Lecture 4.3

Control of Multiple Robots

Here's an experiment studying how to use teams of robots in emergency response:

Imagine there's a 911 call or a natural disaster has occurred and you want eyes and ears on the scene before human first-responders or law-enforcement officials arrive. We can get robots with downward-facing cameras and microphones at the scene, monitoring the situation before the humans arrive at the scene.

We're interested in determining how to coordinate the 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? And how to they collaborate with different types of robots?

Certainly nature has many examples in which individuals coordinate, cooperate and collaborate to perform tasks that they individually can not accomplish. The video showed examples of desert ants cooperating to carry an object back to their nest; half-a-million to a million starlings flocking off the coast of Denmark, coordinating to form incredibly complex patterns in the sky; and army ants creating artificial bridges to escape a flooded region and get to dryer lands.

In all these examples, individuals coordinate, collaborate and cooperate. We want to learn from these examples and build artificial swarms.

Here are three organizing principles for collective behaviour.

1. Each individual acts independently. There is no leader in the swarm that's telling every individual what to do.
2. All actions are based only on local information. Again, there is no mechanism for sharing information so that every individual has the same global picture.

3. Anonymity in coordination. There's a paradigm of anonymity in play here. The individuals are anonymous. It really doesn't matter which individual collaborates or coordinates with which.

When we build robots, we try to preserve these three overarching principles. The video showed an example of robots cooperating and coordinating to build a three-dimensional structure. In this particular experiment, the robots determined on-the-fly, which would pick up what object (each object was a truss-like element), and then who gets to place them in what position. By coordinating their actions, they're able to build complex three-dimensional trusses.

In this example, the robots only have information about the CAD model for the truss. They're able to determine how to coordinate their actions and plan their trajectories in a safe manner in order to accomplish the task. Let's look at the mathematics of this problem.

This is a complex task. Imagine we have n robots & m obstacles. The dimensionality of the state space grows with the number of robots. For a single 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, and the calculations we need to carry out, will grow as n increases.

The number of potential obstacles each robot must consider also increases. The number of potential interactions increases as n^2. In addition, each robot must worry about the obstacles in the environment. If there are m obstacles, then there are m possible interactions. In a team of n robots, the number of such interactions $m \times n$. Thus, the complexity of the task of coordinating motions and avoiding obstacles, goes as $(mn+n)^2$.

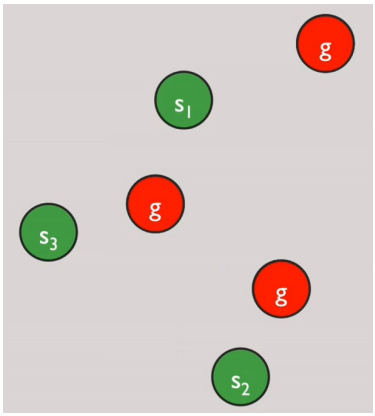
Finally, there's a question of, who does what? This is the so-called ***task assignment problem***. If the robots have a number of goal positions to choose from, the problem of which robot goes to what goal position can be formulated as a combinatorial problem, and the complexity of this grows as n!.

Complexity

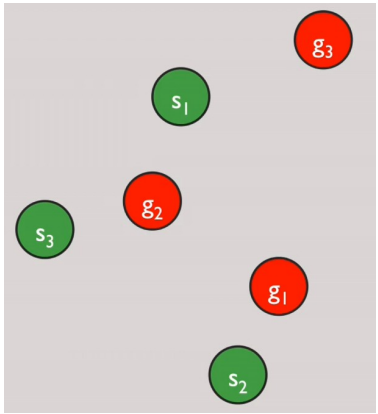
n robots, *m* obstacles

- Dimensionality of the state space increases linearly with *n* — $O(n)$
- Number of potential interactions with neighbors increases as n^2 — $O(mn+n^2)$
- Number of potential interactions with obstacles increases as *mn*
- Number of assignments of robots to goal positions — $O(n!)$

Here is an illustration of that problem. In this case, we only have three robots, starting at positions s_1, s_2, and s_3, and there are three goal positions. The goal positions are not labelled.



