Deutsche
Sporthochschule Köln
German Sport University Cologne

Technology
Arts Sciences
TH Köln

Bachelorarbeit

„Visualisierung von Daten aus der Fußballbundesliga 2011 / 2012 durch Heatmap“

Vorgelegt an der
Technische Hochschule Köln, Campus Gummersbach
Fakultät für Informatik und Ingenieurwissenschaften
und
Deutsche Sporthochschule Köln
Institut für Kognitions- und Sportspielforschung

Im Studiengang Wirtschaftsinformatik

Ausgearbeitet von
Alexandre Wildt Graziani, Matr.-Nr.01104100311

Erstprüfer: Prof. Dr. Heide Faeskorn-Woyke.
Zweitprüfer: Phd Robert Rein.

Köln, Dezember 2016

Kurzfassung

Das Ziel der vorliegenden Bachelorarbeit ist die Konzeption und Entwicklung eine Software, mit den Daten aus der Fußball-Bundeliga 2011–2012 eine Heatmap zu erzeugen. Die Software soll die Position vor einem Offensivspiel zu verschiedenen Zeitpunkten des Spieles veranschaulichen.
Bei der Arbeit wurden zwei Cluster-Methoden in Betracht genommen. *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) und der Kerndichtschätzer (*kernel density estimation*). DBSCAN ist ein Cluster-Algorithmus und wurde mit dem Sofware-Tool WEKA untersucht. Der Kerndichtschätzer ist ein statistisches Verfahren zur Schätzung einer Dichte.
Nach der Analyse von Daten mit dem Algorithmus wurde die Software in der Deutschen Sporthochschule Köln implementiert. Die Software untersucht die Bildung von Clustern bzw. die Dichte mit der Absicht, ein Verhalten oder Muster vor einem Offensivspiel zu erkennen.

Abstract

The purpose of this bachelor thesis is the conception and development of a software (application) to generate a heatmap based on the data of Association Football German Bundesliga 2011-2012. The software should demonstrae the positon before an offensive play at different points in time.

Two cluster-methods were taken into consideration: /Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and the kernel density estimation. DBSCAN is a cluster algorithm that was analyzed with the software-tool WEKA. Kernel density estimation is a statistical procedure(method) for density estimation.

After the analysis of data with the algorithm the software was implemented at the German Sport University Cologne. The software examines the formation of clusters or the density with the intention of recognizing a pattern or behavior before an offensive play.

Inhaltsverzeichnis

1 Einleitung

Das *Institut für Kognitions- und Sportspielforschung* an der *Deutschen Sporthochschule Köln* beschäftigt sich mit der Erforschung von Psychologie und Bewegung in verschiedenen Arten des Hochleistungsports wie Fußball, Volleyball, Handball, Hockey, Baseball, Basketball und Tischtennis. Von besonderem Interessen für das Institut sind die neuen Analysemöglichkeiten anhand von Daten aus Sportspielen mit Hilfe von Mustererkennung, Klassifikationen und Data Mining. In Rahmen dieser Bachelorarbeit wird eine Software zur Analyse von Daten der *Deutschen Fußball Liga* ab Saison 2011/2012 und deren Visualisierung entwickelt. Das Ziel der Analyse ist es festzustellen, ob die Positionen eines offensiven Spielzuges vor einer Torschussgelegenheit oder einem erzielten Tor in einem regelmäßigen Punktecluster oder Raster zu erkennen sind. Die Punktdichte wird auf Grundlage der Anzahl von Punkten auf dem Fußballfeld berechnet, wobei die geclusterten Punkte einen höheren Wert haben. Die Identifikation von Punktclustern oder Hotspots wird durch sogenannte Heatmaps angezeigt, diese ermöglichen es, Daten in Form eines Diagrammes zu visualisieren, indem die Abhängigkeit des Werts von den Daten in Farben dargestellt wird; je mehr Datenmenge eine Fläche aufweist, desto dunkler werden die Farbtöne. Die Software soll die taktische Handlung von Fußballspielen unterstützen, indem Muster von Positionen oder Koordinaten des Balls auf dem Feld vor einem Torschuss festgehalten werden. Mit den Spielkoordinaten wird die Position von Abwehrspielern auf dem Fußballfeld kurz vor dem Torschuss analysiert und verbessert. Das Ziel ist es zu identifizieren, in welchem Bereich des Spielfelds bei Offensivspielzügen mit großer Wahrscheinlichkeit ein Tor getroffen werden kann.

Die Bundesliga Fußball-Daten der Sporthochschule werden von der Firma IMPERI AG, die zum Unternehmen deltatre AG gehört, bereitgestellt. Die Firma deltatre AG beschäftigt sich unter anderem auch mit der Sammlung von sportwissenschaftlichen Statistik Daten aus der Bundesliga. Die Bundesligaspiele werden durch Kameras aufgezeichnet, so wird jede Position eines jeden Spielers auf dem Feld vollautomatisch erfasst. Das Bildverarbeitungssystem erfasst von jedem Spieler 25 Mal pro Sekunde eine (x, y)-Koordinate, so lässt sich bspw. die Länge der von einzelnen Spielern gelaufenen Strecken berechnen, deren Maximalgeschwindigkeit, Sprints, Pässe Zweikämpfe, Fouls, Schüsse, Tore, gelbe und rote Karten und einzelne Ballkontakte. In dem Fußballstadion werden auch drei Scouts oder Spielbeobachter platziert. Zwei Scouts beschäftigen sich mit je einer Mannschaft. Ein dritter Scout ist zuständig für die Beobachtung der Ballpositionen. Die Spielbeobachter sind notwendig, denn wenn die Spieler zu dicht untereinander sind, z. B. bei Torjubel und Zweikämpfen, kann das System die Einzelspieler nicht mehr unterscheiden, dann müssen die Scouts eingreifen.

Die HD-Kameras sind auf jeder Spielhälfte platziert und in allen Stadien auf der Höhe der Mittellinie fest angebracht. Die aufgenommenen Bilddaten werden z. T. über Glasfaserkabel per

Gigabit-Ethernet an vier Bildverarbeitungs-PCs übertragen[1].
Diese Arbeit besteht aus einem theoretischen und einem praktischen Teil.
Im Kapitel 2, Grundlagen, wird beschrieben, welche Werkzeuge und Techniken in dieser Arbeit verwendet werden. Im Vordergrund steht die Anwendung von Heatmaps, die die Analyse und Visualisierung von großen Mengen von Daten unterstützen. Zudem werden die Algorithmen *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) und *Kerndichteschätzer* erläutert, die praktischen Teil der Arbeit verwendet werden. Der Schwerpunkt dieser Arbeit ist die Umsetzung vom Algorithmus auf Heatmaps in einem Java-Programm.

Der Kapitel 3, Konzept, umfasst die Anforderungen an die Software, die Analyse und Auswertung der von der Deutschen Sporthochschule Köln freigestellten Dateien der Bundesliga-Saison 2011/2012 und abschließend wird noch die Konzeption der PostgreSQL-Datenbank vorgestellt.

Im Kapitel 4, Realisierung, wird die Umsetzung und Implementierung des Programms Dangerzone erläutert, indem dessen Modellierung detailliert beschrieben wird. Die Lösungsansätze und Modelle sowie die Probleme, die bei der Implementierung des Programms aufgetreten sind, gehören zu den wichtigen Aspekten dieses Kapitels. In diesem Kapitel wird auch die Erstellung und der Einsatz der Datenbank PostgreSQL sowie deren Funktionalität in Bezug auf das Programm geschildert. Der Import der Daten auf die Datenbank erfolgte nach der Vorbereitung, Transformation und Selektion der relevanten Daten aus den Bundesliga-Dateien. Zum Schluss gibt es eine ausführliche Analyse der Ergebnisse zwischen dem Programm Weka und Danzerzone.

Kapitel 5 – Fazit – schließt diese Arbeit ab, in dem die Erkenntnisse bewertet und reflektiert werden, die im Rahmen dieser Arbeit entstanden sind.

