Projekt 2

Kornknecht

Betreuer: Prof. Dr. H. Litschke

Projekt 2

WS 2020

Team Projekt 2

https://github.com/DavidTaube/Projekt2

Gliederung

1. Aufgabe 2
2. Aufgabe 1
3. Aufgabe 3
4. Aufgabe 4
5. Aufgabe 5

Aufgabe 2

- Quasi der „Kernpunkt“: Messen / Berechnen von Abständen der Marker voneinander.
- Daraus: Bestimmung (und Speicherung) von deren Positionen am Rand des Arbeitsfeldes.
- MÖGLICHE Annahme: Rechteck. (Oder handeln wir uns damit eher Ungenauigkeiten ein?)
    - Wichtige Zusatzinfo: Marker 1-4 sind immer in den Ecken

Ansatz

- Eine Tafel in einem Block hat immer einen Nachbarn sofern die Anzahl der Tafeln > 1 ist
- Darauf können wir eine „Kette“ bilden
  - 1 - 6 - 7 - 2 - 9 - 12 - 3 - 10 - 11 - 4 - 22 - 23
- Die Tafeln befinden sich auf den Seiten eines Rechtecks
- Wenn 1-4 die Ecken sind, können wir an diesen Positionen die Koordinatenachsen wechseln
  - 1 - 6 - 7 - 2
23 9
22 12
4 - 11 - 10 - 3

