to correctly locate the points in a global frame but also to derive the baseline with which that point cloud was created.

SFM - Keyframe Updates

At the end of an iteration the keyframes are updated based on the aforementioned characteristics of image overlap and feature retention quality.

2.2.2 Landing Site Detection (LSD)

LSD - Depth Aggregation

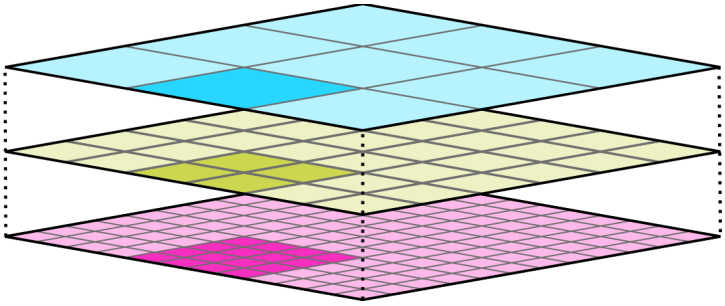


Figure 2.4: Multi Resolution Depth Map

The foundation of the landing site detection mechanism is a rolling buffer based multi resolution depth map as indicated in fig. 2.4. Each base layer cell is represented with 4 cells at a higher resolution layer.
Each point cloud input iteration, the measurements are placed in the respective cells based on the level of detail of the perceived points. In a subsequent step the measurements are pooled up and down the resolution layers in order to make the DEM more consistent and interpolate missing values.
Each cell in this dense elevation map (DEM) is comprised of an optimal mixture of gaussian (OMG) state as described in Proenca et al. [15].
Its update step looks like this:

$$S_t = S_{t-1} + \sigma_{x_t}^{-2} \tag{2.1}$$

$$\mu_t = \frac{1}{S_t}\left(S_{t-1}\mu_{t-1} + \frac{x_t}{\sigma_{x_t}^2}\right) \tag{2.2}$$

$$\sigma_t^2 = \frac{1}{S_t}\left(S_{t-1}(\sigma_{t-1}^2 + \mu t - 1^2) + \frac{x_t^2}{\sigma {x_t}^2} + 1\right) - \mu_t^2 \tag{2.3}$$

Where μ_t is the cell's new mean value, σ_t^2 is the cell's variance and S_t defines an auxilliary variable to keep track of all past variances within a single scalar.
Thus similar to Kalman filters, the OMG cells' uncertainties decline over time as more measurements are entered. Because of this the DEM's terrain estimate converges over time.

LSD - Hazard Segmentation

On the created depth map landing sites can then be detected. This is done using a roughness and slope assessment of the perceived terrain. Roughness defines the maximum absolute altitude difference around a cell in a certain resolution layer and slope is determined by fitting a plane to the vicinity of a considered point.
If the roughness and slope values lie within the acceptance threshold, the spot is recognized as a landing site and marked as such in a binary landing map.

Figure 2.5: Binary Landing Site Map

LSD - Landing Site Selection

After applying a distance transform on the landing site map and performing non-maximum suppression on the landing site sizes, the biggest landing sites are found. Their positions are then refined one last time using a mean shift algorithm that considers roughness, uncertainty and size once again.

Figure 2.6: Binary Landing Site Map after Non Maximum Suppression

LSD - Debug Images

Figure 2.7: Gazebo Simulation Reference

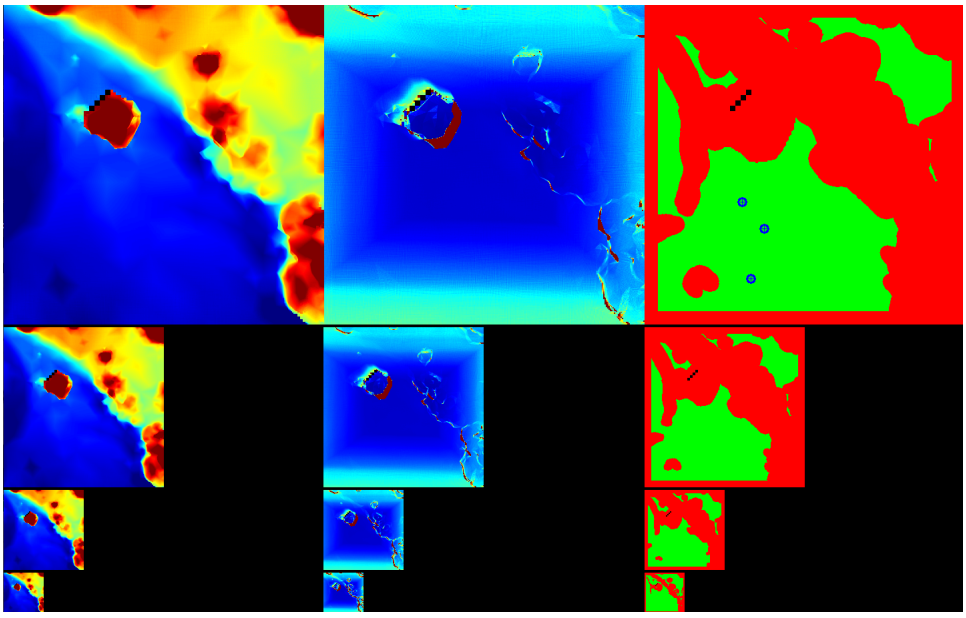


Figure 2.8: LSD Debug Image - Left: DEM, Middle: Uncertainties, Right: LS Map

The landing site detection debug image is a good comprehensive visualization of the landing spot detection procedure.
On the left one can see the multi-resolution map displaying the same terrain area in different resolutions. Red pixels are closer, blue further away.

In the middle one can see the uncertainties of the detected points. For simplification purposes the variance of the detected points is simply the associated stereo depth error.
On the right is the above mentioned binary landing site map. Green indicates valid landing sites, and the blue crosses indicate the chosen non-max suppressed and mean shifted landing sites.

2.3 Autonomy

The autonomous framework was developed within the LORNA project. It is the overarching instance governing all the necessary behaviors and constituting the interface between all the different nodes of the process. It is connected to the flight controller through the Mavros wrapper of the Mavlink protocol. With this connection it can send waypoints and parameter updates to the flight controller. In addition to the in fig. 2.1 shown connections it also communicates with a healthguard node keeping track of the systems health state and alerting in case of anomalies.

