Hauptprojekt: Roboterhockey

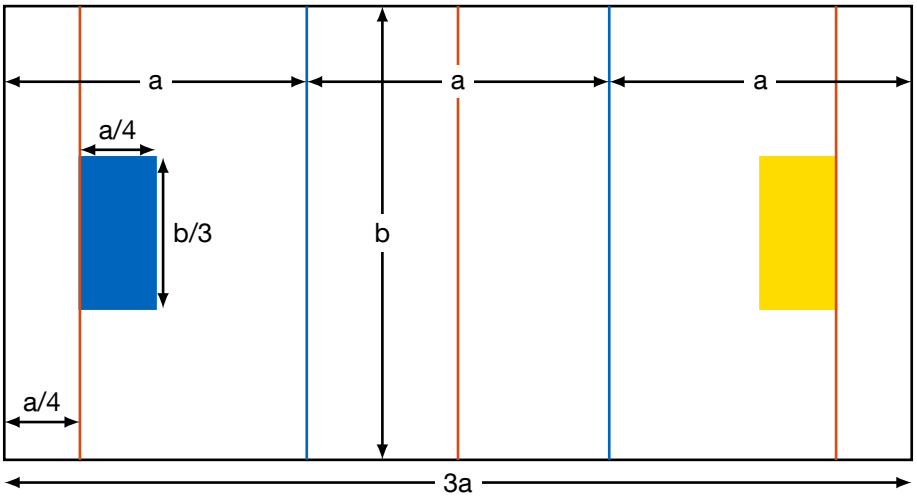
Leistungskurs C++

Martin Knopp

28. November 2017

Im Hauptprojekt des Leistungskurses C++ spielen je zwei Roboter gegeneinander Hockey. Das Ziel ist es dabei so schnell wie möglich 3 Pucks der eigenen Farbe in das gegnerische Tor zu schieben. Voraussetzung dafür ist die Erkennung des Spielfeldes sowie der eigenen Teamfarbe.

1 Das Spielfeld



Das Spielfeld wird durch die Längen a und b definiert. Der Bereich vor den blauen Linien ist die jeweilige Verteidigungszone. Die Ecken bzw. Schnittpunkte der Spielfeldmarkierungen sind zusätzlich durch grüne Pfosten markiert.

2 Phasen des Spiels

Dieses Kapitel beschreibt kurz die einzelnen Phasen des Spiels; genauere Informationen wie die Schnittstelle zu Angelina funktioniert und welche Informationen mit ihr ausgetauscht werden (müssen) finden sich auf den Vorlesungsfolien!

2.1 Initialisierung

Die Software zur Kontrolle der Roboter wird gestartet (via SSH), Angelina wird über den erfolgreichen Start informiert. Der Roboter darf sich in dieser Phase nicht bewegen.

2.2 Erkennung Spielfeld und Teamfarbe

Der Start dieser Phase wird durch Angelina signalisiert. Die Roboter dürfen sich frei in ihrer jeweiligen Verteidigungszone bewegen. Ziel in dieser Phase ist es die Längen a und b zu bestimmen, das Verhältnis (a/b) wird Angelina mitgeteilt, sie antwortet darauf mit den wahren Werten für a und b. Desweiteren muss in dieser Phase auch die Teamfarbe bestimmt und an Angelina gemeldet werden.

Diese Phase endet, bzw. die nächste Phase startet wiederum durch Angelina (auf das entsprechende Signal warten!)

2.3 Das Hockeyspiel

Das eigentliche Spiel. Beide Roboter dürfen sich jetzt frei im gesammten Spielfeld bewegen. Ziel ist es die drei Pucks der eigenen Farbe im gegnerischen Tor zu plazieren. Die gegnerischen Pucks aus dem eigenen Tor wieder herausschieben ist ein erlaubter Spielzug, aber innerhalb der gegebenen fünf Minuten für das gesamte Spiel schwierig.

Nach Ende des Spiels (GameOver-Signal) dürfen sich die Roboter nicht mehr bewegen.

3 Fragen und Antworten / Tipps

3.1 Ist es egal welchen Roboter ich verwende?

Grundsätzlich ja, die Homeverzeichnisse und Benutzer sind aber individuell pro Roboter und werden nicht synchronisiert. Ihr müsst Änderungen also selbst abgleichen, typischerweise über euer git-Repository.

