Autonomous Mobile Robot Design

Topic: Introduction to Path Planning

Dr. Kostas Alexis (CSE)

How do I plan my motion and actions?

Robots employ various algorithms to manage to plan their actions, trajectories, mission execution steps and more. The ultimate goal is to enable mission-level autonomy.

What is Path Planning?

- **Determining the robot path based on a set of goals and objectives, a set of robot constraints and subject to a representation and map of the environment.**

What is Motion Planning?

Hawk Navigation

Cheetah running

Eagle hunting

Nadia Comaneci, First “10”, 1976

Main Topics of Path Planning

Motion Planning

- Geometric representations and transformations
- The robot configuration space
- Sampling-based motion planning
- Combinatorial motion planning
- Feedback motion planning

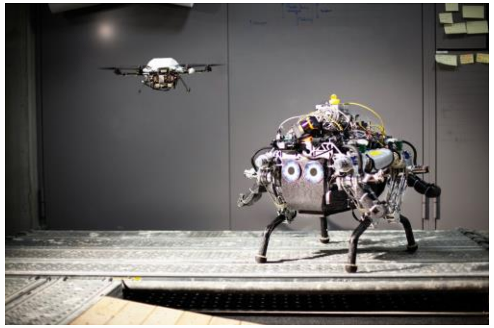
Decision-theoretic planning

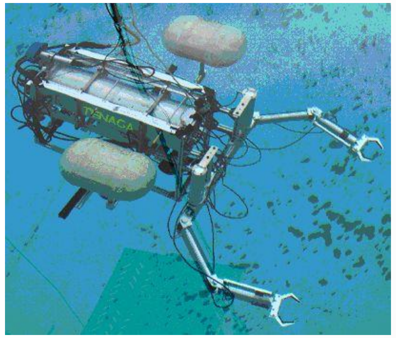
- Sequential decision theory
- Sequential decision theory
- Sensors and information
- Planning under uncertainty

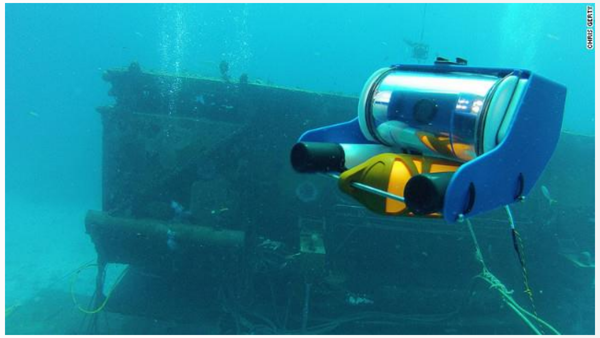
Planning Under Differential Constraints

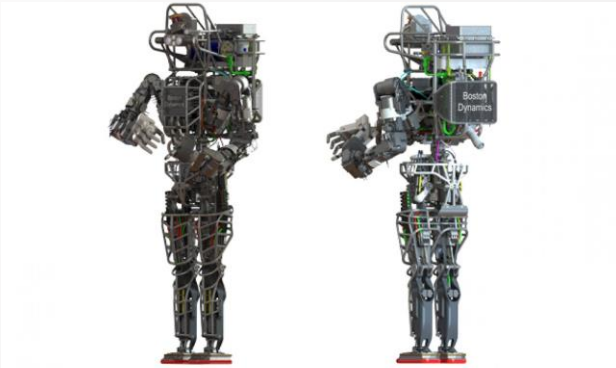
- Differential models
- Sampling-based planning under differential constraints
- System theory and analytical techniques

Robots exist in many configurations









Trends in Robotics/Motion Planning

- **Classical Robotics (mid-70's)**
  - Exact models
  - No sensing necessary
- **Reactive Paradigm (mid-80's)**
  - No models
  - Relies heavily on good sensing
- **Hybrids (since 90's)**
  - Model-based at higher levels
  - Reactive at lower levels
- **Probabilistic Robotics (since mid-90's)**
  - Seamless integration of models and sensing
  - Inaccurate models, inaccurate sensors

