On-Board Estimation, Nonlinear Control and Swarms

Sensing and Estimation

Onboard State Estimation

Robots must have the ability to estimate their state, their position, the orientation and the velocities. They must be able to do it onboard.

Hokuyo Laser Sensor

- allows robots to measure distances to obstacles
- operation in unstructured environment

Paper – Multi-Sensor Fusion for Robust Autonomous Flight in Indoor and Outdoor Environment with a Rotor craft MAV.

motion capture camera

→ able to capture measure different features in the environment with precision & repeatability

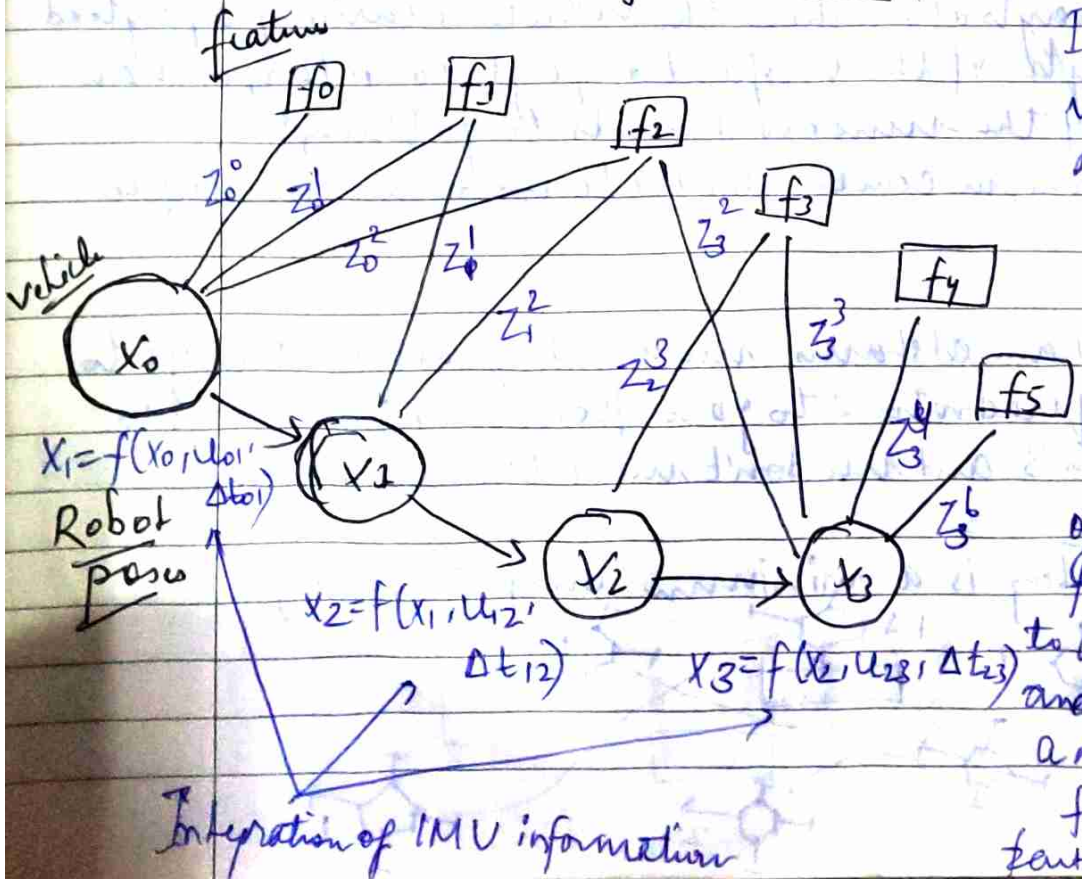
→ Reflective markers – Allowing cameras to estimate the position of quadrotor to within millimeter and orientation of quadrotor to within a fraction of a degree.

→ Then updates can be obtained at rates as high as 200 Hz.

(Vicon Motion Capture System)

Simultaneous Localization and Mapping (SLAM)

(also Structure from Motion)



Imagine we have the vehicle at position X_0. At pos X_0, it measures first three features f_0, f_1 & f_2. After moving to a new position X_1, it measures a subset of those features f_1 & f_2. It moves to another position and now it measures a new feature, f_3 which was not part of its feature set.

Some overlapping with its existing feature set and some contributing to new features. So this goes on & on.

→ The key idea is that we have a graph with two different types of edges.

→ the first set of edges correspond to measurements made by onboard sensors, cameras or laser scanners.

→ the second set of edges has to do with movements that the robot has made. Based on its applied inputs and based on the time elated, it's able to estimate how far it's moved. (In other words, it's able to estimate the difference between X_1 & X_0, X_2 & X_1, X_3 & X_2 and so on).

→ Each of these types of edges corresponds to information gleaned from sensors. This information is noisy. We eventually have a big graph with types of edges. We have equations that describes these edges. — We end up with a big optimization problem that we have to solve.

If we solve it, not only can we recover the positions of these features but also the displacements from $X_0 \rightarrow X_1$, $X_1 \rightarrow X_2$, etc. So, we are about to map the features as well as localize ourself as we fly through the environment. This is the simultaneous localization and mapping problem, also called SLAM.

GPS
laser scanners
pressure altimeter
stereo cameras
a download camera
IMU

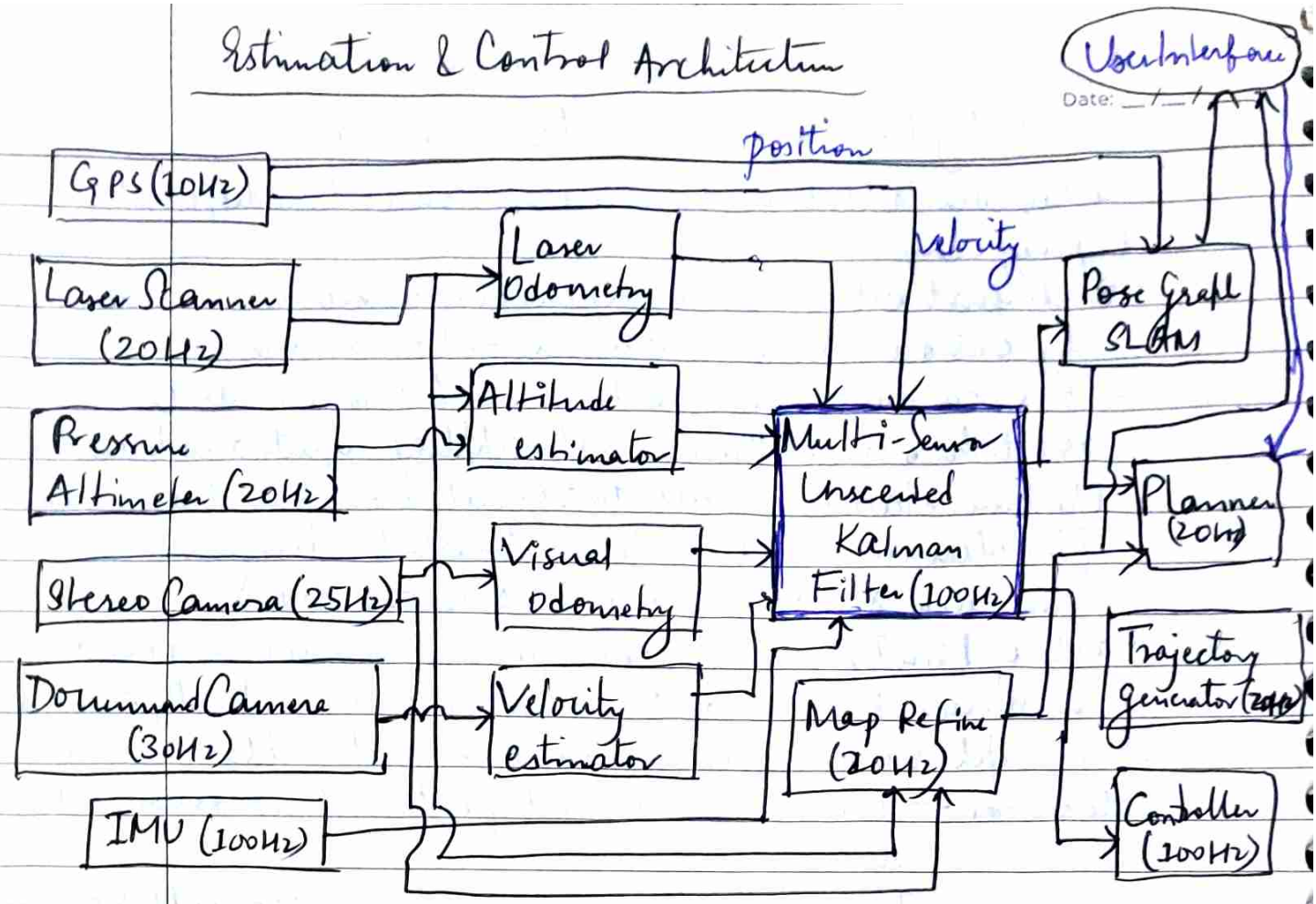
Each of these sensors give us information that's of a different type. These information also comes at different rates. The fastest sensors usually are the IMUs (100 - 200 Hz) while GPS only works around 10 Hz. We combine information from all these sensors using a filter and that filter obtains state estimates at 100 - 200 Hz and this allows us to drive the controller.

