Combining static and dynamic optimization approaches for path planning, with collision avoidance

Dr. Clément Coïc[1] Marco Masannek[1,2]

[1]Siemens Healthineers AG, Germany, {clement.coic, marco.masannek}@siemens-healthineers.com
[2]Lab for mobile Robotics, Nuremberg Institute of Technology, Germany, {marco.masannek}@th-nuernberg.de

Abstract

In response to staff shortages in hospitals, healthcare providers aim at increasingly automating their systems. Defining automated system paths – with collisions avoidance – is a critical step towards automation. In this paper, three different approaches for path planning are investigated: a static, a dynamic and a hybrid approach. The hybrid approach, that sequentially combines part of the static and dynamic approaches, results in improved accuracy against the static approach and improved performance against the dynamic approach.
Keywords: Path planning, C-space, Dynamic Optimization, Hybrid approach

Figure 1. Project AURORA: Target operation room environment (left) and real robot (right)

1 Introduction

The staff shortage in the healthcare industry is a bottleneck when it comes to patient touchpoints. Healthcare system automation can partially solve this issue. Should automated systems reduce the workload of qualified healthcare professionals, these would be available for interventions - where they can add value, save life, care for the patients or plan procedures, to name but a few. There are different tasks currently performed by qualified by healthcare professionals that could be taken over by automation. First, existing systems could perform in a more automated manner. For the sake of simplicity, this paper focuses on robotic systems evolving freely on the hospital floor - though the entire discussion could be extended to any type of system, e.g. ceiling systems. A typical example could be to provide qualified voice command to, for example, a CIARTIC Move (Siemens Healthineers AG 2025) - such as "Take a 3D scan of the patient right foot." - and the Ciartic Move would autonomously evolve in the room until the foot is at the isocenter of the scan, would ask for permission to scan and proceed. In a second step, new robotic systems could be developed to perform actions that are not part of today's offering. For example, the research project Autonomous Robotic Operation Room Assistance (AURORA) (MITI Research Group 2024) aims to assist in operations by performing tasks required by circulators, which focus on actions in the periphery of the operating table, e.g. equipment preparation (see Figure 1).

In both above examples, one key capability the system shall have is determining the path it shall take to safely evolve in its environment. Additionally, the system shall sense its moving environment and adapt the path based on the potential new or moving obstacles along its path. This paper focuses on solving that path planning problem with fixed obstacles. This can be justified by the fact that the authors already mitigated part of the risk associated with moving obstacles by presenting a novel approach relying on a semantic understanding of the environment, and path adaptation can be done with reinitiating the path planning under the new constraints. The remaining of the paper will present three different approaches for solving the path planning problem. In Section 2, a static approach is discussed. This method relies on discretizing the environment and determining the C-space of the robotic system to define the feasible paths. Section 3 tackles the problem from the perspective of the dynamic motion of the robotic system: what is the optimum path under the constraints of the robotic system kinematic. Section 4 discusses a hybrid approach consisting of sequentially applying part of the static and dynamic approaches. Finally, the paper is concluded and perspective to the work are presented.

2 Static Path Planning

In most mobile robotic applications, the problem of finding a collision-free path is approached by first creating a map of the environment using sensors and Simultaneous Localization and Mapping (SLAM) algorithms (Placed et al. 2022). In a second step, the map is then converted into a discretized occupancy grid map, also commonly referred to as cost map, which assigns specific cost values for each

cell depending on the proximity of obstacles. By utilizing these cost values as a basis for node evaluation during graph search algorithms such as A* or RRT, current robotic systems achieve computation times of often less than 100ms even for longer paths.

However, this approach has two major limitations affecting the quality of generated paths, which are the missing consideration of vehicle dynamics and the often-inaccurate representation of valid configurations for non-circular robots, e.g. for rectangular shapes. To overcome the latter, we introduce a method for fast and memory-efficient computation of the collision-free configuration space (C-space) for any shape robots.