Copied from G. Hager http://voronoi.sbp.ri.cmu.edu/~motion

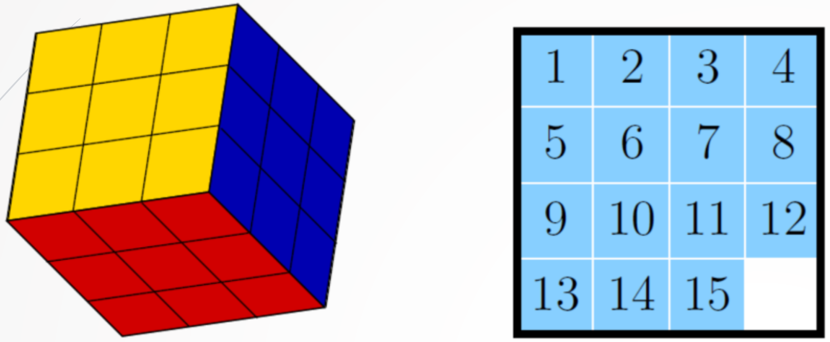
Overview of Concepts

- Planning Tasks
  - Navigation
  - Coverage
  - Localization
  - Mapping

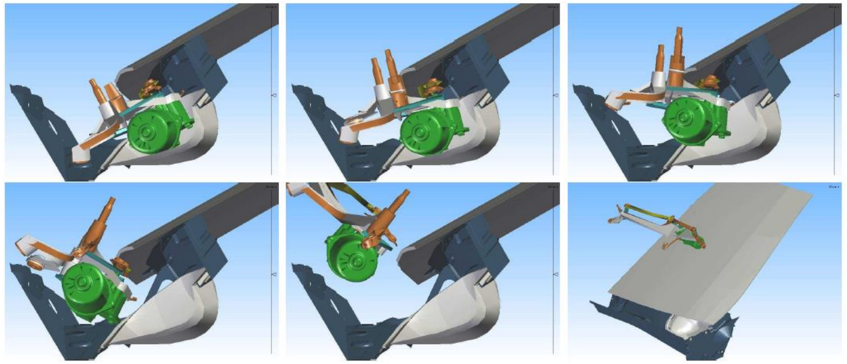
- Properties of the Robot
  - Degrees of Freedom
  - Non/Holonomic
  - Kinematic vs Dynamic

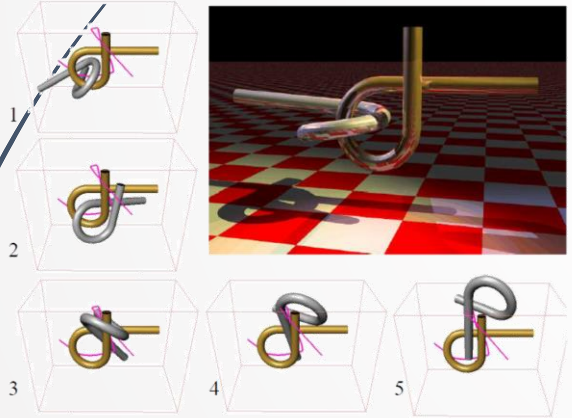
- Algorithmic Properties
  - Optimality
  - Computational Cost
  - Completeness
    - Resolution completeness
    - Probabilistic completeness
  - Online vs Offline
  - Sensor-based or not
  - Feedback-based or not

