Data Fusion Methods and Algorithms in the Context of Autonomous Systems
A path planning algorithms analysis and optimization exploiting fused data

Francesco SAVARESE

DAUIN – Dept of CONTROL AND COMPUTER ENGINEERING
Doctoral Program in Computer and Control Engineering (31° Cycle)

Advisor: Prof. Gianpiero CABODI
Improving Trajectory Generation for Autonomous Car Exploiting GPGPUs

Detecting, Opening and Navigating through Doors: A Unified Framework for Human Service Robots
Improving Trajectories Generation Exploiting GPGPUs
- Background
- Problem Description
- Experimentation
- Limitation & Future Work

Detecting, Opening and Navigating through Doors: A Unified Framework for Human Service Robots
- Background
- Proposed Approach to Door Opening while Navigating
- Experimentation
- Limitation & Future Work
Improving Trajectory Generation for Autonomous Car Exploiting GPGPUs
Necessity of autonomous car:
• Traffic jam and accidents
• People with mobility impairments
• Long commuting time

Navigation in highly dynamical environments:
• Accurate environment understanding
• Efficient manouvre planning
• Reliable decision making
Object recognition, Environment segmentation, map generation already exploit GPGPUs

- Trajectory planning improvements are based on mathematical techniques
- Current algorithms do not exploit GPGPU based technologies
CONTRIBUTION

- Metrics for trajectory planning evaluation proposal

- Study and implementation of a state of the art trajectory planning algorithm based on: U.Schwesinger et al. “A sampling-based partial motion planning framework for system-compliant navigation along a reference path”. In: 2013 IEEE Intelligent Vehicles Symposium

- GPGPU algorithm design

- Performance and behavioural algorithm evaluation
- White for prohibited regions
- Green for allowed regions
New Path from Global Planner
SCENARIO DESCRIPTION (3)

- SEVERAL TRAJECTORIES GENERATED
- ONE SELECTED
An higher number of nodes improves:

- Space exploration
- Trajectory accuracy

The vehicle controller frequency limits the trajectory generation time (20 ms in our scenario)
ALGORITHM DESCRIPTION

Vehicle Kynematic Model

ANALYZED PARAMETERS

Tree Height

Lookahead Time

VEHICLE KYNEMATIC MODEL

Starting Point

Path to Follow

Sampled Goals

Level 0  Level 1  Level 2  Level 3
BEHAVIOURAL EVALUATION METRICS
**EVALUATION METRICS DEFINITION**

**Starting Distance (SD)**

It measures how far from the obstacle ego vehicle reacts. It’s good to measure algorithm reactivity.

**Minimum Obstacle Distance (MOD)**

It measures the vehicle-obstacle distance during the manœuvre execution. It’s good to measure the safety manœuvre.

**RMSE**

Root Mean Square Error to measure the adherence to desired trajectory.

**Generation Time**

Is the time for generating trajectory tree and selecting the best trajectory.
CPU ALGORITHM CONFIGURATION (LOOKAHEAD TIME)

![Graph showing SD (milliseconds) vs. Lookahead Time (seconds)]

![Graph showing Time (milliseconds) vs. Lookahead Time (seconds)]
CPU ALGORITHM CONFIGURATION (Tree Height)

The graphs illustrate the relationship between tree height and two performance metrics: RMSE and Time [ms].

- **RMSE** graph shows a linear increase with tree height, indicating higher RMSE values for taller trees.
- **Time [ms]** graph also shows an increase, with notable jumps at specific tree heights, suggesting potential points of interest for optimization.