[1] Katrin Ahr (2012) http://www.elektroniknet.de/automation/sonstiges/artikel/90403/0/

2 Grundlagen

2.1 Verwendete Techniken

2.1.1 Java

Die Programmiersprache Java ist eine objektorientierte Sprache, die von Sun Microsystems entwickelt, die ihrerseits 2010 von Oracle aufgekauft wurde. Java ist die am weitesten verbreitete Programmiersprache mit aktuell 3 Milliarden Geräten, die mit Java laufen. Java stellt das Java-Entwicklungswerkzeug (JDK) und die Java-Laufzeitumgebung (JRE) bereit. Die Java-Laufzeitumgebung (JRE) besteht wiederum aus einer virtuellen Maschine (JVM), die die Ausführung von Anwendungen ermöglicht, so dass Programme plattformunabhängig implementiert werden können. Damit können Anwendung, die in Java geschrieben sind, unabhängig von dem zugrundeliegenden Rechner und Betriebssystem laufen. Die Entwicklungswerkzeug-Sammlung (JDK) enthält die Bibliotheken und Java-Entwicklungswerkzeuge, wie zum Beispiel den Java-Compiler, Java-Debugger, Java-Dokumentationswerkzeuge usw.

JDBC ist die Abkürzung für Java Database Connectivity und ist eine Schnittstelle für relationale Datenbanken. Diese Schnittstelle ermöglicht es, eine Datenbank in Java anzusprechen, dazu baut JDBC eine Datenbankverbindung auf und leitet dann die SQL-Anfragen an die Datenbank weiter. Die Ergebnisse dieser Anfragen werden dann der Anwendung zur Verfügung gestellt.

2.1.2 Java 2D

Java 2D ist eine API (*Application Programming Interfaces*) und eine Erweiterung der Java Bibliothek AWT (*Abstract Window Tookit*). Die API Java 2D ermöglicht Objekten zu zeichen, verschieden Geometrische Form zu bilden, Bearbeitung von Bildern gegebenenfalls mit Filterung und Rechnenoperationen um Bewegung zu bewirken, 2D Objekten zu erzeugen. Java 2D enthält ein Farb-Management und auch verschieden Methoden für Rendering (Tranformation von Rohdaten), Farbverläufen und Texturen, Zeichnung von Regionen, Compositing und Manipulation von Pixel.
In dieser Arbeit wurde besonderes die BufferedImage und AlphaComposite Klassen benutzt. Die BufferedImage Klasse stellt ein Bild in Form von eine Buffer dar. Bilder können auch hochgeladen werden. Die Farbe von Pixel können ausgelesen oder gesetzen werden.. Die AlphaComposite-Klasse behält verschiedene Compositing Typen, die bestimmen, wie überlappende Objekte gerendert werden. Ein AlphaComposite kann auch einen Alpha-Wert haben, der den Transparenzgrad angibt ,alpha = 1.0 ist völlig opak, alpha = 0.0 total transparent (klar). AlphaComposite unterstützt auch die Porter-Duff-Compositing-Regeln.

2.1.3 Eclipse

Eclipse ist eine in Java open source programmierte integrierte Entwicklungs-Umgebung für Software. Sie wird von einem Konsortium von EDV-Firmen wie IBM, SAP, Oracle, Redhat, SuSe und von vielen externen Entwicklern (Ulrich Cuber) unterstützt. Eclipse IDE ist das meistbenutzte Entwicklungswerkzeug für die Programmiersprache Java und wurde mit dem Ziel bestmöglicher Erweiterbarkeit und Integration von Schnittstellen entworfen.
Die Erweiterung von Werkzeugen in Eclipse erfolgt mittels Plug-Ins, diese ermöglichen die erweiterung von Funktionen durch kleine Programmpakete.

2.1.4 Grafikbibliotheken

ImageJ ist eine open source Software zur Bildverarbeitung und -analyse. Sie wurde vom *National Institute of Health* entwickelt und wird vielfach für medizinische und wissenschaftliche Bildanalysen genutzt. ImageJ wurde vollständig in Java entwickelt und ist damit plattformunabhängig. Das Programm läuft entweder als Online-Applet oder als eine herunterladbare Anwendung auf jedem Computer mit einer virtuellen Maschine für Java 1.5 oder höher.
ImageJ kann Bilder in viele Bildformaten wie TIFF, GIF, JPEG und BMP verarbeiten, speichern und lesen. Das Programm stellt eine Reihe von vorgefertigten Werkzeugen zur Verfügung zum Erzeugen, Verarbeiten, Analysieren, Öffnen und Speichern von Bildern in den verschiedensten Bilddatei-Formaten. Es kann zum Beispiel eine Pixelwertstatistik berechnen, Entfernungen und Winkel messen und bietet Verarbeitungsfunktionen wie Kontrastmanipulation, Schärfen, Glätten, Kantenerkennung und Medianfilterung an.
ImageJ wurde mit einer offenen Architektur entworfen, das ermöglicht die Erweiterung ihrer Funktionalität durch Java-Plugins. Dies sind kleine Java-Programme, die individuell entwickelt werden und mit ImageJ integriert werden können. Als Open-Source-Software steht auch eine Sammlung von dokumentierten Programmbibliotheken zu Verfügung, die verschiedene vorgefertigte Lösungen anbieten.

2.1.5 Erwin

Erwin ist eine Software zur Modellierung von konzeptionellen Daten, mit der Schemata für physische und relationale Datenbanken erstellt werden können. Diese Schemata können dann in Datenbanksystemen wie PostgreSQL eingesetzt werden. Erwin bietet viele Möglichkeiten, einfache Entity-Relationship-(ER)-Modelle zu zeichnen und daraus SQL-Statements automatisch zu erzeugen, was die Erstellung einer Datenbank erleichtert.

2.1.6 Weka

Das *Waikato Environment for Knowledge Analysis* (Weka) ist eine Software für maschinelles

Lernen im Rahmen von Data Mining. Die Software ist ein Projekt der *Machine Learning Group* der Universität von Waikato in Neuseeland. Es handelt sich um eine frei verfügbare Software, die verschieden Algorithmen für maschinelles Lernen enthält. Weka verfügt über Algorithmen wie Naive Bayes, künstliche neuronale Netze, Entscheidungsbäume, Support-Vector-Maschinen, Regression, Assoziationsregeln und Clustering-Verfahren wie den EM-Algorithmus, *k*-Means-Algorithmus, COBWEB und DBSCAN.

Weka hat eine Grafik-Oberflasche und erzeugt verschiedene Statistiken und Berichte. Die Software wurde vollständig in Java geschrieben, ist damit plattformunabhängig und bietet die Möglichkeit, für eigene Anwendungen deren Bibliotheken anzubinden. Es wurde auch eine umfassende Online-Dokumentation erstellt, was das Erlernen von Weka leichtmacht.
Weka unterstützt das Dateiformat *attribute-relation file format,* abgekürzt arff**.** Dies ist eine ASCII-Textdatei. Sie ist unterteilt in einen Header- und einen Data-Teil, im Header stehen die Relation und deren Attribute. Der Relation muss ein Name gegeben werden und für die Attribute müssen die Datentypen festgehalten werden. Der Datenteil beginnt mit dem @data-Schlüsselwort. Es gibt auch die Möglichkeit, die Daten aus einer Datenbank (mit JDBC) oder einer Datei mittels URL auszulesen.

Die Oberfläche von Weka unterteilt sich in die fünf Bereiche: Explora, Experimenter, Knowledge flow, Workbench, Simple Command line interface:

– In dem Bereich *Explora* gibt es die Möglichkeit der Visualisierung und Auswahl von verschiedenen Algorithmen und Daten.

– *Experimentator* ist eine Umgebung für die Durchführung von Untersuchungen und statistischen Tests zwischen Lernsystemen.

– *Knowledge flow*: Diese Umgebung unterstützt im Wesentlichen die gleichen Funktionen wie der Explorer, aber mit einer Drag-and-Drop-Schnittstelle. Ein Vorteil dessen ist, dass es inkrementelles Lernen unterstützt.

- *Simple command line* bietet eine einfache Befehlszeilenschnittstelle, die eine direkte Ausführung von WEKA-Befehlen durch das Betriebssystem erlaubt.

