Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Robotic Systems Lab
Prof. Dr. Marco Hutter

Master Thesis

# 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

# Abstract

An autonomous rotorcraft literally stands or falls on it's reliable landing capabilities. When that same rotorcraft is on Mars, this procedure cannot fail even once. The LORNA (Long Range Navigation) project tackles this problem by introducing a Landing Site Detection (LSD) mechanism which aggregates Structure From Motion (SFM) point clouds into a multi resolution depth map and performs landing site segmentation on the collected depth information. In this master's thesis we incorporated this landing site detection pipeline into an autonomous framework and implemented a behavior tree based landing mechanism to safely and efficiently select, verify and discard detected landing sites. Furthermore the pipeline was enhanced using a stereo camera depth input alternative to SFM for lower altitudes to remove the necessity of lateral motion in order to perceive depth. The software was tested extensively in a gazebo simulation on different synthetic as well as recorded environments and different behaviors were considered and analyzed throughout various Monte Carlo iterations. The contributions in this work aim at enabling future mars rotorcrafts to autonomously and reliably land at safe locations thus enabling a more daring aerial exploration of the red planet.

# Chapter 1

# Setup

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

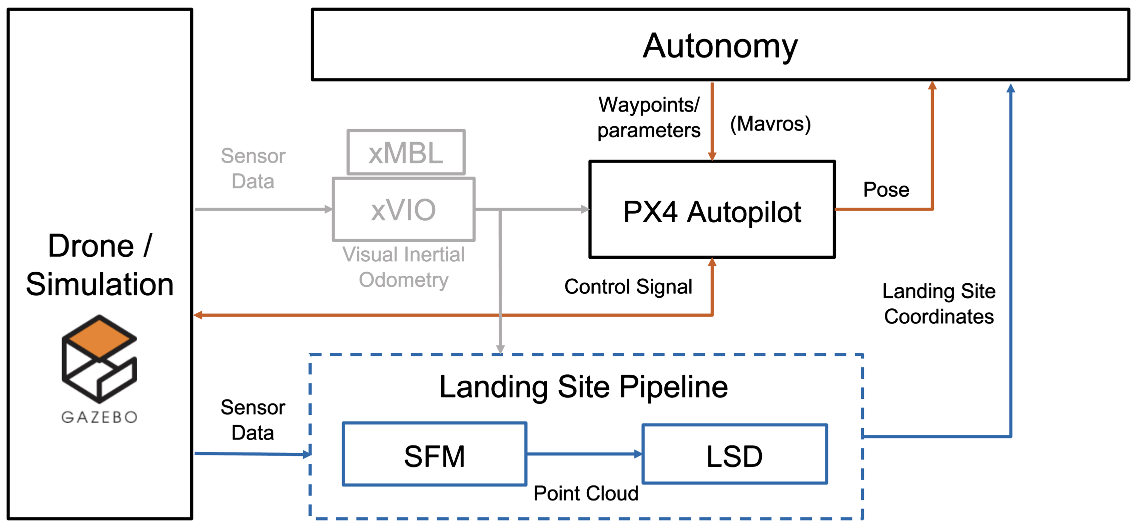

Figure 1.1: LORNA Project Setup

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.

## 1.1 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.

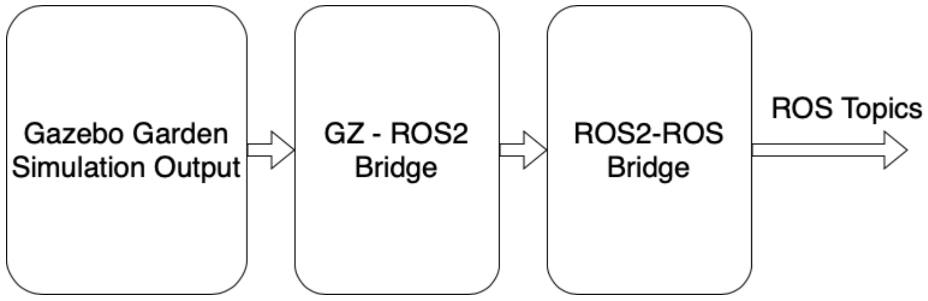

Figure 1.2: Gazebo ROS Bridge

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.

## 1.2 Landing Site Acquisition Pipeline

The landing site acquisition pipeline consists of two nodes. A structure from motion node [1] which creates a point cloud using a keyframe based stereo approach on monocular images and a landing site detector node [2, 3] 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.