2.1 C-Space for non-circular robots

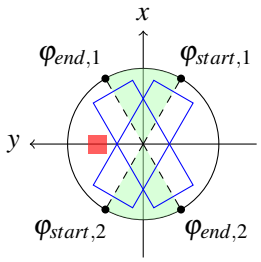
To determine valid poses within a given environment, most approaches start by creating a Euclidian Signed Distance Field (ESDF) which describes the distance to the nearest obstacle for each x-y-cell in the cost map. For circular robots, these values can already be utilized to perform complete collision evaluation by comparing the distance with their own radius. Cells below this threshold are usually marked as colliding, while distances above the threshold mark free and therefore traversable space. Since the orientation (heading) of a circular robot has no effect on its radius and therefore minimal distance threshold, the classification holds true for all headings φ.

However, rectangular or any shape robots need to consider their actual occupied space for each discrete heading φ_h at a given position $p(x,y)$ to reason about whether or not the robot would collide at a specific pose $P(x,y,\varphi)$. The angular resolution r_φ that is required to ensure valid collision checks in the discretized grid can be determined using the cell resolution r_c and the circumscribing radius of the footprint d_{cr} (furthest point of the shape) and often lies in a range from 1° to 10°. Since this creates a necessity for a three-dimensional grid (third dimension being the heading), generated c-spaces can grow very large for big environments with high resolution requirements.

To overcome this challenge, (Lau, Sprunk, and Burgard 2013) introduced the concept of valid angle intervals (see Figure 2), which describe a set $\mathbf{v_I}$ of collision-free intervals of heading for an arbitrary shaped robot. This method allows for efficient computation of collision checks for any pose $P(x,y,\varphi)$ by simply evaluating if the angle φ intersects with any interval of $\mathbf{v_I}$. Cells with a single interval spanning the full angular range can additionally be marked as completely free to save further memory.

2.2 Example C-Space for a hospital floor

Figure 3 shows an exemplary hospital floor map with hallways, patient rooms, treatment rooms and bathrooms. In order to plan paths for a rectangular robot (e.g. a patient bed or the CIARTIC Move) between any two positions, the complete floor needs to be processed. Brute force calculations of the C-space would result in over 180 million cells when given a map of 1280x720 pixels and a required



$$\mathbf{v_I}(\mathbf{x}|\mathbf{y}) = \begin{bmatrix} (60°, 120°) \\ (240°, 300°) \end{bmatrix}$$

Figure 2. Example angle intervals for a rectangular robot (blue) for a close obstacle (red square)

angular resolution of 1.8° (equals 200 angles per rotation). Utilizing the method described in section 2.1, we are able to reduce the size of the c-space map by over 80% to roughly 35 million cells.

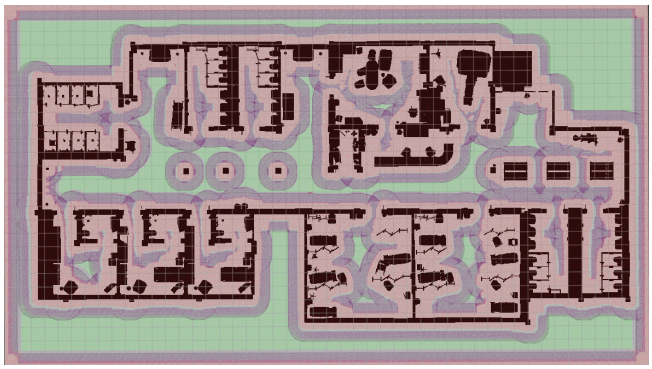


Figure 3. C-Space for a hospital floor and rectangular robot. Black cells mark physical obstacles such as walls, beds or other objects. Green cells are completely free, red cells cause collisions. Purple cells mark cells with valid angle intervals.