2.2 Heatmaps

Heatmaps dienen zur Visualisierung einer Menge von Daten auf einer zweidimensionalen Ebene. Eine wichtige Rolle nehmen die Heatmaps besonders bei der Wahrnehmung der Verteilung der zugrundeliegenden Daten wahr. Diese werden auf eine Dichtekarte verteilt. Die Heatmap repräsentiert die Häufung oder Dichte von Punkten auf bestimmten Stellen der Dichtekarte. Die Häufung oder Dichte wird in eine Farbskala übertragen. Die Farbcodierung muss gewählt werden, üblicherweise werden die wärmeren Farben wie Rot, Orangen, Rot und Gelb für die Stellen der Dichtekarte oder -ebene gewählt, wo große Menge von Punkten ein Muster oder Cluster zeigen. Der Bereich auf der Dichtekarte, wo die Punktdichte geringer ist, wird durch kalte Farben wie zum Beispiel Blau repräsentiert.

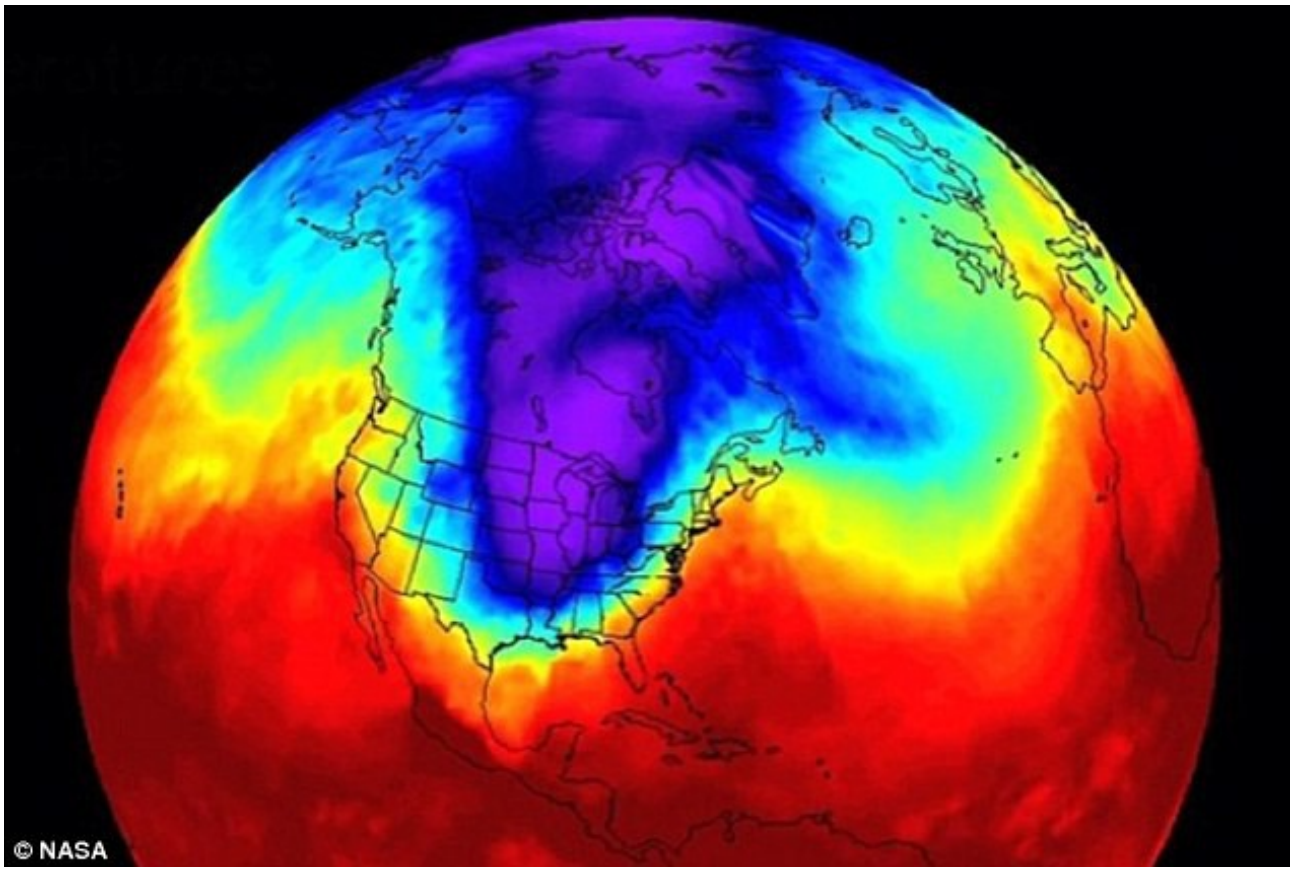

Abbildung 1: Heatmap

2.3 Algorithmen

Data-Mining

Data Mining ist ein automatisches computergestütztes Verfahren, um in großen Datenbanken nützliche Informationen oder Muster zu finden, die sonst unentdeckt bleiben würden. Beim Data Mining kommen integrierte Methoden der künstlichen Intelligenz und der Statistik zum Einsatz.

Der erste Schritt ist die Vorbereitung des Datenbestands. Diesen Prozess nennt man auch

Knowledge discovery in databases (KDD). Er hat vier Schritte:

1: *Data cleaning* Erkennung von inkonsistenten Daten und Ausreißern in dem Datenbestand.

2: *Data Integration* Selektion von Daten sowie deren Konsolidierung und Umwandlung in eine angemessene Form für die Eingabedaten.

3: *Data-Mining* Mit mathematischen Verfahren werden unbekannte Muster oder neue Informationen aus den Datenbestand identifiziert.

4: *Interpretation/Evaluation* Durch Experten werden die gefunden Muster interpretiert und analysiert, auf, dass Erkenntnisse entstehen. Die neuen Erkenntnisse oder Erfahrungen werden dann nach Maßgabe ihrer Verwendbarkeit für die Aufgabenstellung evaluiert.

Data Mining hat das Ziel, Wissen und Strukturen aus bestimmten Datenmengen zu extrahieren. Für das Extrahieren der Daten gibt es verschiedene Verfahren, die verwendet werden können. Die Clusteranalyse stellt ein wichtiges und häufig verwendetes Verfahren dar.

Clusteranalyse

Unter Clusteranalyse versteht man Verfahren, die eine ungeordnete Menge von Objekten oder Daten in Gruppen aufteilt. Mit dieser Gruppierung können die natürlichen Strukturen oder die Zusammenhänge von Objekten erfasst werden. Diese Strukturen oder Zusammenhängen werden als Proximitätsmaße bezeichnet, die die Ähnlichkeit bzw. die Distanz zwischen zwei Objekten bestimmen. Je kleiner die Distanz zwischen den Objekten ist, umso häufiger werden die Objekte gruppiert werden. Der Abstand von zwei Objekten oder Punkten kann dann als Ähnlichkeit (Homogenität) zwischen Objekten interpretiert werden, um Cluster von Objekten zu definieren. In dieser Arbeit wird die Euklidische Distanz angewendet, die besonders bei geographischen Koordinaten genutzt wird. Für zwei Punkte *x* und *y* gilt allgemein die Formel:

$$dist(\rightarrow x, \rightarrow y) = \sqrt{(x_1^2 - y_1^2) + (x_2^2 - y_2^2)}$$

Im Data Mining existieren zahlreiche Clusteralgorithmen, die in partitionierende, hierarchische, dichte-, gitter- und modellbasierte Methoden eingeteilt werden. Im nächsten Abschnitt wird der dichtebasierte Algorithimus DBSCAN näher erklärt.

2.3.1 DBSCAN

Der Algorithmus *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) ist ein dichtebasierter Clusteralgorithmus. Mit dichtebasiert ist hier gemeint, dass mit ihm Regionen in

einem Raum gebildet werden, die eine besonders hohe Anzahl von Objekten aufweisen. Diese Regionen werden von anderen Regionen oder Bereichen, die eine niedrigere Anzahl von Objekten enthalten, abgegrenzt. Die Umgebung oder Region, wo sich die hohe Punktdichte befindet, nennt man Cluster.

Der DBSCAN-Algorithmus definiert die Dichte-Erreichbarkeit von benachbarten Objekten oder Punkten, wenn ein Punkt *P* in einer bestimmten Umgebung ein Minimum von Punkten (MinP ts) mitumfasst. Die Entfernung ε von Punkt *P* ist der höchste Abstand, den alle zugehörigen Punkte haben können. Die minimale Anzahl von Punkten (MinP ts), die zusammen einen Cluster bilden sollen, sowie die Umgebung (ε) werden geschätzt oder ausprobiert.