The data suggests that as the tree height increases, both RMSE and Time [ms] also increase, highlighting the trade-offs in algorithm performance with varying tree configurations.
GPGPU BASED APPROACH AND EVALUATION
PARALLEL APPROACH

Maps Path

Tree Computation

DRAW SAMPLE KERNEL

EXPAND KERNEL

Trajectory Tree

Trajectory Selection

DRAW SAMPLE KERNEL

EXPAND KERNEL
CPU response with:
\[ D = 6 \text{ and } H = 4 \]

GPU response with:
\[ D = 6 \text{ } H = 4, \ D = 5 \text{ } H = 5, \ D = 4 \text{ } H = 6, \ D = 5 \text{ } H = 6 \]
### CPU vs. GPU (2)

<table>
<thead>
<tr>
<th>D</th>
<th>H</th>
<th># Traj</th>
<th>CPU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>T1 [ms]</td>
<td>T2 [ms]</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>216</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>625</td>
<td>26</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1296</td>
<td>45</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3125</td>
<td>77</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>4096</td>
<td>83</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>15625</td>
<td>218</td>
<td>44</td>
</tr>
</tbody>
</table>

- $T_1$ = Time for tree generation
- $T_2$ = Time for trajectory selection
- $T_{tot}$ = Total computation time
CONCLUSIONS

• Quantitative metrics for trajectory planner comparison
• Re-engineering a state-of-the-art randomized sampling based motion planning method

Extensive experimentation:
• Accuracy stability when changing platform
• 5X speedup when using GPGPU

Limitations/future works:
• Parameter selection according to scenario
• Experimentation with different scenario/different vehicle models
• Integration with other GPGPU based modules
• Implementation as ROS module
Detecting, Opening and Navigating through Doors: A Unified Framework for Human Service Robots
Necessity of service robots:
• Aging population
• People with mobility impairments

Navigation of changing environments:
• Unmodified for the robot
• Complexity of the door-opening problem
DOOR OPENING – STEPS INVOLVED

- Door/Handle Recognition
- Door Type Detection
- Handle Grasping/Unlatching
- Door Opening
NO UNIFIED PIPELINE HAS BEEN PROPOSED YET

- In (Rhee et al., 2004), an exclusive robot is adopted, whose hand is specifically designed for the door opening task.

- In (Petrovskaya and Ng, 2007; Rhee et al., 2004; Peterson et al., 2000; Dongwon et al., 2004), handle unlatching and door opening are tackled obviating approaching the door and navigating through it.

- In (Jain and Kemp, 2008), after the handle unlatching the door is pushed to be opened. They do not study the case of pulling door and door traversing.

- In (Fernández-Caramés et al., 2014), a simple method for door detection is proposed, but it is not tested with different types of doors, and door opening is not tackled.

Door characteristics should be pre-input:
- What is the door size?
- Where is the handle?
- What is the door type?
  - Pulling, pushing, etc.
• Door opening action flow (prior door characteristics knowledge is required)

• Deep learning-based automatic door/handle detection
  • MIL-door dataset

• Navigation framework to conduct experimentation

• Implementation on a standard platform
SOFTWARE PLATFORM: State Machine (SM) Approach

LEVEL 2: DOOR OPENING

LEVEL 1: NAVIGATE ENVIRONMENT

TOP LEVEL: FIND USER

SOFTWARE PLATFORM

Error recovery:
- Inaccurate handle detection
- Grasping failures
- ...

VOICE COMMAND

SPEECH TO TEXT

STATE MACHINE CONTAINER

FIND USER

SM FOR DIFFERENT TASKS

TEXT TO SPEECH

VOICE RESULT
Door Opening Algorithm
Single Shot MultiBox Detector (SSD) deep neural network (Liu et al., 2016)
- Real time detection
- Easy to train
- Works well on embedded systems

Door/Handle Detection

- Closed Door?
  - NO: Move Through
  - YES: Door Params Computation
    - Handle Grasping & Unlatching

Color image → DNN → Bounding boxes coordinates + class label
462 images of doors and 318 images of Handles manually annotated.

Validation

<table>
<thead>
<tr>
<th>Object</th>
<th>Detection Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Door</td>
<td>94.7%</td>
</tr>
<tr>
<td>Handle</td>
<td>86.3%</td>
</tr>
</tbody>
</table>

Error Recovery policy:

- Move slightly back and forth, left and right
- When a door marked in the map is not detected
- When the door is detected but the handle is not
Parameters and Door Type Estimation
DOOR PARAMETERS ESTIMATION

Inputs:
• Color image
• Door-Robot Distance
• Bounding boxes

Outputs:
• Door width
• Opening direction

Door/Handle Detection

Closed Door?