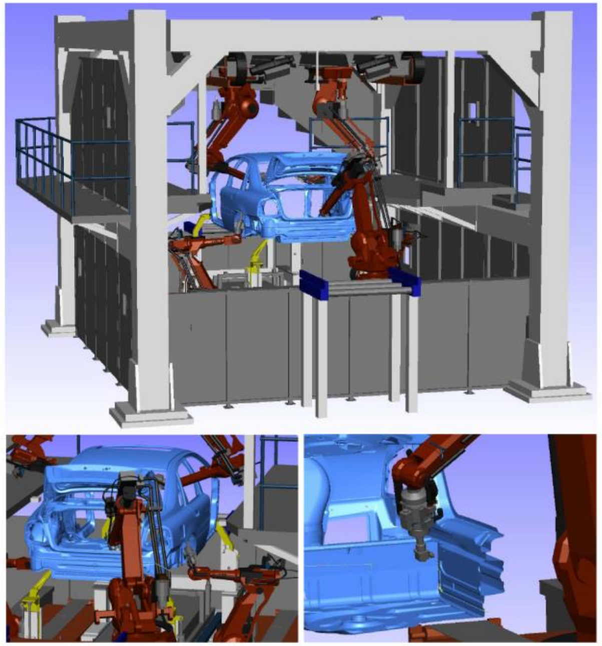
Indicative Examples



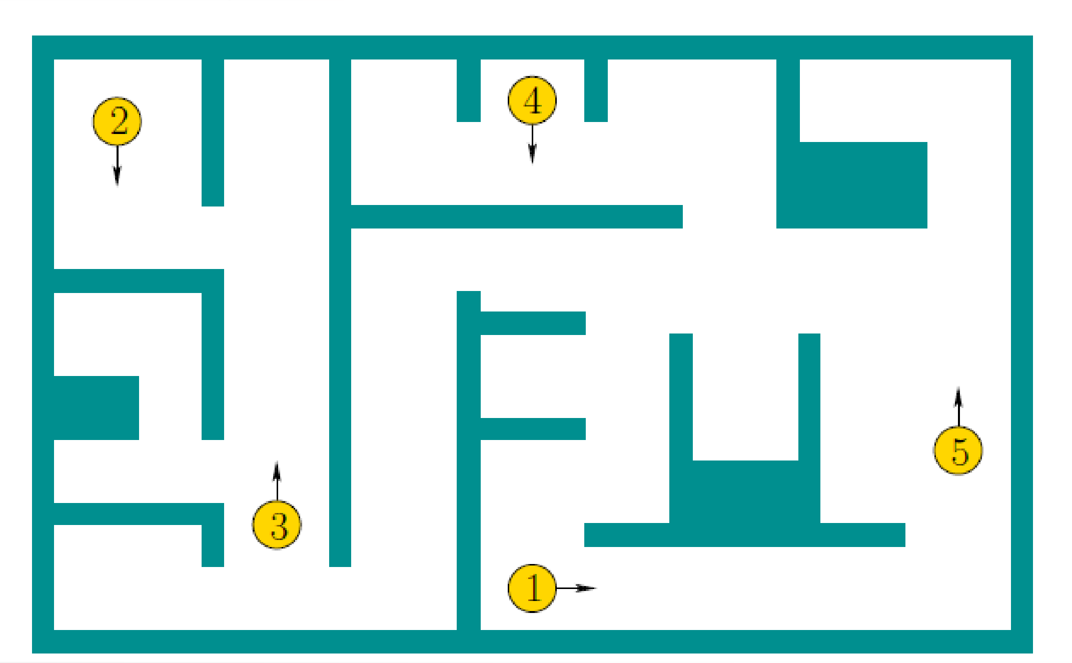
(a) (b)