### 1.2.1 Structure From Motion (SFM)

**SFM - Bundle Adjustment**

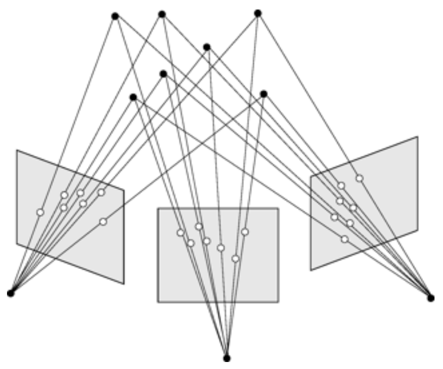

Figure 1.3: Bundle Adjustment Procedure

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.

### 1.2.2 Landing Site Detection (LSD)

#### LSD - Depth Aggregation

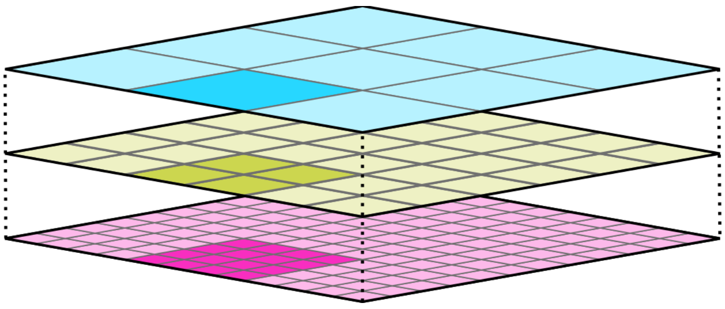

Figure 1.4: Multi Resolution Depth Map

The foundation of the landing site detection mechanism is a rolling buffer based multi resolution depth map as indicated in fig. 1.4. 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. [3].
Its update step looks like this:

$$S_t = S_{t-1} + \sigma_{x_t}^{-2} \tag{1.1}$$

$$\mu_t = \frac{1}{S_t}\left(S_{t-1}\mu_{t-1} + \frac{x_t}{\sigma_{x_t}^2}\right) \tag{1.2}$$

$$\sigma_t^2 = \frac{1}{S_t}\left(S_{t-1}(\sigma_{t-1}^2 + \mu t - 1^2) + \frac{x_t^2}{\sigma {x_t}^2} + 1\right) - \mu_t^2 \tag{1.3}$$

Where $\mu_t$ is the cell's new mean value, $\sigma_t^2$ 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.

