KIT-Fakultät für Informatik
Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H^2T)

Robotik I im WS 2017/18

8. Übungsblatt

Termin: 05. Februar 2018

Prof. Dr.-Ing. Tamim Asfour
M. Sc. Fabian Paus
Dipl.-Inform. Peter Kaiser
Adenauerring 2, Geb. 50.20
Web: http://h2t.anthropomatik.kit.edu

Aufgabe 1 (Symbolische Planung mit STRIPS)

Gegeben sei die folgende STRIPS-Planungsdomäne mit initialem Zustand, Zielzustand und verfügbaren Planungsoperatoren.

```
Initial state: Table(T), Stove(S), Location(T), Location(S),
Gripper(G), Hand(G), Robot(R), Agent(R), Pan(P),
Empty(G), On(P,T), At(R,S)

Goal state:    On(P,S)

Actions:

// Agent A nimmt Objekt O mit Hand H von Ort L
pickup(A, O, H, L)
Preconditions:  Agent(A), Hand(H), Location(L),
Empty(H), On(O,L), At(A,L)
Effects:        !On(O,L), !Empty(H), InHand(O,H)

// Agent A platziert Objekt O mit Hand H auf Ort L
putdown(A, O, H, L)
Preconditions:  Agent(A), Hand(H), Location(L),
InHand(O,H), At(A,L)
Effects:        On(O,L), Empty(H), !InHand(O,H)

// Agent A bewegt sich von Ort L zu Ort M
move(A, L, M)
Preconditions:  Agent(A), Location(L), Location(M), At(A,L), !(L = M)
Effects:        !At(A,L), At(A,M)
```

Hinweis: Ein Ausrufezeichen negiert ein Prädikat.

1. Geben Sie die kürzeste Aktionssequenz an, die den initialen Weltzustand in den Zielzustand überführt. Achten Sie dabei auf die korrekte Parametrisierung der Aktionen.

2. Erstellen Sie einen Planungsoperator

   `moveAndPickup(A, O, H, L, M)`,

   der den Agenten `A` von Ort `L` nach `M` bewegt und von dort das Objekt `O` mit der Hand `H` aufnimmt. Verwenden Sie dabei die STRIPS-Notation und geben Sie geeignete Vorbedingungen und Effekte an.

3. Wie lautet der Weltzustand nach Ausführung der Aktion

   `moveAndPickup(R, P, G, S, T)`

   bei dem vorgegebenen Initialzustand?

Aufgabe 2 (Programmierung, A*)

In dieser Aufgabe sollen Sie den A*-Algorithmus für die mobile Plattform des Roboters ARMAR-III implementieren. Es wird dafür ein Grundgerüst bereitgestellt, in dem Sie die Lücken füllen müssen. Gehen Sie wiefolgt vor:

1. *Installieren Sie Simox.*

   Wir empfehlen die Installation unter Ubuntu:

   `https://gitlab.com/Simox/simox/wikis/Installation-Source-Ubuntu`

   Alternativ können Sie unter Windows 10 auch das Linux-Subsystem verwenden (siehe dazu Simox-unter-Windows.pdf).

2. *Kompilieren Sie den bereitgestellten Skeleton-Code.*

   In der Datei `motion-planning-simox.zip` sind alle notwendigen Projekt- und Quelldateien für diese Aufgabe. Entpacken Sie diese Dateien. Es handelt sich um ein CMake-Projekt, das Sie wie folgt bauen können:

   - `cd build`
   - `cmake ..`
   - `make`

   Es wird eine ausführbare Datei `build/PathPlanning` erzeugt, die eine GUI mit dem Roboter ARMAR-III und eine Szene bestehenden aus mehreren Hindernissen lädt.

3. *Implementieren Sie den A*-Algorithmus als Pseudocode.*

   In der Datei `Planner/AStarPlanner.cpp` sind einige Teile des A*-Algorithmus nicht implementiert. Diese sind im Code durch die folgende Markierung kenntlich gemacht:

```
//////////////////////////////////////////////
// INSERT CODE HERE
//////////////////////////////////////////////
```

Die Pseudocodes können Sie direkt in der Quellcodedatei als Kommentare einfügen.

4. *Implementieren Sie nun den A*-Algorithmus in C++.*

   Über die GUI können Sie ihren Code ausführen, indem Sie auf den Button “Run Planner” klicken.

Aufgabe 3 (Programmierung, RRT)

In dieser Aufgabe werden Sie den RRT-Algorithmus für die mobile Plattform des Roboters ARMAR-III implementieren. Es wird dafür ein Grundgerüst bereitgestellt, in dem Sie die Lücken füllen müssen. Gehen Sie wiefolgt vor:

1. *Installieren Sie Simox.*

   Wir empfehlen die Installation unter Ubuntu:

   `https://gitlab.com/Simox/simox/wikis/Installation-Source-Ubuntu`

   Alternativ können Sie unter Windows 10 auch das Linux-Subsystem verwenden (siehe dazu Simox-unter-Windows.pdf).

2. *Kompilieren Sie den bereitgestellten Skeleton-Code.*

   In der Datei `motion-planning-simox.zip` sind alle notwendigen Projekt- und Quelldateien für diese Aufgabe. Entpacken Sie diese Dateien. Es handelt sich um ein CMake-Projekt, das Sie wie folgt bauen können:

   - `cd build`
   - `cmake ..`
   - `make`

   Es wird eine ausführbare Datei `build/PathPlanning` erzeugt, die eine GUI mit dem Roboter ARMAR-III und eine Szene bestehend aus mehreren Hindernissen lädt.

3. *Implementieren Sie den RRT-Algorithmus als Pseudocode.*

   In der Datei `Planner/RRTPlanner.cpp` sind einige Teile des RRT-Algorithmus nicht implementiert. Diese sind im Code durch die folgende Markierung kenntlich gemacht:

   ```
   //////////////////////////////////////////////
   // INSERT CODE HERE
   //////////////////////////////////////////////
   ```

   Die Pseudocodes können Sie direkt in der Quellcodedatei als Kommentare einfügen.

4. *Implementieren Sie nun den RRT-Algorithmus in C++.*

   Sie können Ihre Implementierung über die GUI testen, indem Sie die Combobox auf “RRT Planner” stellen und dann auf “Run Planner” klicken.