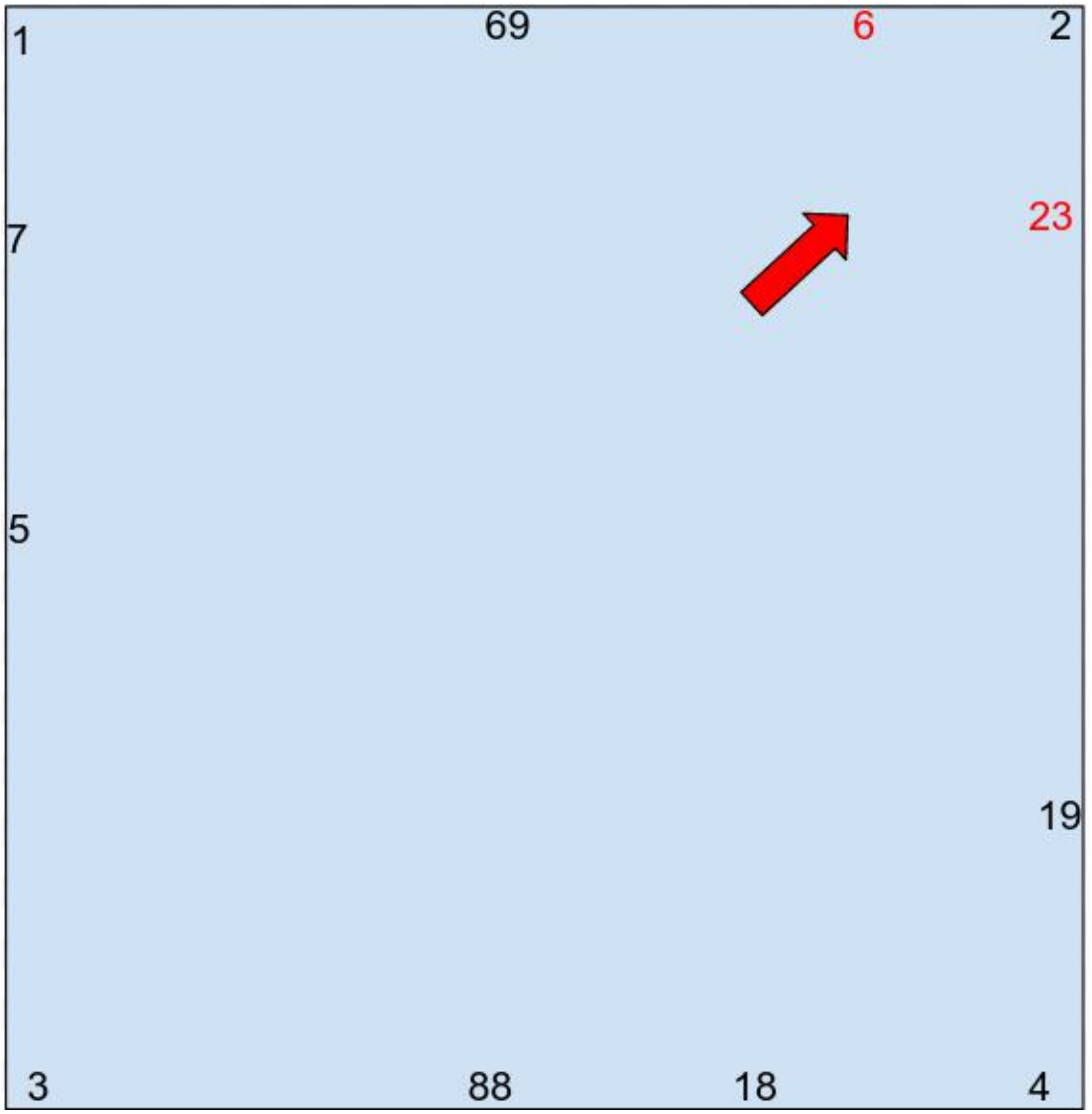
Programmaufbau

- Init
  - Einlesen
  - Blocks in eine Datenstruktur umwandeln
- Divergenz berechnen
- Werte Normalisieren
  - Angleichen der Kamera 0 auf Kamera 1
- rechten Nachbarn für jeden Marker berechnen
  - Mehrfacherkennungen mitteln und aussortieren
  - anhand der Y Achse die Position im Raum analysieren (von links nach rechts)
- Koordinaten berechnen
  - über die Kette iterieren
  - if (Tafel == Ecke) { Winkel += 90° }

Probleme

- Die Koordinaten sind nicht korrekt
  - Messfehler in der Kamera
  - Wir kommen nie auf 0,0 zurück
- Daten unvollständig
  - Es wurde nicht jede Tafel gesehen
- Sichtweite der Kamera
  - Zwischen 2 Tafeln kann eine verschwinden, welches dazu führt, dass der Nachbar mit falschen Daten überschrieben wird

Probleme



Ausblick / mögliche Problemlösungen

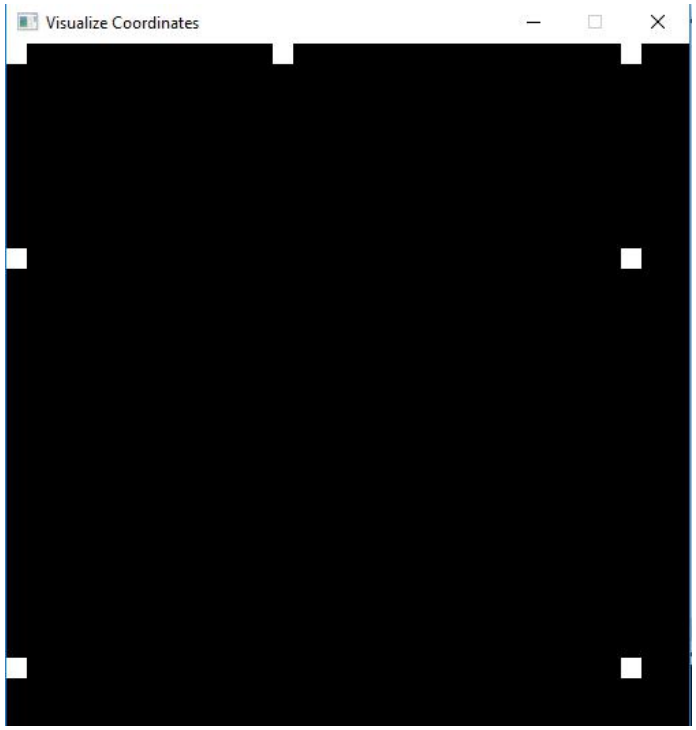
- Memory Funktion / Häufigkeitsanalyse der Nachbarn
  - Der Nachbar mit der höchsten absoluten Häufigkeit wird als Nachbar eingetragen
- LoopCycles durchbrechen
  - Die Stellen, an denen eine Loop entsteht, kappen

Aufgabe 1

- Visualisierung der aus den Logfile entnommenen Koordinatenwerte relativ zu den beiden Kamera-Koordinatensystemen
- Visualisierung anhand der resultierenden Koordinaten aus der Aufgabe 2

Visualisierung

- Funktion: „initWindow()“
- Mittels SDL2 Window und Renderer initiiert
- Koordinaten in Auflösung mit Faktor 10 multipliziert
- Punkte um Faktor 3 skaliert