Multi-Sensor Unscented Kalman Filter (100 Hz)

→ In addition to getting the state information, we are able to get the information from the sensors to create a map and this is the SLAM problem.

#learnthesmarterway

Estimation & Control Architecture



If we have a user that's interacting with the map, he or she can specify goals or intermediate waypoints for the quadrotor and guide the vehicle through a complex environment with a map that's being built by the robot without actually being in the same location as the robot

Paper – Multi-Sensor Unscented Kalman Filter & how we build 3D maps (Shaojie_Shen-Thesis.pdf)

✓ Autonomous navigation in confined indoor environments with a micro-aerial vehicle.

Systems Design Considerations

→ Larger vehicles are more capable (better sensor, processor)
→ Larger vehicles can exhibit longer mission (bigger battery)
→ Smaller vehicles can navigate in more complex indoor environments. Also, they are inherently more agile & maneuverable.

① IMU measurement provides below information —

→ ~~Info about absolute position~~ body-frame acc^n, angular rates & optionally (magnetic field)

→ Info about absolute attitude (orientation) of the robot)

→ Info (via integration) about the relative motion from $x_{i-1} \to x_i$

Commercially available overall, 3D laser scanners don't have the resolution that is required for state estimation for flights at modest speeds.

Nonlinear Control

Until now, we have considered the problem of controlling the quadrotor at states that's not too far from the hover equilibrium configuration. Because of this we are able to make some assumption.

Limitations of Linear control —

Assumption — roll & pitch angles and all velocities are close to 0.

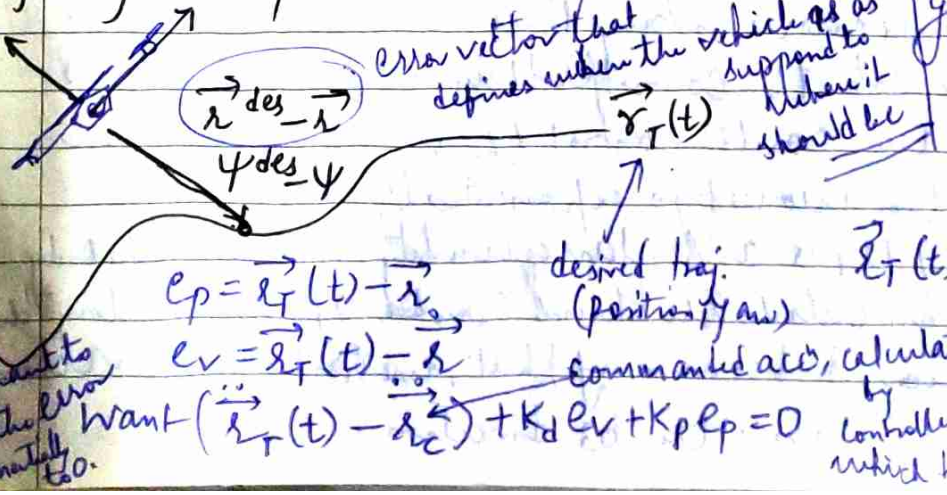
But, as the vehicle maneuvers at high speeds and exhibits roll and pitch angles far away from 0, these kinds of controllers are not likely to work.

We want to overcome these limitations.

→ Specifically, we want to consider nonlinear controllers that allows us to control the vehicle far away from the equilibrium state.

Nonlinear Control — Control the robot at states far away from the equilibrium (hover) state.

Trajectory Control problem (Check previous notes)



Given $\vec{r}_T(t), \dot{\vec{r}}_T(t), \ddot{\vec{r}}_T(t)$
$\vec{r}^{des}(t), \dot{\vec{r}}^{des}(t), \ddot{\vec{r}}^{des}(t)$
$\psi^{des}(t), \dot{\psi}^{des}(t), \ddot{\psi}^{des}(t)$

$$\vec{r}_T(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \psi(t) \end{bmatrix}$$

that can be applied by quadrotor through its motor inputs

Controller design problem

Date: __/__/____

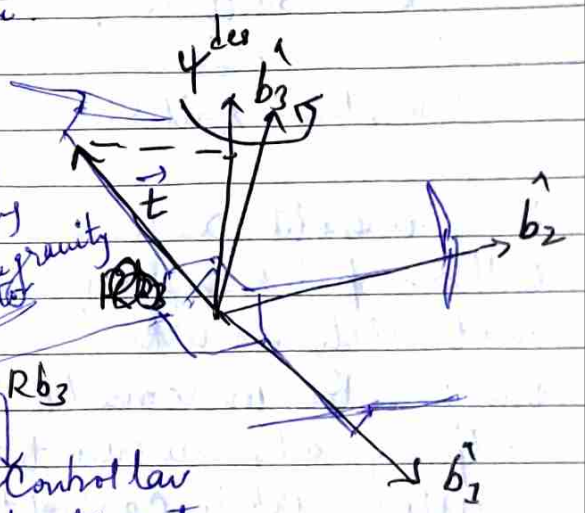
Inner loop addresses the attitude control problem and outer loop then addresses the position control problem.

Compares the commanded orientation with the actual orientation and the angular velocities & determines the input u_2.

Looks at the specified position and its derivatives, compares that to the actual position and its derivatives and computes the input u_1.

We want to do the same thing as above except now we want to look at trajectories that can exhibit high velocities and high roll and pitch angles, i.e., trajectories that are not close to equilibrium or hover state.

$$\xi_T(t) = \begin{bmatrix} x^{des}(t) \\ y^{des}(t) \\ z^{des}(t) \\ \psi^{des}(t) \end{bmatrix}$$



$$u_1 = \left[\ddot{z}^{des} + K_v e_{\dot{z}} + K_p e_z + mg \right] \cdot R b_3$$

Compensating for the gravity vector

$\vec{t}$

$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

- feed forward term which essentially is the acc^n of the specified trajectory
- derivative term
- proportional term
- Control law for the outer position loop (this is essentially a proportional plus derivative control law)

If the system were fully actuated, in other words, if the system could move in any direction, then this would essentially solve the control problem. u_1 would be a 3D vector & we could control the vehicle so that error in position would go exponentially to 0.

However, u_1 is a scalar quantity, in fact, the only direction in which the thrust can be applied is along the b_3 direction. So because of that, we take the vector $(\vec{t})$ & project it along the b_3 direction.

#learnthesmarterway

So that gives us a projection that's close to the desired input, but not quite because $\vec{t}$ is never perfectly aligned with $\hat{b}_3$.

Attitude Control problem

Knowing that we want $\vec{t}$ to be aligned with $\hat{b}_3$ and knowing that we have very little flexibity in what direction it will point in, we do the next best thing which is we rotate the vehicle so that $\hat{b}_3$ is along $\vec{t}$. This gives us the Constant.

