Background and Progress Report

For individual project

Jonas Brami

1 June, 2022

Contents

1. Introduction

The problem aimed to solve during this project is the placement of a sensor on a specific target point on a surface using a fixed manipulator arm mounted on the top of an unmanned quadcopter.

A quacopter is an underactuacted helicopter with four rotors. This UAV (Unmanned Aerial Vehicles) is underactuated because arbitrary configurations and trajectory cannot be realized.
In the last ten years, research about UAV controls accelerated drastically with many applications in the civilian industry in a variety of areas.
Monitoring and sensing tasks are traditionally operated by human but can be complicated (require expert, highly trained technicians), expensive, and dangerous for the human operator (when the point of interest is hard to reach, the human manipulator may need special training and equipement to reach the point of interest). For example, big structures like bridges need to undergo regular inspections to ensure there are no cracks or other signs of structural fatigue.
Using UAVs for such tasks would not only reduce cost of maintance of those structures, but also increase the security of both the human manipulator and the civilians using it.
Sensor placement on surfaces using UAVs is an active field of research and we will propose a solution based on velocity fields.

In this project we present an application of the Passive Velocity Field controller (PVFC) using Velocity field derived from the gradient of a potential or a shaping function; integrated with a depth camera for active obstacle avoidance and a grasping arm for sensor placement. We also provide an analysis of the controller behavior under multiple types of fields and controller parameters.

This project was done in 2 steps: The first one was the implementation of a Passive Velocity Field controller with simple quadcopter dynamics and physics on Python (Thanks Brett!). The simplicity of this simulation has allowed us to easily debug the controller, verify that our implementation is behaving as expected, and get a high level understanding of the parameters of this controller. The second step was implementing the full Sensor Placement pipeline using ROS/PX4/Gazebo. Robot Operating System is a platform to build robot application where task specific node interact together based on subscribing and advertising to topics. Gazebo is a simulator providing real world physics, a variety of plugins that we used to implement a grasping arm to simulate a Sensor Placement task. PX4 is an open source autopilot sotfware integrated with ROS to provide seamless communication between our ROS controller and the quadcopter actuators. Those implementations will be further detailed in section 4

2. Background and Literature Review

2.1. Current State of the art for UAM

There exists 2 main approaches for controls of unmanned aerial manipulators: The first one is the centralized approach where we consider the manipulator and the UAV as a whole whereas the decentralized apporach the manipulator control and UAV control are independant problems. In the case of the Sensor placement with a quadcopter, we use the centralized approach because the arm has no degree of freedom and the force exerted by the tip is coming from the thrust of the UAV. The centralized approach is often built on top of a model-based full state control loop optimized with LQR (Linear–quadratic regulator) around some desired state. In [4], the author present the current state of the research for UAMs. An UAM can be divided into 4 elements: The UAV floating base (in our case, a quadcopter), the robotic arm, a sensor/gripper attached to the end of the arm (in our case, a sensor will be attached to the end of the arm), diverse sensors on UAV to handle perception (the depth camera)

2.2. Velocity field path-planning for single and multiple unmanned aerial vehicles

In [1], the author presents a path-planning technique based on velocity fields generated from potentials solution of Laplace's equation. Two different types of solution to potential V for the Laplace

$$\begin{aligned} \nabla^2 V &= 0 \\ \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} &= 0 \end{aligned} \tag{1}$$

equation are presented in this paper: Type 1 are irrotational solutions to generate sink and source fields and Type 2 solutions are used to build solenoidal fields.

$$V_1 = Q_1 \ln\Big((x_1 - \tilde{x_1})^2 + (x_2 - \tilde{x_2})^2\Big) \tag{2}$$

$$V_2 = Q_2 \arctan\left(\frac{(x_2 - \tilde{x_2})}{(x_1 - \tilde{x_1})}\right) \tag{3}$$

where (x_1, x_2) is the position of the UAV, $(\tilde{x_1}, \tilde{x_2})$ is the position of the obstacle; V_1 and V_2 are respectively type 1 solution (source field) and type 2 solution (vortex field).
The author justifies the use of Laplace solution for building the velocity field for multiple reasons:

- The use of Laplace solution for potential guarantees the uniqueness of the minimum in the field. Specifically, the use of vortex function built from shaping function to circle around obstacle will ensure that only the goal point will be a minimum of the field and that the UAV will not get stuck at some local minimum. As the author states, we can do an analogy with a famous strategy to find the exit of a maze: by keeping a hand on a wall of the maze and walking while always touching the wall, we are ensured to find the end of the maze. This is far from being an optimal solution, however, it can guarantee that the goal will be reached. As a result, those solenoidal fields based on vortex function also provide active collision avoidance.