Die Daten werden in drei Kategorien unterteilt.[2]

Kernpunkt Ein Punkt *P* ist ein Kernpunkt, wenn eine Mindestanzahl von Punkten (MinPts) innerhalb eines Abstands ε (einschließlich des Punktes selbst) erreichbar ist.

Randpunkt Randpunkte sind keine Kernpunkte, befinden sich aber in der Umgebung vom Kernpunkt.

Rauschpunkt Ein Rauschpunkt ist einen Punkt, der sich nicht in der Umgebung vom Kernpunkt befindet und auch selber kein Kernpunkt ist.

Neben den drei Kategorien von Punkten: Kernpunkten, Randpunkten und Rauschpunkten gibt es Definitionen, wie Konnektivität oder Erreichbarkeit, die wie im Folgenden unterteilt werden:

Direkte Dichte-Erreichbarkeit Ein Punkt *p* ist direkt dichte-erreichbar von einem Punkt *q*, wenn *q* ein Kernpunkt ist und wenn ein Punkt *p* direkt in der Umgebung ε des Punktes *q* erreichbar ist. Von dem Punkt *q* aus muss eine minimale Anzahl von Punkten (MinPts) direkt dichte-erreichbar sei.

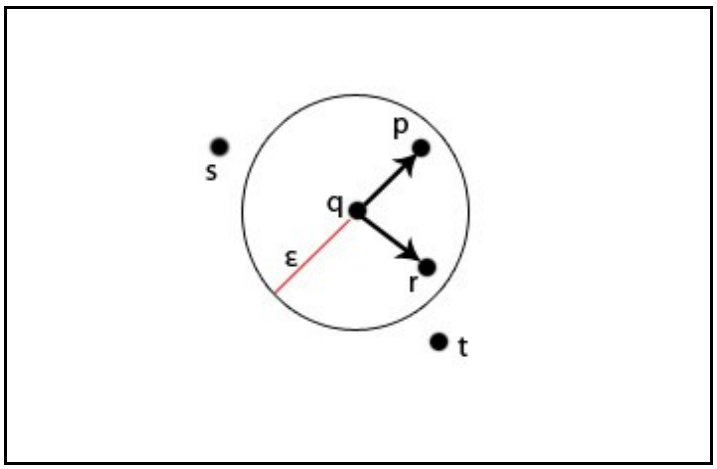

Abbildung 2: Direkte-Dichte-Erreichbarkeit

[2] Tan, Steinbach, Kumar (2014:529)

Dichte-Erreichbarkeit Ein Punkt *p* ist *dichte-erreichbar* von einem Punkt *q*, wenn zwischen den beiden Punkten eine Kette durch den verbindenden Punkt *m* gebildet werden kann, der seinerseits direkt von *p* und *q* erreichbar ist. D. h. die Kette zwischen den Punkten muss untereinander direkt dichte-erreichbar sein, sowie Punkt *q* und *m* müssen Kernpunkte sein.

Dichte-Verbundenheit Eine Kette von dichte-erreichbaren Punkten sind ein Teil von dem Cluster einschließlich der Randpunkte. Zwischen diesen Ketten können Verbindungen entstehen. Ein Punkt *p* ist *dicht verbunden* mit einem Punkt *q*, wenn sowohl *p* und *q* durch eine Kette von Punkten verbunden sind, die jeweils dichte-erreichbar sind.

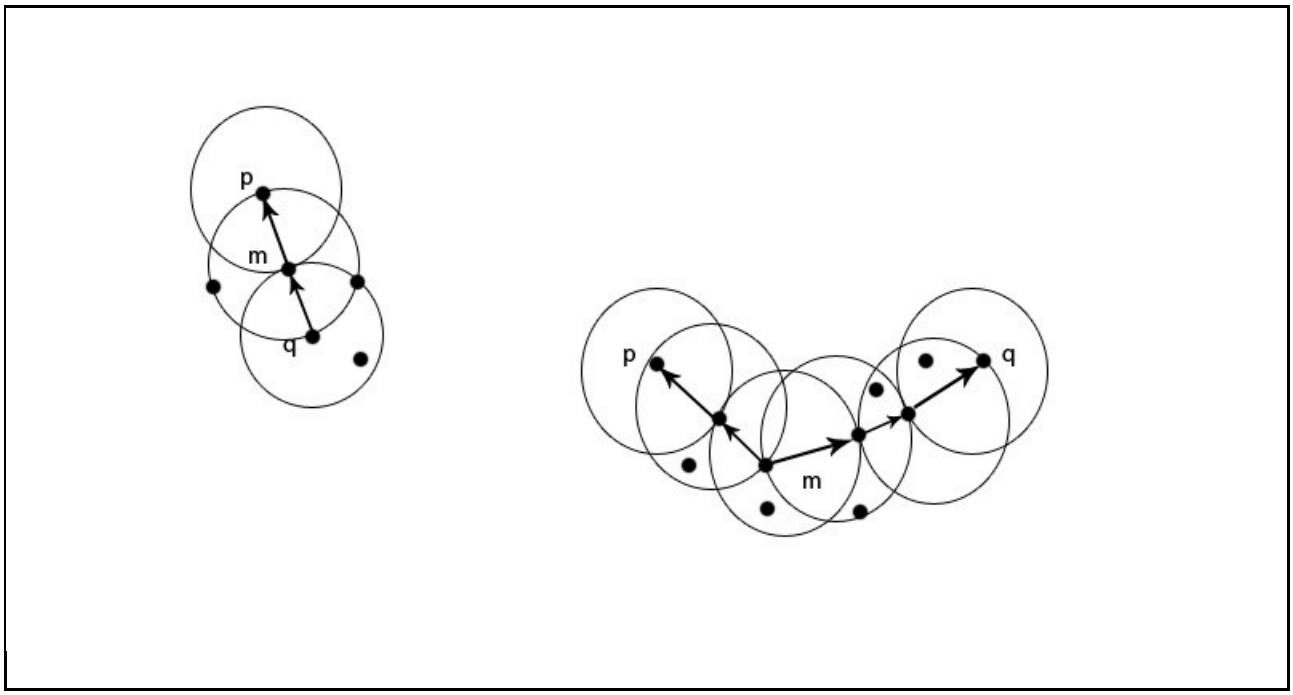

Abbildung 3: Graphische Darstellung von Dichte-Erreichbarkeit links und Dichte-Verbundenheit rechts[3]

Algorithmus: DBSCAN

Der Algorithmus DBSCAN bekommt als Eingabe eine Objektmenge *D* und die Parameter MinPts und ε. Zuerst werden alle Punkten aus der Objektmenge *D* als nicht besucht markiert. Zufällig wird ein Punkt *p* aus *D* ausgewählt und als besucht gezeichnet. Zugleich wird überprüft, ob dieser Punkt ein Kernpunkt ist. Wenn der Punkt kein Kernpunkt ist, wird er als Rauschpunkt markiert. Andernfalls wird ein Cluster für den Kernpunkt *p* erzeugt. Alle benachbarten Punkte *p'* in der Umgebung ε von *p* werden dem Objekt *N* hinzugefügt. Alle Punkte in dem Objekt *N* werden nacheinander daraufhin überprüft, ob sie als nicht besucht markiert sind. Wenn der Punkt noch nicht besucht wurde, wird er als besucht markiert und überprüft, ob er ein Kernpunkt ist. Wenn er ein

[3] Tan, Steinbach, Kumar (2014:529)

Kernpunkt ist, wird er in dem Objekt *N* gespeichert, sonst überprüft man, ob der Punkt schon zu einem Cluster *C* gehört. Dieses Verfahren wird wiederholt, bis alle Punkt in dem Objekt *D* als besucht markiert sind.

Input:

D: enthält eine Menge von *n* Objekten,

ε: ist die Umgebung oder Radius von einem Punkt,

MinPts: die minimale Anzahl von Punkten in der Umgebung ε.

