Computational Geometry

Ausarbeitung für das Praktikum

Studienarbeit von
Maximilian Hempe
Roland Wilhelm

Dozent: Dr. Fischer
Hochschule München
Master Informatik
2. Juli 2013

Inhaltsverzeichnis

Abbildungsverzeichnis

Listingsverzeichnis

1 Einleitung

Diese Studienarbeit beschreibt das Praktikum zur Vorlesung Embedded- und Echtzeitbetriebssysteme. Die Studenten sollen darin den Umgang mit Betriebssystemen von Embedded Systemen kennenlernen. Die grundlegenden Werkzeuge im Praktikum sind ein BeagleBoard, das Echtzeitbetriebssystem QNX und die dazugehörige Entwicklungsumgebung QNX Momentics. Auf dieser Basis werden im Verlauf mehrere Aufgaben erarbeitet. Diese vertiefen die Themen der Vorlesung und gehen auf spezielle Sachverhalte intensiver ein. Ziel ist es zyklisch Threads zu starten und mit Hilfe einer selbstentwickelten zeitverschwende Funktion das Embedded System auszulasten. Diese Auslastung wird mit Hilfe von Momentics dargestellt und es kann analysiert werden, ob die Threads ihre Echtzeitbedingungen einhalten können. Abschließend wird der QNX Kernel noch optimiert, sodass er lediglich die unbedingt nötigen Module enthält.

2 Aufgabe 1

3 Aufgabe 2

4 Aufgabe 3

5 Maximaler Kreis in einer konvexen Hülle

In dieser Aufgabe 4 werden für zwei vorgegebene konvexe Polygone der größtmögliche einbeschreibbare Kreis mit Linearer Programmierung berechnet. Die vorgegebenen Polygone sind in den Dateien *Polygon.txt* und *testpolygon.txt* abgespeichert. Nachdem das Problem beschrieben wurde, wird auf Basis der Testdatei *testpolygon.txt* ein lineares Programm definiert und hergeleitet. Dieses lineare Programm wird schlussendlich auf das Polygon in der Datei *Polygon.txt* angewendet und das Ergebnis dargestellt.

5.1 Herleitung Problem

Gegeben ist ein konvexes Polygon $P \subseteq \mathbb{R}^2$. Bei dem Problem in dieser Aufgabe suchen wir nach dem größtmöglichen Kreis $K \subseteq \mathbb{R}^2$, der vollständig in P enthalten ist; das heißt, es gilt $K \subseteq P$ wobei der Radius von K maximal ist. Damit der Kreis K mit Mittelpunkt $m := (m_x, m_y)$ und Radius r komplett in P enthalten ist, müssen folgende Bedingungen erfüllt sein:

- Der Punkt m hat mindestens den Abstand r von allen Strecken.
- Der Punkt m liegt innerhalb des konvexen Polygons P.

Um den Abstand r des Mittelpunktes m von einer Strecke P_i zu berechnen, wird die folgende Formel genutzt.

$$r = N_{0i} * [m - P_i]$$

Dabei beinhaltet die Matrix N_{0i} in der Zeile i den jeweiligen Normaleneinheitsvektor für die Strecke P_i.

Der Kreis K liegt also genau dann vollständig in P, wenn folgende Ungleichung für alle Strecken erfüllt ist.

$$N_{0i} * [m - P_i] \geq r, i = 1, ..., n$$

5.2 Lösung durch lineare Programmierung

Das Ziel ist es den größten Kreis zu finden. Wir definieren ein lineares Programm mit den Variablen m_x, m_y und r, indem die Variable r unter der Nebenbedingung

$$N_{0i} * [m - P_i] \geq r, i = 1, ..., n$$

maximiert wird. Die entsprechende Zielfunktion wird definiert als

$$f := \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Zur Lösung des lineare Problems wird das Programm **MATLAB** (MATrix LABoratory) verwendet. Nachfolgend werden nur die wichtigsten Programmteile dargestellt und erklärt, dass komplette Programm **incircle.m** kann in der mitgelieferten Datei unter **matlab/incircle.m** begutachtet und getestet werden. Die jeweiligen konvexen Polygone werden durch das Programm als Matrizen in den **Workspace** importiert und stehen anschließend im **Command Window** zur Verfügung. Die Matrizen für die konvexen Polygonen P haben dabei folgende Strukturen:

$$P = \begin{bmatrix} 0 & 0 \\ 10 & 0 \\ 10 & 10 \\ 0 & 10 \end{bmatrix}$$

Die jeweiligen Zeilen der Matrix bilden den Startpunkt einer Strecke ab. Die Anzahl der Zeilen ist zugleich die Anzahl der zur Verfügung stehenden Strecken indem der Kreis eingebettet werden soll. Die Spalte 1 gibt die x-Werte und die Spalte 2 die y-Werte eines Startpunktes wieder. Um aus den einzelnen Punkten ein konvexes Polygon zu erhalten, werden die Startpunkte miteinander verbunden; also der Punkt in der Zeile wird mit dem Punkt in der Zeile 2 verbunden u. s. w. , der letzte Punkt (hier Zeile 4) wird mit dem Punkt in der Zeile 1 verbunden.