- Scalar shaping functions are at the base of these methodology because by crafting them to match the shape of the obstacles, we are able to generate corresponding vortex functions for obstacles of any shape. Since the vortex field is defined for each obstacle, it would be easy to reevaluate the field after addition or removal of an obstacle.

- Finally, irrotational solutions of the Laplace equation allow us to enforce an exclusion radius around obstacles (source field) and to direct the UAV in direction of the target point (sink field). The exclusion radius is encoded using the amplitude Q_1 of the irrotational field.

We can leverage these both types of potentials to derive a velocity field that will guide the UAV to the contact point without colliding with the surface. For example, we could define the exclusion radius to be the distance between the centre of mass (CoM) of the quadcopter and its most distant part on the quadcopter. We will still be able to make contact because the distance between the tip of the arm and the CoM will be longer than this exclusion radius.

3. Design and Theory

A velocity field is a function taking as input a state, q and returns a 3 dimensional desired velocity vector (u, v, w).
It describes a path (a function of position) as opposed to a trajectory (function of position and time).

3.1. Spherical field to maintain desired force

When contact has been made, we suppose that the surface static friction coefficient is high enough to maintain the contact. Since the arm has a fixed size and does not move, this section will describe a velocity field on the surface of a sphere around the target point with a radius defined by the distance between the CoM of the quadcopter and the tip of the arm.
Computing the field on the surface of a sphere is very computationally efficient because we do not have to sample field vectors in the whole 3d space around the point of interest. First we need to compute the feasible position of the CoM to apply the desired force. We know that this position is unique because there is only one vertically stable pitch for a given desired force amplitude. The quadcopter needs to pitch to have a forward velocity because the quadcopter is underactuacted.
We can either compute this position analytically or we can use machine learning techniques such as regression to compute the feasible pitch as a function of the desired force. The latter option would require collecting training data from simulations on Gazebo. Now we can generate the velocity field on the surface of the sphere to point on the tangent direction of the sphere in the direction of the stable pitch position.
This field will have an amplitude proportional to the distance from this point. The planning strategy we described would also allow us to easily define a desirable range for the yaw angle depending on the type of sensor and on the friction coefficient of the target surface. Finally, we use a Passive Velocity Field Controller [2] to follow this field to minimize the loss of kinetic energy to the environment when interacting with the surface. Now we are going to describe how the spherical field is computed. Let us recall that the cartesian to spherical change of variable is defined as

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \arccos\left(\frac{z}{r}\right)$$

$$\phi = \begin{cases} \arctan\left(\frac{y}{x}\right), & \text{if } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi, & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi, & \text{if } x < 0 \text{ and } y < 0, \\ \pi/2, & \text{if } x = 0 \text{ and } y > 0, \\ -\pi/2, & \text{if } x = 0 \text{ and } y < 0, \\ notdefined, & \text{if } x = 0 \text{ and } y = 0 \end{cases} \tag{4}$$

The jacobian defining the curvature of the sphere for at each point for this transformation is

$$J = \frac{\partial(x, y, z)}{\partial(r, \phi, \theta)}$$

$$J = \begin{bmatrix} \sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) & -\sin(\phi) \\ \sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi) & \cos(\phi) \\ \cos(\theta) & -\sin(\theta) & 0 \end{bmatrix} \tag{5}$$

We can now derive this velocity field in 3 steps:

1. Sample a list of point on the surface of the sphere in cartesian coordiate. In Python or Matlab, a meshgrid can be used where we the field will be non zero with

   $$|\sqrt{x^2 + y^2 + z^2} - R_{sphere}| \leq \epsilon$$

   where R_{sphere} is the radius of the mounted arm.
2. For each point in the list, we compute its spherical coordinate according to 4 and we compute the jacobian at the point
3. Let $(\phi_{opt}, \theta_{opt})$ be the desired position on the sphere. For each point (ϕ, θ) on the sphere , we compute the tangeant vector in the the field that will point toward the goal as $J(r, \phi, \theta).(\epsilon, \phi_{opt} - \phi, \theta_{opt} - \theta)^T$ using the jacobian defined in 5

3.2. Passive Velocity Field Control of Mechanical Manipulators

In [2], the author explains the advantages of encoding a contour following task using velocity fields. He later presents the passive velocity field controller whose objective is to "maintain an energetically passive relation-ship between the manipulator under closed loop control and its physical environment, while causing the manipulator to perform the desired task." We will explain each one of those arguments and relate them to our sensor placement task.

