A Fully Autonomous Indoor Quadrotor

Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard

Laboratory for Autonomous Intelligent Systems, University of Freiburg, Germany.

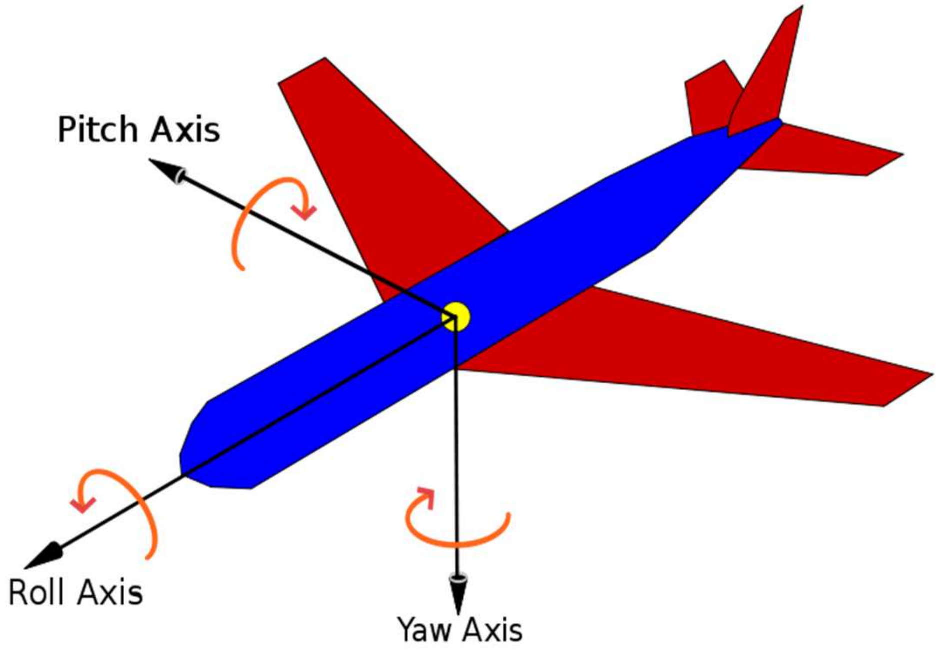
Presented by: Bhanuprasad Patibandla

ECGR 6185 Adv. Embedded Systems

February 20th 2013

Some Basic Terms

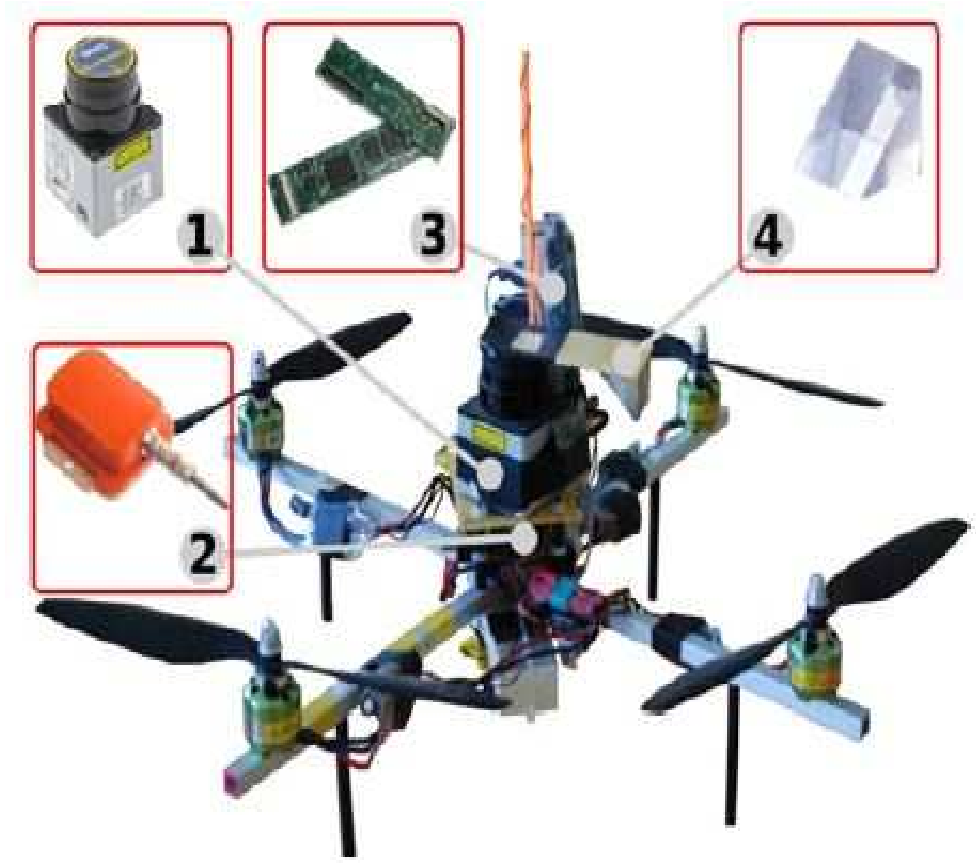
- Pitch
- Roll
- Yaw
- Degrees of Freedom