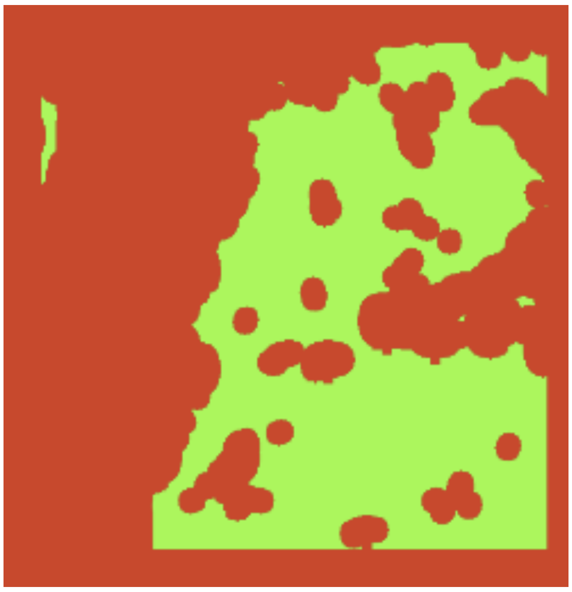

Figure 1.5: Binary Landing Site Map

### 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.

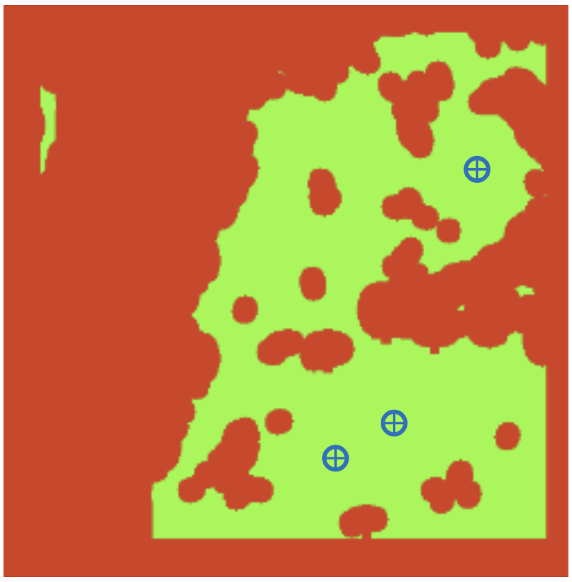

Figure 1.6: Binary Landing Site Map after Non Maximum Suppression

### LSD - Debug Images

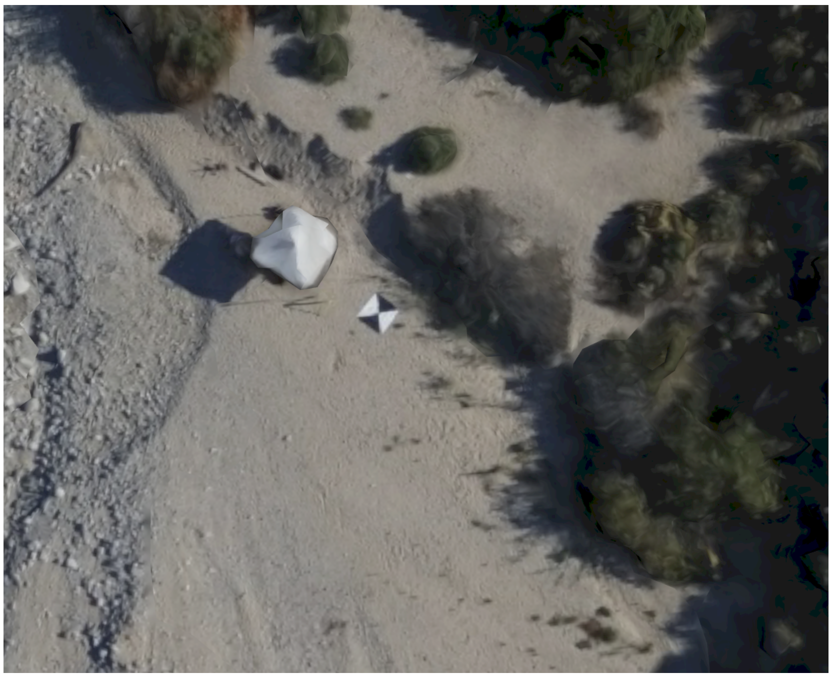
Figure 1.7: Gazebo Simulation Reference

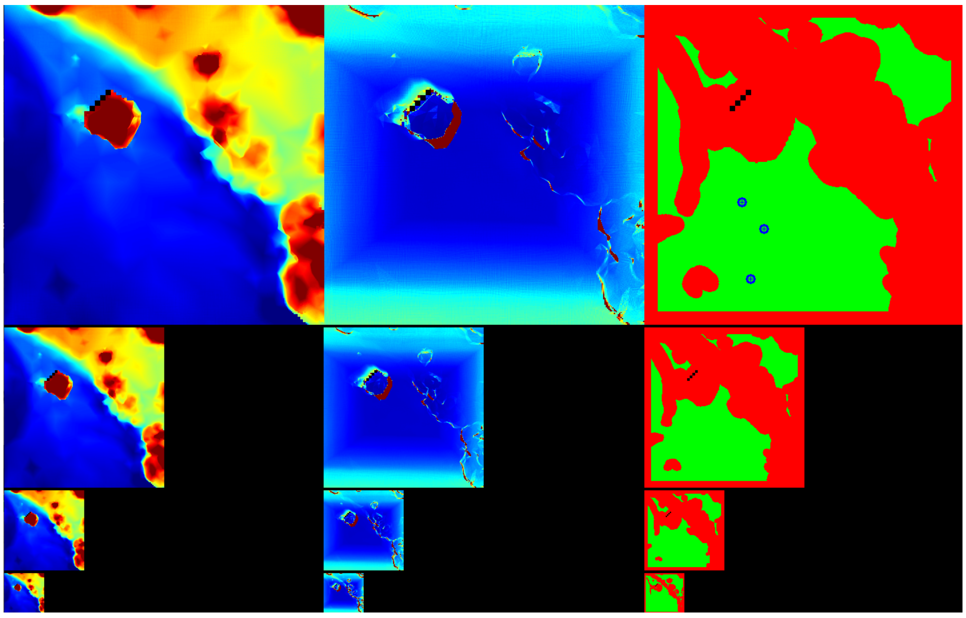
Figure 1.8: LSD Debug Image - Left: DEM, Middle: Uncertainties, Right: LS Map

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.

## 1.3 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. 1.1 shown connections it also communicates with a healthguard node keeping track of the systems health state and alerting in case of anomalies.

