AutoLook Design Doc

Last update: 2019-04-04
Self link: go/proxy-autolook
Author: ammarh@
Reviewer: x-proxy-navigation@
Status: Implemented

Objective

Proxy robots currently navigate around by perceiving their environment using point cloud data. These point clouds may be acquired through the CBr lidar scanner or through the stereo camera pair. The point clouds are processed into the occupancy grid where cells with an obstacle are marked as **occupied**, the ones that have not been observed as **unknown** and the rest being **free**. The CBr cannot be moved and therefore produces a fixed observable area with respect to the robot. In contrast the stereo camera pair is mounted on the head of the robot and can therefore be commanded to look at arbitrary areas of interest near the robot. Even though the field of view per scan for the stereo point cloud is much smaller than the field of view of the CBr, the total field of view obtained through the stereo point clouds is significantly larger when the motion of the head is incorporated. Additionally, just by the nature of their sensing, stereo cameras are capable of producing much denser point clouds leading to better observations in the navigation occupancy maps. Although cameras are more prone to outliers.

A key limitation of the current system is that the navigation interface does not actively control or command the head of the robot as it is driving. The head position is set by the L5 application layer and it points the head to the back of the robot in order to fill in the known CBr blindspot there. However, one can imagine a more intelligent framework where the head is more actively involved in observing areas of most interest to the planner. This design document outlines a proposal for the same, thereby enabling both smoother & safer navigation.

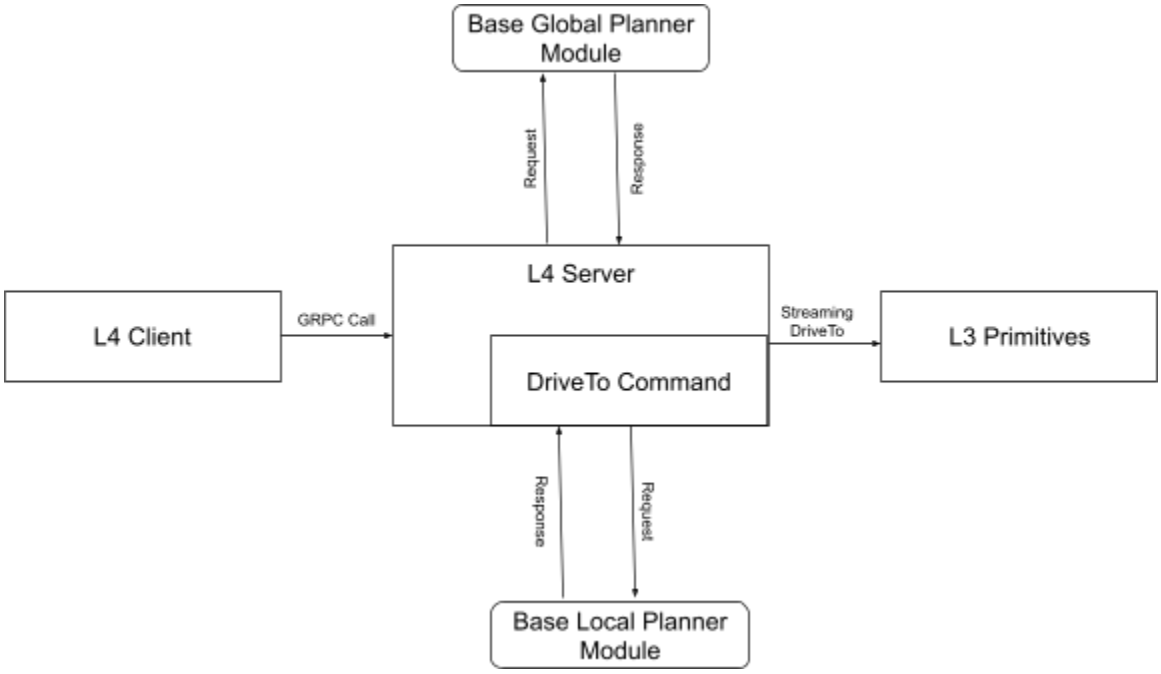
Scope & Exit Criteria

The scope of this project is limited to moving the head of the robot to observe potentially reachable areas in the occupancy map. It does not involve adding any capability into the base planner to create more reachable areas that may either be occluded or in the blindspot of both sensing modalities. A successful exit criteria for this project will be achieved when the base local planner is able to reject all trajectory proposals that intersect unknown regions and yet navigate through free space. As outlined in the Design Ideas section, the project will be approached in three phases of development and in increasing order of complexity. While phase# 1 is the MVP required to meet the exit criteria and is therefore scheduled to complete first in Q2’19, the follow on phases build up on the efficiency &

seamlessness of the system.