Hardware Architecture

Off the shelf Mikrocopter
Includes:

(1) Hokuyo laser range finder
(2) Xsens IMU
(3) Gumstix computer
(4) Laser mirror



Navigation System

- Based on a modular architecture - modules communicate via the network by using a publish–subscribe mechanism.

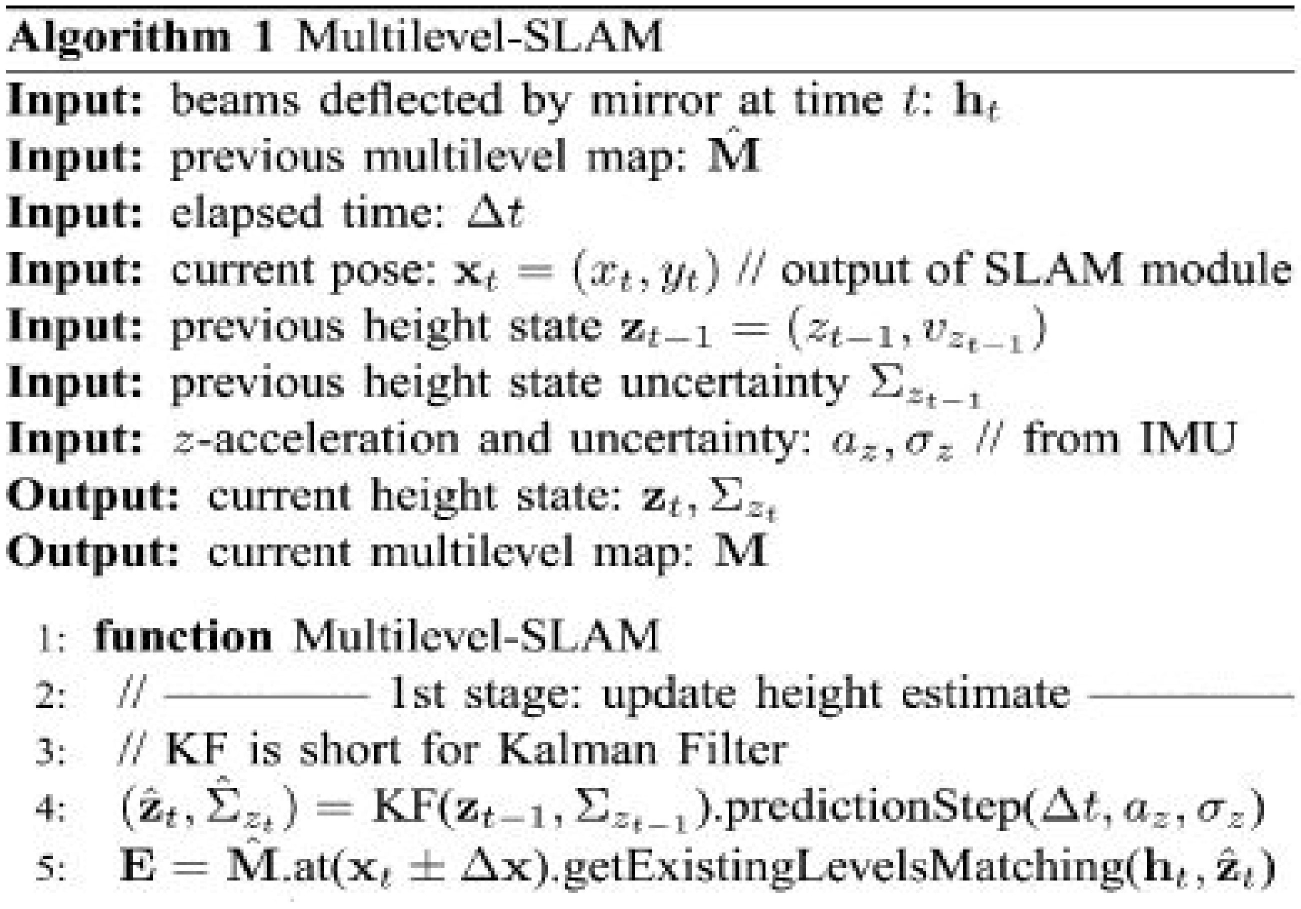
- Device Drivers on-board. Computational Algorithms on a remote PC communicating over wireless with the platform.