3.2.1. Velocity fields for planning

The classical approach to do planning is to encode the task into a timed trajectory $Q : \mathbb{R}_{\geq 0} \to G$ where G is the n-dimensional configuration manifold for the manipulator. Using this strategy, the objective of the controller is to minimize the deviation between $q(t)$ and $Q(t)$ where $q : \mathbb{R}_{\geq 0} \to G$ is the actual coordinate representation of the manipulator. This strategy could be fine if the manipulator was able to never deviate from the timed trajectory defined by Q. However, this is rarely the case and when a deviation occurs, a side effect of this minimization strategy called radial reduction

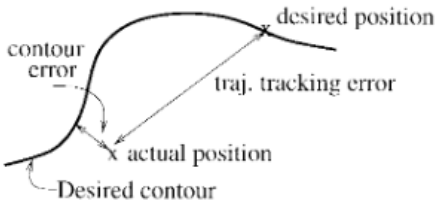


Figure 1: radial reduction from [2]

can occur. The author argues that if following the path of the desired trajectory is more important than the timing in which the manipulator follows this trajectory, a strategy based on velocity field is more appropriate because the velocity of the manipulator will only depend on its current position and will be time invariant. For the sensor placement task, using a timed trajectory strategy with deviation minimization to reach the contact point may lead the UAV to collide with an obstacle. With our strategy based on potential velocity field, the UAV will not try to shortcut the desired path in case of deviation.

3.2.2. Passive velocity field controller

The controller presented in this paper allows storing kinetic energy of the manipulator's actuators in a spring or a flywheel and releasing it when needed so that we can interact with surfaces while minimizing energy loss. This is useful because the UAV has a limited amount of energy stored in its battery and this is one of the main limiting factors for most tasks (including Sensor Placement).
To present this concept, the author first define the notion of a passive dynamic system:
A dynamic system with input $u \in U$ and output $y \in Y$ is passive with respect to the supply rate $s : U \times Y \to \mathbb{R}$, if for any $u : \mathbb{R}_{\geq 0} \to U$ and for any $t \geq 0$ the following relation is satisfied:

$$\exists\, c \in \mathbb{R}$$

$$\int_0^t s(u(\tau), y(\tau))d\tau \geq -c^2 \tag{6}$$

Let us remember that the work W and power P of a force F on a point mass object with velocity v are defined as

$$P = F.v \tag{7}$$

$$W = \int_0^t P dt \tag{8}$$

Therefore the supply rate $s(\tau_{tot}, \dot{q}) = \tau_{tot}^T \dot{q}$ can be seen as the total mechanical power input. Here the input τ_{tot} is the total force exerted on the manipulator and the output $\dot{q}$ is the velocity of the manipulator.

When considering a feedback system interacting with the environement shown in figure 2, we can decompose $\tau_{tot} = \tau_e + \tau$ (where τ and τ_e are respectively the forces generated by the actuators and the external forces, for example the contact force when touching a surface), we can derive the power generated by external forces as $s(\tau_e \dot{q}) = \tau_e^T \dot{q}$. As explained in the paper, a system

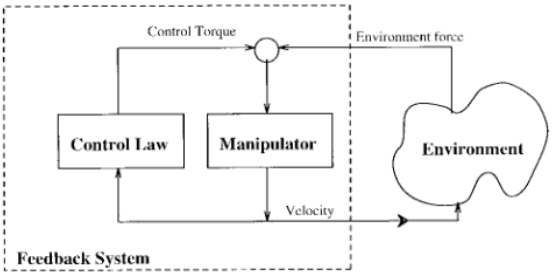


Figure 2: PVFC loop [2]

defined by this supply rate is not passive because obstacles may bring the manipulator to a complete stop for an unbounded amount of time and this loss of kinetic enery cannot be bounded. As a result, the passivity relation