R^{des}

$$R^{des}\hat{b}_3 = \frac{\vec{t}}{\|\vec{t}\|}$$

$$\psi = \psi^{des}$$

the desired rotation matrix should be such that $\hat{b}_3$ should point in the $\vec{t}$ direction.
we also know what the desired yaw angle needs to be.

$e_R(R^{des}, R)$ (error in Rotation) by Comparing the actual rotation & the desired rotation.

$$\longrightarrow \vec{u}_2 = w \times \vec{I}w + \vec{I}(-K_R e_R + K_w e_w)$$

$w \times \vec{I}w$: proportional to error in Rotation

$\vec{I}(-K_R e_R + K_w e_w)$: proportional to the error in angular velocity

we also have to compensate for the fact that the vehicle has a non-zero inertia I and there are gyroscopic terms as we know from the Euler eqns of motion.

So, $u_1 = \left[\ddot{r}^{des} + K_v e_{\dot{r}} + K_p e_r + mga_3\right] \cdot Rb_3$

Control inputs & $u_2 = w \times \vec{I}w + \vec{I}(-K_R e_R - K_w e_w)$

How to determine R^{des}?

↳ desired rotation matrix

Date: __/__/____

→ we are given two pieces of information

(I) we want the $\hat{b}_3$ to be aligned with $\vec{t}$. $\vec{t}$ is known and we want to find R^{des} so that below eqn is satisfied

$$R^{des}\hat{b}_3 = \frac{\vec{t}}{\|t\|}$$

$$\hat{b}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

(II) we want the yaw angle to be equal to the desired yaw angle, ψ^{des}; i.e., $\boxed{\psi = \psi^{des}}$

we know from the theory of Euler angles that the rotation matrix has the following form:

$$R = \begin{bmatrix} C\psi C\theta - S\phi S\psi S\theta & -C\phi S\psi & C\psi S\theta + C\theta S\phi S\psi \\ C\theta S\psi + C\psi S\phi S\theta & C\phi C\psi & S\psi S\theta - C\theta S\phi C\psi \\ -C\phi S\theta & S\phi & C\phi C\theta \end{bmatrix}$$

we should be able to find the roll & pitch angles.

How to calculate the error $e_R(R^{des}, R)$?

Given the current rotation R and the desired rotation R^{des}, what is the rotation error?

→ we cannot simply subtract one matrix from another. If we do so, the resultant matrix will not be orthogonal and will not be rotational matrix.

Correct way

what's the magnitude of the rotation required to go from the current orientation to the desired orientation?

$$R \longrightarrow R^{des}$$

#learnthesmarterway

The required rotation is

$$\Delta R = R^T R^{des}$$

Date: __/__/____

Once we have ΔR, the angle of rotation (which is the magnitude of rotation) and the axis of rotation can both be determined using the Rodrigues formula.

Stability

→ Large basin of attraction

$$tr\left[I - (R^{des})^T R\right] < 2$$

$$\|e_w(0)\|^2 \leq \frac{2 k_R}{\lambda_{min}(I)}\left(1 - \frac{1}{2} tr\left[I - (R^{des})^T R\right]\right)$$

basin of attraction is the region from which it will converge to a desired equilibrium point, as almost all of the space of rotations and a very large set of angular velocities & linear velocities.

controller

Specifies the set of rotations from which it can converge to the hover configuration.

This inequality tells us the range of angular velocities from which it can converge to the hover configuration.

this term $[\lambda_{min}(I)]$ relates to the eigen values of the Inertial tensor.

$\lambda_{min}(I)$ is essentially the smallest eigen value of the Inertia matrix.

The smaller this eigen value, the larger the quantity on the R.H.S, which tells us that as we scale down the inertia matrix, the set of angular velocities from which the robot can converge to the hover state increases. In other words, the vehicle becomes more stable.

Nature example Honeybees are small, they are able to react to collisions & recover from them. Because the inertia is small, they have a controller whose basin of attraction is fairly large and they are able to exploit the advantages of small size and small inertia to recover from large deviations.

#learnthesmarterway

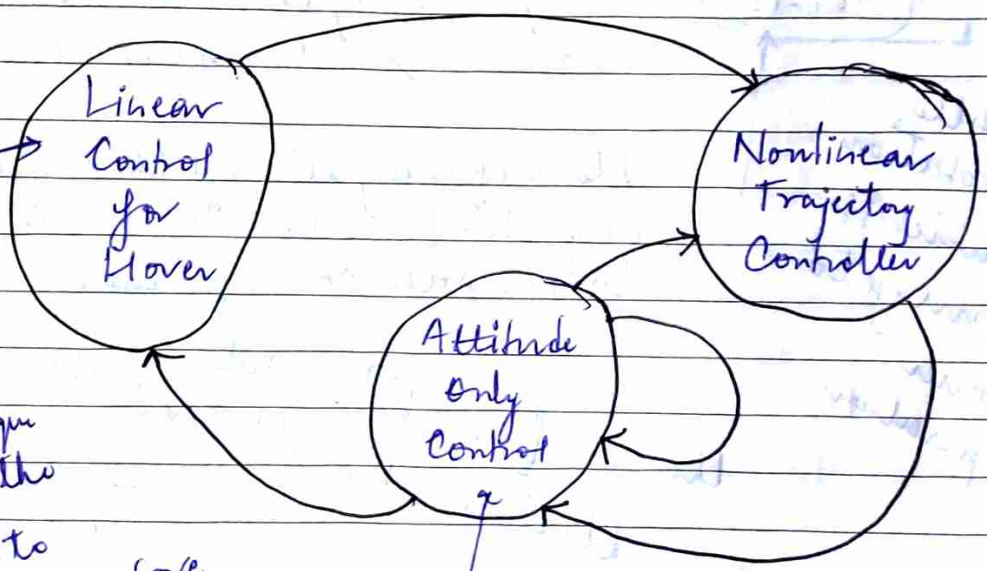
Paper — Design of small, safe and robust quadrotor swarms Date: __/__/____

* Because of large basin of attraction for the non-linear controller that the small vehicles are robust to deviation.

Size of basin of attraction $\sim \frac{1}{L^{5/2}}$ (Smaller the characteristic length L, the bigger the basin of attraction)

Paper — Minimum Snap Trajectory Generation and Control for Quadrotor.

Sequential Composition



linearized hover controller

we can use this technique in order to allow the robot to be able to gather momentum before twisting its body to go through a window whose width is only a few centimeters more than height of the robot.

Thesis — daniel-mellinger (how to use the nonlinear controller and how to plan 3D trajectories)

Trajectory planning

Date: __/__/____

The ability to use nonlinear controllers also allows us to design trajectories that are more aggressive. In order to see that, it's useful to exploit the geometric structure that's inherent in these quadrotor

Input: $u_1, \vec{u}_2$

$$u_1 = \sum_{i=1}^{4} F_i \qquad \vec{u}_2 = L \begin{bmatrix} 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ \mu & -\mu & \mu & -\mu \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}$$

State $(\vec{q}, \dot{\vec{q}})$: pos $\vec{r}$, vel $\vec{v}$, acc $\vec{a}$, jerk $\vec{j}$, snap $\vec{s}$, yaw ψ, yaw dot $\dot{\psi}$, vector of higher yaw $\psi_{..}$, ψ

pos, yaw

1-1 correspondence $\Leftrightarrow$ $\vec{q}, \dot{\vec{q}}, \ddot{\vec{q}}, u_1, \dot{u}_1, \ddot{u}_1, \vec{u}_2$

Eqn of motion

$$u_1 = m(\vec{a}_3 - \vec{g}) \cdot \hat{b}_3 \quad (\vec{a}_3: \text{acc}^n \text{ in } \hat{z}\text{-direction})$$

$$\dot{u}_1 = m j_3$$

$$\ddot{u}_1 = m s_3 + u_1(q^2 + r p)$$

$$p = -\frac{m j_2}{u_1}, \quad q = \frac{m j_1}{u_1}$$

using

What does about mean for us in terms of Trajectory planning?