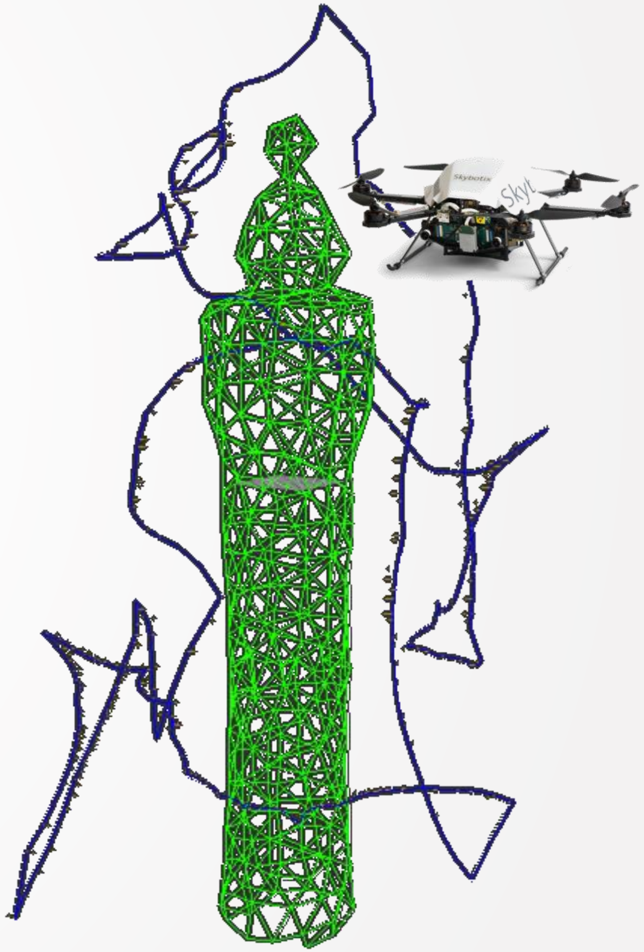






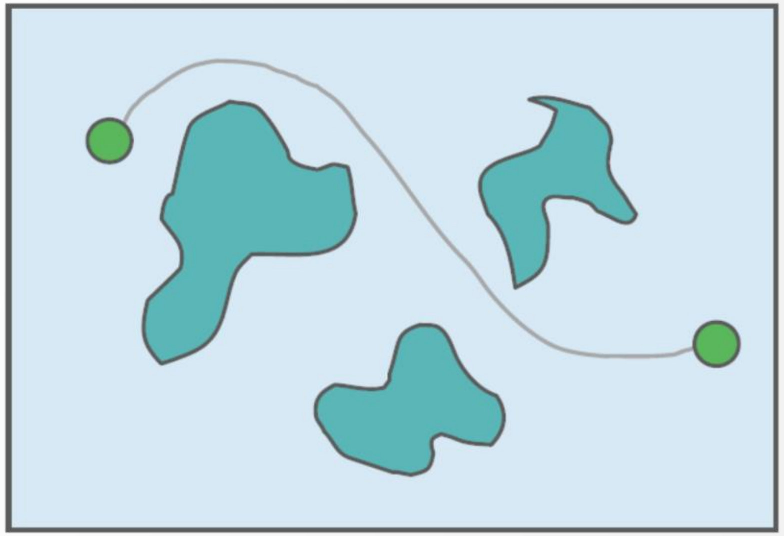
Example of a world (and a robot)





Fundamental Problem of Path Planning

- **Problem Statement:**
  - Compute a continuous sequence of collision-free robot configurations connecting the initial and goal configurations.



- Geometry of the environment
- Geometry and kinematics of the robot
- Initial and goal configurations

Path Planner

Collision-free path

Fundamental Problem of Path Planning

- **Problem Statement:**
  - Compute a continuous sequence of collision-free robot configurations connecting the initial andgoal configurations.
- **Motion Planning Statement for collision-free navigation**
  - If W denotes the robot's workspace, and WO_i denotes the i-th obstacle, then the robot's free space, W_{free}, is defined as: $W_{free} = W - (\cup WO_i)$ and a path c is $c: [0,1] \rightarrow W_{free}$, where $c(0)$ is the starting configuration q_{start} and $c(1)$ is the goal configuration q_{goal}.


ETH zürich
Autonomous Systems Lab
Continuous-Time Trajectory Optimization for Online UAV Replanning
Helen Oleynikova, Michael Burri, Zachary Taylor, Juan Nieto, Roland Siegwart and Enric Galceran