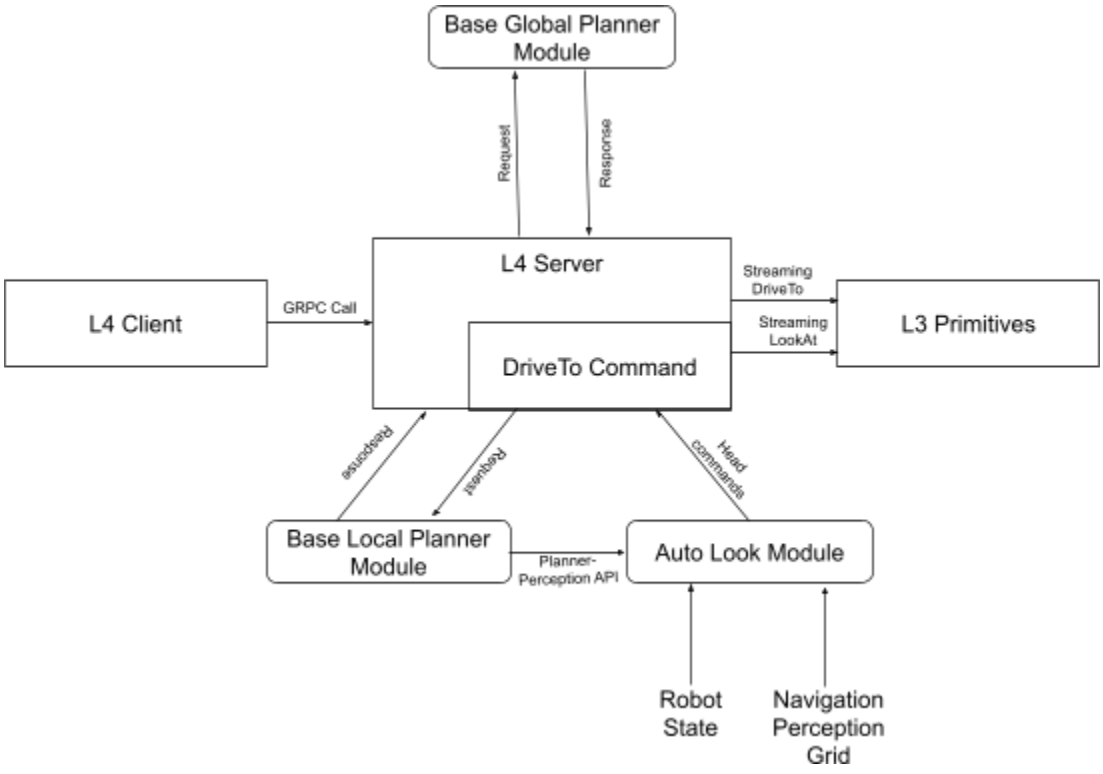
Background

The robot is commanded to navigate between points using the DriveTo (design doc) L4 API command. This command invokes a global planner to obtain a path to the goal by leveraging the map of the environment. The BaseLocalPlanner produces several trajectory proposals that are consistent with the global plan and have a X seconds/meters horizon from the robots. The best collision free trajectory is executed by the robot. The DriveTo command API communicates with both the planning modules via IPC with a Request & Response API. Below is a simple sketch of the architecture.



Design Ideas

The proposed approach is to create an AutoLook Module that subscribes to the current robot state & the navigation perception occupancy grid. Additionally a new API Proto will be defined for the BaseLocalPlanner to communicate its planning state such as clutter ratio, planned trajectory etc. These channels will be processed through a few head motion policies that will be published to DriveTo as various LookAt primitive commands, as shown in the figure below:



Phase 1: Q2 2019

The first phase of this design focuses on building a proof of concept for active perception which will unlock several learnings along the way. The goal at the end of this phase is to implement the system pipeline as described is the figure above. The exploration strategy within the AutoLook module will be fairly naive in terms of efficiency.

