Risk-Based Obstacle Avoidance in Unknown Environments using Scenario-Based Predictive Control for an Inspection Drone Equipped with Range Finding Sensors

Sverre Velten Rothmund* and Tor Arne Johansen*

Abstract— **This paper develops an obstacle avoidance strategy for inspection drones equipped with simple range finding sensors, such as radar or sonar. The obstacle avoidance strategy uses scenario-based model predictive control where the predicted outcomes of a set of possible control actions are evaluated. The action with the best predicted outcome amongst the safe options is chosen. The resulting behaviour is deemed safe if the probability of collision at each time-step in the prediction is lower than a given maximal accepted probability. The probability of collision is calculated by combining a probability density function of the position of the drone with an obstacle probability map generated by the range finding sensors. This constraint is checked at each step over the prediction horizon thus ensuring that the control action will give rise to safe behaviour. The algorithm is implemented in a 2D case and tested with a simple model for the range finding sensors. Simulations show that the drone is able to avoid obstacles and that the drone will change speed or take detours to avoid flying in potentially dangerous areas to mitigate risk. The algorithm is designed for avoiding obstacles along a pre-planned path. The pre-planned path is assumed to be generally good, but might be unsafe or not take some unknown obstacles into account. If the pre-planned path goes through a larger convex area or does not take a large obstacle into account, then this algorithm might not find a way around the obstacle and the drone will stop at a safe distance. The path of the drone must then be re-planned taking these obstacles into account. The resulting obstacle avoidance strategy guarantees safe behaviour which enables higher level controllers or human planners to plan a path based on lacking obstacle information without taking the safety of the drone into consideration.**

I. INTRODUCTION

A. Background and motivation

When moving close to static obstacles, such as for industrial inspection, including the uncertainties in the drone and obstacle positions are essential to avoid collisions. One approach to incorporate uncertainty in the position of the drone is to assume bounded noise and then to ensure that all possible positions where the drone can end up will not be in an obstacle. This is done for the linear case with linear constraints in [1] and for a nonlinear case with a predefined set of manoeuvres in [2]. Another example of bounding is in [3] where the positional variance at each time-step is calculated and a constraint is introduced that requires that obstacles are k standard deviations away from the drone. Bounding the accepted uncertainty or accepted positional offsets makes sure that the probability of the drone colliding is smaller than the probability that the bounds were wrong. This method is conservative, which might be good when the goal is to get to the goal position without colliding with something along the way. But when the goal is to fly close to something for inspection, such conservative bounds might prevent the drone from getting as close to the object as is desired. Another limitation with bounding is that it does not give an obvious answer on what should be done when no action, including aborting the mission, is feasible.

A less conservative approach would be to calculate the probability of collision and then bound this probability. This probability could be set based on maximizing the revenue taking into account the value of the mission and the expected loss if a collision occurred. When no action fulfils the required probability for success, the action that minimizes the probability of collision can be chosen. A method for model predictive control using an upper bound for the collision probability with linearly constrained obstacles is developed in [4]. This work assumes that the position and shape of the obstacles are known.

When the environment is not fully known, uncertainties in the environment must be considered to give a reasonable probability estimate of collision. One approach to describe obstacle uncertainty is to use occupancy grid maps. Each cell in these maps contains information on how likely it is that the current cell contains an obstacle. Mapping with range finding sensors using occupancy grid maps was first introduced by [5]. This work lacked a computationally feasible method for updating the map for non-ideal sensor models. A linear time method that solved this problem was developed in [6].

Different methods for utilizing uncertainty in obstacle information with potential fields is shown in [7] and [8]. [7] utilizes potential fields to push the vehicle further away from objects that have a higher certainty of existence, while [8] pushes the vehicle further away from objects where the positional uncertainty is larger. These works incorporate different aspects of uncertain environment information but do not incorporate the uncertainty in the position of the vehicle itself, and do not consider risk in a probabilistic sense.

B. Contribution

The main contribution of this paper is to develop a risk-based framework for obstacle avoidance in unknown environments that incorporates both uncertainty in the environment and the vehicle's position. The algorithm developed

*Center for Autonomous Marine Operations and Systems (AMOS), Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway

Coresponding author: sverre.v.rothmund@ntnu.no

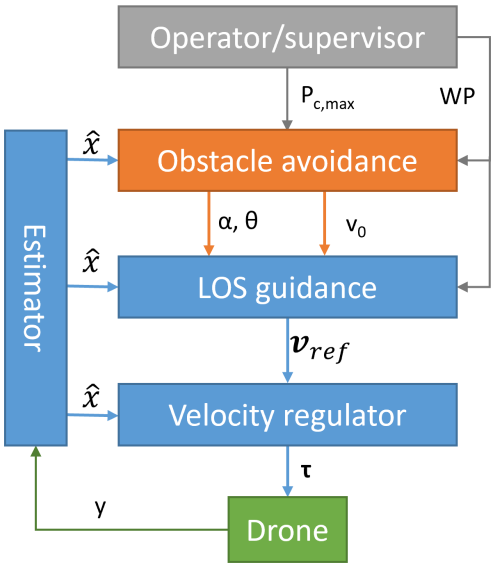


Fig. 1: Control hierarchy. $P_{c,max}$ is the maximal accepted probability of collision at each time-step. WP are the waypoints marking the planned path. α and θ are angle offsets relative the nominal direction of motion. v_0 is the reference speed, $\mathbf{v}_{ref}$ the reference velocity, $\boldsymbol{\tau}$ are motor torques and forces, $\mathbf{y}$ are measurements, and $\hat{\mathbf{x}}$ is the estimated state.

in [9] for collision avoidance between ships is adapted to the inspection drone case and modified to use probabilistic uncertainty models that utilizes uncertainties in the drone state and the environment to calculate the probability of collision. The collision avoidance strategy is generalized to 3D. To achieve this a 3D line of sight guidance strategy is proposed.

II. Overview

The task is to make an obstacle avoidance strategy for execution of industrial inspection missions with an explicit awareness of probability of collision with obstacles. The task of the drone is to collect data while following the straight lines between pre-planned waypoints. There might be unexpected obstacles blocking the planned path of the drone which forces the drone to deviate from the planned path to avoid collision. The drone is equipped with multiple wide-angle range finding sensors that give limited information about the obstacles. Examples of wide-angle range finding sensors are radar and sonar. The large field of view of the sensor and the position uncertainty of the drone at the time of sensing gives an uncertainty in the position and shape of detected obstacles.