Output: Dichte-basierte Cluster

Ablauf:

(1) alle Punkten als nichtbesucht markiert.
(2) **do**
(3) zufällig einen Punkt aussuchen.
(4) markiert den ausgesuchten Punkt *p* als besucht
(5) **if** Punkt *p* in seine Umgebung ε eine minimale Anzahl von Punkten (MinPts) hat,
(6) erzeuge einen Cluster *C* und füge Punkt *p* dem Cluster *C* hinzu.
(7) Alle Punkte in der Umgebung von *p* werden dem Objekt *N* hinzugefügt.
(8) **for each** Punkt *p′* in *N*
(9) **if** *p′* als nicht besucht markiert ist
(10) markiere *p′* als besucht
(11) **if** der Punkt *p′* eine Umgebung (ε) mit einer minimalen Anzahl von Punkten (MinPts) hat,
Füge die Punkte in der Umgebung von *p′* dem Objekt *N* hinzu.
(12) **if** *p′* keinem Cluster zugeordnet ist, füge *p′* dem Cluster *C* hinzu.
(13) **end for**
(14) output *C*
(15) **else markiere** *p* als Rauschpunkt.
(16) **until** alle Punkt sind besucht.
(17)

Es ist zu beachten, dass es eine gewisse Variation geben kann, je nach Dichte des Clusters und der zufälligen Verteilung von Punkten. Normalerweise ergibt sich eine Dichte aus Punkten oder Objekten in der Umgebung (ε) eines Kernpunkts. Für Punkte, die sich nicht in einem Cluster befinden, wie beispielsweise Rauschpunkte, wird die Umgebung (ε) jedoch sehr groß sein. Um zu gewährleisten, dass die Punkte dem Cluster hinzugefügt werden, sollte der Parameter für die

Umgebung (ε) so lange ausprobiert werden, bis eine Dichte von Punkten oder Cluster entsteht[4]. Wenn die Cluster-Dichten nicht grundlegend verschieden sind, wird im Durchschnitt die Variationsbreite nicht sehr groß sein.

DBSCAN erreicht sehr gute Ergebnisse bei klar abgetrennten Clustern. Liegen die Clusterdichten sehr nahe beieinander, werden die Cluster aber als ein Cluster angesehen. Der Algorithmus kann jede Form von Clustern erkennen und die Anzahl von Clustern muss nicht in voraus angegeben werden. Der Zeitaufwand von DBSCAN beträgt $O(n^2)$, denn jeder Punkt muss abgefragt werden, ob der Punkt ein Kandidat zu einem Cluster ist oder ob ein Punkt zu der Umgebung eines Kernpunkts[5] zugehört. Mit komplexeren Strukturen oder einem Index ist es möglich, die Zeitkomplexität auf $O(\log n)$ zu reduzieren.

2.3.2 Kerndichteschätzer

Der Kerndichteschätzer (*kernel density estimation*, KDE) vollzieht eine nichtparametrische Dichteschätzung. Die Idee wurde von Ronsenblat (1956) und einige Zeit später von Parzen (1962) eingeführt.[6] Die Kerndichteschätzer suchen die Wahrscheinlichkeitsdichte für den Wert $f(x)$ der Dichte f an der Stelle x. Nichtparametrisch bedeutet, bei einer unbekannten Wahrscheinlichkeitsverteilung der Stichproben mit geringen Voraussetzungen in Hinblick auf die Wahrscheinlichkeitsdichte zu arbeiten.

Um ein besseres Verständnis von der Idee des Kerndichteschätzers zu erhalten, wird zuerst die Darstellungsform Histogramm erläutert, die älteste Methode der Dichteschätzung. Ein Histogramm ist eine graphische Darstellung in Form von Rechtecken oder Balken der Häufigkeitsverteilung. Die Daten werden in Klassen unterteilt, die eine konstante Breite haben. Die Höhe von den Rechtecken über der Klasse steht für die Häufigkeitsdichte.[7]

Die Daten werden zuerst klassifiziert, das bedeutet, dass die Daten oder Stichprobe in Gruppen aufgeteilt werden. Die Wertebereiche von den Daten sind nebeneinander als Intervalle angeordnet. Über den Klassen werden dann die Balken aufgestellt. Die Häufigkeitsdichte wird berechnet durch die Division der Anzahl von Daten in der Klasse durch die Gesamtzahl der Daten multipliziert mit der Breite der Klassen. Die dargestellten Flächen sind dann direkt proportional zu den relativen Häufigkeiten.

---

4 Busch (2005:13)
5 Marcus Josiger, Kathrin Kirchner(2016)
6 Büning, Trenkler (1994:252)
7 Busch (2005:17)

Das Histogramm zeigt zwanzig Punkte: P1, P5, P9; P11; P13, P15, P18; P20, P21, P23, P24, P25; P27, P28, P29, P31 P33, P36, P38, P45.
Die Punkte werden auf Fünf Gruppen geteilt. Jede Gruppe hat die gleiche Breite. Gruppe 1 [0,10), Gruppe 2 [10,20), Gruppe 3 [20,30), Gruppe 4 [30,40) und Gruppe 5[40,50).
Es gibt drei Punkte (P1, P5, P9) in Gruppe 1, der Gruppe 2 sind vier Punkten (P11; P13, P15, P18) zugeordnet, Gruppe 3 enthält acht Punkte, P20, P21,P23, P24, P25, P27,P28; Gruppe 4 vier Punkte, P31, P33, P36, P38 und Gruppe 5 einen Punkt P45.

h ist die Höhe des Histogramms, *N* ist die Anzahl von Punkten und *b* ist die Breite der Klassen.

h =3/20.10 = 0.015.
h=4/20.10 =0.02.
h=8/20.10 =0.04.
h=4/20.10.=0.02.
h=1//20.10=0.005.

$$h = \frac{n}{N} . b$$

Die Daten wurden dann in die Tabelle eingetragen:

Gruppe	1	2	3	4	5
Anzahl von Punkten	3	4	8	4	1
Histogramm-Höhe	0.015	0.02	0.04	0.02	0.005

Tabelle 1: Beispiel eines Histogramms

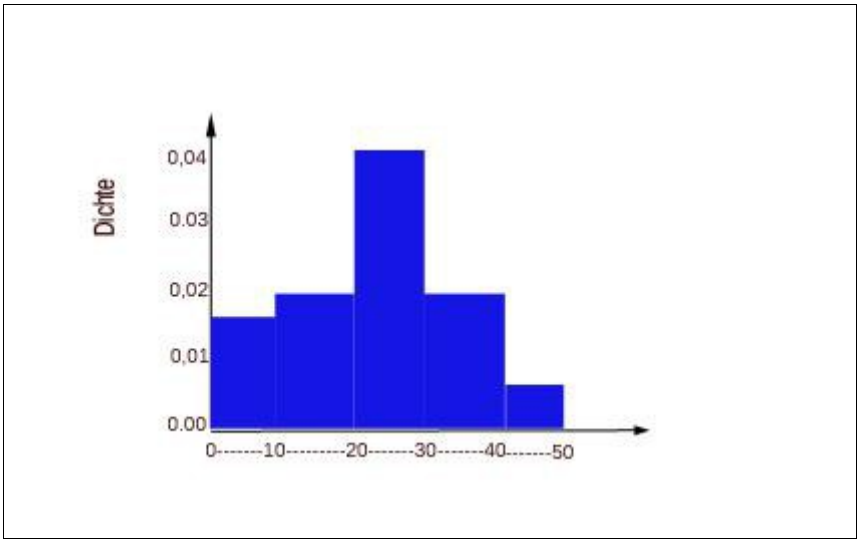

Abbildung 4: Histogramm

Die Fläche der Histogrammbalken repräsentiert die Prozentualität von Punkten in jeder Gruppe. Bei der ersten Gruppe sind es 3 × 0,05 = 0.15 = 15 %, also fünfzehn Prozent von den zwanzig Punkten.

Das Histogramm hat den Nachteil, dass die Wahl der Klassenbreite zufällig erfolgt, der optische Eindruck von den Balken lässt eine falsche Interpretation von den Verteilungen von Stichproben zu. Wenn die Breite zu groß oder zu klein ist, lässt sich die Dichte nur schlecht approximieren[8]

Der zweite Nachteil von dem Histogramm bezieht sich auf die sprunghafte Treppenfunktion. Die Verteilung eines stetigen Merkmals wird durch eine glatte Funktion besser repräsentiert. Die Treppenfunktion kann dazu durch eine Dichtekurve approximiert werden.