3.2 Muss ich wirklich jedesmal dieses komplizierte Passwort eingeben, wenn ich mich mit dem Roboter verbinden will?

Nein, ihr könnt es entweder auf dem Roboter mit dem Befehl `passwd` ändern (Achtung: Müsst ihr auf *jedem* Roboter machen, siehe auch die vorangehende Frage), oder – besser, komfortabler und sicherer – ihr nutzt SSH-Keyfiles[1]. Komfortabler vor allem deshalb, weil ihr sie auch zusammen mit GitLab verwenden könnt und dann auch für git pull/push keine Passwörter mehr braucht. Um den, von eurem lokalen Rechner zwischengespeicherten, Key den git-Befehlen auf dem Roboter zur Verfügung zustellen, verbindet euch mit `ssh -A <nutzer>@<roboter>`.

[1] Tutorial: `https://www.thomas-krenn.com/de/wiki/OpenSSH_Public_Key_Authentifizierung_unter_Ubuntu`

3.3 Wie verbinde ich ROS im Netzwerk?

ROS ist komplett netzwerkfähig, ihr müsst pro Rechner bzw. Roboter nur zwei Umgebungsvariablen passend setzen.

ROS_HOSTNAME setzt den Hostnamen des jeweiligen Rechners, in der automatischen Variante fehlt uns die passende Domain, deswegen muss dieser Wert manuell gesetzt werden. Ein passender Eintrag für die .bashrc oder bevor man rosrun / roslaunch ausführt ist `export ROS_HOSTNAME=$(hostname --fqdn)`. Überprüfen könnt ihr die Variable mit `echo $ROS_HOSTNAME`, das Ergebnis sollte so ähnlich wie „hpc15.clients.eikon.tum.de“ aussehen.

ROS_MASTER_URI setzt die Adresse unter der roscore erreichbar ist und muss auf allen beteiligten Rechnern bzw. Robotern identisch sein. Er wird beim Start von roscore ausgegeben und kann dort kopiert/übernommen werden. Ein passender Befehl für die anderen Rechner sieht ungefähr so aus `export ROS_MASTER_URI=http://hpc15.clients.eikon.tum.de:11311/`. Den Hostnamen müsst ihr natürlich entsprechend anpassen, wichtig ist die .clients.eikon.tum.de-Domain, damit sich die Rechner im Netz finden können.

3.4 Wie verbinde ich die virtuelle Maschine so mit dem WLAN, dass ROS funktioniert?

Das Netzwerk eurer VM muss von NAT auf Bridged umgestellt werden. Klickt dazu in VirtualBox auf *Ändern* und wählt den Punkt *Netzwerk*. Hier ändert ihr bitte *Angeschlossen an:* von *NAT* auf *Netzwerkbrücke*. Damit wird das Feld *Name:* aktiv, hier müsst ihr die Netzwerkkarte wählen, mit der euer Notebook sich mit dem LDV-CPP-WLAN verbindet. Abschließend müsst ihr unbedingt noch bei *MAC-Adresse* auf das Icon mit den rotierenden Pfeilen klicken, um eine neue MAC-Adresse zu generieren. Wenn ihr das vergessen solltet, wird zu sehr schwer diagnostizierbaren Netzwerkproblemen kommen, bei denen ihr und andere nicht mehr in Netz kommen werden.

3.5 Das WLAN ist unbenutzbar langsam

Ihr teilt euch alle ein gemeinsames Netzwerk, das heißt ihr müsst selber ein wenig darauf achten mit den Ressourcen angemessen umzugehn. Dabei spielt vor allem eine Rolle, wie ihr die Kameradaten der Kinect behandelt. In der Standardeinstellung publiziert die Kinect-Node ein unkomprimiertes RGB-Bild, wenn ihr dieses Topic auf einem anderen Rechner abonniert, steigt ab dem zweiten Roboter das WLAN aus. Ihr müsst entweder die Bildverabeitung auf dem Roboter selbst erledigen oder ein komprimiertes Bild übertragen.

Ansonsten gilt: Bitte für Streaming-Video und andere große Downloads nicht das LDV-CPP-WLAN nutzen.

3.6 Mir fehlt ROS-Komponente X, Software Y, Tool Z …

Die installierten Programme bzw. ROS-Komponenten sind ein gewisser Standardsatz, der auf unseren Erfahrungen und Vorlieben bzw. denen anderer Fakultätsmitglieder basiert. Falls ihr euer Lieblingstool vermisst oder im ROS-Wiki eine interessante Komponente gefunden habt, müsst ihr die nicht selber installieren. Gebt uns einfach kurz per Mail Bescheid, wir kucken dann, dass wir das über das zentrale Management auf allen Rechnern (oder Robotern) zur Verfügung stellen.