The proposed control hierarchy is shown in figure 1. The drone is controlled by a velocity controller that ensures that the drone moves in the designated direction at the designated speed. The velocity reference vector needed by the velocity controller is supplied by a line of sight (LOS) guidance law. This guidance law uses the list of waypoints to calculate the velocity reference that gently moves the drone towards and along the planned path.

The obstacle avoidance algorithm uses the scenario based MPC formulation presented in [9] with a probabilistic uncertainty model that calculates the probability of collision over the prediction horizon. This method utilizes the LOS guidance to parameterize different paths using only one parameter in 2D and two parameters in 3D. This method has a limited set of possible control actions making it less complete than optimal control methods with full control over the drone's behaviour, such as in [3]. The major advantage of the method is that the run-time is linear with respect to the number of possible combinations of control-actions, and is easily parallelizable which makes it much faster than full optimal control solutions on multi-core processors. This makes the method feasible for real-time applications. The method might be slower than potential field methods, but it avoids some of the inherent problems with potential field such as unstable motion and getting stuck in potential minima close to narrowly spaced obstacles [10]. The proposed method also opens up for working with the probability of collision, which potential fields do not.

This obstacle avoidance algorithm gives a constant offset angle and reference speed to the LOS-guidance algorithm. The offset angle makes the drone gradually move away from the planned path specified by the waypoints. As the drone moves further away, the LOS guidance vector will point more directly towards the path, counteracting the offset. For angles under 90° the drone will converge towards a line parallel to the planned path. This behaviour makes it possible to give a constant angle offset and still move in the along path direction while executing an evasive manoeuvre. When the angle offset is set back to zero the LOS guidance law will automatically make the drone move back to the planned path made by the waypoints.

A finite set of angle offsets and velocity offsets are defined. A model of the drone system with LOS-guidance and a velocity controller is simulated over a prediction horizon with all the different combinations of angle and velocity offsets. The control action is applied at the initial time-step and kept constant over the entire prediction horizon. The probability of collision with the resulting behaviour is checked against the maximal accepted probability of collision at each time-step. The control action that maximizes the overall mission objective amongst the safe enough options is then chosen.

The probability of collision with the resulting behaviour is calculated by combining a probability map over obstacle positions with a probability density function over the position of the drone. The probability map over obstacles is made online based on the range measurements from a radar, sonar, or lidar. The drone's position is not exactly known at the current time-step due to uncertainties in the sensors used for estimation. When predicting into the future, the position of the drone gets less certain over time as some unknown disturbance might affect it. The drone will have controllers that will counteract these errors, but these controllers have dynamics which makes them unable to instantly counteract disturbances.

III. Drone and control model

A. Open loop model

The drone is for simplicity assumed to be a fully actuated double integrator, driven by an acceleration caused by the control input $\mathbf{u}$, and affected by an additive disturbance $\mathbf{w}_c$. The state is written on the form $\mathbf{x} = [\mathbf{x}_p^\top, \dot{\mathbf{x}}_p^\top]$, where $\mathbf{x}_p$ is the position of the drone decomposed in a North East Down (NED) coordinate frame.

$$\dot{\mathbf{x}} = A_c\mathbf{x} + B_c\mathbf{u} + \mathbf{w}_c \tag{1}$$

$$A_c = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad B_c = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}, \quad E_c = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \tag{2}$$

As this model is linear, an exact discretization for A_c and B_c can be found, these are denoted as A and B. The discrete time white noise process $\mathbf{w}$ is assumed to be Gaussian with a covariance matrix denoted as Q. The notation $\mathbf{x}[k] = \mathbf{x}(k\,dt)$ is used, where dt is the discretization time step. To simplify notation the time-step index is only included in the state update equations.

$$\mathbf{x}[k+1] = A\mathbf{x}[k] + B\mathbf{u}[k] + \mathbf{w}[k] \tag{3}$$

B. Velocity controller

The drone is equipped with a velocity controller.

$$\mathbf{u}[k] = -K(\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \hat{\mathbf{x}}[k] - \mathbf{v}_{ref}[k]) \tag{4}$$

$$\hat{\mathbf{x}}[k] = \mathbf{x}[k] + \mathbf{v}[k] \tag{5}$$

Where K is a gain matrix and $\mathbf{v}_{ref}$ is the reference velocity vector. This controller uses the estimated state, $\hat{\mathbf{x}}$, which is modelled as the true state, $\mathbf{x}$, plus some estimation error, $\mathbf{v}$, that is assumed to be a discrete Gaussian white noise process with zero mean and covariance matrix R.

The closed-loop dynamics is then given as follows

$$\mathbf{x}[k+1] = A_{cl}\mathbf{x}[k] + B_{cl}\mathbf{v}_{ref}[k] - \Gamma_{cl}\mathbf{v}[k] + \mathbf{w}[k] \tag{6}$$

$$A_{cl} = A - BK\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad B_{cl} = BK, \quad \Gamma_{cl} = BK\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{7}$$

C. Line of Sight guidance

A 3D line-of-sight guidance strategy is needed. Four different options are compared in [11]. These methods produce reference yaw and pitch angles. A method that instead produces a reference velocity vector is presented here. This method avoids trigonometric functions which simplifies linearization of the resulting dynamics.

To formulate the LOS-guidance law in 3D an addition coordinate system, called the LOS coordinate system, is defined. This coordinate system is defined as having the x-axis along the line between the previous and the next waypoint, denoted as WP_1 and WP_2. The y- and z-axis can be arbitrarily chosen as long as the LOS coordinate system is a right-hand coordinate system. $\mathbf{y}_{LOS}$ is chosen to be the cross product between $\mathbf{x}_{LOS}$ and the z-axis in the NED frame.