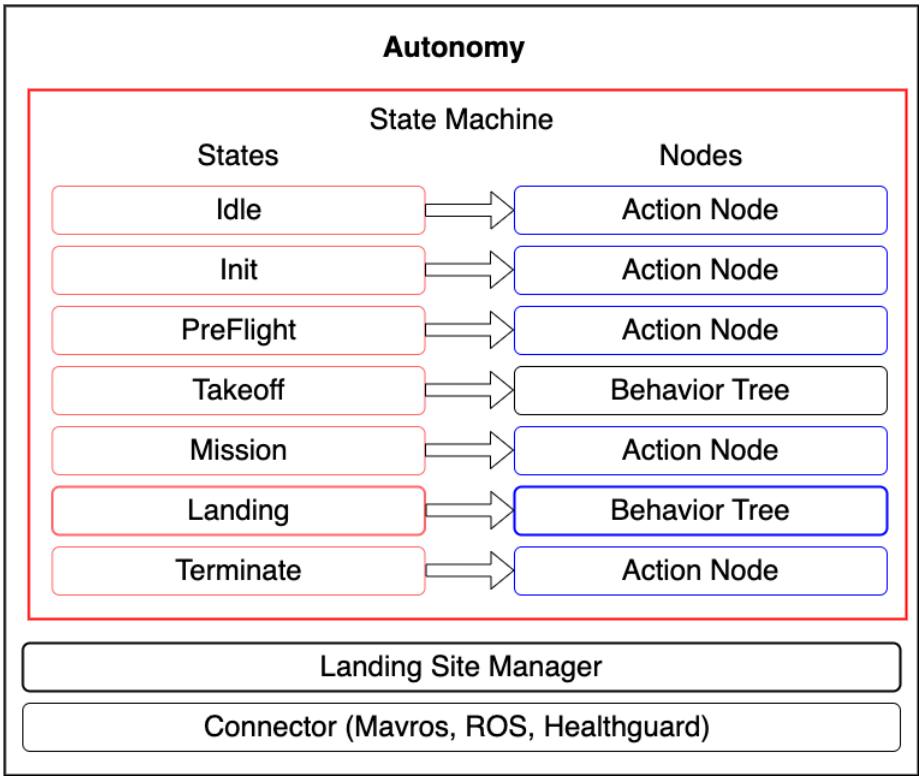


Figure 2.9: Simplified Structure of the Autonomy

The core of the autonomy is the state machine as depicted in fig. 2.9. In each state the respective node is executed which most often is a single action node. For more complicated procedures a behavior tree is used. This is the case for the takeoff as well as the landing node. As indicated in bold, the landing node is the most crucial node in this work as this is where the landing behavior using the landing sites is executed.
In a separate process the landing site manager processes incoming landing sites. Prior to this work the landing site interface was a dummy implementation serving the framework's completeness.

The connector threads interact with the flight controller through Mavros and the other LORNA nodes through a ROS connector.

2.3.1 Behavior Tree

The behavior tree framework within the autonomy is implemented in the traditional way, each comprised of a root node, control flow nodes and action nodes resulting in a modular structure enabling tasks of tremendous complexity. Additionally decorator nodes can be used which alter a single node and function as an add-on on the same level.

Control Flow Nodes

The most imported control flow nodes which were used in this thesis are the following:

- **Fallback Node**: Attempts to execute first child node and if successfull returns a success state. Else it continues to the next child node. If no child node was successfull it returns the failure state. A boolean operator analogy for this would be the logical OR, stopping at the first successful entity.
- **Sequence Node**: Executes one child after another. It only returns the success state if all children nodes ran successfully. Otherwise it returns false. Here an analogy would be the logical AND boolean operator.

Decorator Nodes

The most important decorator nodes are:

- **Inverter Node**: Inverts the output of an action node. Boolean analogon would be the ! (not) operator.
- **Repeat Node**: Repeats a node a number of times until fails or a timeout is reached.
- **Retry Node**: Similiar to the repeat node loop but it repeats the node a number of times only until it succeeds or a defined timeout is reached.
- **Timeout Node**: Adding a timeout to an action node which otherwise wouldn't be temporally limited.

Action Nodes

There are a multitude of actions required for the various subtask a rotorcraft has to perform during a mission. The most important ones for the landing behavior in this work are the following:

- ChangeAltitudeAction: Changes the drone's altitude to the given value. The ascend / descend velocity diminishes upon reaching proximity to the desired waypoint.
- HoldPoseAction: Self-explanatory: the drone holds the current pose for a given time.
- LandingAction: Similar to the ChangeAltitudeAction, it descends to a waypoint which in this case is simply the ground vertically below the drone's current position. Upon reaching a certain proximity to the landing point, the descent velocity is reduced to a minimum in order to accomplish smooth landing.

- NavigateToWaypointAction: Lateral movement to the given waypoint. Same proximity based slow-down mechanism as ChangeAltitudeAction and LandingAction.
- RotateTowardsWaypointAction: Rotates the drone to face the given waypoint.

Chapter 3

Methodology

The autonomous framework[23] allows us to fly independent missions at cruise altitude of 100m+. The structure from motion approach captures 3D information during traversal as its adaptive baseline allows it to perceive high quality depth information also at such high altitudes. This information can be used by LSD in order to detect landing sites during mission.
At low altitudes SFM works as well but surrounded with obstacles, the need for lateral motion poses significant risk. This is because the drone does not retain any hazard information due to the limitations of computational complexity present for mars flights. Therefore it could be said that the drone flies blindly with regards to obstacles. In thesis I present a stereo camera implementation to remedy these shortcomings.

3.1 Stereo Camera

The implementation of the stereo camera sensor itself is very straightforward as simply duplicating an existing camera, offsetting it an adequate distance to resemble the real model and settings the parameters to equal the hardware results in the desired outcome.
Processing this information is different than for the existing SFM node. A new depth generation node was put in place.
Both images as well as the drone's base link pose are given to the node as input and processed. Using openCV's stereoSGBM algorithm, disparity images are created. These are then converted to point clouds using the stereo depth formula 4.1 and coordinate transforms. Lastly they output the point cloud in the world frame as well as the two camera poses used to create it.

3.2 Autonomous Landing Procedure

Having implemented a stereo camera as a low altitude alternative to SFM and after ensuring a correct ground truth comparison, the main contribution of this work could be tackled: Bringing the visual landing site pipeline together with the autonomous framework in order to achieve reliable autonomous landing in unknown terrain.
The method of the landing pipeline can be split into the following parts:

- Landing site detection output Prior to this work, LSD only published a landing site's location. To give the autonomy more information to make adequate

decisions, LSD is changed to also yield additional characteristics which are used during the landing site segmentation process.

- Landing site interface of the autonomy The autonomous framework is changed to correctly receive and handle incoming landing sites. The incoming candidates are ordered according to a novel landing site heuristic and updated upon being redetected.
- Autonomous landing behavior Using the newly expressive landing sites and their handling procedure, the adaptive landing instance is put in place using an existing behavior tree framework within the autonomy. Additional modular action nodes are created and previously existing ones are altered in order to achieve a precise and safe landing procedure.