Das Listing 4 zeigt die Erstellung einer konvexen Hülle. Durch den MATLAB Befehl *convhulln(xy)* wird aus den vorgegebenen Strecken (xy) eine konvexe Hülle erzeugt und die entsprechenden Indizes zurückgegeben. Dadurch können dann in Zeilen 2 und 3 die jeweiligen Start- und Endpunkte einer Strecke bestimmt werden.

```
1  ecken = convhulln(xy);
2  A = xy(ecken(:,1),:);
3  B = xy(ecken(:,2),:);
```

Listing 1: Erstellen einer konvexen Hülle mit MATLAB

Nachdem die Start- und Endpunkte für alle verfügbaren Strecken berechnet worden sind, werden für diese in den Zeilen 1 - 3 der dazugehörige Normaleneinheitsvektor berechnet. Um sicherzustellen, dass alle Normaleneinheitsvektoren zum Mittelpunkt m zeigen, findet in den Zeilen 3 - 6 eine Überprüfung statt. Hier werden aus allen Startpunkten der x- und y-Mittelwert berechnet und als sicherer Mittelpunkt M_0

abgespeichert. Ist bei der anschließenden Subtraktion ein negativer Normaleneinheitsvektor vorhanden, wird dieser um 180 Grad gedreht.

```
1 N_p = (B - A) * [0 1; -1 0];
2 Betrag = sqrt(sum(N_p.^2,2));
3 N_p = N_p./[Betrag, Betrag];
4 M_0 = mean(A,1);
5 index = sum(N_p.*bsxfun(@minus, M_0, A), 2) < 0;
6 N_p(index,:) = -N_p(index, :);
```

Listing 2: Berechnung des Normaleneinheitsvektor mit MATLAB

Nachdem alle erforderlichen Daten berechnet worden sind, wird nun unter Einhaltung der Nebenbedingung, dass Ungleichungssystem aufgestellt. Mit dem MATLAB Befehl **linprog** kann ein Ungleichungssystem in der Form

$$A * x \leq b$$

gelöst werden. Dieser Befehl berechnet das Minimum eines linearen Problems, da hier aber ein maximaler Radius r gesucht wird, muss die Zielfunktion f entsprechend angepasst werden. Soll ein Maximum anstatt eines Minimum berechnet werden, müssen alle Koeffizienten der Zielfunktion f negiert werden. Die angepasste Zielfunktion lautet

$$f := \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

Durch Umformung der Nebenbedingung

$$N_{0i} * [m - P_i] \geq r$$

erhält man das angepasste Ungleichungssystem

$$-N_0 * m + r \leq -N_0 * P$$

für das Programm MATLAB. Das Listing 5 berechnet in den Zeilen 1 und 2 jeweils die linke Seite sowie rechte Seite. In den Zeilen 3 und 4 wird die dazugehörige Zielfunktion definiert.

```
1 b = -sum(N_p.*A, 2);
2 A = [-N_p.*ones(strecken_nr, 2), ones(strecken_nr, 1)];
3 f = zeros(3, 1);
4 f(3) = -1;
```

Listing 3: Erstellung der Ungleichung mit MATLAB

Die erzeugten Parameter werden in der ersten Zeile dem Befehl *linprog* übergeben, dieses liefert uns dann in den entsprechenden Rückgabewerten die Lösung des linearen Problems zurück.

```
1 [result, fval, exitflag, output] = linprog(f, A, b);
2 C = result(1:2)';
3 R = result(3);
```

Listing 4: Starten der Berechnung mit MATLAB

5.3 Ergebnisse

Im Abschnitt 5.2 wurde das entwickelte MATLAB-Programm schrittweise vorgestellt und erklärt. Hier werden nun die entsprechenden Lösungen für die Dateien *Polygon.txt* und *testpolygon.txt* vorgestellt.

Wird das Polygon in der Datei *testpolygon.txt* dem MATLAB-Skript *incircle* übergeben, werden die Werte $m = (5.0, 5.0)$ für den Mittelpunkt und $r = 5.0$ für den Radius des Kreises berechnet. Die Abbildung 1 bildet das Ergebnis grafisch ab.