The first decision that the robots have to make is: who goes to which goal position? This is the labelled problem once the robots have decided:



The labels g_1, g_2 and g_3 designate the goal position: 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. The entry, sub i, j of this matrix is equal to 1 if robot i is assigned the goal destination j, and 0 if the robot i is not assigned the goal destination j.

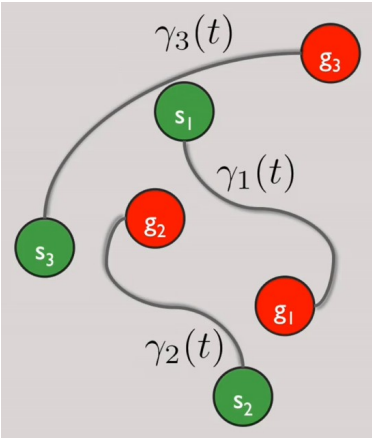
$$\phi_{i,j} = \begin{cases} 1 \text{ if robot i is assigned to goal j} \\ 0 \text{ otherwise} \end{cases}$$

The second sub-problem is planning the trajectories. Here, all the state vectors are stacked into one big state vector:

$$X(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \dots \\ x_N(t) \end{bmatrix}$$

$$\gamma(t) : [t_0, t_f] \rightarrow X9t)$$

The problem of planning trajectories that are safe must be formulated in this joint state space.



As stated earlier, the dimensionality of the state space grows as n. The safety constraint is expressed by this inequality:

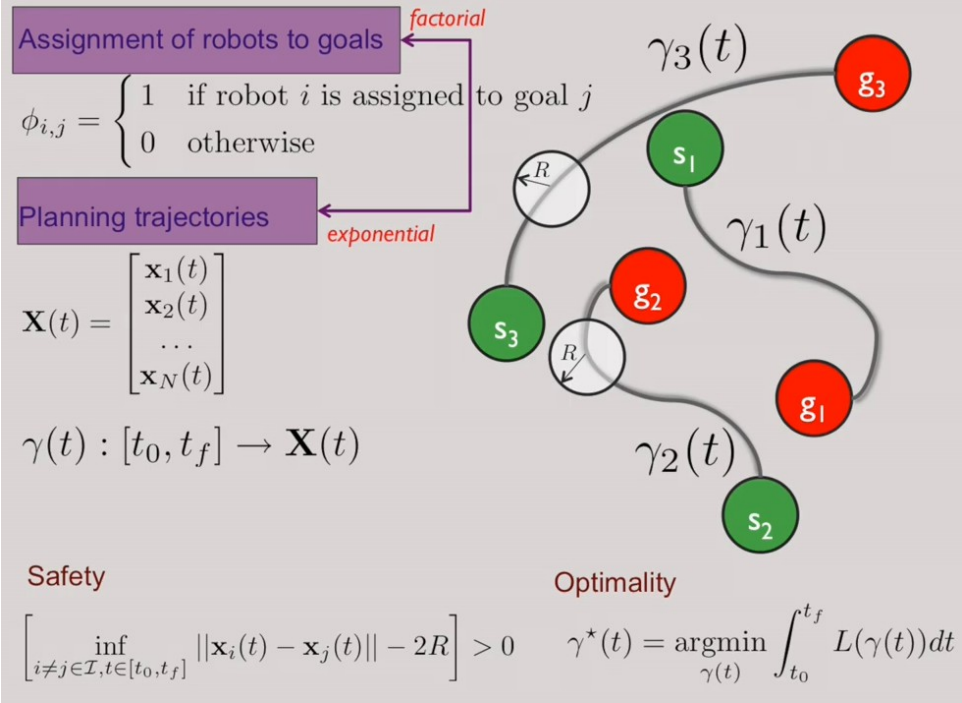
$$\left[\inf_{i \neq j \in I, t \in [t_0, t_f]} \left\| x_i(t) - x_j(t) \right\| - 2R \right] > 0$$

Assuming the robots are circular or spherical, we require robots i and j, to be separated by at least 2× the radius. So for all pairs of robots i and j, this inequality must hold. Further, this inequality must hold at all times.