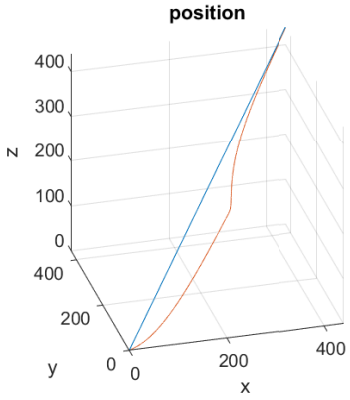


Fig. 2: LOS guidance in 3D. An offset $\alpha = 40^\circ$ in direction $\theta = 160^\circ$ is applied the first half of the simulation, then the offset is turned off and the drone returns to the path.

$$\mathbf{x}_{LOS} = \frac{WP_2 - WP_1}{||WP_2 - WP_1||} \tag{8}$$

$$\mathbf{y}_{LOS} = \frac{\mathbf{x}_{LOS} \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top}{||\mathbf{x}_{LOS} \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top||} \tag{9}$$

$$\mathbf{z}_{LOS} = \mathbf{x}_{LOS} \times \mathbf{y}_{LOS} \tag{10}$$

The position of WP_1 and WP_2 as well as the vectors $\mathbf{x}_{LOS}$, $\mathbf{y}_{LOS}$, and $\mathbf{z}_{LOS}$ are given in the NED frame.

For the special case where $\mathbf{x}_{LOS} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$, where the cross product in (9) is undefined, the alternative formulation is used.

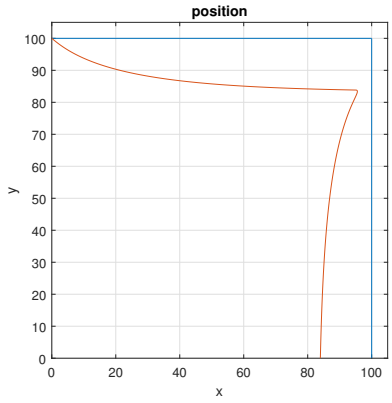
$$\mathbf{z}_{LOS} = \frac{\mathbf{x}_{LOS} \times \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top}{||\mathbf{x}_{LOS} \times \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top||} \tag{11}$$

$$\mathbf{y}_{LOS} = \mathbf{z}_{LOS} \times \mathbf{x}_{LOS} \tag{12}$$

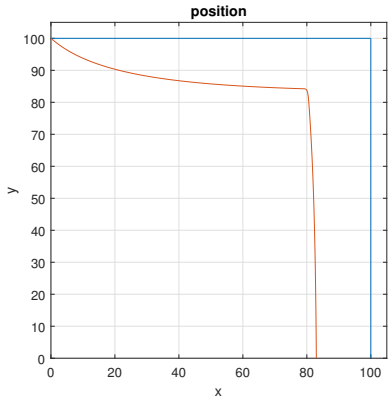
This basis can be used to find the rotational matrix between NED and LOS.

$$R_{LOS}^{NED} = \begin{bmatrix} \mathbf{x}_{LOS} & \mathbf{y}_{LOS} & \mathbf{z}_{LOS} \end{bmatrix} \tag{13}$$

The difference between the drone's position and WP_1 given in LOS frame gives the drone's offset from the path between the two waypoints. The x coordinate is the distance along the path, while the y and z coordinates give the offset across the path. LOS guidance makes the drone at all times follow the vector pointing from its current position to a point Δ ahead on the planned path. In the LOS frame this vector has coordinate Δ in x_{LOS} direction, and the y and z components of the distance from the drone to WP_1 in the y_{LOS} and z_{LOS} direction. By normalizing this vector and multiplying it with the desired speed, v_0, the reference speed



(a) Unwanted behaviour when switching waypoints based on along path distance.



(b) Correct behaviour when switching based on relative distance to the two lines.

Fig. 3: Behaviour with different strategies for changing waypoints. Orange shows the drone position with a control action offset, blue marks the straight line path between the waypoints.

vector that the drone should follow is generated:

$$\chi_{LOS}^{NED} = R_{LOS}^{NED}\left(\begin{bmatrix}\Delta\\0\\0\end{bmatrix} - \begin{bmatrix}0&0&0\\0&1&0\\0&0&1\end{bmatrix} R_{NED}^{LOS}(\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\hat{\mathbf{x}} - WP_1)\right) \tag{14}$$

$$\mathbf{v}_{ref} = v_0 \frac{\chi_{LOS}^{NED}}{||\chi_{LOS}^{NED}||} \tag{15}$$

The estimated drone state, $\hat{\mathbf{x}}$, is given in the NED frame. Note that the line of sight guidance system makes decisions based on the current best estimate of the state, $\hat{\mathbf{x}}$, not the actual state $\mathbf{x}$.

The obstacle avoidance controller introduces an offset angle to the velocity vector. In the 2D case developed in [9], an angle α is is added to the LOS angle. When seen as a vector, this is the same as rotating the vector by α about an z-axis pointing out of the plane. For the 3D case, two parameters are needed, α and θ. α is used for rotating the vector around some axis orthogonal to the $\mathbf{x}_{LOS}$ axis. The angle θ tells us the orientation of this axis relative to the $\mathbf{y}_{LOS}$ axis. This is done using the rotation matrix shown in equation (17).

$$\chi_{LOS,ca}^{NED} = \tag{16}$$
$$R_{LOS}^{NED} R_{ca}\left(\begin{bmatrix}\Delta\\0\\0\end{bmatrix} - \begin{bmatrix}0&0&0\\0&1&0\\0&0&1\end{bmatrix} R_{NED}^{LOS}(\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\hat{\mathbf{x}} - WP_1)\right)$$

$$R_{ca} = R_{x=-\theta} R_{y=\alpha} R_{x=\theta} \tag{17}$$

$$\mathbf{v}_{ref,ca} = v_0 \frac{\chi_{LOS,ca}^{NED}}{||\chi_{LOS,ca}^{NED}||} \tag{18}$$

The resulting behaviour with a constant offset angle is shown in Figure 2. This figure also shows that the drone gently moves back to the path when α is set to zero. How quick the drone should move towards and away from the path is specified by Δ.