Im Gegensatz zu Histogrammen liefert der Kerndichteschätzer also eine stetige Funktion, welche an jedem beliebigen Punkt berechnet werden kann, die sprunghafte Treppenfunktion wird durch eine glatte Funktion approximiert. Dabei soll die Funktion nicht negativ sein, genauso wie bei den Histogrammen, und der Flächeninhalt soll gleich eins sein.
Die Dichtekurve ist folglich eine Funktion $f: R \rightarrow R$ mit

$$f(x) \geqslant 0 \qquad \forall x \in R$$

und $$\int f(x)\,dx = 1 \cdot$$

[8] Rinne (2008:493)

Der Kerndichtschätzer bietet die also Möglichkeit einer anderen Sichtweise darauf, wie die Punkte in Klassen gruppiert werden. Anstatt die Punkte in der zugehörigen Klasse zu betrachten, wird ein beliebiger Punkt in seiner Umgebung untersucht. Für einen Punkt x bildet man somit ein Intervall $[x-h, x+h]$ der Breite $2h$ und approximiert die Dichtekurve an der Stelle x.

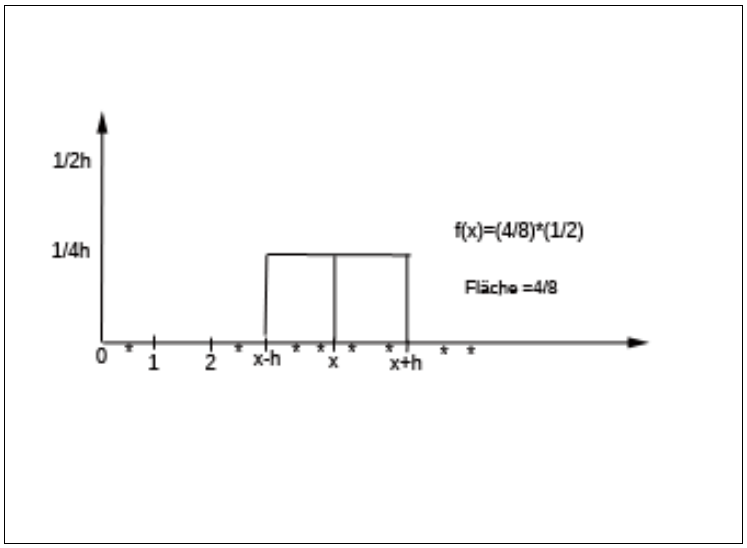

Abbildung 5: Histogramm und Kerndichteschätzer [9]

In der Abbildung werden 5 Daten (*) auf der Ebene x veranschaulicht. Das Fenster $[x-h, x+h]$ beinhaltet 4 Daten (*) zwischen den Punkten $x-h$ und $x+h$. Der Wert $F(x)$ ist wie bei dem Histogramm die Höhe der Kurve.

Für jeden Wert x werden dann die Punkte x_i in dem Bereich $[x-h, x+h]$ aufsummiert und anschließend durch n dividiert. Der Kernfunktion k übergibt ein Muster für die Funktion $f(x)$. Durch einen Kern K wird die unstetige Treppenfunktion geglättet. Einen stetigen Schätzer enthält man also dann durch eine stetige Funktion K.[10]

Der Kern (K) kann verschiedene Formen haben:

Epnechnikov-Kern $K(x)=\frac{3}{4}(1-x)^2$ für - $-1 \leqslant x<1,0, sonst$

Bisquare-Kern $K(x)=\frac{15}{16}(1-x^2)^2$ für $-1 \leqslant x<1,0, sonst$

Gauß-Kern $K(x)=\frac{1}{\sqrt{2\pi}} exp(\frac{-1}{2}x^2)$ für $x\in,$

[9] Fahmeir,Künstler, Pigeo,Tuts (2010:99)
[10] Jiawei, Kamber,Pei (2015:477)

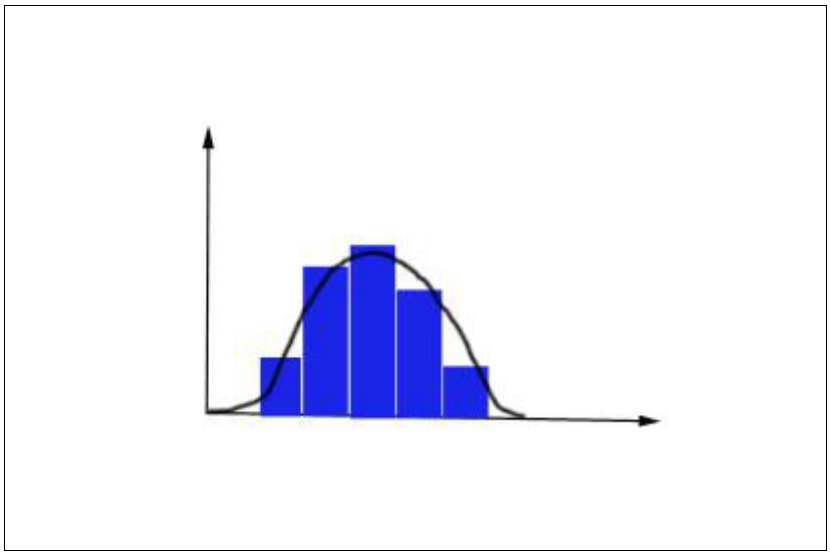

Abbildung 6:gleitendes Histogramm

Mit dem Epnechnikov-, Bisquare- und Gauß-Kern werden die Datenpunkte, die näher bei dem Punkt x liegen, stärker gewichtet als die Punkte, die einen größeren Abstand von dem Punkt x haben. Die Punkte x_i, die näher am Punkt x liegen, haben einen größeren Einfluss auf die Wahrscheinlichkeit von $f(x)$.[11] Mit den drei Kernen erhält man eine stetige Kurve für die Funktion F(x), was eine bessere Darstellung für eine Dichtekurve ist. Die Kerne haben keinen größeren Einfluss auf eine gute Schätzung der Dichte. Die Wahl der Bandbreite (h) hat hingegen, wie bei den Histogrammen, eine entscheidende Rolle. Durch ein großes h wird die Kurve sehr glatt und kleine Details können übersehen werden. Mit kleinem h wird der Dichtekurve sehr unruhig und erzeugt viele Zacken in der Darstellung von $f(x)$.

Um die Breite (h) zu definieren, gibt es keine endgültige Methode. Bei einer festen Anzahl von Punkten n, ist die Funktion $\hat{f}(x)$ sehr empfindlich. Eine Möglichkeit ist, dass die Bandbreite (h) selber von dem Benutzer festgelegt wird (Fahmeir, Künstler Pigeot, Tutz). Die Anpassung der Bandbreite (h) hängt auch zusammen mit der Zufälligkeit der Verteilung der Stichprobe. Die Feststellung von h kann anhand des optischen Eindrucks von Benutzer gewählt werden.

Eine andere einfache Methode für die Breitenbestimmung besteht darin, die Normalverteilung ins Spiel zu bringen. Mit der Normalverteilungsdichte von $F(x)$ und der Varianz σ^2 erhalten wir $h = 1.06\,\sigma\, n^{(\frac{-1}{5})}$ für die Bandbreite.

[11] Fahmeir,Künstler,Pigeo,Tuts (2010:101)

2.3.2.1 Implementation des Kerndichteschätzers als Lösungsansatz

Der Kerndichteschätzer ermöglicht die Dichteschätzung auf einer Fläche von Interesse, in dieser Arbeit ist es das Fußballfeld. Er schätzt die Wahrscheinlichkeit von Ereignissen pro Flächeneinheit. Die Ereignisse in unser Fall sind die Positionen des Fußballs vor einem Torversuch oder einem tatsächlichen Tor. Die Überlegung ist, dass man auf dem gesamten Fußballfeld eine Reihe von festen Punkten markiert. Jeder von diesen *Ankerpunkten* hat auch einen Radius, dieser umfasst einen Bereich auf dem Fußballfeld.