- 4 Degrees Of Freedom consisting of the 3-D position (x, y, z) and the yaw angle ψ.

- The only sensor that is used to estimate these 4DOF and detecting obstacles is the laser range scanner.

- No odometry measurements. Incremental movements in (x,y,ψ) are estimated by 2-D laser scan matching.

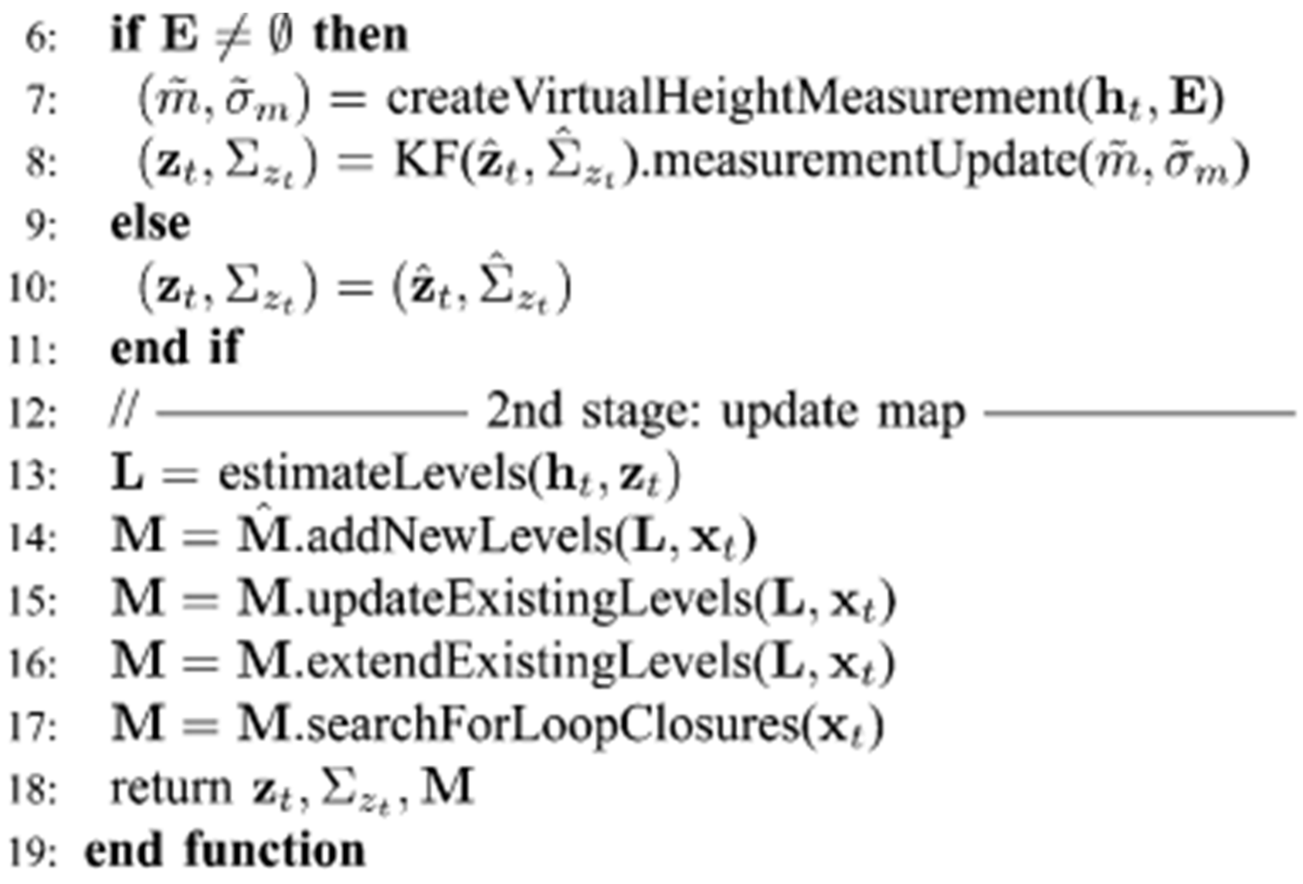
The SLAM Algorithm-1



Algorithm 1 Multilevel-SLAM

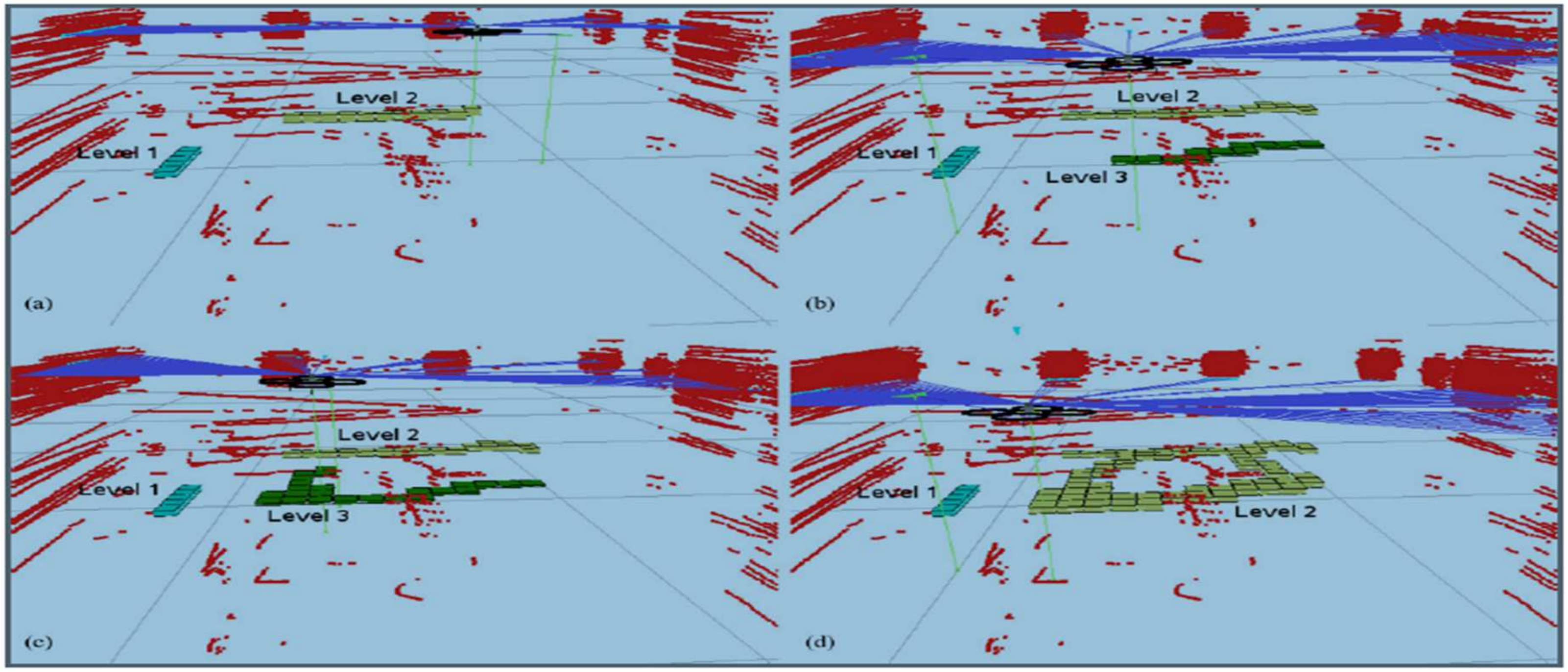
```
Input: beams deflected by mirror at time $t$: $\mathbf{h}_t$
Input: previous multilevel map: $\hat{\mathbf{M}}$
Input: elapsed time: $\Delta t$
Input: current pose: $\mathbf{x}_t = (x_t, y_t)$ // output of SLAM module
Input: previous height state $\mathbf{z}_{t-1} = (z_{t-1}, v_{z_{t-1}})$
Input: previous height state uncertainty $\Sigma_{z_{t-1}}$
Input: $z$-acceleration and uncertainty: $a_z, \sigma_z$ // from IMU
Output: current height state: $\mathbf{z}_t, \Sigma_{z_t}$
Output: current multilevel map: $\mathbf{M}$

1: function Multilevel-SLAM
2:   // ———— 1st stage: update height estimate ————
3:   // KF is short for Kalman Filter
4:   $(\hat{\mathbf{z}}_t, \hat{\Sigma}_{z_t}) = \mathrm{KF}(\mathbf{z}_{t-1}, \Sigma_{z_{t-1}})$.predictionStep($\Delta t, a_z, \sigma_z$)
5:   $\mathbf{E} = \hat{\mathbf{M}}$.at($\mathbf{x}_t \pm \Delta \mathbf{x}$).getExistingLevelsMatching($\mathbf{h}_t, \hat{\mathbf{z}}_t$)
```