2.3 Collision graph from angle intervals

In order to plan a path through the occupancy grid including its angle intervals, a customized graph structure needs to be created that accounts for the presence of multiple angle intervals on the same position $p(x,y)$. We therefore create an adapted neighborhood discovery step during expansion in our graph search (utilizing A*) which discovers all valid intervals of a neighboring cell. Connections are established only when intervals overlap to ensure that a continuous path can be created in the end. Since completely free cells have overlaps with all intervals, connections between them can always be established (see Figure 4).

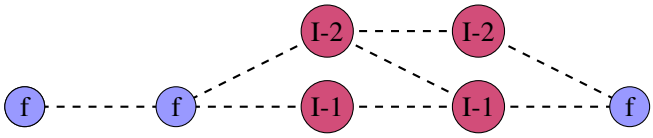


Figure 4. Example edges between free and and orientation interval nodes

DOI
10.3384/ecp21873

2.4 Limitations of grid-based planning

Figure 5 shows a typical path planning problem in hospitals, where a patient bed must dock to a magnetic resonance imaging system in a narrow room.

While our static path planning is able to produce collision-free paths for non-circular robots even through the narrow sections (see Figure 5), the generated trajectories are not smooth and also imply following using an omni-directional movement model (e.g. shopping cart).

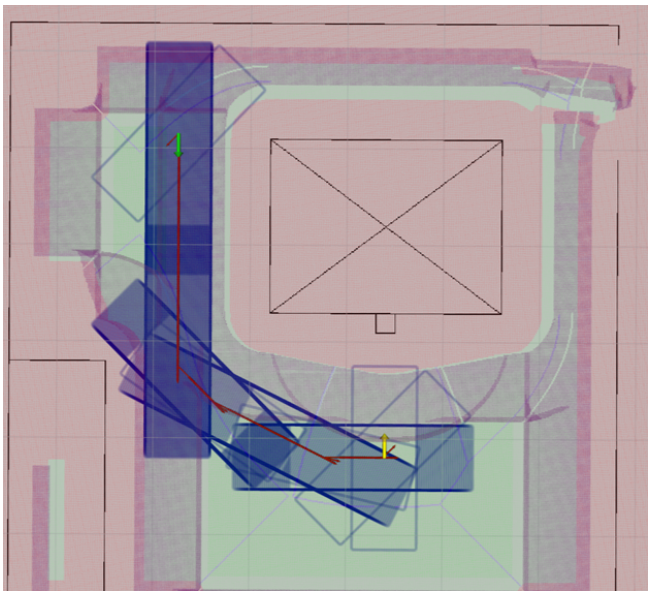


Figure 5. Planned path using our custom collision graph. The arrows mark the start (green) and goal (yellow) pose for the robot; the blue phantoms depict the footprint along the path.

3 Dynamic Optimization

The static approach to path planning proved to be very efficient in finding a solution. However, it does not guarantee that a robotic system, with constrains on its kinematic motion, could actually achieve the found path. This might not be an issue for some robotic systems - holonomic, such as the CIARTIC Move, which drive system allows for any type of motion on the 2D plan (i.e. rotation on the spot, translation in all directions (even lateral) and combination of both motions). However, many other types of drive systems are non-holonomic - i.e. they have constrained kinematics that should be accounted for when planning for their path.

Dynamic optimization (Åkesson 2008) is leveraged here to solved for the optimum path of a behavioral model of the robotic system, under its environment constraints. Four steps have been followed to implement this solution. First, a simplified behavioral model of the robotic system is developed. Then, a simplified model of the environment is introduced together with the collision assessment with the robot. In a third step, the dynamic optimization function is implemented. Finally, the resulting path is run in a forward simulation of a more detailed behavioral model. These four steps are detailed below, after a quick discussion on the need for these two fidelities of behavioral models.

3.1 Behavioral Model Purposes

Models are developed with a purpose in mind. Here, our aim is to easily assess the behavior of different drivetrain set ups for our robotic systems and their impact on the trajectory they can achieve. We collected the two different set of requirements associated with this purpose.

