Efficient Autonomous Exploration Planning of Large Scale 3D-Environments

Magnus Selin
magns76@student.liu.se
IDA, Linköping University and CAS, Royal Institute of Technology

Mattias Tiger
mattias.tiger@liu.se
IDA, Linköping University

Daniel Duberg
dduberg@kth.se
CAS, Royal Institute of Technology

Fredrik Heintz
fredrik.heintz@liu.se
IDA, Linköping University

Patric Jensfelt
patric@kth.se
CAS, Royal Institute of Technology

Abstract
Exploration is an important aspect of robotics, whether it is for mapping, resource estimation or path planning in an unknown environment. Frontier exploration planning (FEP) and Receding Horizon Next-Best-View planning (RH-NBVP) are two different approaches with different strengths and weaknesses. FEP explores a large environment consisting of separate regions with costs, but is slow or missing full exploration due to switching back and forth between regions. RH-NBVP shows great potential and is able to find new regions with low cost, but is tied to the current information gain and is incapable of exploring large environments not exploring all regions. In this work we present a method that combines both approaches, with FEP as a global exploration planner and RH-NBVP for local exploration. We also present techniques to estimate potential information gain faster, to make previously estimated gains and to exploit these to efficiently estimate new queries.

Overview
The Autonomous Exploration Planner (AEP) is a further development from the Receding Horizon Next Best View Planner (RH-NBVP). When developing AEP, we looked at the shortcomings of RH-NBVP:

• RH-NBVP does not scale well with map resolution.
• RH-NBVP can get stuck in already explored dead ends.

AEP solves these problems with sparse ray-casting to estimate the potential information gain, caching and re-use of already calculated information gains and frontier exploration as a global exploration strategy, while using RH-NBVP for local exploration.

Sparse ray casting
The estimation of \( g \) given a position \( x \) is efficiently done by performing sparse ray tracing. We look at small volume elements \( V_i \) in spherical coordinates, which are defined as:

\[
V_i = \{ r, \theta, \phi | \theta \in [\theta_i, \theta_i + \Delta\theta], \phi \in [\phi_i, \phi_i + \Delta\phi], r \in [0, \max\text{ range} - \text{hit an obstacle}] \}
\]

Rays are cast, in the field of view, outwards from the sensor until they hit an obstacle or the max range is reached. Every part of a ray that passes through unmapped space, will contribute with \( g \) to the potential information gain.

For a given yaw direction \( \gamma \), the potential information gain is the sum of the potential information gain of all volume elements inside the field of view.

\[
g(x) = \sum_{i=1}^{N} g(V_i)
\]

Best yaw calculation
The sample space for new queries is reduced from four dimensions \( x, y, z, \gamma \) to three \( x, y, z \) by sampling only the position and optimizing for the best yaw. This is done by performing ray-casting \( 360^\circ \) around the agent and using window summation to find the best yaw.

Gaussian process interpolation of \( g \)
We cache calculated information gains from earlier iterations (Fig 5a). These are used as data points in a Gaussian process (Fig 5b). Whenever a new point \( x \) is queried, we first check the result of the Gaussian process. If the posterior variance is low enough, we accept the interpolated data, otherwise we calculate it explicitly and add it to the cache.

Table 1 shows that we save time by querying the Gaussian process instead of calculating the values explicitly. We can also see that the time for explicit gain estimation grows quickly when the rays are denser, while for the GP the time naturally stays the same.

Frontier exploration
When testing RH-NBVP, we saw the behavior that it sometimes got stuck in dead ends. We have tackled this problem by combining RH-NBVP with frontier exploration. RH-NBVP works as a local exploration strategy and frontier exploration takes over when there is no new information to be gained nearby.

Results in office environment

Acknowledgements
This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) and the SSF project FACT.