NO

Move Through

YES

Door Params Computation

Handle Grasping & Unlatching

DOOR WIDTH

OPENING DIRECTION
Pull Door of 5 cm

Pulling Succeeded?

Push Door of 5 cm

Pulling Succeeded?

Door Locked

End
PULLING DOOR TRAJECTORY

Compute Pulling Trajectory

Pull Door

Compute Pushing Trajectory

Push Door

OPENING DIRECTION

Door Width

80 Deg

8 Deg
PUSHING DOOR – COLLISION CHECKING

Compute Pulling Trajectory

Push Door

Compute Pushing Trajectory

Push Door
PULLING DOOR – COMPLETE TASK

39

PULLING DOOR – COMPLETE TASK
Experimentation Setup and Map Manager
Goal: To evaluate our door opening framework in a realistic navigation scenario

EXPERIMENTS

The navigation is repeated 50 times.

Goal reaching success rate: 100%
Closed/Open door discrimination success rate: 100%

Robustness of the error recovery protocol
  • Door/handle detection
SEMANTIC NAVIGATION – MAP MANAGER

RVIZ based Environment description Generator

Environment Description

Location Description

MAP MANAGER

- Place By Attribute
- Get Room from Coordinates
- Get Room From Location
- Save a Temporary Position
- List locations in a room
- ...

SEMANTIC NAVIGATION – MAP MANAGER
Goal: To evaluate the robustness of our handle grasping/door opening algorithm

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Handle Type</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulling non-spring loaded door</td>
<td></td>
<td>14/20</td>
<td>18/20</td>
<td>16/20</td>
<td>18/20</td>
</tr>
<tr>
<td>Pulling spring loaded door</td>
<td></td>
<td>12/20</td>
<td>16/20</td>
<td>16/20</td>
<td>18/20</td>
</tr>
<tr>
<td>Pushing non-spring loaded door</td>
<td></td>
<td>20/20</td>
<td>16/20</td>
<td>20/20</td>
<td>16/20</td>
</tr>
</tbody>
</table>

100% of the cases the door type is successfully discriminated.

Experiment failed when an error arose: detection, grasping, opening.

T1: Slippery handle on the door left side.
T2: Slippery handle on the door right side.
T3: Not Slippery handle on the door left side.
T4: Not Slippery handle on the door right side.
Unified framework for opening doors while navigating:
• Automatic detection of door parameters
• Implementation on a standard platform

Promising experimental results:
• Robustness and flexibility
• High applicability

Limitations/future work:
• Spring loaded pushing doors
• Narrow Spaces
• More types of doors: sliding doors, knobs, ...
• Move away obstacles
Data Fusion Methods and Algorithms in the Context of Autonomous Systems
A path planning algorithms analysis and optimization exploiting fused data

Francesco SAVARESE

DAUIN – Dept of CONTROL AND COMPUTER ENGINEERING
Doctoral Program in Computer and Control Engineering (31° Cycle)

Advisor: Prof. Gianpiero CABODI

THANKS FOR YOUR ATTENTION
PUBLISHED PAPERS

Cabodi, G.; Camurati, P.; Garbo, A.; Giorelli, M.; Quer, S.; Savarese, F.

*Detecting, Opening and Navigating through Doors: A Unified Framework for Human Service Robots*.

OTHER PUBLISHED PAPERS

*Embedded Systems Secure Path Verification at the HW/SW Interface*. In: *IEEE Design & Test*. - ISSN 2168-2356

Sterpone, Luca; Cabodi, Gianpiero; Finocchiaro, Sebastiano Fabrizio; Loiacono, Carmelo; Savarese, Francesco; Du, Boyang (2016)
*Scalable FPGA Graph model to detect routing faults*. In: IEEE International Symposium on On-Line Testing and Robust System Design.

Cabodi, Gianpiero; Camurati, Paolo; Finocchiaro, Sebastiano Fabrizio; Loiacono, Carmelo; Savarese, Francesco; Vendraminetto, Danilo (2016)