1. Path planning: the behavioral model shall represent the system dynamics, while being as simple as possible, to allow for a fast optimization. As directional derivatives are leveraged by the dynamic optimization solver, it is of high importance that the behavioral model is C2-continuous, i.e. continuous, derivable and the derivative is continuous too.

2. Behavioral path following: the behavioral model shall allow for modeling the different drivetrain in more details to assess how they perform when following a specified path.

In order to meet both requirements, the choice was made to have two separate models - one for each purpose. The behavioral model dedicated to path planning is a point-heading motion model, with potential constraints on the trajectories it can follow. The behavioral model dedicated to studying the path following response is a multi-body model with detailed kinematics of the drivetrain.

3.2 Point-heading Optimization Model

A point-heading motion model, as the names indicates, consists in modeling the system as a point with a heading, and some constraints on its motion. In addition, the robot envelope around the point is also known so that we can easily check for collision of the real system - and not only the point. At this stage, two different cases are made possible:

1. The robotic system is holonomic, i.e. it can evolve freely in all directions and motion types (like the CIARTIC Move).

2. The robotic system is non-holonomic, i.e. it can only continuously change the path curvature with respect to its reference frame.

The former case does not constrain the path with respect to the robot heading. For example, a path forming a 90° angle with the heading corresponds to a sideway translation. The latter means that the rate of change (derivative) of the path heading cannot be higher than the point steering capability. In other words, the path and point headings shall align. These two cases can easily be modeled by adding a constraint on the path to heading angle that should be 0 in case of non-holonomic robots.

Listing 1. Point-heading Optimization Model

```
model PointHeading
 // Cartesian coordinates
 Modelica.Units.SI.Position x "Global x
     position";
 Modelica.Units.SI.Position y "Global y
     position";
 Modelica.Units.SI.Angle phi(start=phistart,
     fixed=true) "Global heading";
 // Change to local coordinates
 //...
 parameter Boolean isHolonomic = true "false
     , if heading should be fixed =0";
 // Defining robot body
 // ...
equation
 // motion ...
end PointHeading;
```

The resulting Modelica model shown in Listing 1 is very simple and includes some additional variables for changing the reference frame from global to local and for defining the body of the robotic system around the point. It remains C2-continuous and suitable for a fast dynamic optimization application.

3.3 Simplified Environment Model

The environment where the robot evolves is a bounded rectangle. Internally, it is composed of the floor set-up - including walls, defining the rooms and corridors - and all types of objects and persons that could be on the way - e.g. an MRI or CT machine, a bed with patient, or even a plant. From the perspective of the robot, these are all seen as obstacles. Therefore, there is no need to differentiate them in this approach, and it was decided rather to allow for modeling a couple of shapes - specifically rectangles and circles - as footprint on the floor. As discussed in section 2, circles are efficient as there is no need to model their orientation - the Euclidian distance is the same for any angle. Rectangles require their orientation to be modeled. For this purpose, specific classes have been developed - Circle, Rectangle -, that contain their dimensions and take the position of the object in motion to check for collision. As many obstacles as desired can be introduced in the environment and a combination of them can represent more complex shapes. For example, a square room with an open-door access can be represented with 4 rectangle obstacles. (Alternatively, should a scenario happen exclusively within a rectangular room, it is possible and more efficient to define the bounds of the room as constraints to the robot coordinates in the optimization problem.)

3.4 Optimization Problem

Now the point-heading model and its environment are defined, the next step is to look at the path the robot shall follow. Note again that the path and the trajectories of the robot model (x, y, phi) are not necessarily the same: specifically, for non-holonomic robots, the heading of the robot and of the path may not be aligned. Therefore, a heading variable for the path itself, phi_p, is introduced. Additionally, we define a velocity along the path, v_p, its projections on the x and y axes, v_x and v_y, and its time integral gives us the length of the path d_p.

The cost function of our optimization model is a combination of the path length and the rate of change in path curvature. The goal is to minimize the distance while avoiding abrupt changes in curvature. Obviously, this cost function can easily be adapted based on the needs and would lead to different optimum paths.