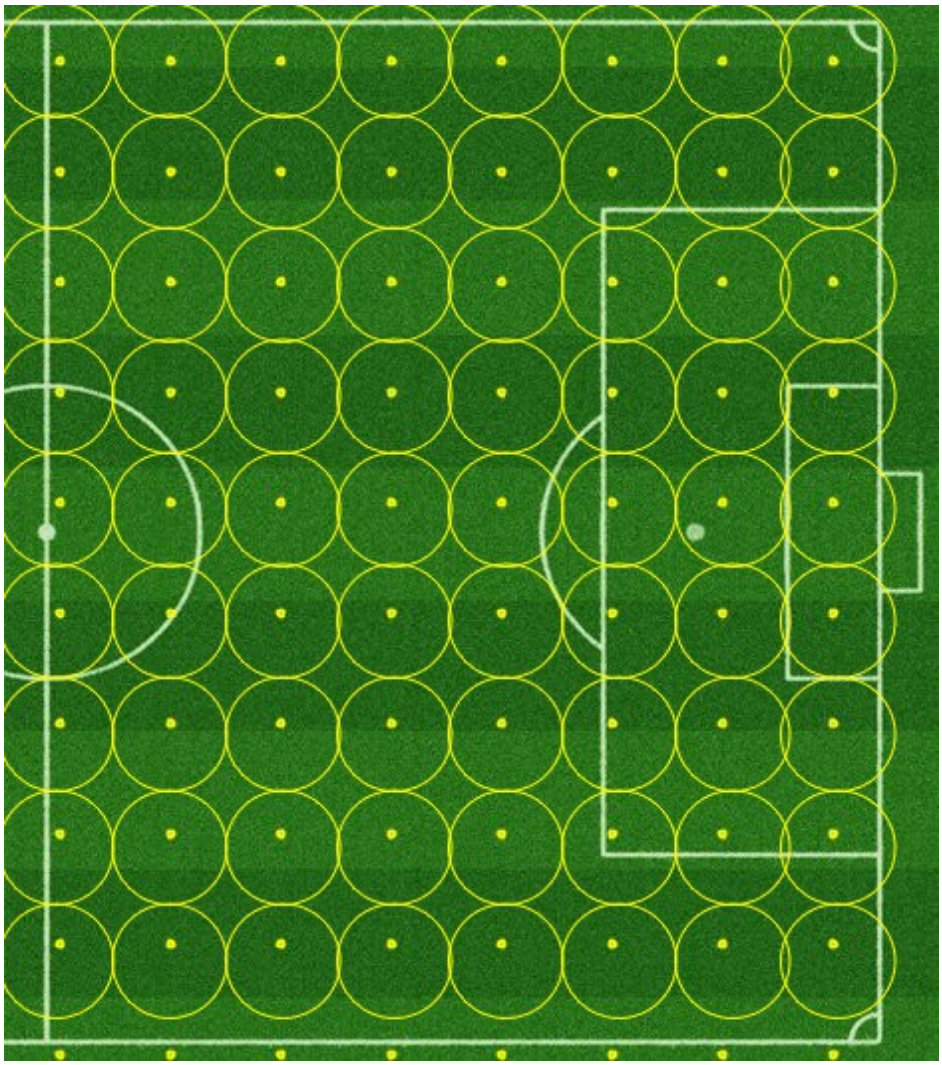

Abbildung 7:Fußballfeld mit Ankerpunkten mit Radius

In einem zweiten Schritt wird für jeden Ankerpunkt der Abstand von diesem Punkt bis zu benachbarten Punkten (Torversuch oder ein tatsächliches Tor) berechnet. Die benachbarten Punkte sind Punkte, die sich im Einflussradius von den Ankerpunkten befinden. Der Radius von dem Ankerpunkt definiert, wie viele Punkte für die Berechnung von dem Kerndichteschätzer gezählt und aufsummiert werden. Die Zahl von Punkte, die sich auf einem Einflussradius befinden, beeinflusst die Dichte auf der Fläche, wo sich die Ankerpunkte befinden.

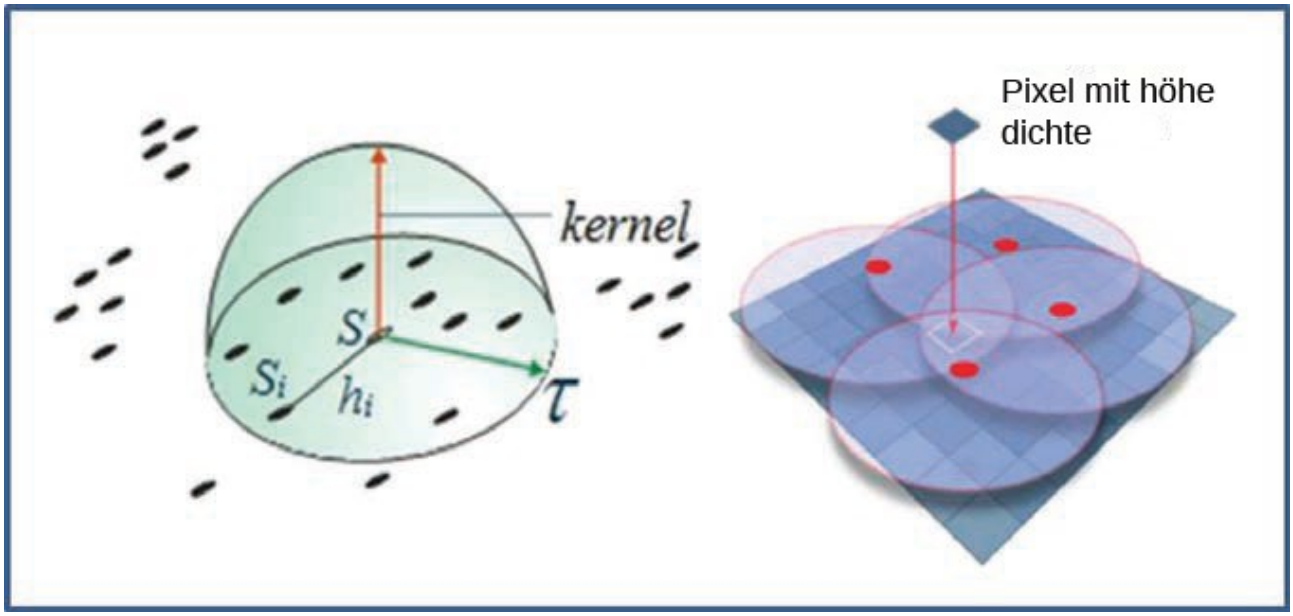

Abbildung 8:Kernel und Pixel Höhe[12].

Das linke Bild zeigt den Ankerpunkt, in der Abbildung repräsentiert durch *S*, die Punkte S_i repräsentieren die Punkte, die unter dem Einflussradius von S stehen. h_i ist der Abstand zwischen S und S_i. Der Kern ist die Funktion, die einen Wert für den Punkt *S* gibt. Das rechte Bild stellt die Kerne auf dem Fußballfeld dar. Jeder von diesen Kernen hat eine Gewichtung, beeinflusst von der Dichte in dem Punkt *S*. Die Kerne werden dann auf der Fläche des Fußballfelds mit dem entsprechenden Radius untersucht. Die Schnittmenge von den Kernen identifiziert auf dem Fußballfeld eine Fläche, wo mit größerer Wahrscheinlichkeit die Dichte von den Punkten S_i höher ist.

Mit den Koordinaten von den festen Ankern, die auf dem Fußballfeld verteilt sind, und mit den Koordinaten von den Torversuchen und gezielten Toren werden wir für jeden Ankerpunkt einen Einflussradius und eine Gewichtung in Bezug auf die Verteilung von Torversuchen und gezielten Toren ermitteln, diese können dann auch dargestellt werden als Punkte mit Koordinaten auf der Fläche von den Fußballfeldern.

