ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

RSL 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

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.
Utilizing vision based approaches is also not a new idea. [**?** **?**] and [**?**] use artificial landing markers as indications of valid landing sites.
[**?** **?** **?**] and [**?**] 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 [**?** **?** **?**] and [**?**] show. As 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[**?**]. 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.
[**?**] use a similar approach as the one used by LORNA. Compared to LORNA's Landing Site Detection [**?** **?**] 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 [**?** **?**] and [**?**] use a 2.5D terrain representation similar to the setup used in this project.
Other novel approaches use learning based methods as did [**?** **?**] and [**?**]. 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.
Test [**?**] Test 2 [**?**]

Chapter 2

preparation - 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 Heuristic
- Conceptual Behavior
- Behavior Tree Implementation

2.1 Landing Site Heuristic

As mentioned in **??** the autonomy receives landing sites from LSD 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

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

$$L_{LS} = w_{dist}L_{dist} + w_{rough}L_{rough} + w_{var}L_{var} + w_{size}L_{size} + w_{verAlt}L_{verAlt} \quad (2.1)$$

2.1.1 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.

2.1.2 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.

2.1.3 Uncertainty - L_{var}

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

2.1.4 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. (2.1), 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.

2.1.5 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.

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

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.

2.2 Conceptual Behavior

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

2.2.1 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.

2.2.2 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(2.1) 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.

2.2.3 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,

2.3 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. (2.3), 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.

$$\Delta z = \frac{z^2}{f \cdot b} \Delta d \tag{2.2}$$

$$\Delta z = \frac{100m^2}{256 \cdot 15m} 0.5 = 1.302m \tag{2.3}$$

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(**??**).

2.3.1 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. (**??**) 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 2.3

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 **??** 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

$$z_{\text{clear}} = z_{\text{obst.}} + z_{\text{buffer}} + d_{\text{drone-LS}} \tag{2.4}$$

$$z_{\text{clear}} = \min\left(z_{\text{clear}}, z_{\text{failsafe}}\right) \tag{2.5}$$

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 2.1 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 **??**. Optimally however, this would be performed as long as the battery state permits it.

Bibliography

[1] M. Vukobratović and B. Borovac, "Zero-moment point — thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[2] M. Raibert, *Legged Robots That Balance*. Cambridge, MA: MIT Press, 1986.

[3] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1995, pp. 3137–3181.