Special 2D case: The vector-based 3D line of sight formulation can easily be used in 2D as well but requires some special notation as the cross product and rotational matrices are not defined for 2D.

$$\mathbf{x}_{LOS} = \frac{WP_2 - WP_1}{||WP_2 - WP_1||} \tag{19}$$

$$\mathbf{y}_{LOS} = \begin{bmatrix}0&-1\\1&0\end{bmatrix} x_{LOS} \tag{20}$$

$$R_{LOS}^{NED} = \begin{bmatrix}\mathbf{x}_{LOS} & \mathbf{y}_{LOS}\end{bmatrix} \tag{21}$$

$$\chi_{LOS,ca}^{NED} = R_{LOS}^{NED} R_{ca}\left(\begin{bmatrix}\Delta\\0\end{bmatrix} - \begin{bmatrix}0&0\\0&1\end{bmatrix} R_{NED}^{LOS}(\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\hat{\mathbf{x}} - WP_1)\right) \tag{22}$$

$$R_{ca} = \begin{bmatrix}\cos(\alpha) & -\sin\alpha\\ \sin(\alpha) & \cos(\alpha)\end{bmatrix} \tag{23}$$

Switching between waypoints: Two common ways of switching waypoints in line-of-sight guidance are presented in [12]. One method is to change waypoint when the vehicle is within a given radius of the waypoint (circle of acceptance). As the anti-collision control actions might take us far away from the waypoints, this might lead to the waypoint being missed. The other strategy is to change waypoint when the along path distance to the next waypoint is small enough. This strategy works well when the goal is to follow the desired path closely but leads to unwanted behaviour when there is a wanted offset due to the control action made by the obstacle avoidance system. The drone might then switch waypoint closer to the next path segment than the designed offset. This is shown in figure 3a.

This can be solved by switching waypoints when the drone is closer to the next path segment than to the current path segment. This will avoid making the drone move closer and then further away from the path segment. A margin can be implemented to compensate for the drone dynamics, making the drone switch waypoint a bit before its equally close to

both path segments. The distance to the path segment should be the closest distance to any point on the infinite line going through the waypoints. The shortest distance a point $\mathbf{x}$ is away from the infinite line going through points $\mathbf{a}$ and $\mathbf{b}$ can be calculated as

$$s(\mathbf{a},\mathbf{b},\mathbf{x}) = (\mathbf{x}-\mathbf{a})^\top \frac{(\mathbf{b}-\mathbf{a})}{||\mathbf{b}-\mathbf{a}||} \tag{24}$$

The behaviour of this waypoint switching algorithm is shown in Figure 3b.

IV. Heading dynamics

For the fully actuated double integrator drone model, the heading does not affect the position and velocity dynamics, as the drone is able to fly in any direction independent of the heading. But as both the payload sensors (e.g. camera) and range finding sensors may be predominantly placed in one direction, the drone may have to turn the sensors towards the movement direction to be able to detect obstacles in its way. A simple heading dynamic is implemented in the simulator to include this behaviour.

$$\psi[k+1] = \gamma\psi[k] + (1-\gamma)\psi_{ref}[k] \tag{25}$$
$$\psi_{ref} = \text{atan2}(V_{ref,ca,x}, V_{ref,ca,y}); \tag{26}$$

Where ψ denotes the heading. The parameter $\gamma \in <0,1>$ decides how quick the heading dynamics will be.

V. Covariance propagation

A. Covaraince formulation

The LOS guidance law is nonlinear due to the normalization of the $\chi^{NED}_{los,ca}$ vector in equation (18). Nonlinearities will distort a Gaussian probability distribution making it difficult to propagate the covariance. The system is linearized to avoid this problem.

First, the guidance law (18) is re-written.

$$\chi^{NED}_{LOS,ca} = E - F\hat{\mathbf{x}}_p \tag{27}$$
$$= E - F\mathbf{x}_p - F\mathbf{v}_p \tag{28}$$
$$E = R^{NED}_{LOS} R_{ca} \left(\begin{bmatrix} \Delta \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} R^{LOS}_{NED} W P_1 \right) \tag{29}$$
$$F = R^{NED}_{LOS} R_{ca} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} R^{LOS}_{NED} \tag{30}$$
$$\mathbf{x}_p = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}, \quad \hat{\mathbf{x}}_p = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \hat{\mathbf{x}}, \quad \mathbf{v}_p = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{v} \tag{31}$$

Both $\mathbf{x}$ and $\mathbf{v}$ are stochastic variables where $\mathbf{x}$ is the state and $\mathbf{v}$ describes the uncertainty due to measurement errors. Inserting (28) into (18) yields

$$\mathbf{v}_{ref,ca} = v_0 \frac{E - F\mathbf{x}_p - F\mathbf{v}_p}{||E - F\mathbf{x}_p - F\mathbf{v}_p||} \tag{32}$$

This equation is linearized around the estimated expected position evaluated at time-step k denoted as $\mathbf{x}_k$. This state should be propagated through the closed loop state space equations (6) using the nonlinear LOS guidance equations (18) for the velocity reference vector. The linearization is done around $\mathbf{v} = 0$ as $\mathbf{v}$ is assumed to have zero mean. The linearization results in the following equations

$$\bar{\mathbf{v}}_{ref,ca} = v_0(\bar{E} - \bar{F}\mathbf{x} - \bar{F}\mathbf{v}) \tag{33}$$
$$\bar{E} = G + H \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}_k \tag{34}$$
$$\bar{F} = H \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \tag{35}$$
$$G = \frac{E - F \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}_k}{||E - F \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}_k||} \tag{36}$$
$$H = \frac{F}{||E - F \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}_k||} - \left(\frac{(E - F \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}_k)(E - F \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}_k)^\top F}{||E - F \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}_k||^3} \right) \tag{37}$$

Inserting the linearized velocity reference vector into the state equation (6) leads to

$$\mathbf{x}[k+1] = A_{LOS}\mathbf{x}[k] + \Gamma_{LOS}\mathbf{v}[k] + \mathbf{w}[k] + C_{LOS} \tag{38}$$
$$A_{LOS} = A_{cl} - B_{cl}v_0\bar{F}, \quad \Gamma_{LOS} = -\Gamma_{cl} - B_{cl}v_0\bar{F} \tag{39}$$
$$C_{LOS} = B_{cl}v_0\bar{E} \tag{40}$$

We now have a linear state space formulation. With the assumption that $\mathbf{v}[k]$ and $\mathbf{w}[k]$ are independent white noise processes, all the input terms in (38) are uncorrelated and the covariance matrix of $\mathbf{x}[k+1]$ can be calculated as