$$\int_0^t \tau_e^T \dot{q} d\tau \geq -c^2 \tag{9}$$

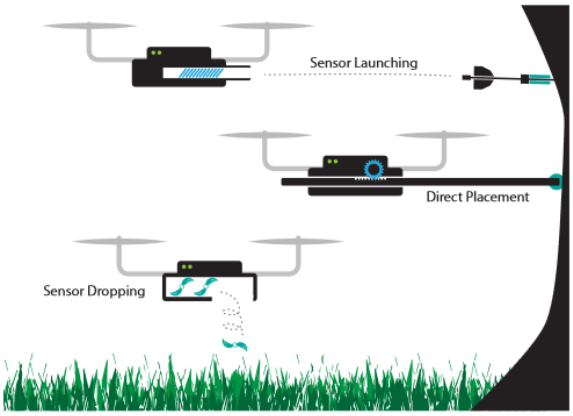
is not valid here because the l.h.s represents the total amount of energy lost to the environment and, as stated before, it cannot be bounded. This issue motivates the introduction of an augmented system: a fictitious flywheel is added to the system that acts as an energy storage element. The dimensionality of the manifold is then increased by one to include the state of the flywheel and this augmented state will be noted as

$$\bar{q} = [q_1, ..., q_n, q_{n+1}] \tag{10}$$

The paper later presents the dynamics of this augmented state and designs a field for the fictitious flywheel such that "the kinetic energy of the augmented system remains constant"

Diverse ways of placing sensors using UAVs have been explored in the past, including but not limited to:

- Direct Placement: Using a fixed arm manipulator on an UAV, we use the force exerted by the thrust of the UAV to provide enough pressure on the tip of the arm to place the sensor on the target point.
- Sensor Launching [1]: Using the energy stored in a spring, the UAV ejects the sensor at the desired velocity to reach and attach to the target (Unmanned Aerial Sensor Placement for Cluttered Environments). This strategy is very useful when it is not physically possible for a mounted arm to reach the target however it suffers from small payload capacity.
- Drop from flight: We simply drop the sensor above the target point. When target accuracy is not a priority and we are aiming at a non-vertical surface and there is no occlusion above the target, this sensor placement strategy is the most effective.



	Direct [7]	**Drop** [2]	**Launch** [i]
Accuracy	±0.025 m	±4 m	±0.1 m
Safety distance	0 m	>10 m	4 m
Payload	1.85 kg	10 kg [ii]	0.65 kg
Sensor number	single	multiple	expandable

Figure 3: Different types of sensor placement from [1]

The characteristics of each one of mentioned methods are shown in 3. We decided to go through with the Direct Placement strategy because despite its simplicity, it provides good accuracy and is able to place a large variety of payloads.

In this section, we will present the field we previously described with illustrations. The field in figure 4 was drawn by computing the gradient of V_1 with $(\tilde{x_1}, \tilde{x_2}) = (0, 0)$.

The field in figure 5 is a sink field at $(0, 1)$, it is similar to the source field in figure 4 but with opposite sign.

The field in figure 6 has been generated by computing the gradient of the shaping function of a superquadratic:

$$H = (x_1 - \tilde{x_1})^n + (x_2 - \tilde{x_2})^n \tag{11}$$

$$F = \frac{1}{1 + (\frac{1}{L} H^{\frac{1}{n}})^m} \tag{12}$$

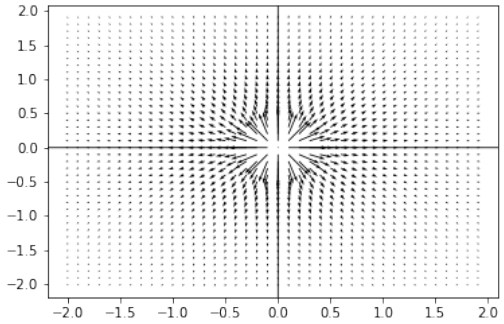


Figure 4: Simple Type 1 irrotational source

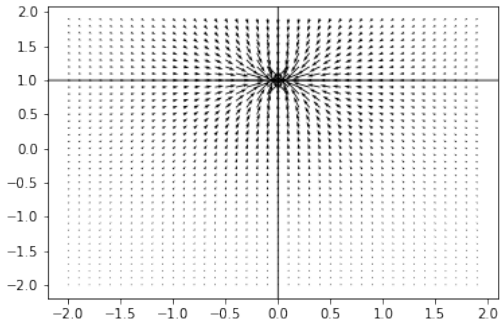


Figure 5: Simple Type 1 irrotational sink

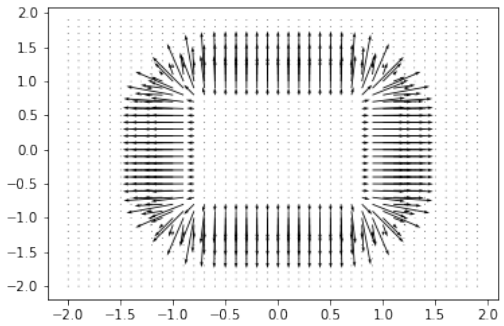


Figure 6: irrotational field from shaping

As stated in [3], when $m \gg 1$, the edge of the shaping function gets thinner. Higher values of n result in more rectangular shapes whereas $n = 2$ describes the shaping function of a sphere.

Both these fields are type 1 irrotational solutions of the Laplace equation.

By adding the irrotational sink from figure 5 and the irrotational field from shaping function in figure 6 we obtain a good representation in figure 7 of what the velocity field will look like when close to the target point.

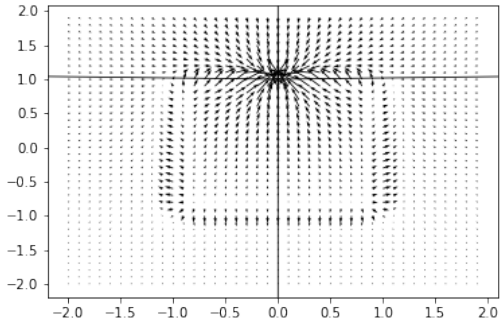


Figure 7: irrotational field from shaping with sink

The shaping function is mainly used to generate a vortex field around an obstacle, as explained in the paper. It is given by:

$$v_2 = -\frac{\partial F}{\partial x_2} e_1 + \frac{\partial F}{\partial x_1} e_2 \tag{13}$$

Here, e_1 and e_2 are an orthonormal basis.
We can see in figure 8 an example of vortex field around a super quadriatic.

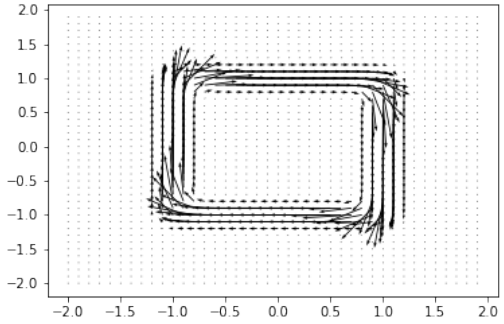


Figure 8: Solenoidale field from shaping function

As explained in section 2, after we made contact, we stop using the potential based field and we use the spherical field to maintain contact with the desired force. Figure 9 represents an example of such a field where the red point is the optimal position to apply the pressure, the blue arrows are the velocity field vectors and the center of the sphere is the point where the pressure is applied

4. Implementation

As was stated in the introduction, we have two main implementations:

Simplistic Python implementation
The Python implementation uses simplistic quadcopter dynamics only defined by:

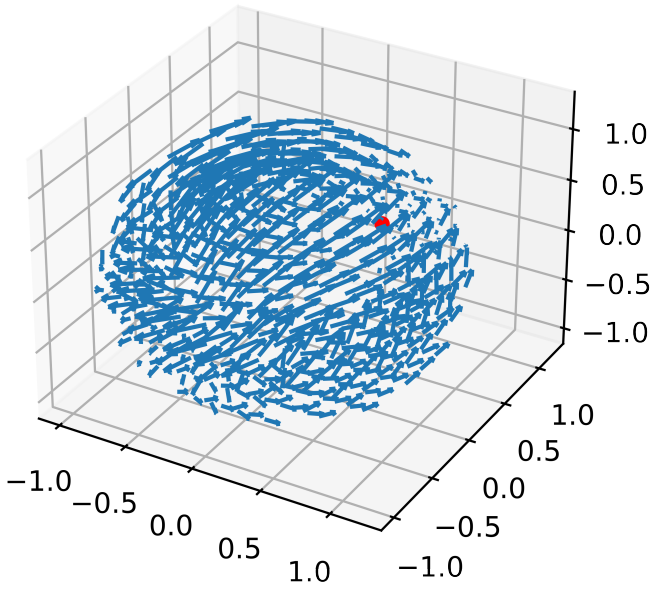


Figure 9: Spherical contact field

- Quad mass: m_b
- Moments of inertia: I_{xx}, I_{yy}, I_{zz}
- drag coefficient: A_x, A_y, A_z
- pd attitude gains: K_{ϕ_p}, K_{ϕ_d}, K_{ψ_p}, K_{ψ_d}, K_{θ_p}, K_{θ_d}

We use ODEint to evolve the state of the system by doing to following on each iteration:

- First we compute the current Mass Matrix M, Coriolis Matrix C, Control Matrix B, gravity Matrix G
- Then we pass to the PVFC controller those dynamics together with the desired velocity field and compute a desired thurst vector according to (cite pvfc equations)
- For a given desired thrust vector, The quad uses a simple attitude PD control law to compute the state derivates of the quad

The desired velocity fields are computed using Sympy (A Python library for symbolic) for easy field and field gradient computation. Since most fields we used are derived from potential functions, we can easily obtain the field and field jacobian required by PVFC by symbolically differentiating their potentials or shaping functions.

Full Sensor Placement pipeline in ROS/Gazebo/PX4

In this implementation, we are using a real world quadrotor called IRIS quad provided in the PX4 SITL package. The dynamics are evolved and computed by Gazebo. The implementation is divided in 2 ROS Nodes, 1 ROS-PCL Node, 1 Gazebo-ROS plugin, and 1 PX4/MAVROS node:

- Velocity Field ROS Node: This node subscribe to the mavros odometry topic, computes the velocity Field at the current position and publishes the desired field and field gradient. We implemented all the velocity field using the symbolic math library SymEngine so that all gradient computations from potential or shaping function would be done automatically. In addition, this node is implementing a state machine such that in each state, a specific field is being used. Specifically, the first state gives a sink to get altitude, the second state is a sink field to approach the target, the third state represents the contact phase with the wall(the node subscribes to a topic called "grasping" to know if the grasping arm is currently activated) it starts when an engagement signal is received from the grasping arm and it finishes when a disengagment signal is received from the

grasping arm. The last state is to get away from the wall and land, we use a sink field to implement it. We implemented an obstacle avoidance system for generic superquadratic obstacles, upon obstacle detection, we compute the avoidance field for this obstacle from the appropriate shaping function, an irrotational field is generated to avoid the obstacle. This node subscribes to an obstacle topic to know when new obstacle have been detected.

- PVFC ROS Node: This node subscribes to the desired velocity field and to the MAVROS odometry and publishes a thurst vector on the MAVROS attitude topic computed my the PVFC controller.
- MAVROS Node :This node is the interface between the PX4 firmware and ROS
- PCL Object Segmentation Node: This node reuses the code of a PCL tutorial performing cylinder segmentation on point cloud data(cite). We added a ROS integration to to the velocity field topic the position of the detected object and also we implemented the transforms from camera point of view to FCU frame, and from FCU frame to world frame.
- Grasping arm Gazebo-ROS plugin: This node reuses the code of the vaccuum arm of Gazebo, but it was modified so that could be "attached" to the moving IRIS model and attach to a wall in Gazebo.

5. Results, Evaluation and Experiments

- Calibrate thrust
- introduction of compensation terms for friction
- finetuning (Ax Ay Az). too high cause speed to diverge, too low the quad lose all its energy and stop.
- beta error
- presents experiences in different worlds
- using wind world helped lead the way to understand the need for compensation

6. Conclusion, Future Work

- Evolutionary algorithm to select the optimal gains and Ebar
- Estimate the friction coefficient
- Smart sampling the field to find the best Ebar
- More obstacle types
- desired yaw velocity field