The inputs to optimize are the heading derivatives of the path and of the bed. Note that because the velocity is a set, these headings fully define the trajectories. (Alternatively, the velocity could be considered an input.) However, the headings derivatives are selected here as these are needed for the cost function (no abrupt changes in heading) and because the headings are easily obtained by further integrations.

To fully define the optimization problem, the following constraints are added:

- The robot remains within the bounds of the environment.
- Potential alignment of headings for non-holonomic robots (see end of subsection 3.1).
- Checking that the distance to obstacle remains positive.
- Ensuring that the initial and final target positions are met.

The optimization problem is written in the Optimica language, allowing us to extend the existing Modelica behavioral and environment models, similarly to (Coïc, Budinger, and Delbecq 2022).

Listing 2. Pseudo Point-heading Optimization Model

```
optimization PointHeading
 objective = d_p(finalTime) + cost(finalTime
     ),
 finalTime(
  free=true,
   min=1,
   max=10,
   start=5)

 // Floor or Rooms bounds
 parameter Modelica.Units.SI.Position xMin
     =0;
 parameter Modelica.Units.SI.Position yMin
     =0;
 parameter Modelica.Units.SI.Position xMax
     =10;
 parameter Modelica.Units.SI.Position yMax
     =10;

 extends PointHeading(
  // modifiers to pass floor bounds as min,
      max
 );
```


DOI
10.3384/ecp21873

```
parameter Integer n_rectangles=5;
Rectangle[n_rectangles] rectangles(
 //modifers
 );


// Start and end point positions-headings
// ...

// Path variables
Modelica.Units.SI.Angle phi_p "Path heading
    ";
Modelica.Units.SI.Length d_p(start=0,fixed=
    true) "Path length";
Modelica.Units.SI.Velocity v_p=1 "Velocity
    along path";
parameter Modelica.Units.SI.Angle max_phi_p
    =if isHolonomic then Modelica.Constants
    .pi else 0;

// Inputs for trajectory optimization
input Modelica.Units.SI.AngularVelocity
    der_phi "Point heading derivative";
input Modelica.Units.SI.AngularVelocity
    der_phi_p "Path heading derivative";

equation
 der(phi)    = der_phi;
 der(phi_p)  = der_phi_p;
 v_p         = der(d_p);
 cost        = d_phi^2 + d_p^2;
 // ...

constraint
 phi_p <=  max_phi_p;
 phi_p >= -max_phi_p;
 // ...
 x(finalTime) = x_end;
 y(finalTime) = y_end;
 phi(finalTime) = phi_end;
end PointHeading;
```

Figure 6 is an example of a rectangular holonomic robot (in grey) entering a challenging room. The blue rectangles represent walls with small tolerances. The room dimensions are above the minimum recommendations, yet the door is placed with an angle, which limits the robot in its movements. This simulation proved that the non-holonomic robot was not able to find a path in this challenging environment (only by few centimeters), yet the holonomic could manage.

The path planning was implemented as web application within Modelon Impact - as in (Coïc, Andreasson, et al. 2020) - to easily generate scenarios. The resulting paths are stored as csv artifact that can easily be imported into the workspace. This results very handy as the next step is to validate that our point-heading model simplification is not too simplistic. This is verified by simulating a higher fidelity (multibody) path following robotic system.

3.5 Multibody Drive Model

For the purpose of accurately modeling the kinematics of the robotic system, it is enough to model its drive system,