Chapter 4

Stereo Camera Depth

First and foremost, to emphasize the advantages of range information given by a stereo camera the following comparison is performed:

4.1 Stereo Camera - SFM Comparison

The specific advantage of a stereo camera implementation when compared to SFM can be summarized in the following points:

- No necessity of lateral motion
- Hardware depth perception
- DEM conversion
- Efficiency

4.1.1 Lateral Motion

As already mentioned above the need for lateral motion in itself is an undesirable necessity for a rotorcraft in unknown terrain.
In this setup the structure from motion approach is based on a keyframe buffer which needs to be filled with image-pose pairs at different horizontal positions in order to start acquiring depth information. The current setting in the implementation Domnik et al. [22] uses 6 keyframes. Therefore for a single point cloud it is necessary to move laterally 6 times in order to start perceiving depth. Following the depth error formula from a stereo disparity image (5.3) and assuming an altitude of 2.5m above ground with a focal length of 256 pixels and a disparity error of 0.5 pixels, the necessary baseline in order to keep the depth error below a critical 5cm is:

4.1.2 Software vs Hardware Depth Perception

Structure from Motion, being a software node that relies on camera poses supplied by a state estimator, is by design subject to inaccuracies. A depth node based on a stereo camera on the other hand works with a fixed rigid baseline between the camera views. Thus for low altitude flights that bear the danger of collision, a more robust hardware approach is preferred.

4.1.3 DEM Conversion

As described in section 2.2.2 the multi-resolution DEM used for depth aggregation in LSD is based on Optimal Mixture of Gaussian cells and thus converges over time. According to section 2.2.2 the landing sites chosen are likely on terrain with low uncertainty. Because of this landing sites are more likely to be detected and have in general a better quality when the terrain perceived has been viewed.
When a landing site has been selected we need to make sure that the landing site is actually correctly detected and of good quality. For this we would like to (re-)detect landing sites on rather converged terrain. Structure from Motion needs constant lateral motion for this. A stereo camera depth node simply hovers in place for any given amount of time.

4.1.4 Efficiency

All in all the stereo camera setup allows us to perceive a landing site at course altitude and after having traversed horizontally to that location, we can simply descent to a stereo camera friendly altitude for the verification. Compared to repeated lateral coverage of the area in question this is a huge increase in efficiency.
Looking in depth at the stereo alternative of depth generation, we can first analyze the theoretical threshold of this system.

4.2 Theoretical Analysis

When it comes to depth perception the obvious drawback of a stereo camera is its limited baseline. It only perceives depth accurately for objects within a certain proximity to the lense.
Assuming a perfectly calibrated and rectified camera there is still always an inaccuracy in the depth estimation arising from the disparity error.
From a given disparity estimate the depth error is derived as follows:

$$z = \frac{f \cdot b}{d} \tag{4.1}$$

Where b is the z is the depth estimate, b is the baseline, f is the focal length and d is the disparity value.
Taking the derivative of z w.r.t. d we get

$$\frac{\partial z}{\partial d} = -\frac{f \cdot b}{z^2} \tag{4.2}$$

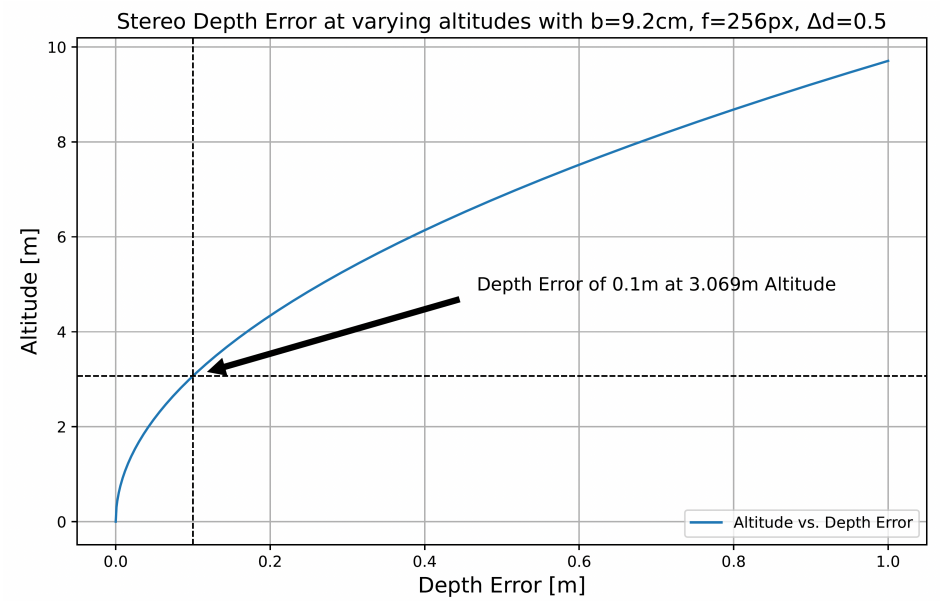
And substituting (eq. (4.1)) we get:

$$\partial z = \frac{z^2}{f \cdot b} \partial d \tag{4.3}$$

Where the sign was left away as for our application there lies equal danger in a point being perceived too close and too far away.
For the maximum altitude given a maximum allowable depth error this yields:

$$z_{\max} = \sqrt{\frac{\Delta z_{\max} \cdot b \cdot f}{\Delta d}} \tag{4.4}$$

Where Δz is the depth error and Δd the disparity error.
The stereo camera mounted on the drone in JPL's aerial vehicle lab had a baseline of about 10cm and a focal length of 256.



With these properties and estimating a subpixel precision disparity error of 0.5 pixels the depth error at varying altitudes looks as follows:
Let's assume we allow a maximum depth error of 10cm. Considering this constraint we can fly at a maximum altitude of about 3m as indicated in section 4.2.
This limitatin has to be kept in mind. However it is neither too surprising nor is it too restrictive as the stereo camera is simply a depth alternative for low altitude flight maneuvers. In the context of an entire science mission it is almost exclusively used for landing site verification purposes.

4.3 Implementation

Like Structure from Motion, the stereo depth instance is a ros node which is given images and image poses from the xVIO state estimator. As the state estimator was in its final development stages during my thesis, camera images and a ground truth camera pose from the simulation were used instead as input for the stereo algorithm. Note that only one camera pose is given as the second one is derived in a straight forward manner, given the fixed baseline.
The stereo depth implementation was done using opencv's StereoSGBM algorithm. As stereo depth generation is a widely known topic I won't go into the details here. The final output of the node is a generated pointcloud in the world frame together with two poses representing the camera locations of the generated pointcloud.