→ If we know the position and the yaw angle & it's derivatives we can determine the state & the inputs.

Eg: Let's consider the eqns of a planar quadrotor in which it's easier to visualize this mapping between these 2 vectors.

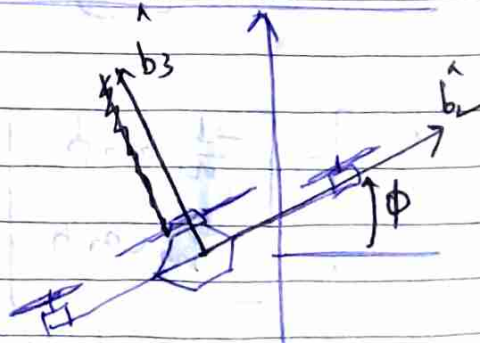
Planar Quadrotor

Inputs u_1, u_2

$$u_1 = F_2 + F_4$$

$$u_2 = (F_2 - F_4) L$$

$$q = \begin{bmatrix} y \\ z \\ \phi \end{bmatrix}$$



Equation of motion

$$\begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

this equivalence between the two vectors is due to a property called differential flatness

Differential Flatness

Date __/__/____

→ All state variables and the inputs can be written as smooth functions of flat outputs and their derivatives

Planar Quadrotor

Diffeomorphism

$$\begin{bmatrix} y \\ z \\ \dot{y} \\ \dot{z} \\ \ddot{y} \\ \ddot{z} \\ \dddot{y} \\ \dddot{z} \\ y^{(iv)} \\ z^{(iv)} \end{bmatrix} \longrightarrow \begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ u_1 \\ \dot{u}_1 \\ \ddot{u}_1 \\ u_2 \end{bmatrix} \longleftarrow$$

The flat outputs for a quadrotor are simply the y and z position and the other way around. If we know the state vector, input u_1, its first & second derivatives and the input u_2 we can recover the flat outputs and their derivatives.

The flat outputs and their derivatives can be written as a function of the state, the inputs & their derivatives -

Flat outputs	State	Input
$\begin{bmatrix} y \\ z \end{bmatrix}$	$\begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \end{bmatrix}$	$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$

$$\begin{bmatrix} \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m}\sin\phi \\ \frac{1}{m}\cos\phi \end{bmatrix} u_1$$

$$\begin{bmatrix} y^{(iii)} \\ z^{(iii)} \end{bmatrix} = \frac{1}{m}\begin{bmatrix} -u_1\dot{\phi}\cos\phi - \dot{u}_1\sin\phi \\ -u_1\dot{\phi}\sin\phi + \dot{u}_1\cos\phi \end{bmatrix}$$

$$\begin{bmatrix} y^{(iv)} \\ z^{(iv)} \end{bmatrix} = \frac{1}{m}\begin{bmatrix} -\sin\phi & -\frac{u_1}{I_{xx}}\cos\phi \\ \cos\phi & -\frac{u_1}{I_{xx}}\sin\phi \end{bmatrix}\begin{bmatrix} \ddot{u}_1 \\ u_2 \end{bmatrix} + \frac{1}{m}\begin{bmatrix} -2\dot{u}_1\dot{\phi}\cos\phi + u_1\dot{\phi}^2\sin\phi \\ -2\dot{u}_1\dot{\phi}\sin\phi - u_1\dot{\phi}^2\cos\phi \end{bmatrix}$$

#learnthesmarterway

$$u_1 = m(\ddot{y}^2 + \ddot{z}^2)$$

$$\phi = \text{atan2}\left(\frac{-m\ddot{y}}{u_1}, \frac{m\ddot{z}}{u_1}\right)$$

$$\dot{u}_1 = m(-y^{(iii)}\sin\phi + z^{(iii)}\cos\phi)$$

$$\dot{\phi} = -\frac{m}{u_2}\left(y^{(iii)}\cos\phi + z^{(iii)}\sin\phi\right)$$

Date: __/__/____

$\dot{u}_1 = \ldots$

$\dot{\phi} = \ldots$

$u_2 = \ldots$

Planar Quadrotor

$$\begin{bmatrix} y \\ z \\ \dot{y} \\ \dot{z} \\ \ddot{y} \\ \ddot{z} \\ y^{(iv)} \\ z^{(iv)} \end{bmatrix} \longrightarrow \begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ u_1 \\ \dot{u}_1 \\ \ddot{u}_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ u_1 \\ \dot{u}_1 \\ \ddot{u}_1 \\ u_2 \end{bmatrix} \longrightarrow \begin{bmatrix} y \\ z \\ \dot{y} \\ \dot{z} \\ \ddot{y} \\ \ddot{z} \\ y^{(iv)} \\ z^{(iv)} \end{bmatrix}$$

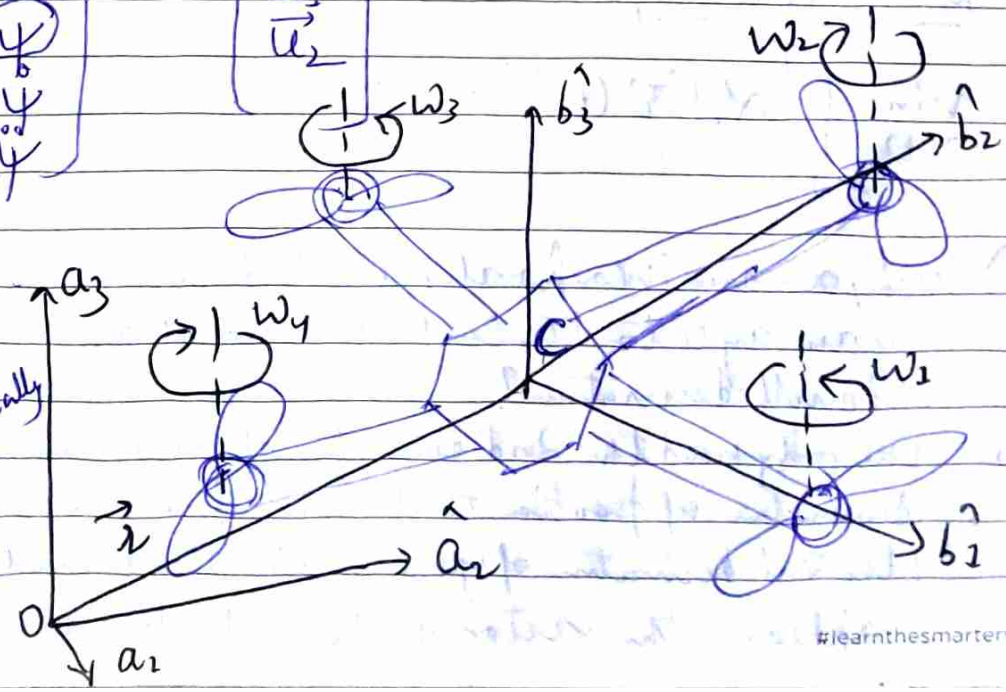
this relationship between two vectors is called diffeomorphism. The term diffeomorphism refers to a smooth map between these two vectors.

3D Quadrotor

$$\begin{bmatrix} \vec{r} \\ \vec{v} \\ \vec{a} \\ \dot{\vec{a}} \\ \vec{s} \\ \psi \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} \longleftrightarrow \begin{bmatrix} \vec{q} \\ \dot{\vec{q}} \\ u_1 \\ \dot{u}_1 \\ \ddot{u}_1 \\ \vec{u}_2 \end{bmatrix}$$

If we know the flat variables & their derivatives, we can recover the state variables & the inputs.