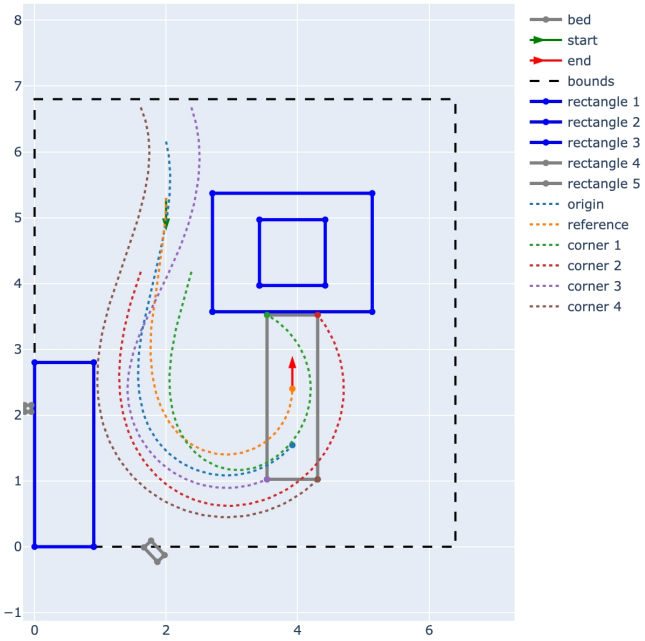


Figure 6. Optimum path planning example

and the rest as point-mass located in the center of gravity - as shown in Figure 7. Each wheel module - see Figure 8 - is a submodel that allows for representing different types of wheels: steered, driven, free, with a caster trail, etc. The tyre model is based on (Pacejka 2012). To each wheel is connected a controller that computes their commands in steering and driving - should these be activated. This makes a very flexible model that allows for investigating a variety of different robotic drive system.

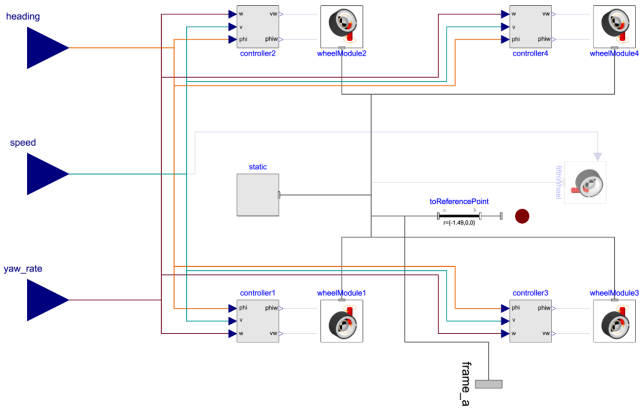


Figure 7. Multibody Model of the Robotic System Drive

As an example of holonomic robot, the four lateral wheels can be active with two caster wheels and two wheels that are both steered and driven. A robot with a fifth wheel in the middle that is locked in steering and driven while one of the other four wheels would be steered only would lead to a non-holonomic configuration - as the robot could only turn in forward motion and around the fifth wheel point of contact.

Resimulating the optimum path from Figure 6 - of the

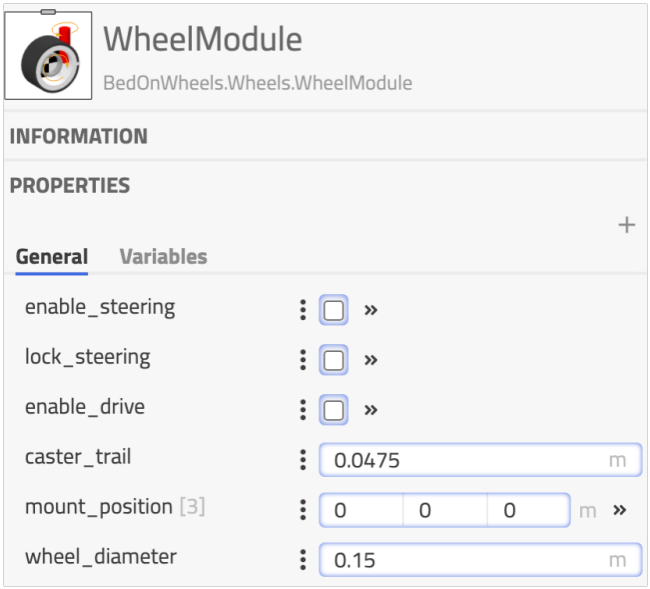


Figure 8. Fully configurable wheel module

reference point, in orange - with a holonomic drive robot, it is observed that the followed path is nearly identical to the optimum one (see Figure 9). And the small divergence could potentially be due to the path following controller. For this case, it is possible to conclude that the point-heading model is a reasonable simplification for the optimization problem.