4.3.1 Transform Overview

A critical part of navigation is always the consistency of the coordinate systems in which quantities are represented. Hereafter is a display of the present coordinate systems in the stereo camera setup.

4.3.2 Landing Site Detection withouth Lateral Motion

Taking off vertically with the drone in the simulation, the first landing site without lateral motion was found.

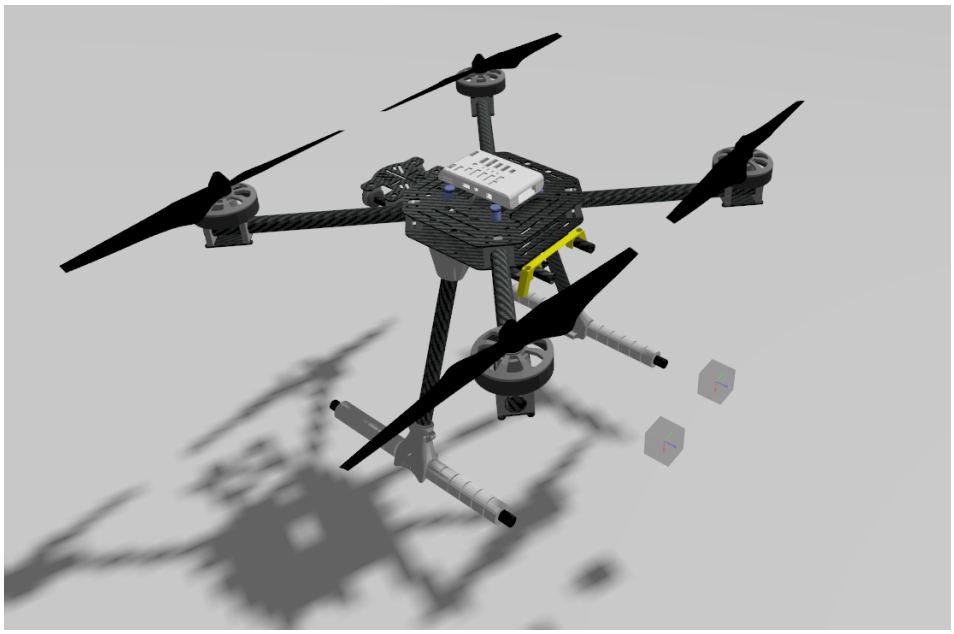


Figure 4.1: Stereo camera on drone indicated by opaque boxes

Figure 4.2: Drone during vertical ascent in simualation

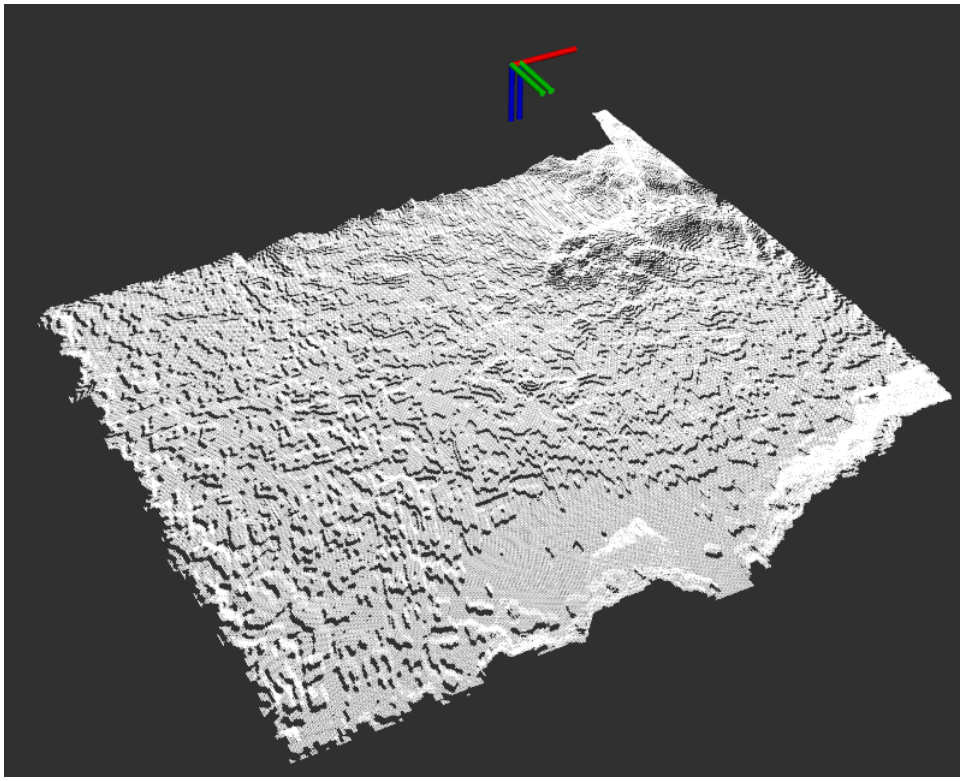


Figure 4.3: Rviz visualization of created point cloud from stereo camera

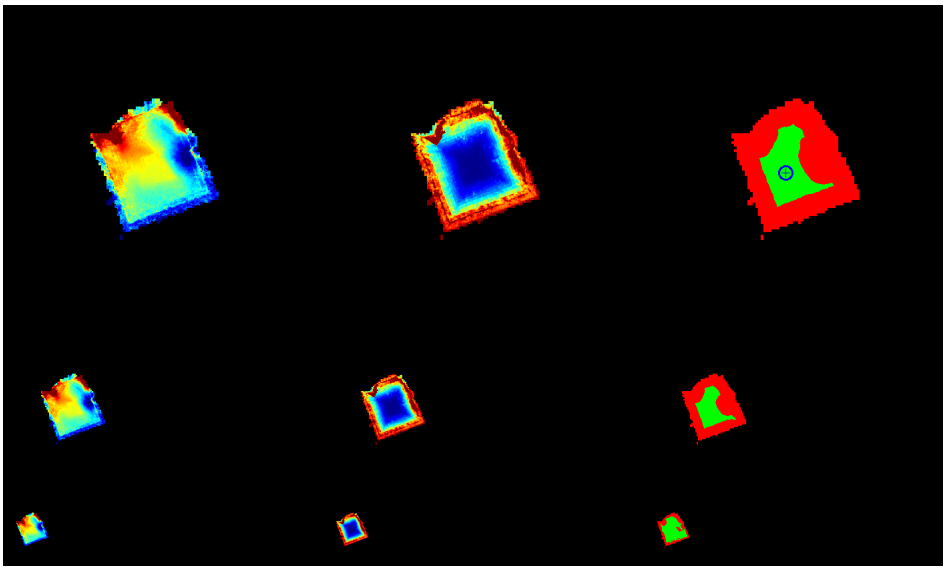


Figure 4.4: LSD Debug output displaying LORNA's first detected landing site during vertical motion