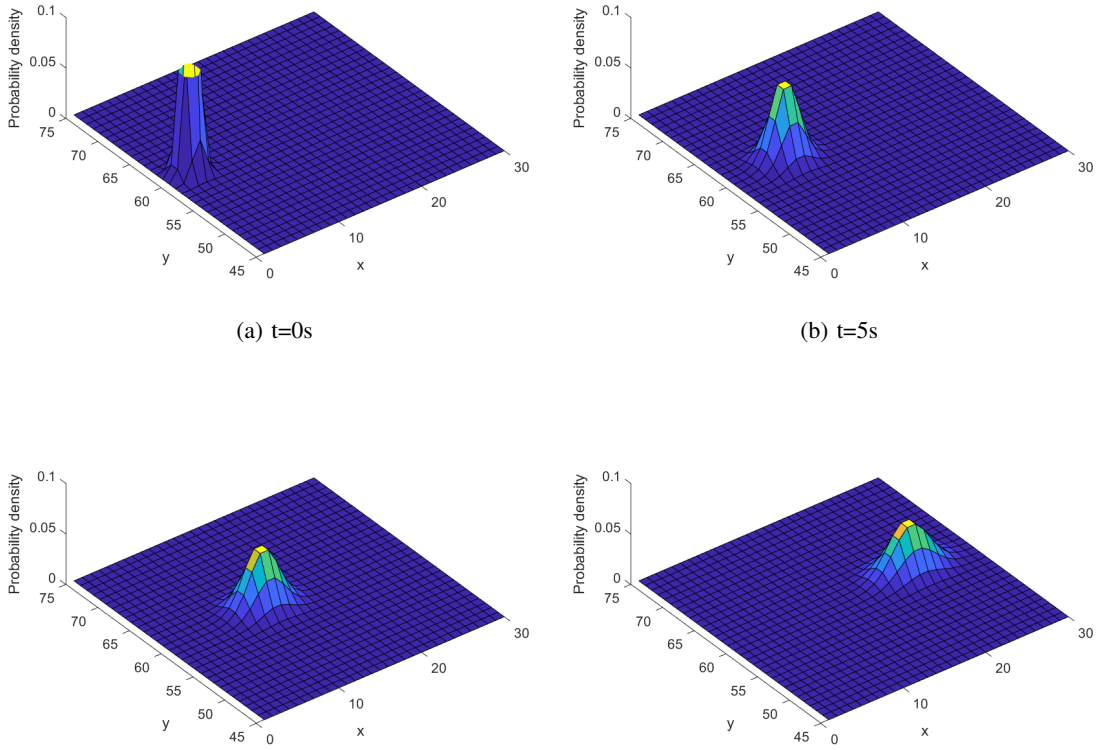
$$\text{var}(\mathbf{x}[k+1]) = A_{LOS}\text{var}(\mathbf{x}[k])A^\top_{LOS} + \Gamma\text{var}(\mathbf{v}[k])\Gamma^\top + \text{var}(\mathbf{w}[k]) \tag{41}$$
$$\text{var}(\mathbf{x}[k+1]) = A_{LOS}\text{var}(\mathbf{x}[k])A^\top_{LOS} + \Gamma R\Gamma^\top + Q \tag{42}$$

The initial variance is equal to the state estimator variance, R.

$$\text{var}(\mathbf{x}[0]) = R \tag{43}$$

B. Resulting covariance dynamics

Figure 4 shows how the uncertainty in position varies over time in the prediction. The figure shows that the uncertainty in predicted position will start low and then gradually increase. It is interesting to note that the probability density function becomes elongated over time, having a larger uncertainty in the along path direction than in the across path direction. This is a direct consequence of LOS guidance only counteracting across path error, making it asymptotically stable in across path direction but only marginally stable in the along path direction.



(a) t=0s

(b) t=5s

(c) t=10s

(d) t=20s

Fig. 4: The probability density function of the drone's predicted position at different time steps into the future.

VI. Mapping

To be able to avoid obstacles, a map has to be made based on the data from the range finding sensor. This work assumes that one or more body fixed sonar or radar is used. A laser range finder would be a special case where the field of view of the range finding sensor is just one line, and the different measurements in a scan can be incorporated as separate sensors with different headings. The sensor is assumed to return the shortest distance to any object within a cone with the width equal to the field of view of the sensor. A radar would return multiple reflections, but only the first is used as it is uncertain whether later reflections are caused by the radar-wave leaving or entering into a new material. The exact location of the object inside the field of view is unknown, only the distance from the sensor and the fact that it is inside the field of view is known. This model is quite simplistic and does not incorporate specular reflections or multipath. Specular reflection is when the entirety of the emitted signal is reflected away from the sensor, which will not give a distance measurement. Multipath is when the signal is bounced off multiple surfaces before it reaches the sensor, which makes the measured distance longer than the true distance to the target. One method for handling specular reflections is presented in [13]. There will be uncertainties in the position of a measured obstacle as there will be uncertainties in the range measurement and in the position of the drone at the time of the measurement. The uncertainties are assumed to be Gaussian. The variance in position in the direction of the measurement is added together with the variance of the sensor output to give the measurement uncertainty, σ^2.

The occupancy grid map concept developed by [5] bases itself on assuming that all the cells are independent and then uses Bayesian interference to updated the map. The probability of a cell c_i being occupied given the measured range r can be calculated as

$$P(c_i|r) = \frac{p(r|c_i)P(c_i)}{p(r)} \tag{44}$$

Where $P(c_i)$ is the a-priori map value and $p(r|c_i)$ is the inverse sensor model. The inverse sensor model can be calculated by taking the weighted sum over all possible map-states where cell c_i is occupied, weighted by how likely that map-state is, given the a-priori information. For each possible map-state, the probability of getting the measurement r has to be considered. A map-state, M, is one realization of the map where each cell is either occupied or empty. The relevant information in each map-state is the location of

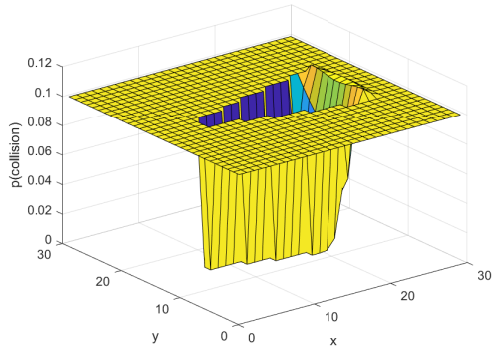


Fig. 5: Occupancy grid map updated with one measurement.