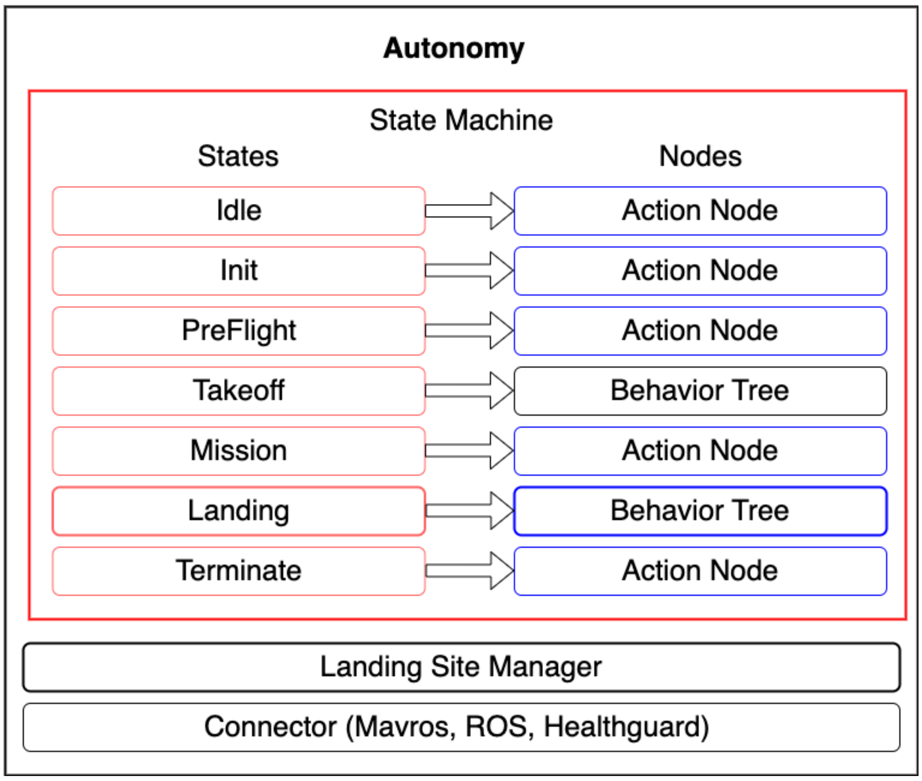

Figure 1.9: Simplified Structure of the Autonomy

The core of the autonomy is the state machine as depicted in fig. 1.9. 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.

### 1.3.1 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 2

# Methodology - Autonomous Landing Procedure

With the enhanced structure of the LSD output, 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 faced: Bringing the visual landing site pipeline together with the autonomous framework in order to achieve reliable autonomous landing in unknown terrain.
This implementation can be split into the following parts:

- Landing Site Handling
- Landing Site Heuristic
- Landing Behavior

## 2.1 Landing Site Handling

As shown in section 1.3 the autonomy is structured in a hierarchical and modular way. The current state of the state machine determines the specific task to be executed in that state's execution node. As the landing sites are constantly received alongside the mission tasks performed, they have to be processed in a separate thread. This is handled by a landing site manager (LSM) singleton class.

### 2.1.1 Landing Site Manager

Incoming landing sites are ranked according to a loss function and stores them in a heap buffer.

## 2.2 Landing Site Heuristic

As mentioned in **??** the autonomy receives landing sites with the following properties:

- Position
- Roughness
- Uncertainty
- Size

- Obstacle Altitude

From these properties the characteristics that comprise the final heuristic are:

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

### 2.2.1 Current Distance to Drone

Each iteration the current distance to the drone's position is calculate for each retained landing site. The distance is then normalized by dividing it by the cruise altitude which is 100m. Note that in practice landing sites fell off when farther away than 100m which yields a valid normalization.

### 2.2.2 Roughness

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

### 2.2.3 Uncertainty

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

### 2.2.4 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.

### 2.2.5 Verification Altitude

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.

## 2.3 Landing Behavior

The final landing behavior is implemented in the form of a behavior tree which allows adaptive decision making.
The autonomous framework(1.3)

### 2.3.1 Action Definition

For the finished behavior tree the following additional actions were defined. See section 1.3.1 for the actions defined prior to this work.

#### AscendToClearAltitude

#### ChangeAltitudeLSAction

#### GetLandingSiteAction

#### GetNextPatternCenterWPAction

#### LandingSiteSearchAction

#### LandingSiteVerificationAction

# Bibliography

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

[2] 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.

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

# Chapter 3

# Appendix: Simualation Depth Camera Misalignment

As described in **??** the implementation of the depth camera was wrong in the Gazebo Garden source code.
Hereafter are further comparative images.