Finally, as before, we want these trajectories to be optimal. Once again, we assume that there's a functional that can describe our measure of optimality, and we're looking for trajectories that minimize the value of this functional.

$$\gamma^*(t) = \arg\min_{\gamma(t)} \int_{t_0}^{t_f} L(\gamma(t))dt$$

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

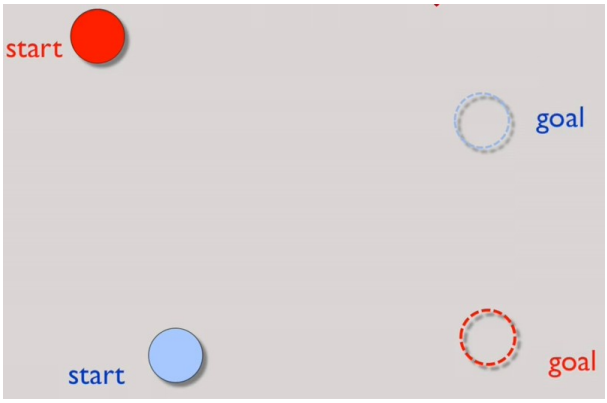


There are four key ideas that we use to solve this problem:

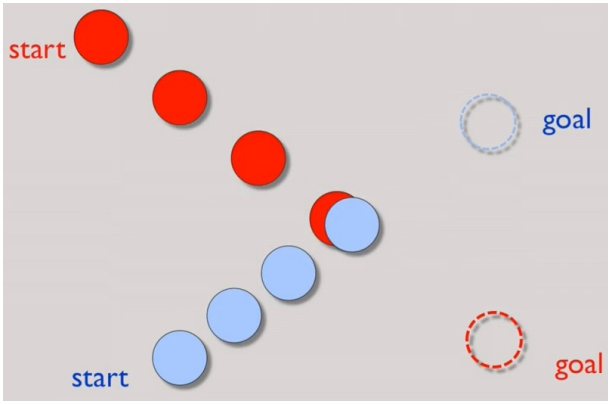
1. Concurrent assignment of goals and determining optimal trajectories.
2. Leader-follower networks to formulate interactions between robots and their neighbours.
3. Anonymity. Robots can be exchanged with each other so that the specific identities of robots don't matter.
4. Sharing Information. Enabling robots to make decisions based only on local information.

Let's see how each of these ideas works out in practice.

First, we want to be able to assign goals and plan collision free trajectories. Let's consider a very simple example, in which we have two robots, a red robot and a blue robot starting off from designated start positions with designated goal positions.

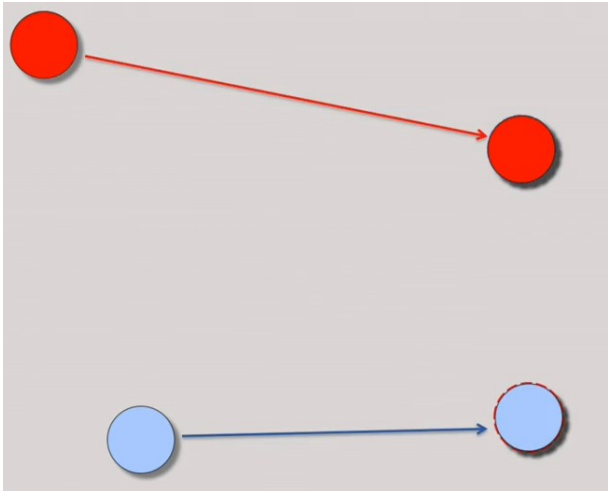


If we make a bad assignment, even if we planned the best trajectory possible (in this case we'll assume the straight-line is the best trajectory), we might get a collision, and that's because these straight-line trajectories intersect:



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. This assignment combined with the optimal trajectories, turned out to be harmful because the robots got in each other's way.