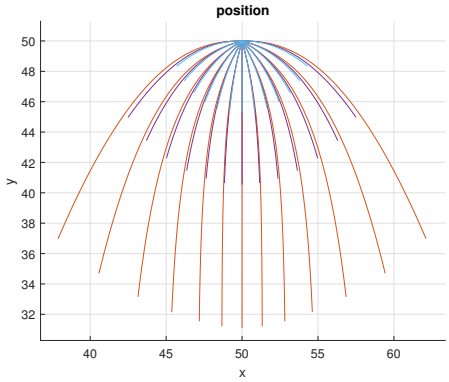


Fig. 6: All candidate paths from a stationary initial position. The planned path goes straight down.

the closest occupied cell as this cell is assumed to have given the measurement. $p(r|M)$ would then be the probability density value of getting the measurement r from the closest occupied cell in map-state M. By assuming a Gaussian sensor model, this value could be found by setting the expected value of the sensor model at the start of the first occupied cell in M and then finding the probability density value of getting measurement r . A recursive method for solving this seemingly exponentially complex problem in linear time was introduced by [6]. This method was developed for a laser sensor where we get a 1D line of cells that could be in the path of the sensor. This method can be extended to the 2D case of a range finding sensor with a larger field of view by first finding all cells within the field of view and then sorting them based on their distance to the sensor. This would produce a 1D array of cells, where cells with lower indexes would block cells with higher indexes as they are closer to the sensor. This 1D array can be directly inserted into the method of [6]. Figure 5 shows how the map looks like after one update with a measured distance of 20 meters with $\sigma^2 = 1m^2$ and the grid cell size of 1x1 meters.

Measurements from different sensors and at different time-steps can simply be incorporated by examining them one at the time. The resulting map from one updated should be used as the a-prior map for the next sensor update. The map is initialized to each cell having a uniform chance of containing an obstacle. If a map over the environment is known, then the map could instead be initialized with a higher value where obstacles are expected to be.

As we want to limit the actual probability of collision, the assumptions in the mapping method must be discussed. The main assumption in an occupancy grid map is that all the cells are independent. This assumption does not hold as all the updated cells from one measurement will be dependent as they give information on where one object is located. If it turns out that the object is not in one cell, then the probability that it is in another cell is increased as the object has to be somewhere. For different measurements of different objects the independent assumption holds, as not colliding with one obstacle does not affect the probability of colliding with another obstacle. Occupancy grid maps attempt to merge dependent sensor information and independent information of multiple measurements in one map with one value at each cell. A problem is that the cell probability value is dependent on the cell size. For small cell sizes the probability does not act as predicted and the map converges to the a-priori value much closer than the measured range. This is highlighted for the 1D case in [14]. As the problem is dependent on the number of possible map-states it gets significantly worse with the number of dimensions making this strategy unfit for 3D.

VII. Scenario-based MPC formulation

The scenario-based MPC strategy will compare different control actions and choose the action that maximizes the mission objective while having an acceptable probability of collision. For the 2D case the list of candidate control actions is defined as follows.

$$\begin{aligned}\alpha = [\ & -90 \quad -75 \quad -60 \quad -45 \quad -30 \quad \cdots \\ & -15 \quad 0 \quad 15 \quad 30 \quad 45 \quad 60 \quad 75 \quad 90\ ]\end{aligned} \tag{45}$$

$$v_0 = \begin{bmatrix} v_0^* & 0.5v_0^* & 0.25v_0^* \end{bmatrix} \tag{46}$$

Where v_0^* is the nominal speed of the drone. The resulting possible trajectories from a stationary start-point along a straight path going downwards are shown in Figure 6.

The drone model will be used to predict future behaviour when applying the different combinations of angle and velocity control actions. The predicted state, as well as the variance in the estimate at each future time-step is used to check the constraint and calculate the cost. The control action that optimizes the objective (see section VII-B) is chosen among the feasible actions that fulfill the constraints (see section VII-A). If no action is feasible then a default action will be taken (see section VII-C). The procedure is repeated at regular sampling intervals.

Only essential constraints and objectives are implemented to highlight how this algorithm works. Other objectives could be added and tuned to give better mission-specific performance.

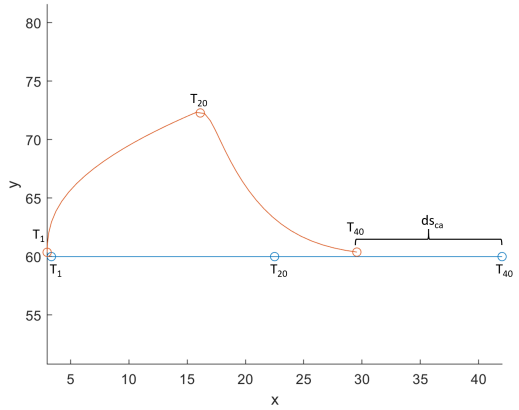


Fig. 7: Red marks the path followed by the drone if a control action of $\alpha = 90^\circ$ is set for the first 20 time-steps, and then turned off for the next 20. Blue marks the corresponding path with the nominal control action. ds_{ca} marks the resulting reduction in traversed distance.

A. Constraint - Probability of collision

To ensure that the chosen route is safe, the probability of colliding at each time-step has to be sufficiently low. This probability is calculated for each cell by multiplying the probability that the drone is within the cell by the probability that there is an obstacle in the cell. This results in the probability that both the drone and an obstacle are present in that cell. The probability of collision is summed up over all cells, giving the probability that the drone will collide with any obstacle cell. The probability of collision has to be lower than a specified maximum probability for collision, $P_{c,max}$, for all time-step. If this is the case, then that control action is deemed feasible.

The constraint is checked at each time-step over a prediction horizon. This horizon is set equal the time needed to identify obstacles, re-plan, and execute an evasive manoeuvre, thereby ensuring that the probability of collision is acceptable over the reaction time of the obstacle avoidance algorithm.