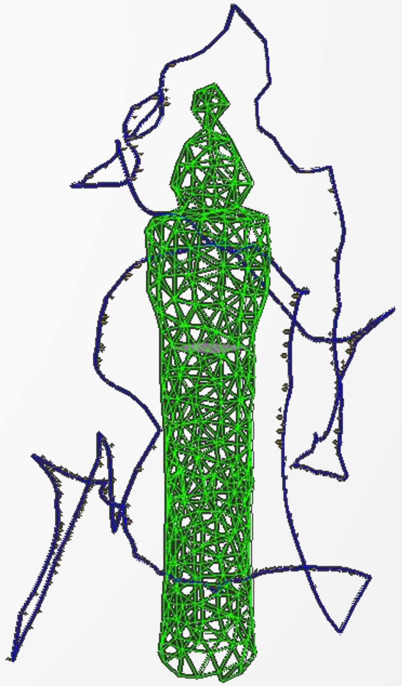
Coverage Path Planning Problem

- **Problem Statement:**
  - Consider a 3D structure to be inspected and a system with its dynamics and constraints and an integrated sensor, the limitations of which have to be respected. The 3D structure to be inspected is represented with a geometric form and the goal is to calculate a path that provides the set of camera viewpoints that ensure full coverage subject to the constraints of the robot and the environment.

- Geometry of the environment
- Geometry and kinematics of the robot
- Structure to be inspected