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



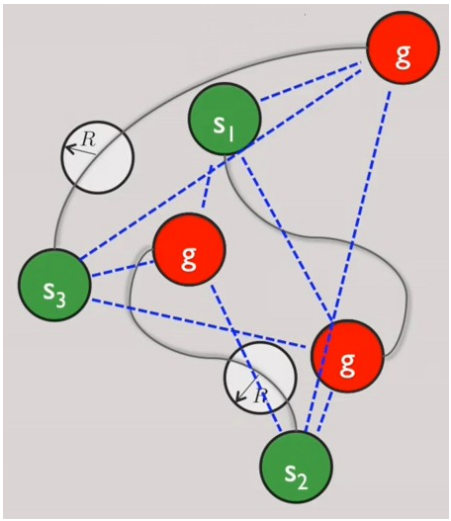
Now the optimal, straight-line trajectory, is one in which neither robot gets in the other's way. So, we have two possible assignments to consider, $\phi 1$, where the trajectories intersect, resulting in collisions, and $\phi 2$, where the trajectories don't

intersect. Instead of first choosing the assignment and then solving for the optimal trajectory, we solve both these problems concurrently.

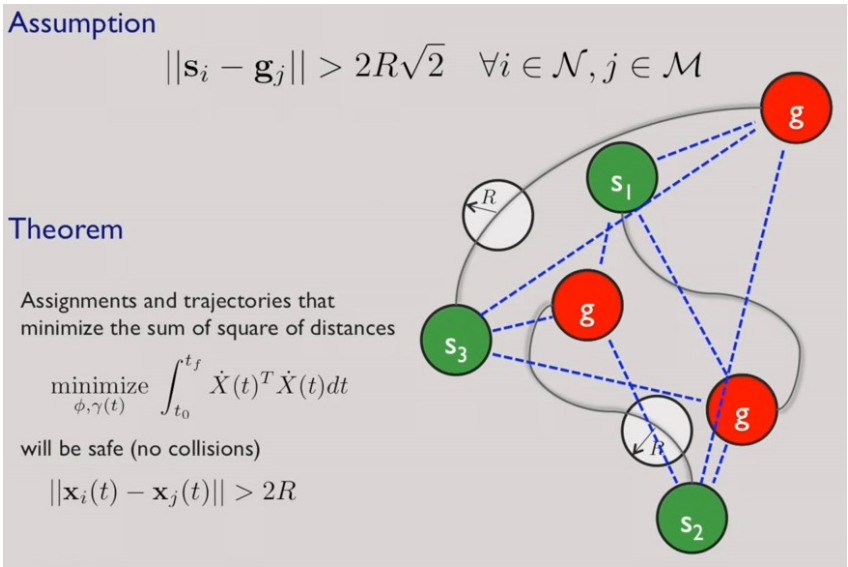
Before solving these problems concurrently, we assume that the start & goal positions for the robots are not too close to each other.

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

In other words, for 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 separated by a distance that's proportional to R. The constant of proportionality here happens to be $2\sqrt{2}$. With this assumption, we explore all possible assignments of robots to goal positions. For three robots and three goals, there are six such assignments. Now we find the best possible trajectory, this time we're minimizing the functional over all possible trajectory and all possible assignments:



In practice, we go through each of these assignments, there are six of them, and for each assignment we find the best possible trajectory:

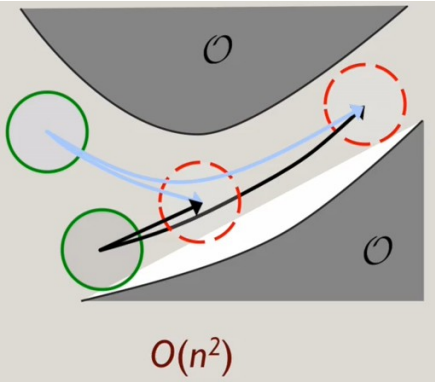


It turns that out if we solve this problem, then the resulting solution will be safe and we're guaranteed not to have collisions.