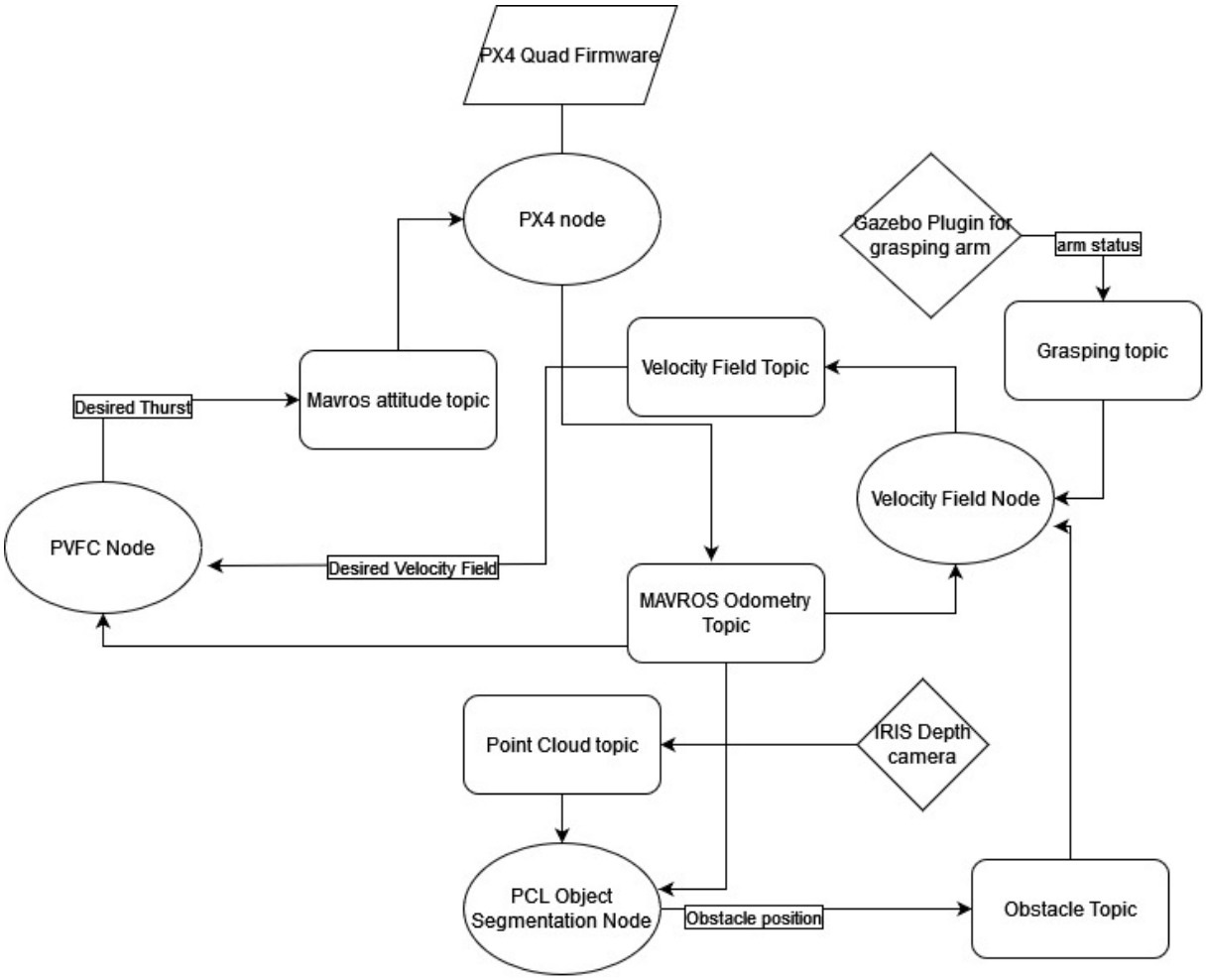


Figure 10: implementation diagram

References

[1]A. Farinha, R. Zufferey, P. Zheng, S. F. Armanini, and M. Kovac, «Unmanned aerial sensor placement for cluttered environments», IEEE Robotics and Automation Letters **5**, 6623–6630 (2020).

[2]P. Y. Li and R. Horowitz, «Passive velocity field control of mechanical manipulators», IEEE Transactions on robotics and automation **15**, 751–763 (1999).

[3]C. McInnes, «Velocity field path-planning for single and multiple unmanned aerial vehicles», The Aeronautical Journal **107**, 419–426 (2003).

[4]F. Ruggiero, V. Lippiello, and A. Ollero, «Aerial manipulation: A literature review», IEEE Robotics and Automation Letters **3**, 1957–1964 (2018).