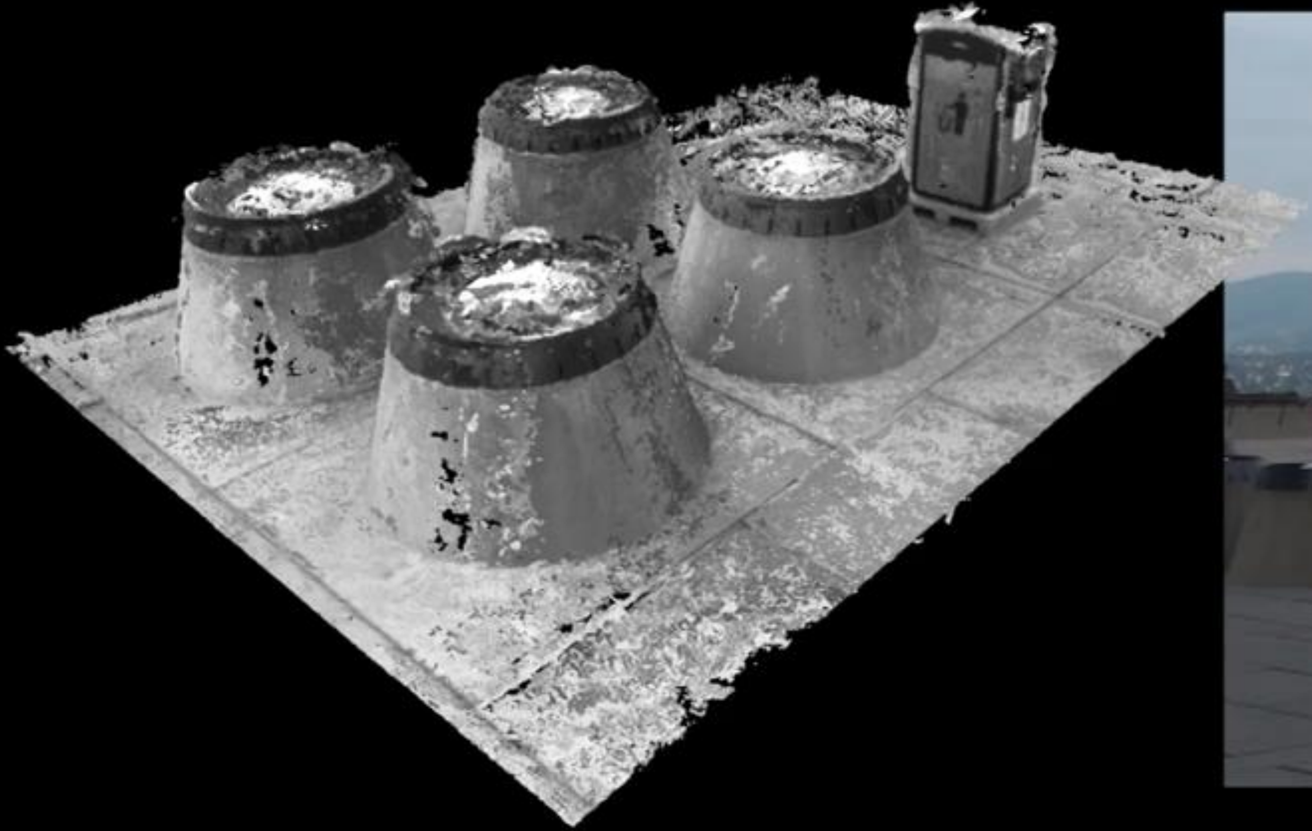
Path Planner

Full coverage path



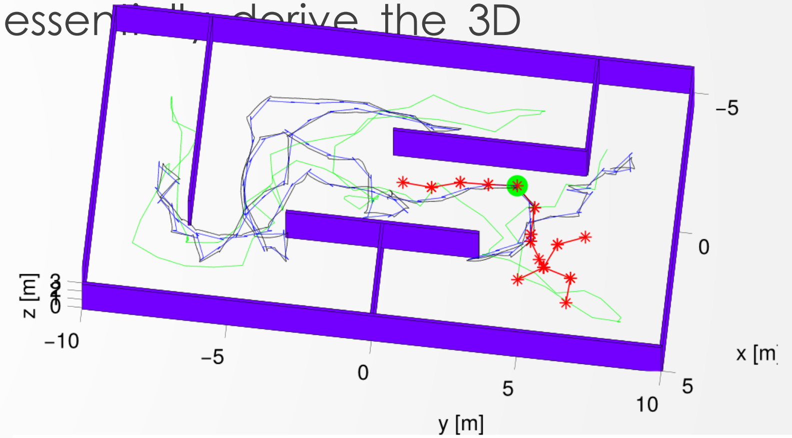
Three-dimensional Coverage Path Planning via Viewpoint Resampling and Tour Optimization using Aerial Robots