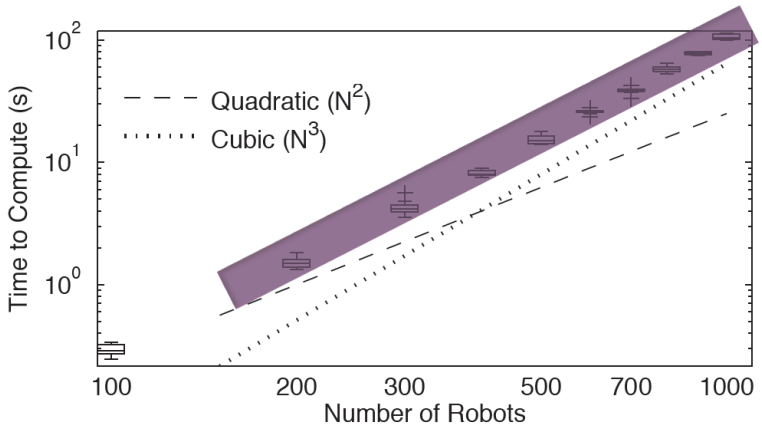
This is a simple result, but it allows us to solve large-scale problems without having to consider the collision avoidance problem. This significantly reduces the complexity of the planning problem.

Of course, it's not practical to explore every possible assignment for large numbers. Fortunately, there's a well known algorithm called the ***Hungarian Algorithm*** that allows us to search through the set of assignments provided we know the 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. Note that each computation is for a single robot:

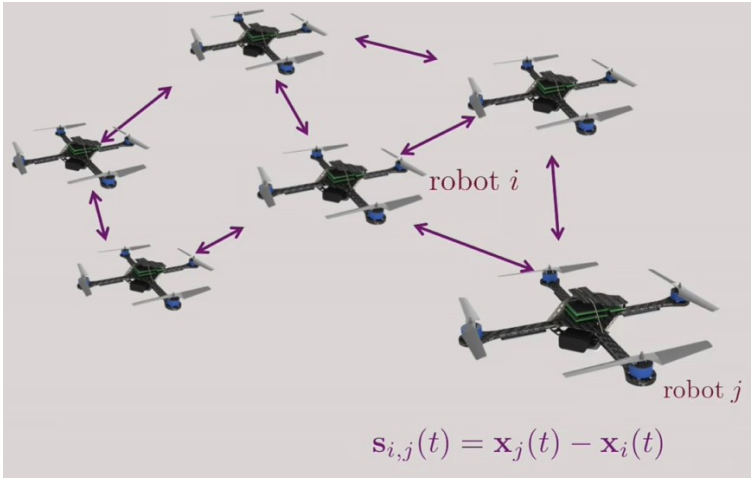


For n robots and n goals, there are n^2 such computations. We then find the assignment with the lowest total cost. The Hungarian algorithm, which is an $O(n^3)$ algorithm, gives us the solution for the best assignment. So if we solve CAPT, using this approach, we find that the time required to compute the solution does not grow exponentially or as n!:



In fact there are two curves here: two straight lines that show the cubic-growth of computational time and the quadratic-growth of computational time as a function of the number of robots. Over hundreds of experiments, the results that we've obtained suggested the actual complexity lies somewhere between quadratic and cubic.

Now, let's move on to the second key idea which is the concept of incorporating leader-follower networks. The idea is simple: each robot monitors the distance with respect to its neighbours. More precisely, it looks at the separation as a vector. So if you take the robot in the middle, call it i, for every neighbour j, it monitors the difference between x_i and x_j:



Of course there are many robots, and there are many neighbours. Each robot needs to regulate its position so that the separations from its neighbours are at a desired value. This is done using the control-algorithms that we described earlier.

Now, in practice, in addition to this we may have to have designated leaders in the team that actually use absolute information and not just relative information with respect to their neighbours. We might have one leader, or a second leader, or even many leaders, but if we don't have at least one leader we will not have an absolute sense of where the team is.

The video showed an experiment where an external user manipulated a single, designated leader. Since all the robots monitored relative separations, they were able to regulate that and follow the leader as the leader was being manipulated by the human user.