The SLAM Algorithm-2



```
6:  if $\mathbf{E} \neq \emptyset$ then
7:    $(\tilde{m}, \tilde{\sigma}_m)$ = createVirtualHeightMeasurement($\mathbf{h}_t, \mathbf{E}$)
8:    $(\mathbf{z}_t, \Sigma_{z_t})$ = KF($\hat{\mathbf{z}}_t, \hat{\Sigma}_{z_t}$).measurementUpdate($\tilde{m}, \tilde{\sigma}_m$)
9:  else
10:   $(\mathbf{z}_t, \Sigma_{z_t}) = (\hat{\mathbf{z}}_t, \hat{\Sigma}_{z_t})$
11: end if
12: // ———————— 2nd stage: update map ————————
13: $\mathbf{L}$ = estimateLevels($\mathbf{h}_t, \mathbf{z}_t$)
14: $\mathrm{M} = \hat{\mathrm{M}}$.addNewLevels($\mathbf{L}, \mathbf{x}_t$)
15: $\mathrm{M} = \mathrm{M}$.updateExistingLevels($\mathbf{L}, \mathbf{x}_t$)
16: $\mathrm{M} = \mathrm{M}$.extendExistingLevels($\mathbf{L}, \mathbf{x}_t$)
17: $\mathrm{M} = \mathrm{M}$.searchForLoopClosures($\mathbf{x}_t$)
18: return $\mathbf{z}_t, \Sigma_{z_t}, \mathrm{M}$
19: end function
```

How it works



(a) Initially, it recognizes two levels (Level 1 and Level 2), corresponding to a chair and a table. (b) Subsequently, it flies away from the table, turns back, and flies over a different region of the same table. (c) This results in the creation of the new Level 3. Then, the robot keeps on hovering over the table until it approaches the extent of Level 2 that has the same elevation of Level 3, which is originated by the same table. This situation is shown in (c). Finally, the robot enters Level 2 from Level 3. (d) System recognizes these two levels to have the same elevation. Accordingly, it merges them and updates the common elevation estimate.

High-Level Control for Pose and Altitude

- The pitch and the roll are controlled by two independent PIDs that are fed with the *x* and the *y* coordinates of the robot pose.

$u\varphi = Kp \cdot (x - x*) + Ki \cdot ex + Kd \cdot vx$

- The yaw is controlled by the following controller:

$u\psi = Kp \cdot (\psi - \psi*)$

- The altitude is controlled by a PID controller that utilizes the current height estimate *z*, the velocity *vz* , and the current battery voltage *Ut* , respectively.

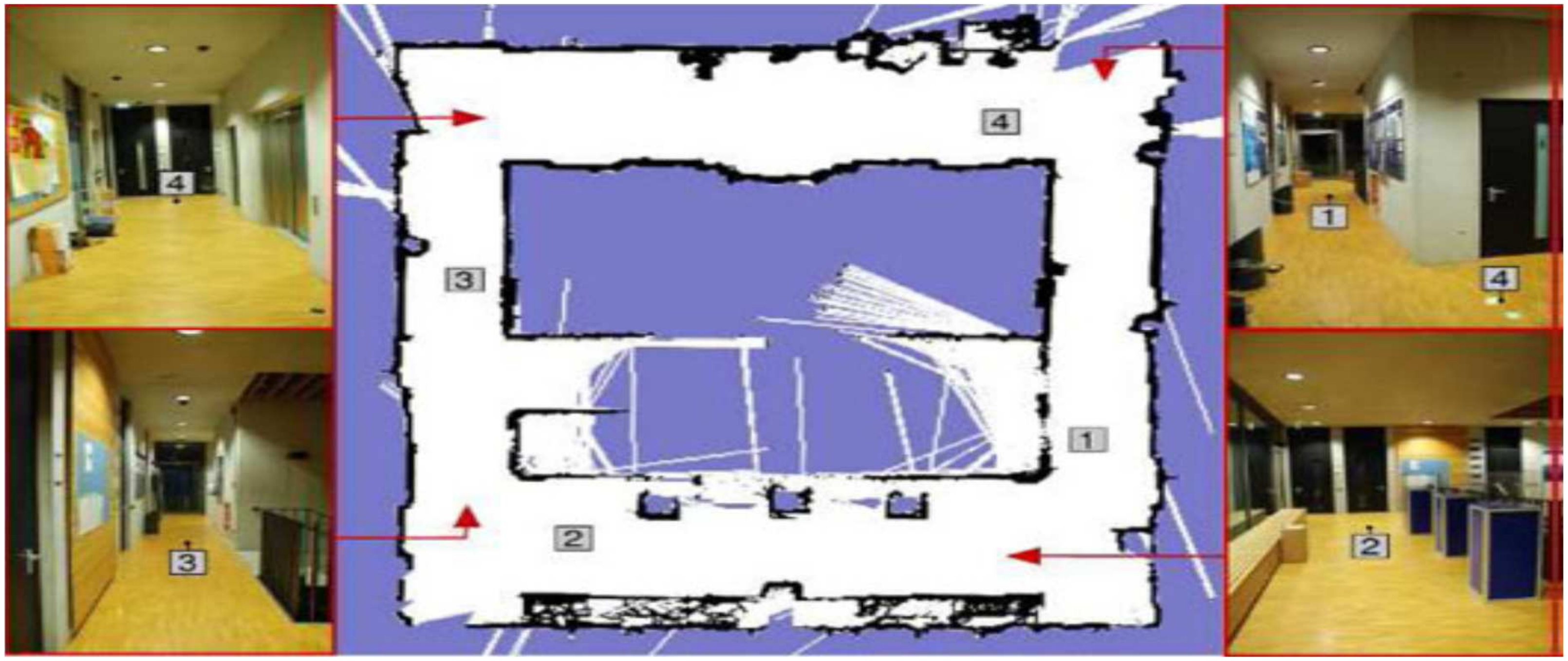
$uz = C(Ut) + Kp \cdot (z - z*) + Ki \cdot ez + Kd \cdot vz$