A. Bircher, K. Alexis, M. Kamel, M. Burri, P. Oettershagen, S. Omari, T. Mantel, R. Siegwart

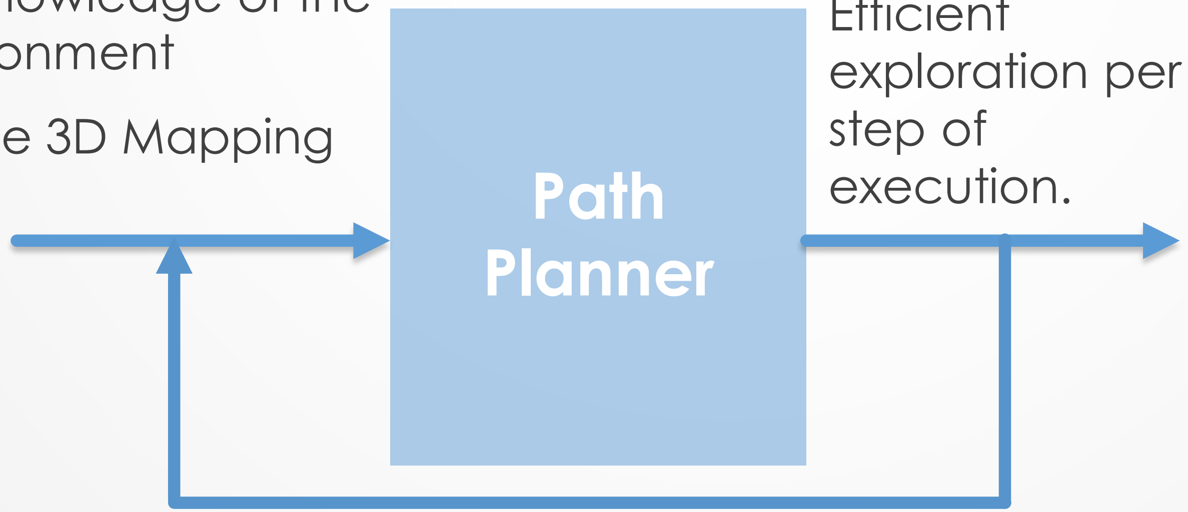


Exploration of Unknown Environments

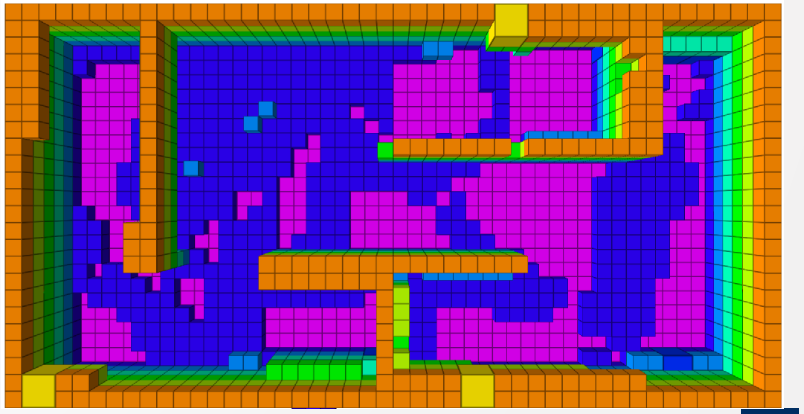
- **Problem Statement:**
  - Consider a 3D bounded space V unknown to the robot. The goal of the autonomous exploration planner is to determine which parts of the initially unmapped space are free V_{free} or occupied V_{occ} and essentially derive the 3D geometric model of the world.



- No knowledge of the environment
- Online 3D Mapping



Efficient exploration per step of execution.



BVS: Holonomic Robot

- Explicit solutions to the problem of point-to-point navigation of a holonomic vehicle operating within an obstacle-free world are straightforward. More specifically, a 6-degrees of freedom (DOF) vehicle that can be approximated to assume only small roll and pitch angles can be approximated using a very simple Boundary Value Solver (BVS). Considering an approximate state vector **ξ = [x,y,z,ψ]** (where x,y,z are the 3 position states and ψ the yaw angle), the path from the state configuration **ξ0** to **ξ1** is given by:

$$\xi(s) = s\xi_1 + (1-s)\xi_0,\ \xi \in [0,1]$$

And considering a limitation on the possible rate of change of the yaw angle \dψ/*dt*|*max* and the maximum linear velocity *υmax*, the execution time is:

$$t_{ex} = \max(d/\upsilon_{max}, \|\psi_1 - \psi_0\|/\dot{\psi}_{\max}$$

with d used to denote the Euclidean distance.

BVS: Holonomic Robot

File: HoverModeMain.py

```
#       __HOVERMODEMAIN__
#       This is the main file to execute examples of the Hover mode
#
#       Authors:
#       Kostas Alexis (kalexis@unr.edu)

from HoverFunctions import *
from PlottingTools import plot3
import numpy as np
import time
import sys

pi = np.pi
verbose_flag = 0
plot_flag = 1

point_0 = np.array([0,0,0,0])
point_1 = np.array([10,10,10,pi/4])

class ExecutionFlags(object):
    """
        Execution flags
    """
    def __init__(self, verbose_flag, plot_flag):
        self.verbose = verbose_flag
        self.plot = plot_flag

class VehicleParameters(object):
    """
        Vehicle Parameters
    """
```

- http://www.kostasalexis.com/holonomic-vehicle-bvs.html

BVS: Nonholonomic Robot

- Dubins airplane is an extension of the classical Dubins car model for the 3D case of an airplane. The specific implementation provided here relies on the formulation presented in:

  (b)Mark Owen, Randal W. Beard and Timothy W. McLain, "Implementing Dubins Airplane Paths on Fixed-Wing UAVs"

  and essentially (as described in this paper) corresponds to a modification of the initial model proposed by *Lavalle et al.* so that it becomes more consistent with the kinematics of a fixed-wing aircraft. Dubins airplane paths are more complicated than Dubins car paths because of the altitude component. Based on the difference between the altitude of the initial and final configurations, Dubins airplane paths can be classified as low, medium, or high altitude gain. While for medium and high altitude gain there are many different Dubins airplane paths, this implementation selects the pat that maximizes the average altitude throughout the maneuver.