4.3.3 Switching

In order to achieve the final desired perception mechanism of flying laterally with SFM and using a stereo camera depth node at low altitudes, one needs to switch between the two alternatives.
The obvious flag to use in the switching mechanism is the current altitude above ground. This could be achieved by analyzing the generated point cloud at a given iteration to the determine the median altitude which indicates the altitude above ground. This however is avoidable computational overhead.
As mentioned in **??** the drone has a laser range finder on board. This allows us to get an estimate of the altitude above ground at any given moment without the need for image processing.
Therefore the switching is performed by using a separate ros subscriber which continuously checks the lrf's measurement and activates or deactivates the SFM node and stereo node respectively.

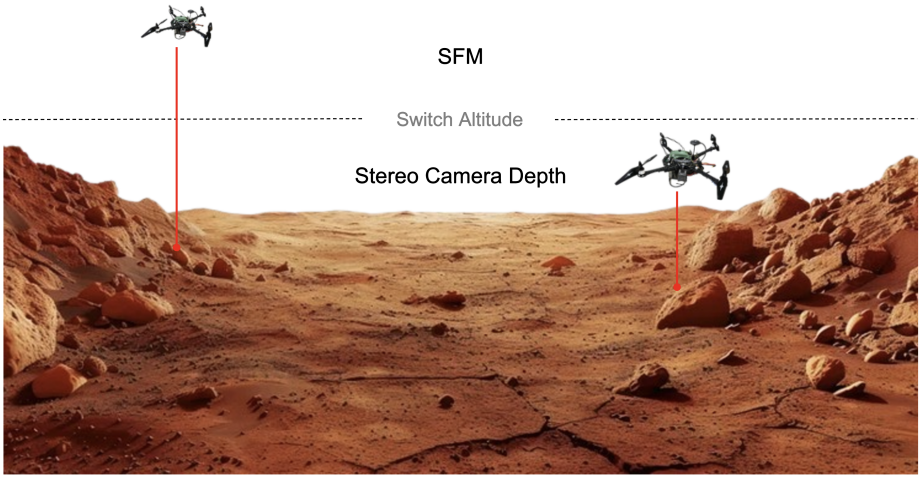


Figure 4.5: Laser Ranger Finder Based Switch between Depth Sources

4.4 Qualitative Practical Analysis

Once implemented the landing site detection instance could be supplied by the stereo depth node. The result thereof can be seen below:

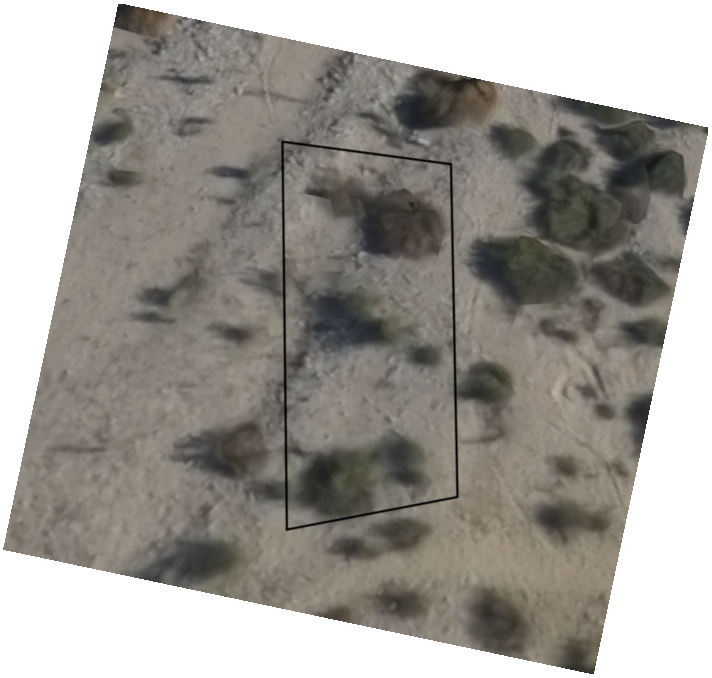


Figure 4.6: Considered terrain patch in Gazebo simulation

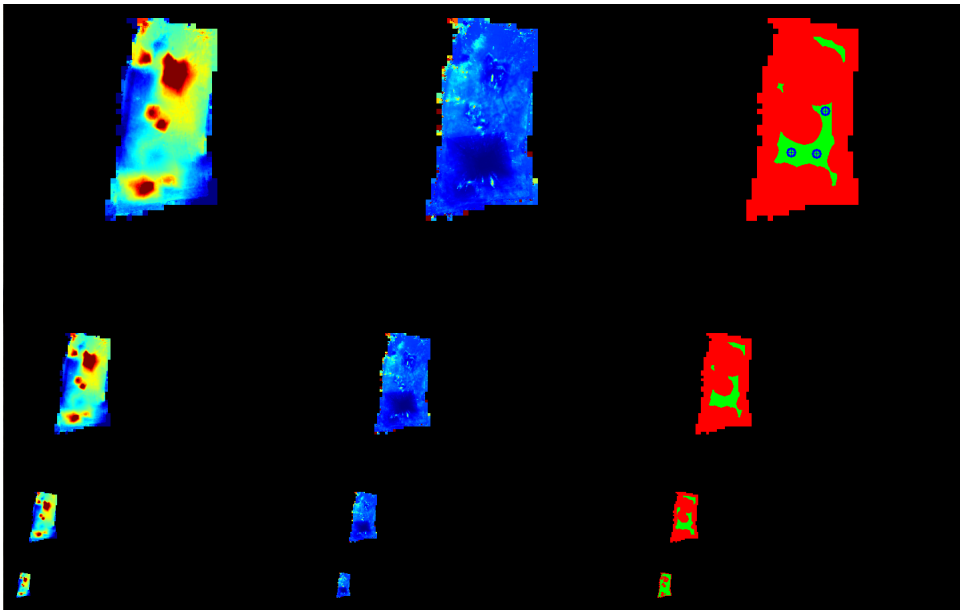


Figure 4.7: Stere camera depth supplied LSD debug image at 2.5m altitude

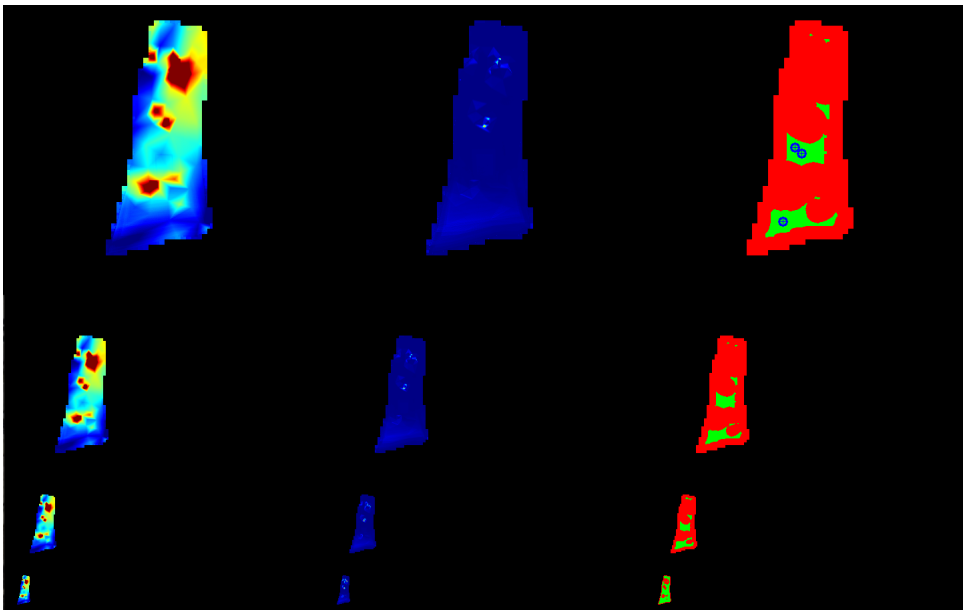


Figure 4.8: GT depth supplied LSD debug image at 2.5m altitude

When comparing the result to the ground truth LSD output it can be seen that LSD creates a very accurate DEM from the stereo camera depth input. The landing sites detected are reasonable when compared to the terrain reference.

4.4.1 Ground Truth

Throughout the entirety of this project, the simulation's ground truth pose has been used which is the pose of the drone's base link. When applying the static camera transform to it, this yielded the ground truth camera pose. For stereo and SFM evaluation purposes however, also ground truth point clouds are required.
Obviously the same approach could be taken for a point cloud created by a depth camera in the simulation. However implementing such a depth camera with equal camera parameters as the reference stereo camera showed different results.

Figure 4.9: Simulation Reference

Figure 4.10: GT depth image with wrong calibration

Looking at fig. 4.9 and fig. 4.10 the error looks like a simple translation error or is even hard to see at all. Focus for instance on the edge of the canape in the top left. When comparing the extrinsic parameters of the cameras, they were perfectly aligned. Further tests showed however that there was an actual error in the Gazebo Garden source code.[1] No matter what camera parameters were passed, the depth camera had the same fov. Changing this depth camera implementation finally allowed the usage of a gazebo depth camera as a supplier of ground truth depth information.

[1] For more footage on this see **??**

Chapter 5

Autonomous Landing Procedure

This implementation can be split into the following parts:

- Landing Site Heuristic
- Conceptual Behavior
- Behavior Tree Implementation

5.1 Landing Site Handling

5.1.1 LSD Properties

With the low altitude depth alternative in place, the connection of the autonomy with the landing site detector could be tackled.
Before this work the output of the landing site dection algorithm was merely the location of a found landing site. However as described in section 2.2.2 the landing site detection algorithms segments hazards based on roughness and slope. Subsequently it considers the size of a landing site as well as the uncertainty associated with a certain selected location.
Simply outputting the location of a landing site is therefore a waste of information when so many characteristics are at hand to make an informed selection.
I decided on the following properties to be output alongside the site's location:

- Uncertainty
- Roughness
- Size
- Obstacle Altitude

The final landing site detection output is a custom landing site ROS message containing the above mentioned characteristics of the detected spot.

Roughness

The roughness value the exact value already used for the hazard segmentation step in the landing site detection.

Uncertainty

The uncertainty value is also a product of the landing site detection algorithm. It denotes the averaged uncertainty across the area around a given landing site. The uncertainty of a single map cell denotes the stereo depth error estimates merged over time.

Size

To determine the size of a landing site, the landing site detection algorithm performs a distance transform on the created landing site map in order to find the closest non-landing site for any found landing site. This returns the radius of the largest valid landing circle around a landing site. Calculating the physical value, the metric radius is returned as the size of a landing site.

Obstalce Altitude

The obstacle altitude was newly introduced in this work. It defines the currently highest point of the aggregated DEM's highest resolution layer. As no actual object detection is performed and no hazard information is retained in this visual pipeline, this value serves the autonomy as an indication of the obstacles heights to avoid in the vicinity of a certain landing site. More on this in section 5.1.

5.1.2 Landing Site Heuristic

The autonomy processes the in section 5.1.1 listed values in order to arrive at the following final landing site properties:

- Current Distance to Drone
- Roughness
- Uncertainty
- Size
- Verification Altitude

The final heuristic defining the quality of a landing site is in fact a square loss function:

$$L_{LS} = w_{dist}L_{dist} + w_{rough}L_{rough} + w_{var}L_{var} + w_{size}L_{size} + w_{verAlt}L_{verAlt} \quad (5.1)$$

Current Distance to Drone - L_{dist}

Well likely the single most important characteristic of an LSD-prefiltered[1] landing site. Each iteration the current distance to the drone's position is calculated for each retained landing site. The distance is then normalized by dividing it by the cruise altitude which is 100m. In practice there were easily enough landing sites found while moving to allow landing sites to fall off when being farther away than 100m.

[1] Each received landing site has already undergone a threshold filtering regarding slope and roughness.

Roughness - L_{rough}

The roughness property is the unaltered roughness value received from LSD. It is already normalized and enters the loss function as it is.

Uncertainty - L_{var}

The same holds for the uncertainty. It is already normalized by design and enters the loss function unaltered.

Size - L_{size}

Analogous to the roughness and uncertainty properties the size comes from the landing site detection directly. However unlike the two preceeding properties it is not normalized but simply denotes the metric radius of the largest circle of valid landing area that can be fit around a given landing site. This is achieved in LSD by performing a distance transform on the created landing site image.
In order to normalize this value the maximum landing site size is retained and each landing site's size is divided by it in order to achieve normalized size information.
Also as can be seen in eq. (5.1), the size contribution enters the loss function with a negative sign. This is due to the fact that compared to all other characteristics, the size defines a property that we would like to maximize.