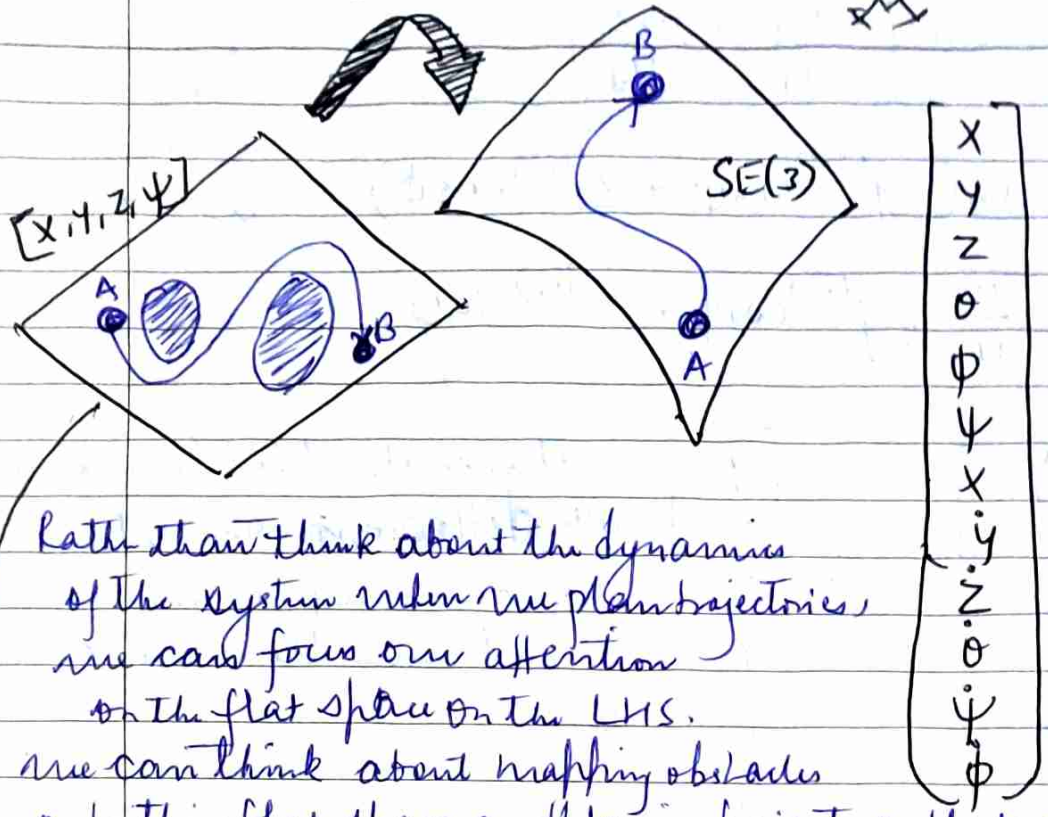
the 3-D quadrotor is differentially flat.



#learnthesmarterway

Trajectory Planning

Date: __/__/____



$$\begin{bmatrix} x \\ y \\ z \\ \theta \\ \phi \\ \psi \\ x \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\phi} \end{bmatrix}$$

Rather than think about the dynamics of the system when we plan trajectories, we can focus our attention on the flat space on the LHS.

We can think about mapping obstacles onto this flat space and planning trajectories that are safe, provided these trajectories are smooth. If they are smooth (since these are the flat outputs) we are guaranteed to be able to find feasible state vector and inputs corresponding to these trajectories. Once we find smooth trajectories, we prefer to make these trajectories minimum snap trajectories, we can then transform these trajectories into the real space with state vector and inputs. Again we are guaranteed that these state vector & inputs will satisfy the dynamic equations of motion.

Minimum snap trajectory

$$\min_{\sigma(t)} \int_0^T \alpha \|\dddot{\vec{x}}(t)\|^2 + \beta \ddot{\psi}(t)^2 dt$$

(Q) Why are we incorporating the 2nd derivative of the yaw angle to the cost functional and not the fourth derivative?

Ans We only need the 2nd derivative of yaw which the 4th derivative of position to obtain expression for the inputs. (the 2nd derivative of yaw & the 4th derivative of position appear in the vector of flat outputs & derivatives)

If we know how to synthesize smooth curves going through specified waypoints, we can easily transform these curves into dynamically feasible trajectories. Trajectories that a non-linear controller can execute easily.

Date / /

Read The minimum snap traj generation problem is a quadratic programming problem. If we have constraints on the inputs & state variables (for eg, the max thrust or the max velocity), we can add these constraints to our optimization problem.
— daniel-mellinger-thesis (chapter 7)

Control of Multiple Robots

Date: __/__/____

Swarms (how to coordinate actions of teams of robots and how to create artificial swarms?
→ what information must robots communicate with each other?
→ How do they coordinate their actions?
→ How do they cooperate to perform tasks that they individually cannot perform?
→ How do they collaborate with different types of robots?)

Three Organising Principles for Collective Behaviours

(I) Each individual acts independently.
→ there is no leader in the swarm that's telling every individual what to do.

(II) Actions are based on local information.
→ there is no mechanism for sharing information so that every individual has the same global picture.

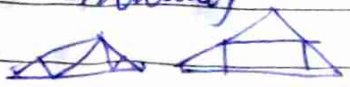
(III) Anonymity in coordination.
→ The individuals are really anonymous. It doesn't matter who an individual is collaborating with or who the individual is coordinating its actions with.

When we build robots, we try to preserve these three overarching principles.

a cooperative task in which robots coordinate the actions to build a 3D structure

transportation and construction — the robots determine who gets to pick up what object and who gets to place them in what position. By coordinating their actions, they will be able to build complex 3-D trusses. (a Truss is an assembly of members such as beams, connected by nodes that creates a rigid structure)

→ In the discussed example, the robots only have info about CAD model for the truss and they are able to determine how to coordinate actions and how to plan their trajs. in a safe manner in order to accomplish the task.



#learnthesmarterway

Complexity

Date: __/__/____

n robots, m obstacles

→ If we look at the state space, the dimensionality of the state space grows with the number of robots.

→ for 1 robot, we might have a state space consisting of its configuration and its velocity.

→ for n robots, we essentially have n copies of the state space.

So, the Complexity of whatever operation we need to perform, whatever calculations we need to conduct, will grow as n increases. The number of potential obstacles each robot must need to consider also increases.

Summary

n robots, m obstacles

→ Dimensionality of the state space ↑ linearly with n $O(n)$

→ Number of potential interaction with neighbors ↑ as n^2

→ Number of potential interactions with obstacles ↑ as mn.

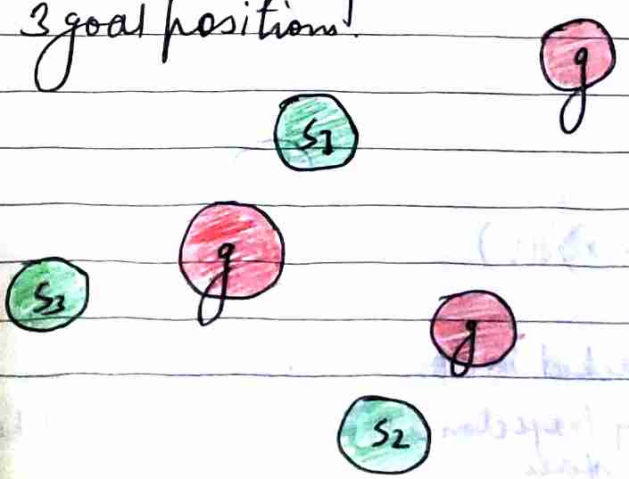
Thus, the Complexity of the task of coordinating motions and avoiding obstacles goes as $(mn + n^2)$

Who gets to do What? ⟶ Task assignment problem

→ If the robots have a number of goal positions to choose from, the problem of which robot gets to go to what goal position can be formulated as a common problem and the Complexity of this grows as $(n!)$.

i.e., Number of assignments of robots to goal positions $O(n!)$

Eg. 3 robots, starting at positions S_1, S_2 and S_3 and there are 3 goal positions.