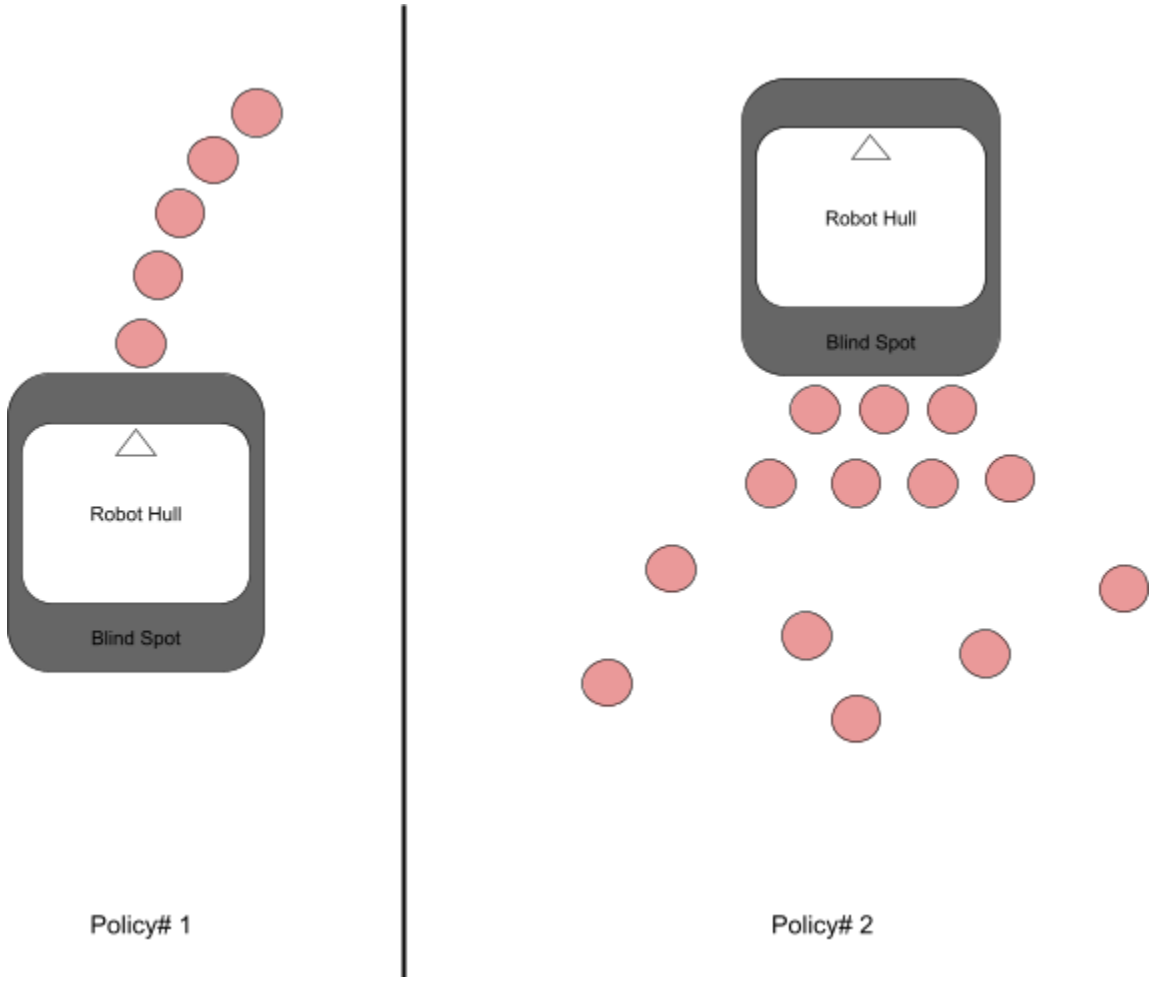
Exploration Strategy:

Policy# 1

When the robot is navigating at moderately high speeds through relatively uncluttered space, active perception will focus on updating the space in front of the robot by sweeping through the trajectory waypoints. The BaseLocal planner will publish these trajectory points in the PlannerPerception API proto.

Policy# 2

While navigating in cluttered environments the AutoLook module will focus on clearing "unknown" areas around the robot. These areas are primarily in the rear of the robot due to the CBr blindspot.



The figure above illustrates these policies. The gray box is the blindspot hull that is not visible by either the stereo camera or the CBr. Therefore this area is fundamentally unobservable when the robot is stationary. The circles are nodes in the exploration graph that need to be observed. In the simple case they would be the grid cell coordinates in the navigation perception grid. The policies will be switched by a flag from the BaseLocalPlanner given the planning difficulty due to clutter. For this phase a simple greedy exploration algorithm will be implemented that scans from one node to the other using a proximity based cost function.