BVS: Nonholonomic Robot

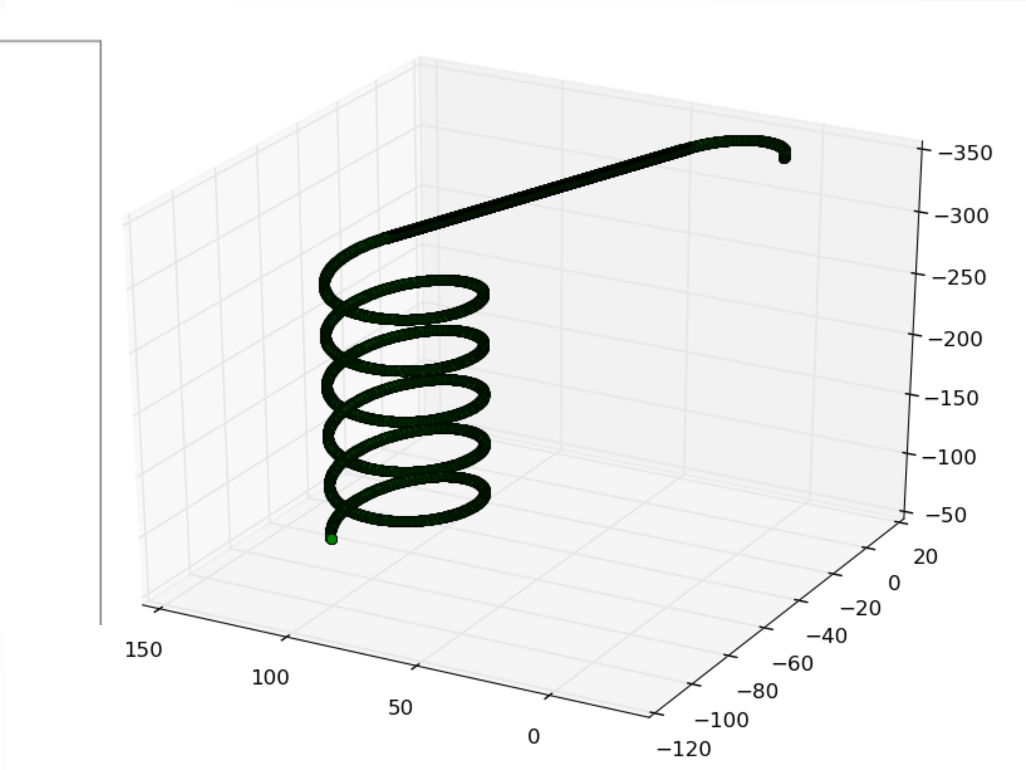
Main File: **DubinsAirplaneMain.py**

```
#       __DUBINSAIRPLANEMAIN__
#       This is the main file to execute examples of the Dubins Airplane mode
#       that supports 16 cases of possible trajectories
#
#       Authors:
#       Kostas Alexis (konstantinos.alexis@mavt.ethz.ch)

from DubinsAirplaneFunctions import *
from PlottingTools import plot3
import numpy as np
import time
import sys

pi = np.pi
dubins_case = 0
verbose_flag = 0
plot_flag = 1

class ExecutionFlags(object):
    """
        Execution flags
    """
    def __init__(self, verbose_flag, plot_flag):
        self.verbose = verbose_flag
```



- http://www.kostasalexis.com/dubins-airplane.html

Code Example

- Python examples on Boundary Value Solvers
  - https://github.com/unr-arl/DubinsAirplane/tree/52ce13e4a6dea9005da702095e6b0acbb175e008
  - https://github.com/unr-arl/autonomous_mobile_robot_design_course/tree/master/python/DubinsCar
  - https://github.com/unr-arl/autonomous_mobile_robot_design_course/tree/master/python/HAV_BVS

Find out more

- http://www.kostasalexis.com/autonomous-navigation-and-exploration.html
- http://www.kostasalexis.com/holonomic-vehicle-bvs.html
- http://www.kostasalexis.com/dubins-airplane.html
- http://www.kostasalexis.com/collision-free-navigation.html
- http://www.kostasalexis.com/structural-inspection-path-planning.html

- http://ompl.kavrakilab.org/
- http://moveit.ros.org/
- http://planning.cs.uiuc.edu/

Thank you!

Please ask your question!