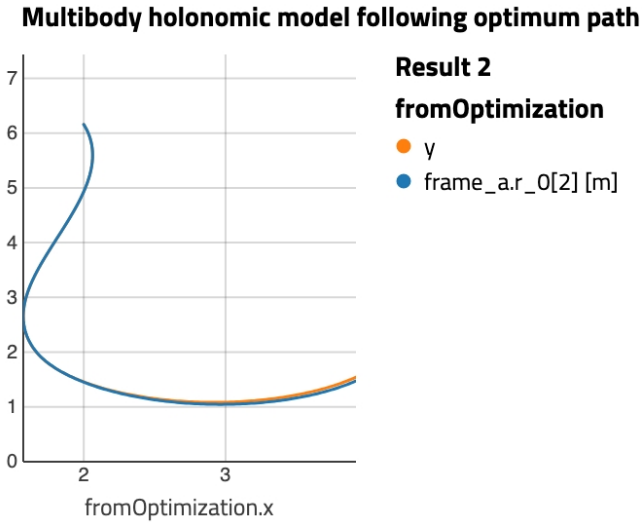


Figure 9. Multibody drive model following optimum path

4 Hybrid Approach

The dynamic optimization solution, combined with multibody path following verification, yields good results. Nevertheless, the static approach proved to be much faster - thanks to the pre-computation of the C-Space. Naturally, the next step in this research is thus to combine both approaches, to get an improved speed with respect to the dynamic optimization solution while conserving its accuracy.

The benefit of defining the environment in a full fledge programming language are multiple. Especially, the environment can be much more complex and programmatically scripted from existing hospital plans. And the C-Space can be easily precomputed with the desired discretization. However, such a discretization would introduce discontinuities, which are not optimization-friendly.

Here, the decision was taken to include the C-Space as part of the cost function (see Figure 10), instead of having it as constraints (as previously in the dynamic optimization implementation). The C-Space is imported into the Optimica model as a table with the three coordinates as inputs - (x, y, phi) - and output an associated cost. The cost is extremely high if the input result in a collision, moderately high if the collision is near and zero if the desired margin is ensured. This cost decrease is made continuous and the table is configured to have a smooth interpolation - making the C-Space optimization friendly.

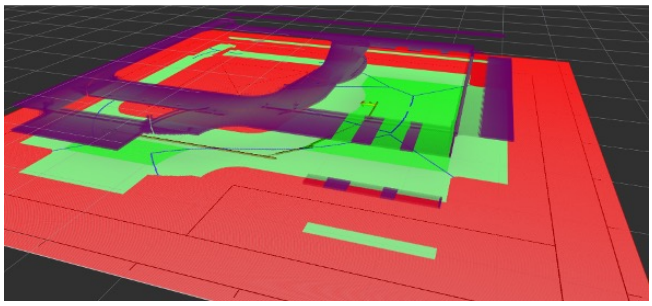


Figure 10. C-Space as a cost "heat map"

This solution also allows for an easy enhancement of the C-Space - e.g. leverage semantic knowledge of the environment to keep a larger distance from people than from fixed objects.

Figure 11 shows two results of the same scenario, with different C-Space costs. The yellow path ensures the desired margin on the persons and finds an optimum that reduces the total length of the path. The green path includes additional costs for the persons and thus leads to a longer path as optimum, with increase safety for the personal. (It is worth noting that a different cost has been applied to the chair - represented in blue -, and thus the green path is nearer to the chair than to the personal.)

As expected, dynamic optimization converges much faster with a good initial trajectory (Delbecq et al. 2021). In this case, it is trivial to provide the one found by running the static approach - which might not satisfy all motion constraints and yet ensure collision-avoidance.