Verification Altitude - L_{verAlt}

A site's verificaiton altitude is the smallest vertical distance between the drone and the landing site at which that site was (re-) detected.
The verification altitude is a useful property because of numerous reasons.

Further Indication of Certainty

First of all similar to the uncertainty metric the verification altitude indicates how certain we can be about a detected landing site as spots detected at lower flight altitudes are more likely correct due to the reduced depth error. Even though it might seem overlapping with the uncertainty property in this regard, these two characteristics are quite complementary as the uncertainty takes OMG convergence and camera specifics into consideration while the verificaiton altitude is a purely location based metric.

Landing Site Property Updates

As the verification altitude yields a simple and good estimation of the trustworthiness of an incoming landing site, it can be used as a flag to know, when a landing site's properties should be updated. When a landing site is redetected with a verification altitude lower than the previously stored one, the algorithm trusts it more and alters the previously stored properties to the new ones received.

Verification

Continuously updating the verification altitude upon redetection allows us to determine the lowest altitute, at which a landing site was redetected. This information can be used to verify that a given site was considered a valid landing spot even at low altitudes.

5.2 Conceptual Behavior

Bringing everything together and emphasizing landing aspects of an autonomous flight perspective, we arrive at the following procedure:

5.2.1 Takeoff

The necessary checks and initializations are performed. This includes the created mission waypoint plan, the ros-, as well as the mavros-connection setup with the initial setting of mavros parameters.
Then the drone takes off vertically until it reaches the first waypoint's target altitude.
During this phase the stereo camera feeds depth images into LSD until the laser range finder switches to SFM which results in a stop of depth supply as SFM does not detect depth during vertical motion.

5.2.2 Prerequisite - Landing Site Handling

Landing sites are constantly received by the autonomy's ROS connector which constitutes the ros interface with the landing site detector. The sites have to be processed in a separate thread. This is handled by a landing site manager (LSM) singleton class.

Landing Site Manager

The incoming landing sites are ranked according to a loss function(5.1) and stored in a max-heap buffer in order to easily switch out the worst landing site for a new one at any given time.

5.2.3 Transition into Landing

During a mission the drone flies to different waypoints. During the lateral motion periods of these flights, the drone continuously processes the incoming landing sites using the LSM.
Once the last mission waypoint has been reached, the transition to the landing behavior occurs.
At first,

5.3 Behavior Tree Implementation

As a mission is flown at 100m altitude we would like to verify it closer to the ground where we can be more certain about measured terrain.
To this end the drone moves laterally to a chosen landing site and then descends blindly to a certain verification altitude above that spot. We can do so safely as, yes, the initial landing site estimate from 100m altitude might not be a good choice, however as can be seen in eq. (5.3), given an approximate baseline of 15m at 100m altitude with a focal length of 256 and an assumed subpixel disparity error of 0.5, the structure from motion algorithm yields depth measurements with an approximate depth error of 1.3m.

$$\Delta z = \frac{z^2}{f \cdot b} \Delta d \tag{5.2}$$

$$\Delta z = \frac{100m^2}{256 \cdot 15m} 0.5 = 1.302m \tag{5.3}$$

Therefore in the very worst case szenario of a depth error of 1.3m we are still safe when we descend to a verification altitude of 2.5m above a detected landing site. Avoiding SFM verification patterns at intermediate altitudes saves a tremendous amount of time and energy.
The verification is performed using the stereo camera(**??**).

5.3.1 Action Definition

The final landing behavior is implemented in the form of a behavior tree which allows adaptive decision making.
The autonomous framework already defined the core control flow nodes as well as some of the action nodes required for the landing behavior. (2.3.1) Hereafter displayed is a list of additional actions needed to be defined in this work. It should be noted, that they define individual actions and should not be a description of the entire behavior. For this consider section 5.3

CheckLandingSite

Simple utility action which checks whether any landing sites have been found by querying the LSM's landing site buffer length.

ChangeAltitudeLSAction

This action was implemented with the purpose to allow us to descend to a certain fixed altitude above a chosen landing site. The creation of another action for this purpose is simply a utility which let's us avoid having to pass a function pointer in the action defition as the arguments passed in a behavior tree are always evaluated at the time of the creation.

GetClearAltitude

Once a landing site has been chosen, the failsafe mechanism to go to that landing site is the following:

1. Ascend to a safe altitude.
2. Traverse laterally to the landing site's xy-position.
3. Descend to the landing site or verification altitude.

The question remains however, what the adequate clearing altitude is. The goal is to find an altitude high enough to fly safely without the risk of collision yet low enough as to not waste energy for the increased ascent / descent distances.
This is where the aforementioned obstacle altitude from section 5.1.1 comes in. The obstacle altitude gives us an indication of the height to clear around the landing site. Therefore we can take this as the start altitude for the derivation.
As the DEM created by LSD covers only a limited area, the obstacle altitude is only an indication of the highest terrain present at the chosen landing site. The worst case szenario would be the detection of the landing site at the very edge of the DEM shortly before significantly higher terrain starts. Therefore one has to anticipate this case.
In practice a safe altitude buffer is implemented by assuming a worst case 45-degree incline of the terrain starting immediately at the landing site. Thus the necessary clearing altitude can be derived by linearly interpolating the final value from the initial obstacle altitude and the distance between the drone and the landing site which needs to be covered.

In the end to not ascend to excessive heights, the clearing altitude is capped at a predetermined, terrain-based failsafe altitude

$$z_{\text{clear}} = z_{\text{obst.}} + z_{\text{buffer}} + d_{\text{drone-LS}} \tag{5.4}$$

$$z_{\text{clear}} = \min\left(z_{\text{clear}}, z_{\text{failsafe}}\right) \tag{5.5}$$

Above we can see the derivation of the clearing altitude where z_{clear} defines the clearing altitude, z_{buffer} a safety buffer on top of the obstacle altitude, $d_{\text{drone-LS}}$ the drone's current distance to the site and z_{failsafe} the failsafe altitude at which the clearing altitude is capped.

GetLandingSiteAction

Upon leaving the mission state and entering the landing state, the autonomy attempts to select the currently best landing site according to the in section 5.1 introduced loss function.
This is done by using the landing site manager in order to rank the landing sites in an ascending sorted list (as opposed to a max-heap) and selecting the first entry with the lowest loss.
The selected landing site is then stored in a blackboard interface which allows easy data sharing between all the actions within the autonomy.
In the end the GetLandingSiteAction invokes the GetClearAltitude action to also store the clear altitude on the blackboard. This is used by the NavigateToWaypointAction to move to that site on the derived safe altitude.