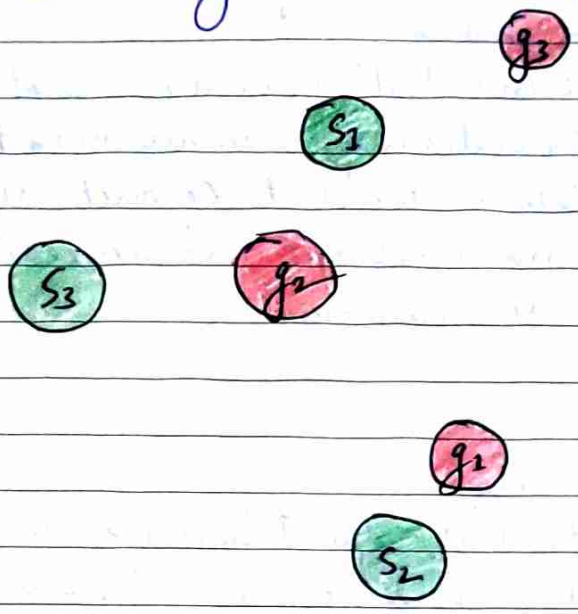


The goal positions are not labeled.

#learnthesmarterway

Date: __/__/____

① The first decision the robots have to make is, who goes to which goal position?
This is a labeled problem where the robots have decided the labels g_1, g_2 and g_3 designating the goal positions g_1 to robot 1, g_2 to robot 2 and g_3 to robot 3.



→ Mathematically, this can be described by an assignment matrix filled with 0s and 1s.

factorial → **Assignment of robots to goals**

$$\phi_{i,j} = \begin{cases} 1 & \text{if robot } i \text{ is assigned to goal } j \\ 0 & \text{otherwise (if robot } i \text{ is not assigned the goal destination } j) \end{cases}$$

both of them need to be done concurrently

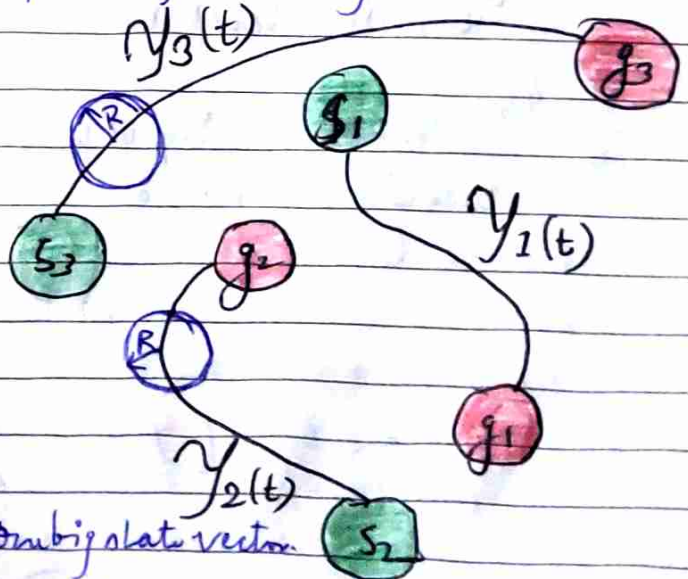
② the second sub problem is planning the trajectories.

exponential → **Planning trajectories**

state vector

$$X(t) = \begin{bmatrix} X_1(t) \\ X_2(t) \\ \cdots \\ X_N(t) \end{bmatrix}$$

$$\gamma(t) : [t_0, t_f] \rightarrow X(t)$$



→ all state vectors are stacked into one big state vector.
The problem of planning trajectories that are safe must be formulated in this joint state space.

→ The safety constraint is expressed by below inequality. Assuming the robot are circular or spherical, we require robots i & j, to be separated by at least two times the radius.

Safety

$$\left[\inf_{i \neq j \in I,\, t \in [t_0, t_f]} \|x_i(t) - x_j(t)\| - 2R\right] > 0$$

So, for all pairs of robots i & j, above inequality must hold. Further above inequality must hold at all instance of time. And eventually we would like these trajectories to be optimal.

→ Once again, we assume that there is a functional that describes our measure of optimality and we are looking for trajectories that minimize the value of this functional.

Optimality

$$y^*(t) = \underset{y(t)}{\operatorname{argmin}} \int_{t_0}^{t_f} L(y(t))\, dt$$

→ The key problem is to find the right assignment and to plan the trajectories. And both of these need to be done concurrently. One grows as $n!$, the other grows exponentially with n.

Four key ideas that can be used to solve this problem

(I) Concurrent assignment of goals and trajectories
→ The problem of assigning goals to robots and the problem of determining the optimal trajectories.
Combining these two individually difficult problems.

(II) Leader-follower networks
→ to formulate interactions between robots and their neighbours

(III) Anonymity
→ we embrace the paradigm of anonymity where robots can be exchanged between each other so that the specific identities of the robots don't matter.

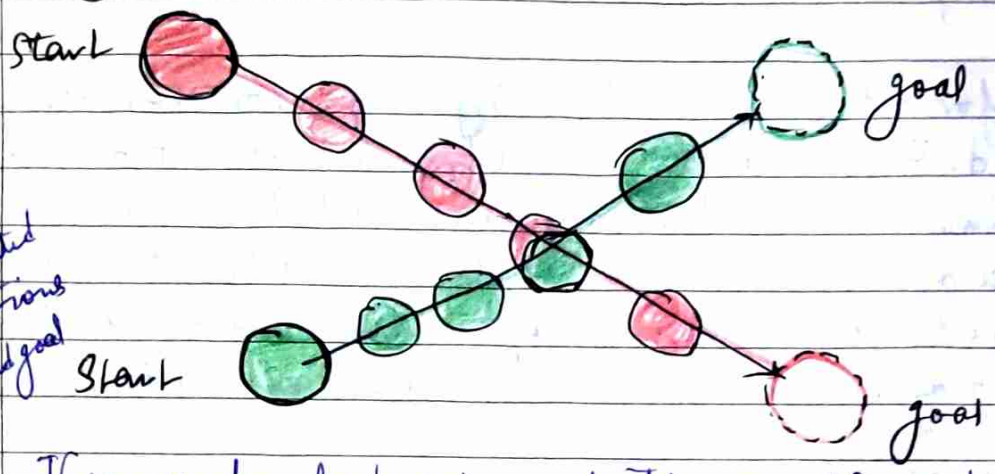
(IV) Sharing information

Date: __/__/____

→ enabling the robots to make decisions based only on local information.

Assignment of Goals and Collision free Trajectories

we have 2 robots, a red robots a blue robot starting off from designated start positions with designated goal positions.

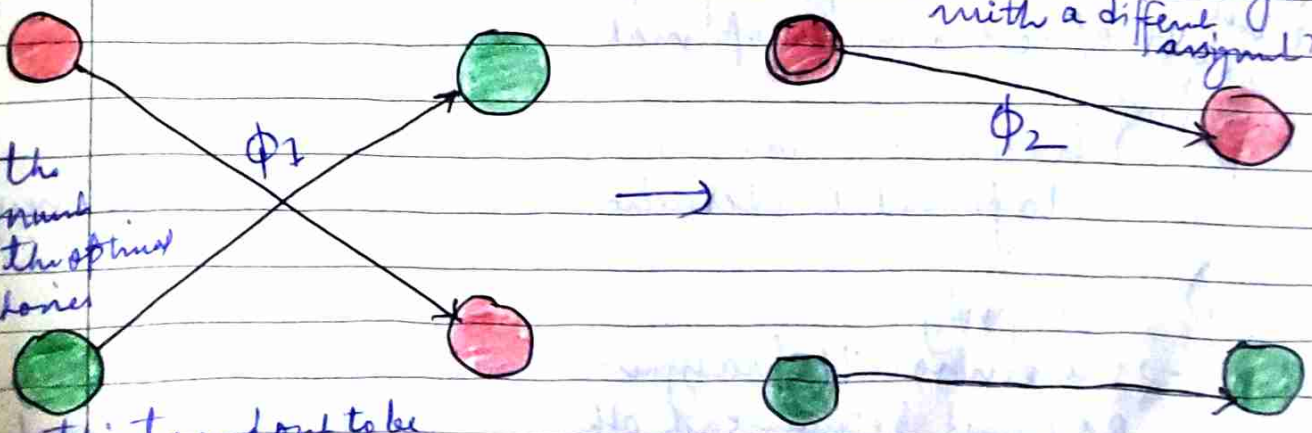


If we make a bad assignment, then even if we planned the best trajectory possible (in this case let's assume the St. line is the best trajectory), we might get a collision and that's because these straight line trajectories intersect.