Path Planning and Obstacle Avoidance

- Path planning module is based on D-Lite algorithm which can reuse previous solutions to correct an invalid path and is done in two steps.
- A path in the x-y-z plane is computed but only action in x-y plane are considered. The known elevation of the surface below is used to determine a possible change in altitude the robot would have to take when moving to a nearby cell.
- Once this trajectory is calculated with D Lite its is augmented with the ψ component.

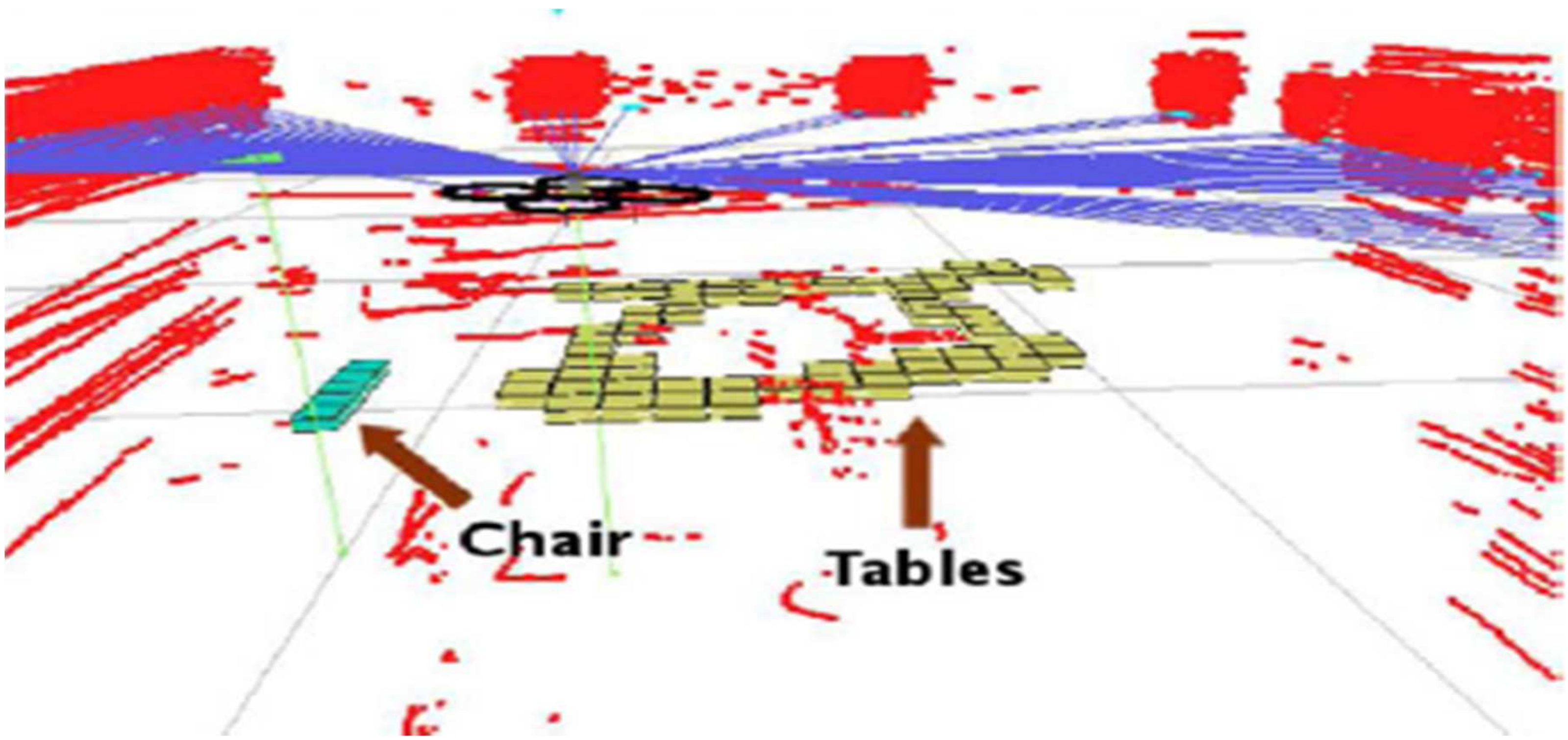
- ## Experimental Results

SLAM



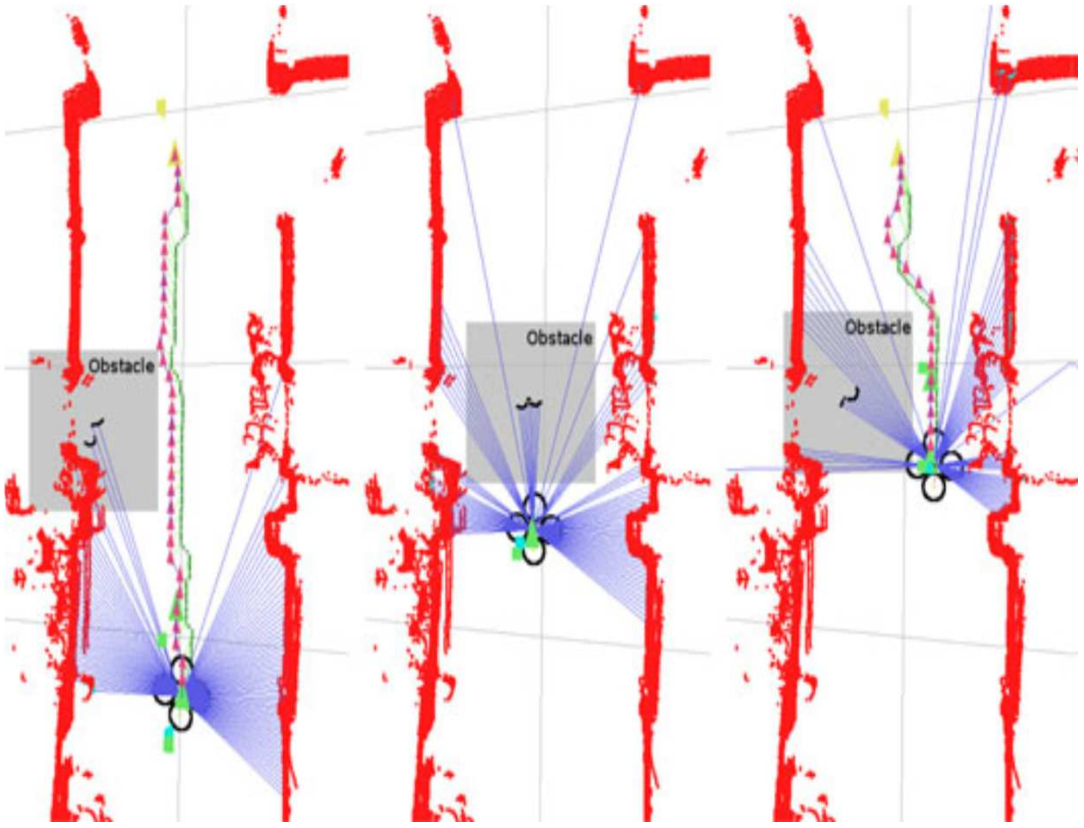
The mapping system was evaluated by letting the quadrotor fly four loops (approximately 41 m each) in a rectangle shaped building of approximate corridor size 10×12 m. The result of our SLAM algorithm is shown in Fig.

