Planning to Plan:
New Flexible Navigation Interfaces

David V. Lu!!
ROSCon 2018
Central Dogma of ROS Navigation

Current Location + Goal Location

**Global Costmap + Planner**

Global Plan

**Local Costmap + Planner**

Command Velocities
Expressing Thought: Improving Robot Readability with Animation Principles

Leila Takayama
Willow Garage
68 Willow Road
Menlo Park, CA, USA, 94043
takayama@willowgarage.com

Doug Dooley
Pixar Animation Studios
1200 Park Avenue
Emeryville, CA, USA, 94608
dooley@pixar.com

Wendy Ju
Willow Garage
68 Willow Road
Menlo Park, CA, USA, 94043
wendyju@willowgarage.com

ABSTRACT

The animation techniques of anticipation and reaction can help create robot behaviors that are human readable such that people can figure out what the robot is doing, reasonably predict what the robot will do next, and ultimately interact with the robot in an effective way. By showing forethought before action and expressing a reaction to the task outcome (success or failure), we prototyped a set of human-robot interaction behaviors. In a 2 (forethought vs. none: between) x 2 (reaction to outcome vs. none: between) x 2 (success vs. failure task outcome: within) experiment, we tested the influences of forethought and reaction upon people's perceptions of the robot and the robot's readability. In this online video prototype experiment (N=273), we have found support for the hypothesis that perceptions of robots are influenced by robots showing forethought, the bicycle riders to avoid pedestrians (as noted by [19]), let us clarify who we are speaking to [9].

discomfort, for instance, when other Robots that operate in public settings (airports, marketplaces) can be safer and more effective if they are designed and programmed to perform readable behaviors [4]. By making the robots more readily apparent to both observers and bystanders, we can improve people’s awareness of their actions with that of robots, making it easier to coordinate our actions with one another.

One of the biggest challenges is to make robots that are not busy “thinking” or planning to act. It is not for robots to remain still while computing planning movement, but the tendency to act suddenly from dead stop to action of

Animating with the Navigation Stack

**Forethought**
Get the Global Plan, Look that Direction, Start Driving

**Reaction**
If the robot failed, How it failed, When it failed

**Planning**
Plan around people, for people
Navigation is often about more than driving from point A to point B efficiently. Context is key.
Planning to Talk about "Planning to Plan"

- The State of Navigation
- Navigation Interface Design
- New Global Planner
- Locomotor
- ROS2: The Next Generation
Why the fork?

Hard to swap out single components

Feature creep of the universal solution

Move slow and don't break things

[GitHub link](https://github.com/ros-planning/navigation)
State of the Navigation Packages

6DOF Poses Everywhere

- **nav_core** - Interfaces haven't changed much in almost a decade
- **costmap_2d** - forces you to use layers, transmits OccupancyGrids

Global Planners

- **navfn** is prematurely optimized, hard to edit
- **global_planner** has bugs

Local Planners - difficult to debug/customize

- **move_base** - Black box, static list of recovery behaviors
Design Methodology

Build reusable pieces

Meant to be extensible

No black boxes
**nav_grid**

**nav_grid::NavGridInfo**

```cpp
unsigned int width = 0;
unsigned int height = 0;
double resolution = 1.0;
std::string frame_id = "map";
double origin_x = 0.0;
double origin_y = 0.0;
```

relative of nav_msgs::MapMetaData
NavGrid\(<T>\) Operations

- Abstracted Data Storage
- Template Based Typing
- Coordinate Translation
- New Message Types
  - NavGridOfChars
  - NavGridOfCharsUpdate
  - NavGridOfDoubles
  - NavGridOfDoublesUpdate
- Iterators
nav_core2
it's plugins all the way down

BaseGlobalPlanner
BaseLocalPlanner
RecoveryBehaviors
CostmapLayer
DWBLocalPlanner
and more...
class Costmap : nav_grid::NavGrid<unsigned char>
{
    void initialize(NodeHandle parent, string name,
                     TFListenerPtr tf);
    void update();
    mutex_t getMutex();
    bool canTrackChanges();
    UIntBounds getChangeBounds(string ns);
};
dwb_local_planner

**Velocity Iterator & Trajectory Generator**

**Trajectory Critic[]**

**Goal Checker**
class GlobalPlanner
{
public:
    void initialize(NodeHandle parent,
                     string name,
                     TFListenerPtr tf,
                     Costmap::Ptr costmap);

    Path2D makePlan(Pose2DStamped start,
                    Pose2DStamped goal);
};
dlux_global_planner

Potential Calculator

- dlux_plugins::Dijkstra
- dlux_plugins::AStar

Traceback

- dlux_plugins::VonNeumannPath
- dlux_plugins::GridPath
- dlux_plugins::GradientPath
The many paths of dlux global planner
move_base

- **Four Components**
  - Global Costmap
  - Global Planner
  - Local Costmap
  - Local Planner

- Pass Global Plan to Local Planner
- Try to Recover when Planning Fails
- Extensible Path Planning Coordination Engine
- Control what to do when path planning succeeds and fails
- Built on nav_core2 Interfaces
- Leverages ROS Callback Queues
Single Thread Control Flow

- doGlobal
  - CostmapUpdate
- makeGlobalPlan
- doLocal
  - CostmapUpdate
- makeLocalPlan
Dealing with Failure

- `doGlobalCostmapUpdate`
  - `InvalidStartPoseException` → `Clear Global Costmap`
  - `InvalidGoalPoseException` → `Choose New Goal`

- `makeGlobalPlan`
Double the Threads - Double the Fun

Flowchart:
- doGlobal
  - CostmapUpdate
  - makeGlobalPlan
- doLocal
  - CostmapUpdate
  - makeLocalPlan
Four Shall Be the Number of the Timers

- doGlobal
  - CostmapUpdate
- makeGlobalPlan
- doLocal
  - CostmapUpdate
- makeLocalPlan
locomove_base

- Drop in replacement for move_base
- Uses nav_core_adapter::CostmapAdapter
- Loads nav_core::RecoveryBehaviors
Plugin Mux

- Load Multiple Pluginlib Plugins
- Not nav_core2 specific
- Switch Plugin via C++ method or ROS Service
- Triggers Callback when Switching
Showing Forethought in Navigation

- `makeGlobalPlan`
- `doGlobalCostmapUpdate`
- `Look in Direction of Global Plan`
- `doLocalCostmapUpdate`
- `makeLocalPlan`
move_base Actions

# move_base_msgs/MoveBase.action
geometry_msgs/PoseStamped target_pose
---
---
geometry_msgs/PoseStamped base_position
Locomotor Action

nav_2d_msgs/Pose2DStamped goal
---
int64 state_info
---
nav_2d_msgs/Pose2DStamped current_position
nav_2d_msgs/Twist2D current_speed
nav_2d_msgs/Path2D global_plan
float32 percent_complete
float32 distance_traveled
float32 estimated_distance_remaining
int64 state_info
ROS 2

- Full reimplementation of navigation stack
  - DluxGlobalPlanner
  - DWBLocalPlanner
  - Locomotor
  - LayeredCostmap and CostmapLayers
  - NavGridServer / Saver
- Out by year's end

https://github.com/ros-planning/navigation2

Matt Hansen, Intel