→ Q. If we were instead to think about the problem of finding the assignment and the outcome of the trajectories concurrently. we might come up with a different way of approaching the problem.

Concurrent Assignment and Planning of Trajectories: CBRT

What if we instead decided to go with a different assignment?



Here is the assignment with the optimal trajectories

This turned out to be harmful because robots come in each other's way.

#learnthesmarterway

So, we have 2 possible assignments to consider,

ϕ_1 → when trajectories intersect, resulting in collisions.

ϕ_2 → when trajectories don't intersect.

→ Instead of first finding the assignment and then solving for the optimal trajectories, we solve these problems concurrently.

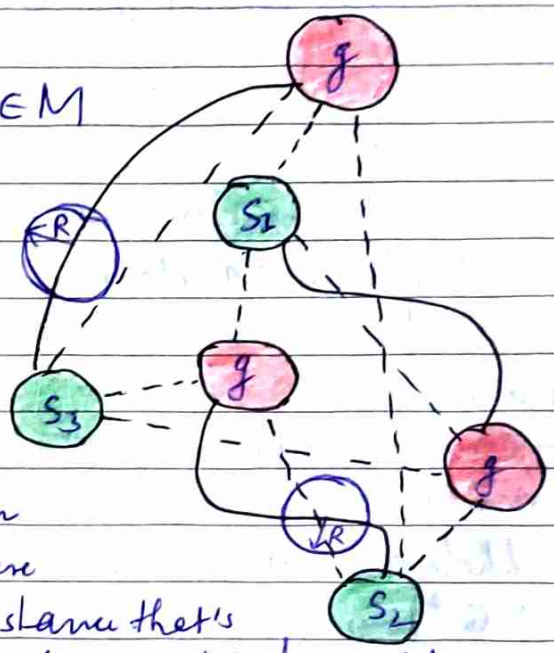
Concurrent Assignment and Planning

Assumption

$$\|s_i - g_j\| > 2R\sqrt{2} \quad \forall i \in N, j \in M$$

(the start and the goal positions for the robots are not too close to each other.

In other words, if we take any start position s_i for the i^{th} robot and any goal position g_j for the j^{th} robot, we want these positions to be seperated by a distance that's proportionate to R, the const. of proportionality here happens to be $2\sqrt{2}$)



→ with the above assumption, we explore all possible assignments of robots to goal position (denoted by dotted lines) → for 3 robots, 3 goals 6 such assignments

Theorem

→ Assignments and trajectories that minimize the sum of square of distance

$$\underset{\phi, Y(t)}{\text{minimize}} \int_{t_0}^{t_f} \dot{X}(t)^T \dot{X}(t)\, dt$$

will be safe (no collisions)

$$\|X_i(t) - X_j(t)\| > 2R$$

→ Now we find the best possible ~~assignment~~ trajectory but this time we are minimizing the functional over all possible trajectory and all possible assignments.

→ the resulting sol^n will be safe and we are guaranteed not to have collisions

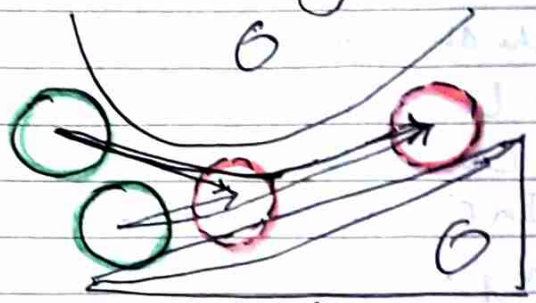
#learnthesmarterway

This is a simple result, but the best result allows us to solve large scale problems without considering the collision avoidance problem. This happens to reduce the complexity of the planning problem. (of course, it's impractical to explore all possible assignment for large number)

Hungarian Algorithm → allows us to search through the set of assignments, provided we know the optimal trajectories for each assignment.

(a) **Compute optimal trajectories for each assignment**

In the new algorithm, we first find the set of optimal trajectories from each start position to all goal positions.

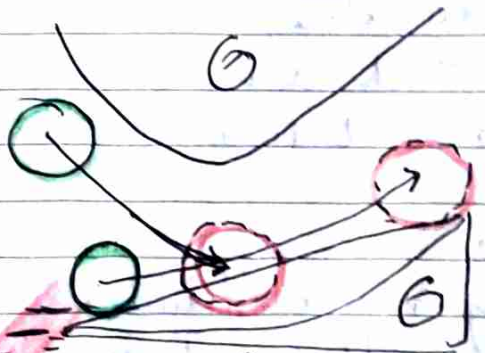


$O(n^2)$

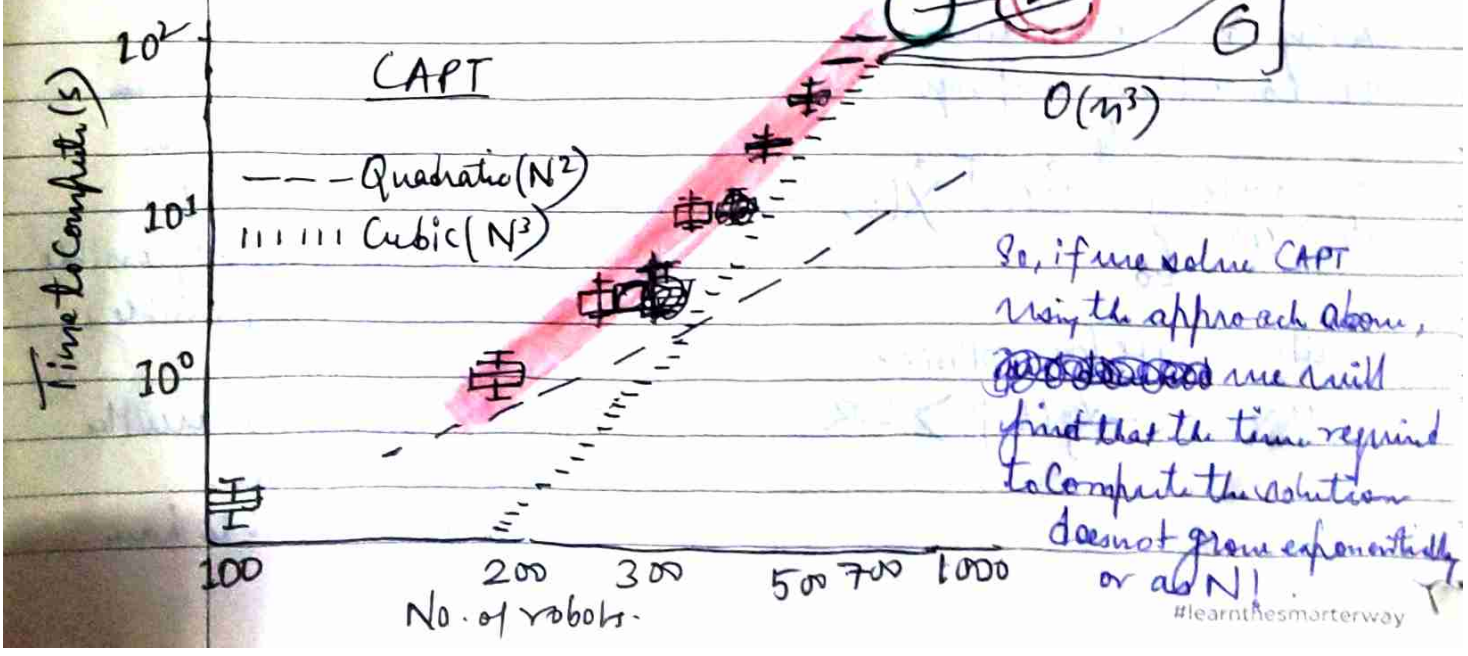
Note – each computation is for a single robot. For n robots and n goals, there are n^2 such computations.