SLAM with Altitude Estimation



The chairs have a height of 48 cm and the tables are arranged next to each other having a height of 77 cm. The estimated heights of the chair and the tables were within 2.7cm and 2.8cm respectively

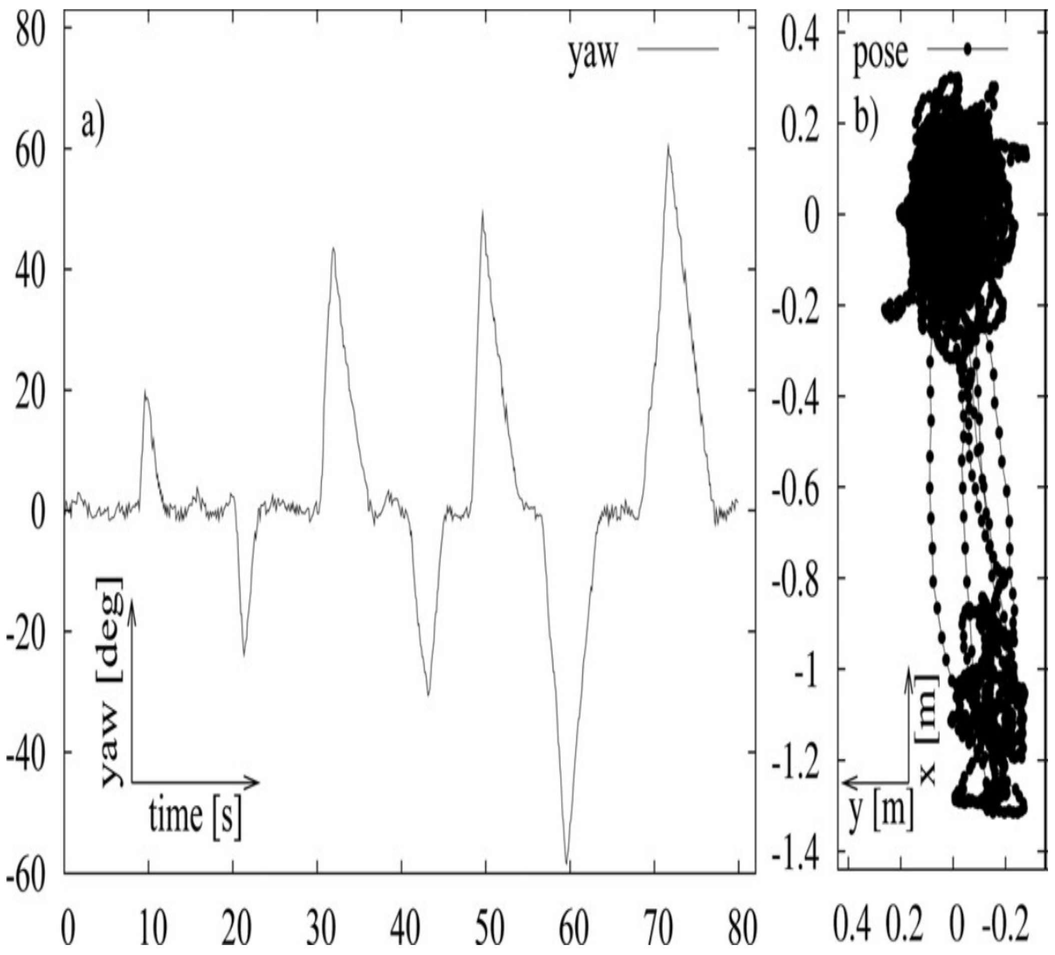
Path planning and Dynamic Obstacle Avoidance



The quadrotor is given a goal point 5 m in front of it.

- The planned path is shown in the left image.
- A person enters the corridor (shaded area) and blocks the robot's path, which results in an invalid plan. The quadrotor therefore hovers around the last valid way point (second image).
- In the third image, the person moved back, leaving the quadrotor enough space for a de-tour.

Pose Control



- During the yaw stabilization experiment, the quadrotor was required to rotate to 0◦, while the user manually turned the robot once in a while to a random orientation.
- Within the pose stability experiment, the quadrotor was set to hover at (0, 0) but was manually moved backwards once in a while and required to fly autonomously back to the initial pose.

Conclusion

- A complete navigation solution that approaches different aspects of localization, mapping, path-planning, height estimation, and control has been achieved.
- Since the system does not rely on characteristics of the flying platform like the dynamics model, it can be easily adapted to different flying vehicles.
- The whole system can run on-board with a more advanced processor like Intel Atom processors.
- Future plans include integrating a time of flight camera into the system.

References

[1] Aircraft Principal Axes
[Web Photo] http://en.wikipedia.org/wiki/Aircraft_principal_axes

[2] Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard, "A Fully Autonomous Indoor Quadrotor" in IEEE Transactions on robotics, vol. 28, No. 1, February 2012