Aufgabe 3

- Bestimmung der eigenen Position aus bekannten Abständen von den Markern
- Bestimmung der eigenen Position anhand der resultierenden Koordinaten aus der Aufgabe 2 und Erzeugung von Pseudo Scans

- 1 Scan beinhaltet 2 Boards
- Boards beinhalten Position und Abstand zum Kornknecht
- Mehrere Scans durchgeführt

```
for (int x = 1; x < 5; x++) {
    FakeLogBlock block;

    //insert data in first board
    FakeGeoboard board1;
    board1.Panel = x;
    board1.X = posOfFirstBlock[x-1][0];
    board1.Y = posOfFirstBlock[x-1][1];
    board1.distToKornknecht = posOfFirstBlock[x-1][2];
    block.boards.push_back(board1);

    //insert datain second board
    FakeGeoboard board2;
    int neighbour = Coordinates[x].rightNeighbour;
    board2.Panel = neighbour;
    board2.X = Coordinates[neighbour].X;
    board2.Y = Coordinates[neighbour].Y;

    //cheack if boardpanel is 12
    if (board2.Panel > 7) {
        board2.distToKornknecht = posOfFirstBlock[7][2];
    }
    else{
        board2.distToKornknecht = posOfFirstBlock[neighbour - 1][2];
    }
    block.boards.push_back(board2);

    FakeBoards.push_back(block);
```

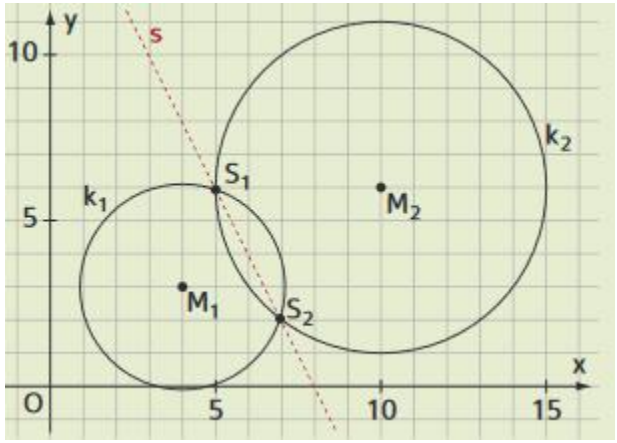
Ansatz 1

- Bildung eines Dreiecks mittels der Abstände von Boards und Kornknecht zueinander
- Berechnung der X und Y Koordinaten

```
for (int i = 0; i < FakeBoards.size(); i++) {

    //cheack if 2 boards were found
    if (FakeBoards[i].boards.size() < 2) {
        continue;
    }

    double a, b, c, alpha, beta, gamma;

    // get length of dist to create triangle
    a = FakeBoards[i].boards[0].distToKornknecht;
    b = FakeBoards[i].boards[1].distToKornknecht;
    c = Coordinates[FakeBoards[i].boards[0].Panel].distNeighbour;

    // check if sides creat triangle
    if (a + b <= c || a + c <= b || b + c <= a) {
        std::cout << "Position couldnt be calculated(cant create triangle)" << std::endl;
        continue;
    }

    //calc angle
    alpha = acos((((a * a) - (b * b) - (c * c)) / (-2 * b * c)));
    double alphagrad = alpha * (180/PI);

    beta = asin((b / a) * (sin(alpha)));
    double betagrad = beta * (180/PI);

    gamma = 180 - alphagrad - betagrad;

    //kornknecht.X = (c*(tan(beta)))/(1+tan(beta));
    //kornknecht.Y = kornknecht.X * tan(alpha);

    //calc X/Y position
    kornknecht.X = b*(cos(alpha));
    kornknecht.Y = b*(sin(alpha));
```