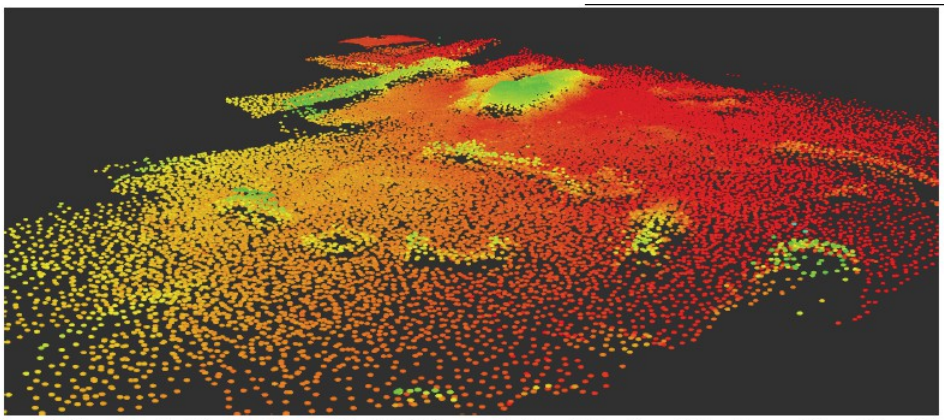
The third key idea is this paradigm of anonymity. The video showed an experiment where the robots had been asked to maintain a circular formation by maintaining a desired separation. As robots were added to, or removed from, the formation, their presence or absence was detected by neighbours who then adjusted their positions to ensure that relative separations were maintained. The robots are agnostic to who their neighbours are; they only react to the presence or absence of neighbours, and the physical position of the neighbours relative to where they are.

If we combine some of these ideas now, we can require that the shape be specified and changed as a function of time. The video, showed a series of experiments in which the shape of the formation changed, and the position of the group changed as a function of time. This is potentially useful if we think about practical tasks such as creating robot first responders.

So here you see a team of outdoor robots. They're able to react to, for example, a 911 call, and they do that by positioning themselves around the building from which they think that the 911 call has originated:

Robots like this could also respond to active shooters without humans having to tell them where to go. The minute a gunshot occurred, the robots could take up positions outside the building where they think the source of the gunshot is.

The video showed an example of this behaviour with robots using the concurrent assignment and planning algorithm together with the paradigm of anonymity. They really don't care about specific identities. The video from cameras mounted on the robots was assimilated into a mosaic, and a three-dimensional reconstruction of the scene was developed by the robots:



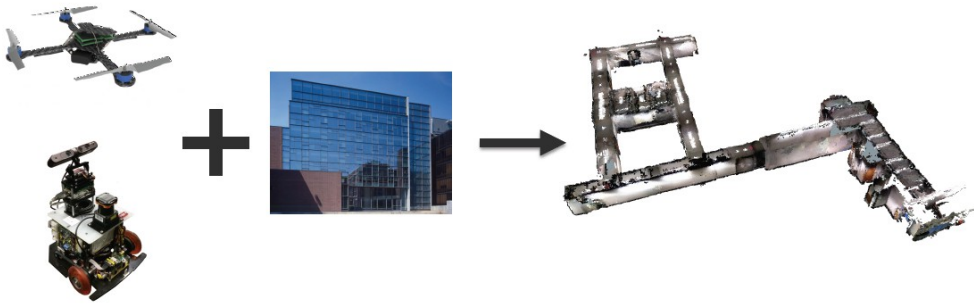
Kartik Mohta, Matthew Turpin, Alex Kushleyev, Daniel Mellinger, Nathan Michael, and Vijay Kumar, "QuadCloud: A Rapid Response Force with Quadrotor Teams," *Int. Symp. on Experimental Robotics* (ISER), 2014.

So far, we've been talking about tasks which required the robots to coordinate with each other to perform a task more efficiently. In each of these examples, individual robots might have been able to perform the task on their own. The presence of multiple robots just resulted in greater efficiency and better coordination guarantees that the efficiency improves.