LandingSiteVerificationAction

Once a landing site is chosen, we have to make sure it's a good spot to land before attempting to do so.
The LandingSiteVerificationAction does this by using the verification altitude property of a given landing site:
When a landing site is detected by SFM at 100m altitude it will be overwritten when redetected at 2.5m above the ground.
This is the mechanism exploited in order to verify a landing site. The drone hovers above the landing site for a predetermined duration in place and attempts to redetect the chosen landing site. This means that the LSM continuously processing incoming landing sites and if one is close enough to an existing one, it is considered a redetection. In that case, the landing site is verified and the landing action can be triggered.
In case of verification failure, the landing site is not only removed but actively banned in order to prevent future false positives at high altitudes.
Also as previously mentioned the verification hovering at low altitudes leads most probably to the detection of close by landing sites. This is arguably as important as the verification of a previous landing site as it yields a good candidate in close proximity. In practice, it turned out, it is equally likely to verify a landing site as it is to not verify one but detect a high quality landing site close by.

LandingSiteSearchAction

The previously described actions are core implementations of the nominal behavior in the landing sequence. However what happens when no landing sites are found, either through fault of the landing site detection setup or due to really unfavorable terrain?

The most intuitive answer seems a good idea - simply look further for a site. The technical implementation of this task at high altitudes requires the detection by SFM and therefore lateral motion. So an easy solution is to fly a pattern at a new location with the exact same landing site handling procedure as in the nominal case. The LandingSiteSearchAction implements this through a predefined rectangle of waypoints which are flown through. This sub-mission is cancelled upon detecting a single landing site. In that case the usual landing procedure is continued.

GetNextPatternCenterWPAction

In case of failure when looking for additional landing sites, the adaptive procedure is to simply move to a new location and try anew.
The drone picks a random position around the final mission waypoint, flies there and again moves laterally in a rectangle shape in the hope of detecting landing sites. Note: This procedure is repeated a fixed number of times using the retry control node described in section 2.3.1. Optimally however, this would be performed as long as the battery state permits it.

Bibliography

[1] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3. IEEE, 2002, pp. 2799–2804.

[2] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza, "Vision-based autonomous quadrotor landing on a moving platform," in *Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics*, Shanghai, China, 2017.

[3] L. Mu, Q. Li, B. Wang, Y. Zhang, N. Feng, X. Xue, and W. Sun, "A vision-based autonomous landing guidance strategy for a micro-uav by the modified camera view," *Drones*, vol. 7, no. 6, p. 400, 2023.

[4] S. Bosch, S. Lacroix, and F. Caballero, "Autonomous detection of safe landing areas for an uav from monocular images," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing: IEEE, 2006.

[5] R. Brockers, P. Bouffard, J. Ma, L. Matthies, and C. Tomlin, "Autonomous landing and ingress of micro-air-vehicles in urban environments based on monocular vision," in *SPIE Defense, Security and Sensing*, 2011.

[6] V. Desaraju, M. Humenberger, N. Michael, R. Brockers, S. Weiss, and L. Matthies, "Vision-based Landing Site Evaluation and Trajectory Generation Toward Rooftop Landing," *Autonomous Robots*, vol. 39, no. 3, pp. 445–463, 2015.

[7] R. Brockers, M. Hummenberger, S. Weiss, and L. Matthies, "Towards autonomous navigation of miniature uav," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 645–651, 2014.

[8] N. Trawny, A. Huertas, M. Luna, C. Villalpando, K. Martin, J. Carson, and A. Johnson, "Flight testing a real-time hazard detection system for safe lunar landing on the rocket-powered Morpheus vehicle," in *Proc. AIAA SciTech Conference*, 2015.

[9] M. Luna, E. Almeida, G. Spiers, C. Villalpando, A. Johnson, and N. Trawny, "Evaluation of the simple safe site selection (s4) hazard detection algorithm using helicopter field test data," in *AIAA Guidance, Navigation, and Control Conference*, 2017.

[10] A. Johnson, A. Klumpp, J. Collier, and A. Wolf, "Lidar-based hazard avoidance for safe landing on Mars," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 25, no. 5, 2002.

[11] S. Scherer, L. Chamberlain, and S. Singh, "Autonomous landing at unprepared sites by a full-scale helicopter," *Journal of Robotics and Autonomous Systems*, 2012.

[12] A. Johnson, N. Villaume, C. Umsted, A. Kourchians, D. Sternberg, N. Trawny, Y. Cheng, E. Giepel, and J. Montgomery, "The Mars 2020 lander vision system field test," in *Proc. AAS Guidance Navigation and Control Conference (AAS-20-105)*, 2020.

[13] A. Johnson, J. Montgomery, and L. Matthies, "Vision guided landing of an autonomous helicopter in hazardous terrain," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2005.

[14] P. Schoppmann, P. F. Proenca, J. Delaune, M. Pantic, T. Hinzmann, L. Matthies, R. Siegwart, and R. Brockers, "Multi-Resolution Elevation Mapping and Safe Landing Site Detection with Applications to Planetary Rotorcraft," 2021.

[15] P. F. Proenca, J. Delaune, and R. Brockers, "Optimizing Terrain Mapping and Landing Site Detection for Autonomous UAVs," 2022.

[16] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.

[17] C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza, "Continuous on-board monocular-vision based elevation mapping applied to autonomous landing of micro aerial vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[18] S. Daftry, M. Das, J. Delaune, C. Sorice, R. Hewitt, S. Reddy, D. Lytle, E. Gu, and L. Matthies, "Robust vision-based autonomous navigation, mapping and landing for MAVs at night," in *International Symposium on Experimental Robotics (ISER)*, 2018.

[19] F. Neves, L. Branco, M. Pereira, R. Claro, and A. Pinto, "A Multimodal Learning-based Approach for Autonomous Landing of UAV," 2024.

[20] S. Abdollahzadeh, P.-L. Proulx, M. Allili, and J.-F. Lapointe, "Safe landing zones detection for uavs using deep regression," in *2022 19th Conference on Robots and Vision (CRV)*, 2022, pp. 213–218.

[21] H. Tovanche-Picon, J. González-Trejo, A. Flores-Abad *et al.*, "Real-time safe validation of autonomous landing in populated areas: from virtual environments to Robot-In-The-Loop," *Virtual Reality*, vol. 28, no. 66, p. 66, 2024.

[22] M. Domnik, P. Proenca, J. Delaune, J. Thiem, and R. Brockers, "Dense 3D-Reconstruction from Monocular Image Sequences for Computationally Constrained UAS," 2021.

[23] L. Di Pierno, R. Brockers, and R. Hewitt, "Autonomous Long-Range Flight Execution for Future Mars Rotorcraft," 2024.