While the initial development of this approach was not trivial, the results are very satisfying in terms both of speed and accuracy. The automation of the environment import, via the C-Space cost table, allows for easy scenario generation and thus increases the representativeness of this solution.

DOI
10.3384/ecp21873

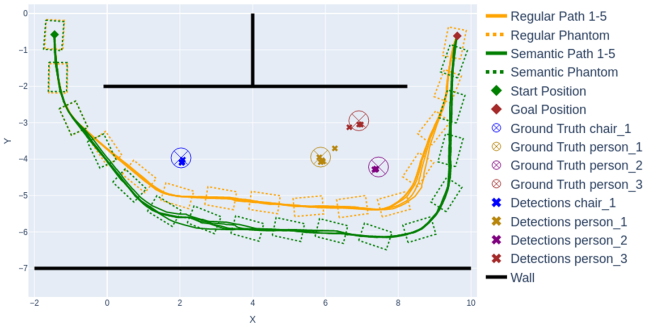


Figure 11. Semantic path planning / social path planning

5 Conclusion and Perspectives

Developing a fast and accurate path planning with collision avoidance is mandatory for automation of robotic systems. This paper presented three different approaches.

1. The static approach proved to be very fast, over a full map of a hospital floor, and yet did not ensure that path found could necessarily be achieved by a non-holonomic robot.

2. The dynamic optimization solution showed achievable trajectories, for simplified scenarios, while requiring much higher convergence times.

3. The hybrid approach showed the best trade-off between accuracy, speed, and flexibility.

As of now, only the static approach has been validated on a real robotic system. Implementing the hybrid solution - relying on dynamic optimization - on the embedded hardware is a next step to this work. This might not be trivial, and a first proof of concept might rely on IoT communication protocols (such as MQTT or OPC UA) to communicate with Modelon Impact cloud software.

Acknowledgements

The first version of the dynamic optimization and multibody models, as well as the associated web application, presented in Section 3 were developed in the scope of a paid project with Modelon AB. The authors would like to thank especially Peter Sundström for his great contributions to this project.

References

Åkesson, Johan (2008). "Optimica—An Extension of Modelica Supporting Dynamic Optimization". In: *6th International Modelica Conference, Bielefeld*.

Coïc, Clément, Johan Andreasson, et al. (2020-10). "Collaborative Development and Simulation of an Aircraft Hydraulic Actuator Model". In: *Proceedings of the Asian Modelica Conference, Tokyo*. DOI: 10.3384/ecp202017467.

Coïc, Clément, Marc Budinger, and Scott Delbecq (2022-10). "Multirotor drone sizing and trajectory optimization within Modelon Impact". In: *Proceedings of the American Modelica Conference, Dallas*. DOI: 10.3384/ECP2118656.

Delbecq, Scott et al. (2021-01). "Trajectory and design optimization of multirotor drones with system simulation". In: *Proceedings of the American Institute of Aeronautics and Astronautics, SciTech*. DOI: 10.2514/6.2021-0211.

Lau, Boris, Christoph Sprunk, and Wolfram Burgard (2013). "Efficient grid-based spatial representations for robot navigation in dynamic environments". In: *Robotics and Autonomous Systems*. DOI: 10.1016/j.robot.2012.08.010.

MITI Research Group, Technical University of Munich (2024). *AURORA: Autonomous Robotic OR Assistance*. https://www.pm.mh.tum.de/en/miti/research/projects/aurora/.

Pacejka, Hans B. (2012). *Tyre and Vehicle Dynamics*. Butterworth-Heinemann. ISBN: 978-0-08-097016-5.

Placed, Julio A. et al. (2022). "A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers". In: DOI: 10.48550/arXiv.2207.00254.

Siemens Healthineers AG (2025). *CIARTIC Move: Robotic C-Arm*. https://www.siemens-healthineers.com/surgical-c-arms-and-navigation/mobile-c-arms/ciartic-move.