Phase 2: TBD

For the second phase, the AutoLook module will implement a smarter exploration for the graph of “unknown” nodes. This would be done by adding a more sophisticated weighting function per node that includes its proximity to robot and time last observed. In order to obtain the time last observed for a grid cell, the navigation perception occupancy grid will need to be augmented to publish this information. Additionally there would be a cost edge

between these nodes that is weighted by the distance between them. In other words, 2 adjacent nodes will have very low cost, while a node on the left of the robot will have a high cost compared with one on the right. The AutoLook module will attempt to traverse through this graph while minimizing traversal cost. It must be noted that this mainly improves Policy# 2.

Another key requirement to satisfy in Phase 2 is to clear areas potentially occupied by dynamic obstacles. This will be done through augmenting the policies to observe areas that have not been observed in a while. It is a logical requirement within this phase, given the fact that the perception occupancy grid will also publish the "last observed" field for each grid cell.

Phase 3: TBD

The third phase for AutoLook will attempt to unify the exploration by merging Policy# 1 with Policy# 2 using a unified cost function. This cost function will factor in the clutter ratio supplied by the planner and weight both the lookahead nodes as well as the "unknown" nodes into a single graph to traverse.

Outstanding Issues

- b/124933293, b/69372097: In the current system the robot footprint is not considered within navigation perception. This causes the unobserved areas, within the robot footprint, of the occupancy grid to eventually decay to unknown if the robot is stationary for a certain period of time. These issues will need to be resolved to successfully meet the exit criteria for this project.
- b/130054283: In order for camera point clouds to be seamlessly integrated into the system, the ego hull needs to be filtered out from the images so that outliers in the stereo point cloud for ego regions do not affect the perception system by spoofing it with obstacles. This a critical requirement for AutoLook to work. Some bug reports where ego hits have caused navigation failure are: b/129984705, b/130027644
- Part Locking concerns - If another application attempts to lock during DriveTo?

Implementation Steps

Outlined in this section are roughly the steps needed to accomplish Phase 1 for AutoLook.

- ❏ Create an AutoLook module that streams LookAt commands to the DriveTo L4 command
- ❏ Resolve the outstanding issue regarding ego hull masking for stereo point clouds.
- ❏ Create the PlannerPerception API proto and populate it with a header, trajectory waypoints, clutter ratio & policy flag.
- ❏ Implement Policy# 1 by sending LookAt commands based on the trajectory waypoints.
- ❏ Resolve the currently outstanding issues regarding robot footprint, as outlined

earlier.

- ❏ Subscribe to the navigation perception grid and robot state in the AutoLook module.
- ❏ Compute unknown grid cells and add them to a priority queue.
- ❏ Pop nodes from the priority queue to send LookAt commands.
- ❏ Update the priority queue given the most recent observations in the occupancy grid. That is if a node was added as unknown but has now been observed because its neighbor was observed then it must be cleared out from the queue.

Alternatives Considered

- The AutoLook module computes head motion commands primarily by fusing the planner state with the navigation perception occupancy grid. One alternative that was considered was to have the active perception logic within BaseLocalPlanner since technically that module already processes similar information. This option was discarded in favor of reducing the complexity within the planner module and increasing the modularity of the system by having an external module responsible for exploring using the head. Additionally this design can be extended in the future to connect with the BaseGlobalPlanner and/or the arm planner for manipulation, in order to actively explore both local & global trajectories.
- Another alternative that was considered was to add the head exploration logic inside the DriveTo command API. The BaseLocalPlanner output is already consumed here and can easily be extended to subscribe to the navigation perception grid. This alternative was also discarded in favor of reduced complexity and added modularity by creating a dedicated AutoLook module.

Risks

- Being robust against changes to camera settings: There have been several recent bug reports (b/130037267) that have been filed due to noise in stereo point clouds. This was most likely triggered due to changes in camera rectification settings. We would need to create some form of QA or automated tests that guard against producing ghost obstacles around the robot.
- The default position of the arm on the robot is to be tucked back & behind the robot hanging over the tray. This position occludes a large part of the floor behind the robot which is also in the CBr blindspot. Therefore to truly clear out unknown areas the arm might need to be moved out of the way to scan the rear of the robot. The AutoLook module is contained entirely within DriveTo command, so that would mean DriveTo would need to be extended to lock & move the arm.

Feedback

If you read this document please provide your short general feedback in the section below.

Username	Date	Comment
nacorn	4/11/2019	+1 - looks great