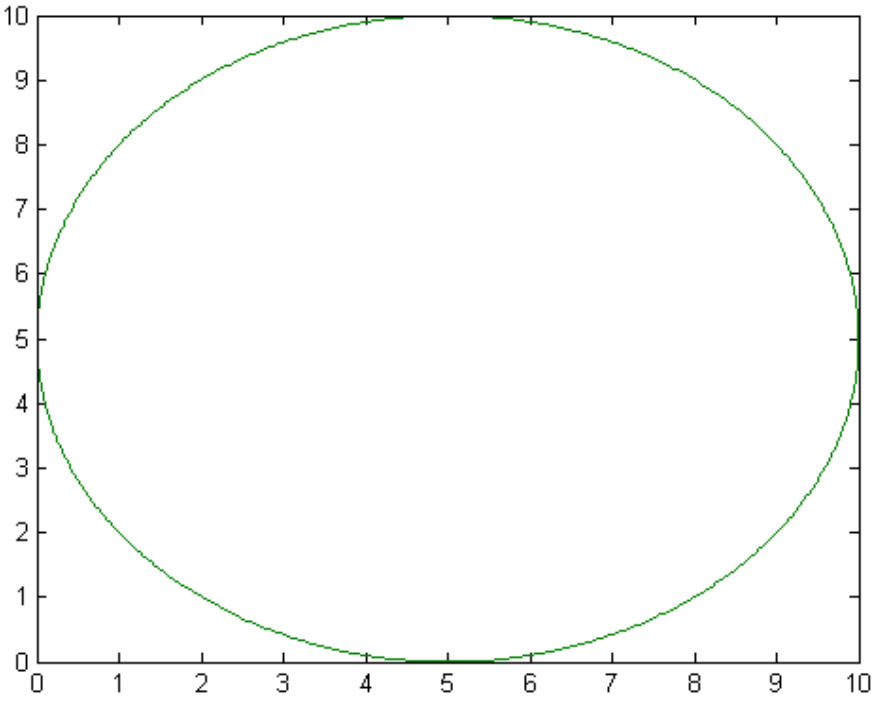

Abbildung 1: Testpolygon mit Kreis

Mit der Datei *testpolygon.txt* werden die Werte $m = (472.5705, 476.6642)$ für den Mittelpunkt und $r = 438.5922$ für den Radius des Kreises berechnet. Die entspre-

chende Grafik wird in der Abbildung 2 abgebildet. Das konvexe Polygon wird blau und der größtmögliche einbeschreibbare Kreis grün dargestellt.

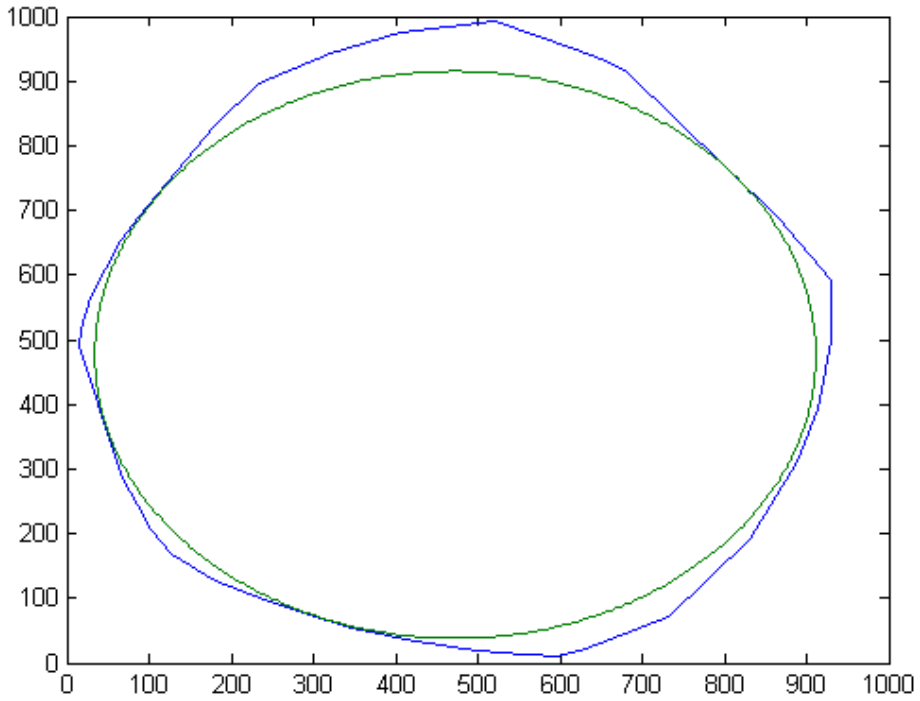

Abbildung 2: Polygon mit Kreis

6 Berechnung von konvexen Hüllen mit qhull

7 Fazit

7.1 Zusammenfassung

7.2 Lessons Learned

Literatur