(b) **Find the assignment with the lowest cost**

The Hungarian algorithm which is an order of n^3 algorithm, gives us the solution for the best assignment.



$O(n^3)$



So, if we solve CAPT using the approach above, we will find that the time required to compute the solution doesn't grow exponentially or as N!

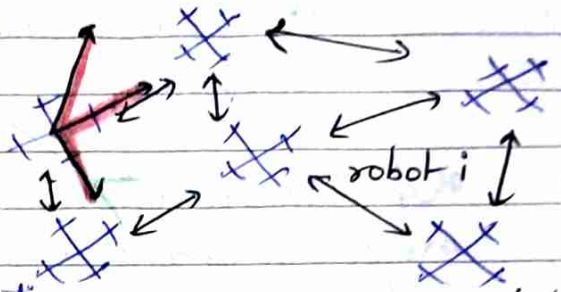
#learnthesmarterway

In the plotting on the previous page, there are two curves, two st. lines that show the cubic growth of computational time and the quadratic growth of computational time as a function of the number of robots.
The complexity is somewhere between quadratic and cubic.

Leader-Follower Networks

→ Each robot monitors the distance with respect to its neighbour.

$$S_{i,j}(t) = X_j(t) - X_i(t)$$



→ More precisely, it looks at the separation as a vector. So, if we take the robot in the middle (Let it be 'i'), for every neighbour j, it monitors the difference between X_i and X_j.
→ There are many neighbors and there are many robots. Each robot needs to regulate its position so that the separations from its neighbours is at a desired value. This is done using the control algorithm.
→ Now in addition to this from a practical standpoint we may have to have a designated leaders in the team that actually use absolute information and not just relative information w.r.t. their neighbors.
→ we may have one leader, or a second leader or many leaders.
→ But without having at least one leader we will not have an absolute sense of where the team is? where the formation is?

Anonymity

→ the robots have been asked to maintain a circular formation so they try to maintain the desired separation in order to form a circular formation.
→ As robots are placed into the formation or removed from the formation, their presence is detected by the neighbors. The neighbors then adjust their position to ensure that the relative separation is maintained. They are agnostic to who their neighbors are, they only react to the presence of the neighbors and again the physical position of the neighbors relative to where they are.

Q The key benefits of using the paradigm of anonymity are:
→ the swarm is robust to failures of individuals in the team
→ robots can avoid easily collisions with the neighbors
→ robots in the swarm can be interchangeable
→ Complexity of coordinating robots is greatly reduced
→ true, because the CAPT algorithm reduces the complexity to $O(n^3)$

Q The shape of the formation change and also the position of the group changes as a function of time. Robots start off from a rectangular formation, change into an ellipse, flatten out into a st. line before taking into an ellipse and then into a circular shape.

Applications

(1) Robot first responders — paper – Quadcloud: A rapid Response Force with Quadrotor teams.
→ they are able to react to, for eg. a 911 call & they do that by positioning themselves around the building from which they think that the call has originated.
→ responds to active shooters without telling them where to go. The minute a gunshot occurs, we can imagine the robots taking up positions outside the building where they think the source of the gunshot is. Robots coordinate themselves to take up position
→ They are using concurrent assignment & planning algorithm to take up positions. They use this paradigm with anonymity. They really don't care the identities.
They really don't care about the identities of their neighbors and they are able to take up positions to get information about the building.

(11) Enabling cooperation — where the robots actually cannot perform the task on their own. Imagine payloads that are too heavy for individual robots to carry. So, robots cooperate to pick up heavy objects & then transport them.

#learnthesmarterway

How do we get robots Cooperate in tasks like them?

Date: __/__/____

Q. In cooperative manipulation or transportation tasks, which of the following are true:

(I) the methods of traj. control presented earlier may not be adequate. ⇒ for eg. depending on the compliance of the gripper, it maybe necessary to explicitly control the forces applied by each robot to successfully manipulate the payload.

(II) Robots in the swarm cannot be interchangeable.
⇒ It is beneficial to allow them to be interchangeable to increase robustness

(III) there is a minimum number of robots required to perform the task.

(IV) We cannot rely on the framework of leader follower methods.
⇒ the robots are usually commanded to maintain desired separation with their neighbours.

(III) **Active Mapping**

Autonomously create 3D map of an unknown environment with ground and aerial robots collaboration.

How do robots like them collaborate to autonomously create 3D map?

Control Policy → Reduce uncertainty of map by maximizing information gain
→ we achieve collaboration by thinking about this task in terms of information a
→ the robots are driven by the common goal of reducing uncertainty about the map.
→ At every step, they think about the information that they gain by positioning their sensors to get new information.

$$\chi_T^* = \underset{\chi_T \in X}{\operatorname{argmax}} \frac{I_{CS}[m; z_T | \chi_T]}{D(\chi_T)}$$

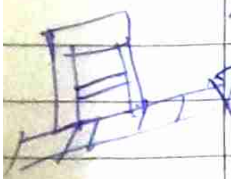
→ eqn that describes the essential strategy being used by the robots

Paths

In this eqn, they're trying to determine their paths. χ_T designates the path of the robot and they choose the path from the a set of available paths

#learnthesmarterway

The available paths are those that conform to the environment and are safe and they are paths that respect the dynamics of the robot.



$$x_T^* = \underset{x_T \in X}{\arg\max} \frac{I_{cs}[m; z_T | x_T]}{D(x_T)}$$

Map → Information

Duration. measurement

→ In the numerator, we see the quantity which represents information gain. So, they are trying to find the path that maximizes the information gain.

→ The information gain is described by 2 random variable -

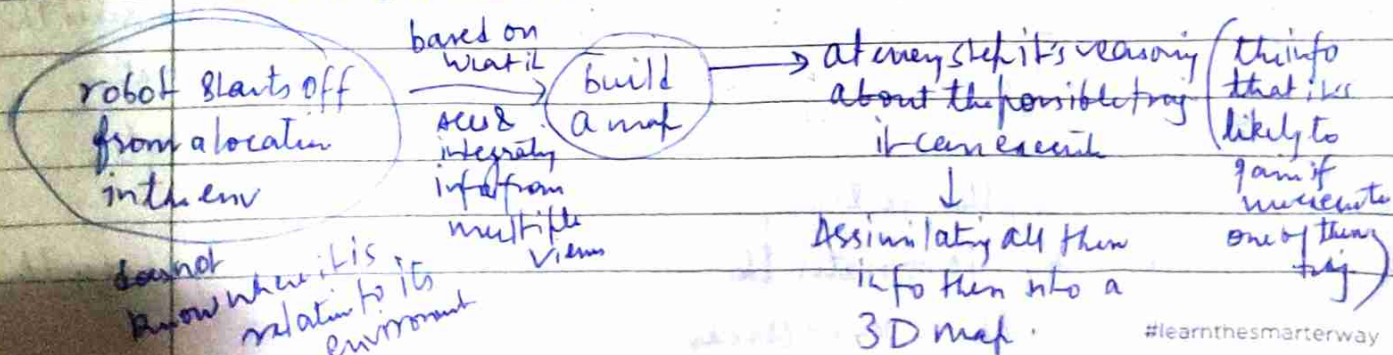
(i) the first random variable is a map. the map is a complex object but it consists of many different elements, each representing feature in the environment. It's a random variable because everyone of these features is a random variable. There is uncertainty associated with everyone of the variables.

(ii) The second random variable that we want to consider is the measurement. Everytime the robot moves, it guarantees measurement.

What the robot really needs to do is to maximize the information gained through its motion so that the measurements yield maximum information about the map. In fact, these measurement should reduce the uncertainty in the map.

Duration → Eventually, we want to minimize the time taken to maximize the information gained.

Thus, in the numerator the duration of the trajectory is also considered.



Search & Rescue

Paper – Collaborative mapping of an earthquake damaged building via ground/aerial robots.

↳ An aerial & a ground robot collaboratively to build a map of a multi-story building that has been damaged by the earthquake.

→ In the process, they share information & they are able to build a complete 3D map.

→ Swarms of robots can build a map like this very quickly & really change the way we think about emergency response.