What about situations where the robots actually cannot perform the task on their own? Imagine payloads that are too heavy for a single robot to carry. Here is an example of a situation where robots are cooperating to pick up heavy objects and then transport them:

How do we get robots to cooperate in tasks like these?

Here's another example where a flying robot must collaborate with a ground robot in order to build a map in the interior of a building:



How do robots like these collaborate to autonomously create three dimensional maps?

Remember, an aerial robot can view the scene from a vantage point that is very different to that of a small ground robot. Likewise, the ground robot can position its sensors underneath tables to get viewpoints that the aerial robot cannot. We achieve collaboration by thinking about this task in terms of information.

The robots are driven by the common goal of reducing uncertainty about the map. At every step, they think about the information that they can gain by positioning their sensors to get new information. Here's the equation that describes the essential strategy being used by the robots:

$$x_\tau^* = \arg\max_{x_\tau \in \chi} \frac{I_{CS}\left[m; z_\tau | x_\tau\right]}{D(x_\tau)}$$

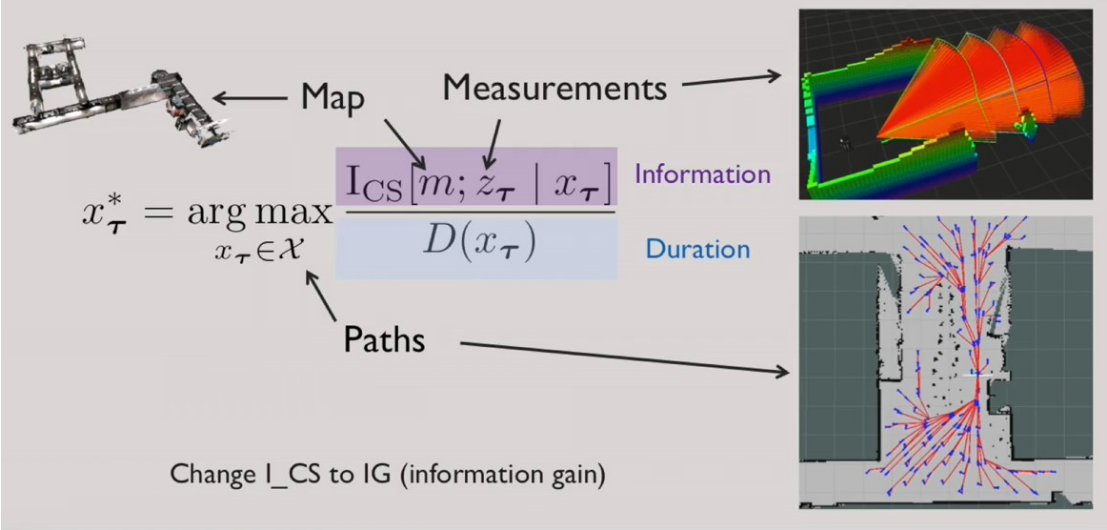
In this equation, they're trying to determine their paths. x_τ designates the path of the robot, and they choose the path from a set of available paths. The available paths are those that conform to the environment and are safe, and they are paths that respect the dynamics of the robot.

In the numerator you see a quantity which represents information gain. So they're trying to find the path that maximizes the information gain. The information gain is described by two random variables.

The first random variable is a map. The map is a complex object, but it consists of many different elements, each representing features in the environment. It's a random variable because every one of these features is a random variable. There's an uncertainty associated with every one of the variables.

The second random variable that we have to consider is the measurement. As the robot moves, it generates measurements. And 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 measurements should reduce the uncertainty in the map.