B. Objective

The constraint ensures that the chosen path is safe. The objective function can now be freely chosen based on the objective of the mission. One objective that ensures progress along the path must be implemented. This can be done by maximizing the traversed distance along the path. Deviations make the drone travel orthogonal to the path reducing the traversed along path distance. The drone must fly back to the path at some point which is introducing further delays. This is illustrated in Figure 7. To figure out the delay introduced by a control action, the drone is first simulated as normal with the control action, and is then further simulated with the nominal control action until it reaches back to the path. The nominal action is the behaviour with $\alpha = 0$ and $v_0 = v_0^*$. This second simulation is done without variance propagation and checking the risk constraint. This will significantly speed up the second predictive simulation. By simulating the behaviour from the control action plus the behaviour on returning to the path, the delay introduced by the control action is captured. The traversed along path distance is compared to the case where only the nominal action is applied. The method is described in Algorithm 1 and the resulting predicted reduction in along path distance introduced by the control actions is minimized.

Algorithm 1: Calculation of relative distance along the path.

Let $S_0(t)$ denote the along path distance for the nominal action at time t.
Let $T_{0,max}$ denote the latest time that is predicted for the nominal action.
Let $S_{ca}(t)$ denote the traversed distance for control action (ca) at time t.
Let $T_{ca,max}$ denote the latest time that is predicted for the control action ca.
Let T_h be the time until a new control action will be chosen by the obstacle avoidance algorithm.

Simulate $S_0(t)$ for $t = 0$ to $t = T_h$

for all control actions ca **do**
  Simulate $S_{ca}(t)$ for $t = 0$ to $t = T_h$
  while cross track error at $T_{ca,max} > \delta$ **do**
    Simulate $S_{ca}(t)$ for $t = T_{ca,max} + 1$ without variance propagation
  end
  if $T_{0,max} < T_{ca,max}$ **then**
    Simulate $S_0(t)$ for $t = T_{0,max}$ to $t = T_{ca,max}$ without variance propagation
  end
  $ds_{ca} \leftarrow S_0(T_{ca,max}) - S_{ca}(T_{ca,max})$
end

The prediction should stop when the drone is sufficiently close to the nominal path. The acceptable cross track error is denoted as δ. This has to be done as the drone will asymptotically move towards the nominal path, but may never completely hit it. With δ small, the resulting loss in along path distance is negligible. The larger δ is, the quicker the simulation is finished. A trade-off between computational time and precision has to be made.

C. Default action

When no action is feasible then the safest action among all the given actions and the stop action should be taken. Often when the drone gets stuck, stopping might be the safest choice. But if measurement errors lead to the current drone position being dangerous, then it might be safer to take a non zero action that will move the drone further away from obstacles. If the stop action is chosen then the drone should start rotating to improve the obstacle map.

VIII. Simulation

The simulations were done with 5 sensors, pointing forward, 30 degrees to the side, and 60 degrees to the side. The field of view of the sensors were set to $\pm 45^\circ$ and the range set to 30 meters. The position and velocity variance returned from the estimator, R, and the model variance, Q, were set to $0.1m^2$. The variance in measured range is set to $0.5m^2$. The accepted probability of collision per time-step is set to 0.1%. The map is initialized with a 10% a-priori probability of containing an obstacle. The parameters were set to give interesting behaviour that highlights the workings of the algorithm.

Figure 8 shows that the drone successfully manages to safely fly around a corner and into a tight corridor. An object is present in the blind-spot around the corner. The drone chooses to take a slight detour to acquire new information about the corridor before it flies into it. This ensures that it avoids the obstacle. In the corridor the drone reduces its speed as the wide field of view of the range finding sensors makes it hard to distinguish walls from the safe area in-between. Reducing the speed gives the drone more time to acquire new data and make a new plan.

Figure 9 shows that the drone manages to find the way through a narrow opening. Figure 10 shows the limitations of this algorithm. Figure 10a shows that the drone manages to fly around smaller obstacles, but unable to circumvent larger obstacles as in Figure 10b. This is caused by there not being an option that lets the drone fly in a large enough arch with the given dynamics and look-ahead distance Δ. Figure 10c shows that the drone can get stuck in convex hulls without being able to escape. A solution to this problem would be to include more extensive control action candidates to to have a supervisory controller that detects that the drone is stuck and re-plans the path taking the new obstacle information into account.

IX. Conclusion

This work has looked into obstacle avoidance based on probability of collision and developed an obstacle avoidance strategy that ensures that the probability of collision is at an acceptable level for all time-steps. The essential constraint, which is that the path is safe over the critical time needed to re-plan and stop, is implemented. The essential objective which is to traverse the path is implemented as well. The simulation study showed that the proposed rules ensured that the drone was safe at all times and that the drone managed to avoid smaller obstacles. The strategy worked with the limited mapping capabilities of range-finding sensors with a wide field of view. It produced the behaviour of looking around corners before entering and flying slower when only information about a limited area is known. The strategy forced the drone further away from the walls when the planned path incurred to much risk. The strategy is greedy, making it in some cases unable to find a path around larger obstacles and out of convex hulls. The proposed strategy ensures that the drone will be safe for all paths, and will stop or move to a safer point when no path is feasible. This enables a supervisory controller, or human planner, to plan a path based on a simplified map without taking the safety or dynamics of the drone into consideration.

The main work that needs to be done before the algorithm can be tested out in practice is within mapping. A new method for mapping that generates more realistic probability values and is able to handle 3 dimensions should be developed. The map must be able to handle specular reflections and multipath. The proposed strategy should be tested out in 3D with a more advanced model of the drone.

X. Acknowledgments

This work was done as a part of the UNLOCK project at the Norwegian University of Science and Technology. The project is funded through the Research Council of Norway FRINATEK (The Granting Committee for Mathematics, Physical Sciences, and Technology), project number 274441, and the Center for Autonomous Marine Operations and Systems (AMOS) grant number 223254.

References

[1] D. Q. Mayne, M. M. Seron, and S. V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[2] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *International Journal of Robotics Research*, 36(8):947–982, 2017.

[3] G Garimella, M Sheckells, and M Kobilarov. Robust obstacle avoidance for aerial platforms using adaptive model predictive control. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5876–5882, 2017.