Ansatz 2

- Erzeugen von 2 Kreisen und berechnen der Position mittels einer Kreisgleichung




```
double ax0, ax1, ax2, ax3, ay0, ay1, ay2, ay3;
double bx0, bx1, bx2, bx3, by0, by1, by2, by3;

//rechte Seite
double ar = FakeBoards[i].boards[0].distToKornknecht * FakeBoards[i].boards[0].distToKornknecht;
double br = FakeBoards[i].boards[1].distToKornknecht * FakeBoards[i].boards[1].distToKornknecht;
double y,yx0, yx1;

//ohne x/y
ax0 = FakeBoards[i].boards[0].X * FakeBoards[i].boards[0].X;
ay0 = FakeBoards[i].boards[0].Y * FakeBoards[i].boards[0].Y;

// mit x/y
ax1 = FakeBoards[i].boards[0].X * 2;
ay1 = FakeBoards[i].boards[0].Y * 2;

//ohne x/y
bx0 = FakeBoards[i].boards[1].X * FakeBoards[i].boards[1].X;
by0 = FakeBoards[i].boards[1].Y * FakeBoards[i].boards[1].Y;

//mit x/y
bx1 = FakeBoards[i].boards[1].X * 2;
by1 = FakeBoards[i].boards[1].Y * 2;

//y
y = (-ay1) - (-by1);

if (y == 0) {
    //x0
    yx0 = (ar - br - (ax0 - bx0) - (ay0 - by0));
    //x1
    yx1 = (0 - ((-ax1) - (-bx1)));
}
else {
    //x0
    yx0 = (ar - br - (ax0 - bx0) - (ay0 - by0)) / y;
    //x1
    yx1 = (0 - ((-ax1) - (-bx1)) / y);
}
//qudratische Gleichung
double cx2 = 1 + (yx1*yx1);
double cx1, cx0;
if (yx0 < 0) {
    cx1 = (-ax1) - (2 * (yx1 * yx0)) + (ay1 * yx0);
}
else {
    cx1 = (-ax1) + (2 * (yx1 * yx0)) + (ay1 * yx0);
}
cx0 = ax0 + ay0 + (yx0 * yx0) + yx0 - ar;

cx1 = cx1 / cx2;
cx0 = cx0 / cx2;

double p = cx1;
double q = cx0;

double L1X, L1Y, L2X, L2Y;

L1X = ((p / 2) * (-1)) + (sqrt(((p / 2) * (p / 2) - q)));
L2X = ((p / 2) * (-1)) - (sqrt(((p / 2) * (p / 2) - q)));

L1Y = (L1X * L1X) + (cx1 * L1X) + cx0;
L2Y = (L2X * L2X) + (cx1 * L2X) + cx0;

if (L1X < 0 || L1Y < 0) {
    kornknecht.X = L2X;
    kornknecht.Y = L2Y;
}
else if (L2X < 0 || L2Y < 0) {
    kornknecht.X = L1X;
    kornknecht.Y = L1Y;
}*/
```

Probleme

- Vorgegebener Abstand zum Kornknecht auf berechneten Position nicht erfüllt

Aufgabe 4

- Statistische Analyse von Messfehlern aus den Daten des Logfiles

Ansatz

- Befinden sich Tafeln doppelt in einem Frame, berechnen von
  - Extrempunkten
  - Durchschnitt
- Berechnen der Spannweite, welche durch abweichende Messwerte entstehen

Programmaufbau

- Komplette Fehlerermittlung für jeden Frame
  - Iteration über das gesamte Logfile
  - Speichern der Abweichungen im struct
- Maximum ermitteln
- Minimum ermitteln
- Durchschnitt berechnen
- Spannweite berechnen
  - Systemrelevante Funktionen mit Extremwerten durchlaufen

Probleme

- Funktionsparameter
  - Unklarheiten in der Aufgabenstellung führten zu nicht abgesprochenen Methodenparametern, weshalb die Spannweite vorerst nicht mit den gegebenen Funktionen berechnet werden kann

Ausblick / mögliche Problemlösungen

- Anpassung von Funktionsparametern
  - Damit die Extremwerte zum Aufruf der Methode benutzt werden können
- Erstellen von Diagrammen
  - Anzeigen der gesichteten Tafeln mit Abständen über Zeit
  - Abweichungen im Diagramm erkennen

Aufgabe 5

- Messfehler Teil 2: Überlegungen zu Messfehlern anhand von Beispielbildern der Kameras
- Mit welchen (Un-)Genauigkeiten muss prinzipiell also in den Koordinaten-Daten gerechnet werden?

- Ausführlichere Ausarbeitung im Repository „Task_5.pdf“

Ergebnisse

- durch Personen, Gegenstände und Staub kann die Sicht beeinträchtigen bzw. Messfehler generiert werden
- (Un-)Günstige Wahl der Belichtungszeit der Kamera, bei einem zu geringen Abstand zu den Schildern können diese überbelichtet sein
- ein schiefer Haufen kann zum herunterrutschen des Kornknechts führen und somit kann dieser aus dem Arbeitsbereich heraus geraten
- Schilder könnten umkippen bzw. umgefahren werden und somit zu Lücken in der Begrenzung führen



Fragen?