Finally, we want to minimize the time taken to maximize the information gained. Thus, in the numerator, the duration of the trajectory is also considered. On the bottom-right in the figure below, we can see the robot reasoning about the multiple trajectories it has at its disposal:

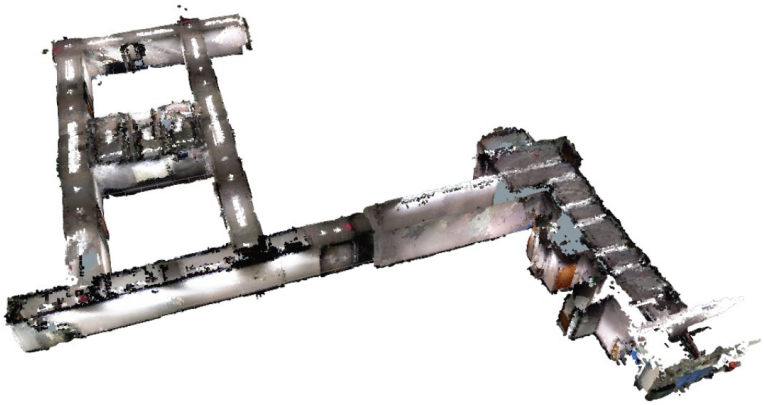


The video showed a small ground being used for active mapping in the lab. The robot started from a location in the environment, without any prior knowledge of where it is relative to its environment. Based on what it sees, and integrating information from multiple views, the robot built a map. At each step it reasoned about the possible trajectories it could execute and the information that it was likely to gain if it executed one of these trajectories. All this information was assimilated into a three dimensional map.

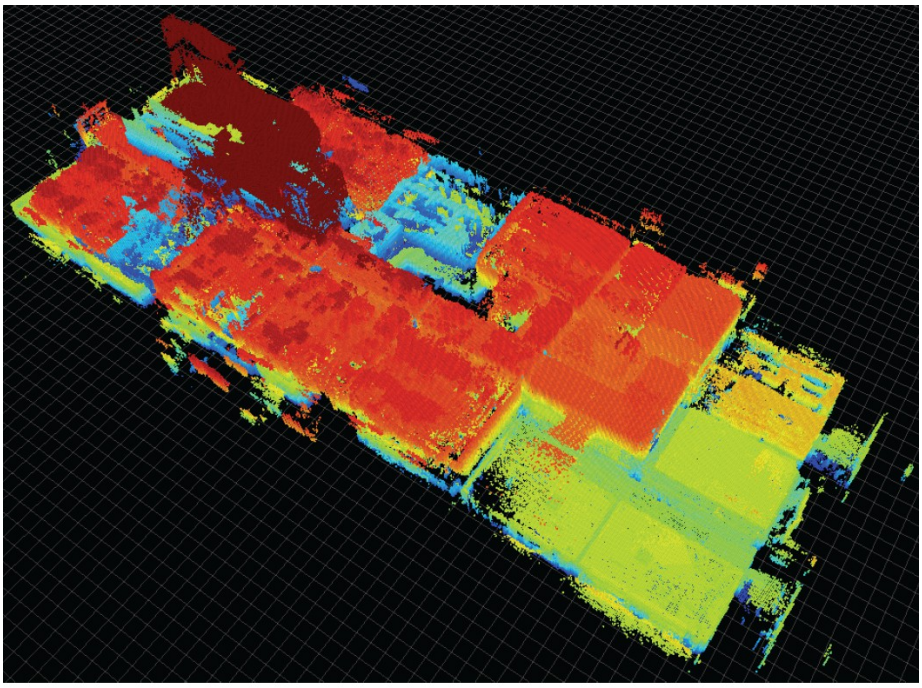
Of course, using this strategy with a flying robot will yield different information. The video also showed the same thing being done by a flying robot. The trajectories available to it are different from those available to a ground robot. The kinds of information that it can gather are also different. However, when it collaborates with the ground robot, they ensure that they can cover the entire environment and build a complete three dimensional map.

The video next showed an aerial and a ground robot collaborating to build a map of a multi-storey building that had been damaged by an earthquake. The robots travelled as a team, but whenever the ground robot got obstructed by collapsed debris, the aerial robot could take off and traverse areas that were inaccessible to the ground robot. In the process, they shared information, and were able to build a complete three-dimensional map.

Maps like these can be built autonomously without any intervention by a human operator. This is a map of a multi-room, single-storey building:



Here's another map, this time of a multi-storey building:



3 floors of a 9 story building

Swarms of robots can build maps like these very quickly and have the potential to really change the way we think about emergency response.