[4] Lars Blackmore, Hui Li, and Brian Williams. A probabilistic approach to optimal robust path planning with obstacles. In *2006 American Control Conference*, page 7 pp. IEEE, 2006.

[5] A Elfes. Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception. In *6th Conference on Uncertainty in AI*, 1990.

[6] Evan Kaufman, Kuya Takami, Taeyoung Lee, and Zhuming Ai. Autonomous Exploration with Exact Inverse Sensor Models. *Journal of Intelligent & Robotic Systems*, 92(3):435–452, 2018.

[7] J Borenstein and Y Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[8] Kyel Ok, Sameer Ansari, Billy Gallagher, William Sica, Frank Dellaert, and Mike Stilman. Path planning with uncertainty: Voronoi Uncertainty Fields. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4596–4601. IEEE, 2013.

[9] T A Johansen, T Perez, and A Cristofaro. Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3407–3422, 2016.

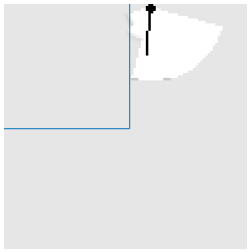
[10] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation, 2002.

[11] Guilherme V Pelizer, Natassya B.F. Da Silva, and Kalinka R.L.J. Branco. Comparison of 3D path-following algorithms for unmanned aerial vehicles. In *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, pages 498–505, 2017.

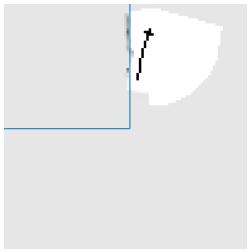
[12] T I Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011.

[13] Kurt Konolige. Improved Occupancy Grids for Map Building. *Auton. Robots*, 4(4):351–367, 1997.

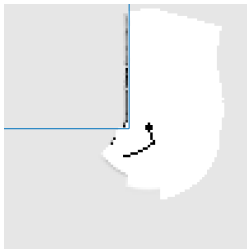
[14] Roxana Dia, Julien Mottin, Tiana Rakotovao, Diego Puschini, and Suzanne Lesecq. Evaluation of Occupancy Grid Resolution through a Novel Approach for Inverse Sensor Modeling. *20th IFAC World Congress*, 50(1):13841–13847, 2017.



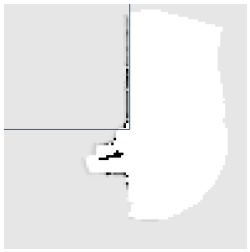
(a) t=0. The initial plan of the drone. The size of the sensor cone makes it impossible to determined if flying towards the path i safe.



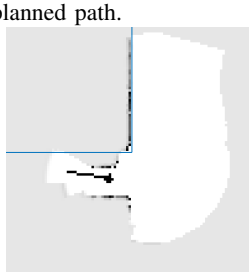
(b) t=12.After mapping for a few seconds a better map over the world is achieved. It is now deemed safe to fly a bit closer to the planned path.



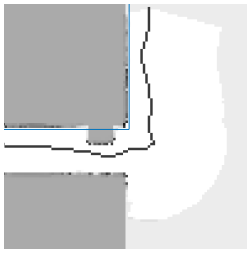
(c) t=51. The drone is tasked to fly around the corner. To avoid flying into unknown territory the drone makes a larger turn.



(d) t=92. The drone enters a narrow corridor where the sensors large field of view makes it difficult to distinguish walls from open space. This forces the drone to reduce its speed.

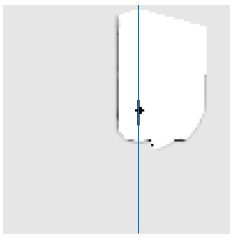


(e) t=94. The corridor widens enabling the drone to make a better map over its environments. The drone increase its speed.

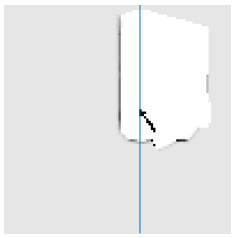


(f) The true obstacles superimposed on the occupancy grid map showing a good fit. There was an obstacle hidden around the corner that the drone managed to avoid by taking a larger turn.

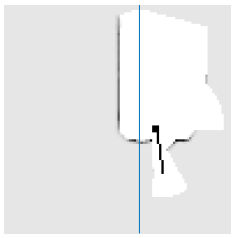
Fig. 8: The drone is tasked to follow the blue lines downwards and to the left. The planned path is too close to the wall to fly safely. The drone is marked as a black dot, and the planned path as the black line. The length of the line indicates the speed of the drone. The background color indicates the state of the occupancy grid, white is safe and darker color indicates a higher probability of the cell containing an obstacle.



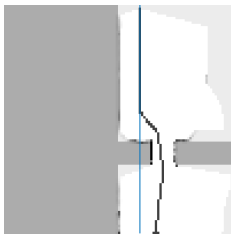
(a) The sensor cone size makes it difficult to detect if there is an opening in the wall The drone reduces its speed and aproaches slowly.



(b) The drone moves towards a possible opening.

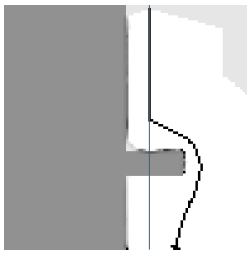


(c) An opening is found and the drone flies through at full speed.

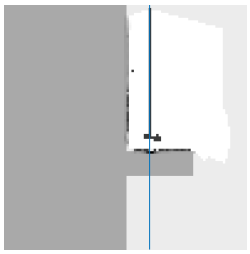


(d) The final path the drone followed. The true obstacles are superimposed on the occupancy grid map.

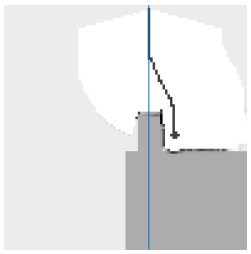
Fig. 9: The drone is tasked to fly straight down, but the planned path does not take the small opening into account.



(a)



(b)



(c)

Fig. 10: Figure (a) and (b) show that the drone manages to circumvent small obstacles, but fails at larger obstacles. Figure (a) is the largest obstacle the drone is able to circumvent with the current control actions and Δ. Figure (b) is one pixel larger hindring the drone form cicumventing it. Figure (c) shows that the drone can get stuck on the wrong side of an obstacle.