[12] Prof. Flávio Henrique M de A Freire (2016)

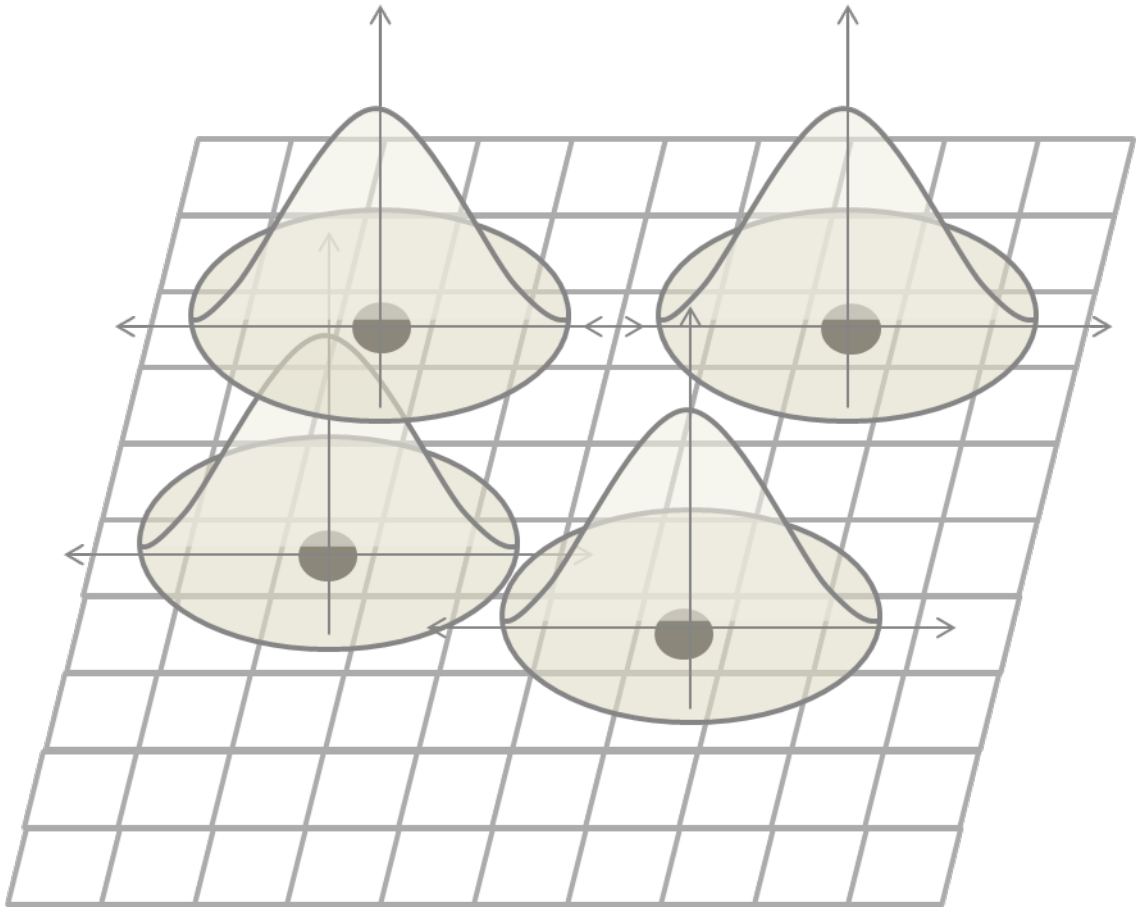

Abbildung 9:Kernel auf dem Fußballfeld[13]

2.4 Die Datenbank PostgreSQL

Die Datenbank PostgreSQL ist ein objektrelationales Datenbanksystem. Es wurde an der Informatikfakultät der Universität von Kalifornien in Berkeley entwickelt. Die erste Version wurde 1987 fertiggestellt und seit 1997 steht die Datenbank als Open-Sorce-Software zur Verfügung. PostgreSQL berücksichtigt die SQL-Standards und es existieren verschiedene Module, die mitinstalliert werden können, um die Funktionalität von PostgreSQL-Datenbanken zu erweitern. Das Modul PostGIS etwa dient der Speicherung und Verarbeitung von geografischen Daten. PostGIS enthält über 200 Funktionen aus dem Bereich der räumlichen Analyse und Geometrie-Objekte, wie Linien und Punkte. Diese räumlichen Funktionen ermöglichen beispielsweise die Berechnung von Flächen und Distanzen, Verschneidung, Berechnung von Pufferzonen, Analyse von Raster- und Vektordaten bis zu Funktionen für die Erstellung von Geometrien.

[13] Rodolfo Maduro Almeida (2016)

3 Konzept

3.1 Fußball und Data Mining

In Durchschnitt gibt es in einem Fußballspiel 3000 Ereignisse, wie Pässe, Fouls, Torschüsse, Ecken, Freistöße, Zweikämpfe, Ballkontakte; dazu zählen noch lange Spielzüge und es gibt fast keine Unterbrechung. Das Spiel findet mit 22 Spielern auf einer Fläche von 7140 Quadratmetern mit einer Dauer von 90 Minuten statt. Das macht das Ergebnis eines Fußballspiels unberechenbar. Die digitale Sammlung und Aufzeichnung von Daten gibt nun die Möglichkeit, verschiedene Methoden des Data Mining und auch der Statistik zu nutzen, um aus einer bestimmen großen Menge von Daten Wissen zu erlangen oder Mustermerkmale zu erkennen. Das Ziel ist, sich einen Vorteil zu verschaffen, indem man vorhersieht, welche Gruppentaktik die Gegner verwenden, wie sie bei Angriff oder Verteidigung reagieren, welches die typischen Verhaltensmuster von bestimmten Mannschaften sind. Auch kann man falsche Positionierungen im Raum von Spielern sowie die Schwächen der gegnerischen Mannschaft erkennen sowie auch Erkenntnisse über die eigene Mannschaft gewinnen. Anschließend geht es darum, aus der Analyse Schlussfolgerungen zu ziehen, um die Spieltaktik zu optimieren.

Die ersten Versuche, Fußballspiele zu analysieren, geschahen im Jahr 1950. Der Engländer Charles Reep hat mehr als 2200 Spiele untersucht und auch die Weltmeisterschaft von 1958. Er ist zu dem Ergebnis gekommen, dass die Mannschaften durchschnittliche 9 Torschüsse brauchten, um ein Tor zu erzielen.

Die Mannschaften haben 80 Prozent mehr Chancen, ein Tor zu erzielen, wenn sie weniger als 5 Ballkontakte hatten. Charles Reep ist deshalb zu dem Schluss gekommen, dass die Mannschaften keine Zeit mit vielen Ballkontakten verlieren sollten, sondern dass man den Ball direkt mit langen Pässen in den gegnerischen Bereich werfen sollte. Diese Strategie wurde als „the long-ball game“ bekannt.

Im Jahr 2005 haben Ian Franks, Professor an der University of British Columbia, und Mike Hughes, ein Mathematiker, sich die Daten von zwei WM-Turnieren angeschaut. Auf dem ersten Blick waren die Daten von Reep kompatibel mit den Studien von Franks und Hughes. Aber nach genauerem Hinsehen zeigte sich, dass die meisten Torchancen nach vier oder sechs Ballkontakten passieren, weil einfach die Spielabläufe vom Fußball so gestrickt sind und nicht, weil die Torchancen mit weniger Ballkontakten besser waren.

Franks und Hughes haben auch festgestellt, dass 90 Prozent der Spiele, die von Reep analysiert wurden, Spiele waren, die fast nur auf 4 oder 5 Ballkontakte bis zum erzielten Tor reduziert waren. Die Tatsache, dass Brasilien in der Zeit 1958 die erfolgreichste Mannschaft war, ohne längere Bälle zu spielen, wurde von Reep auch ignoriert. Ein anderes Beispiel ist das Spiel Deutschland gegen

Brasilien bei der Weltmeisterschaft 2014 in Brasilien. Deutschland hat Brasilien 7:1 besiegt. Wenn sich man die Statistik anschaut, hatte Brasilien die besseren Zahlen. Trotz der hohen Niederlage hatte Brasilien in diesem Spiel 52 Prozent Ballbesitz und Deutschland 48 Prozent, Balleroberung hatte Brasilien 46 und Deutschland 34 Prozent, Brasilien hatte 18 Torschüsse, Deutschland 14, Brasilien hatte 7 Ecken und Deutschland 4, Brasilien hatte 22 Flanken und Deutschland 10.

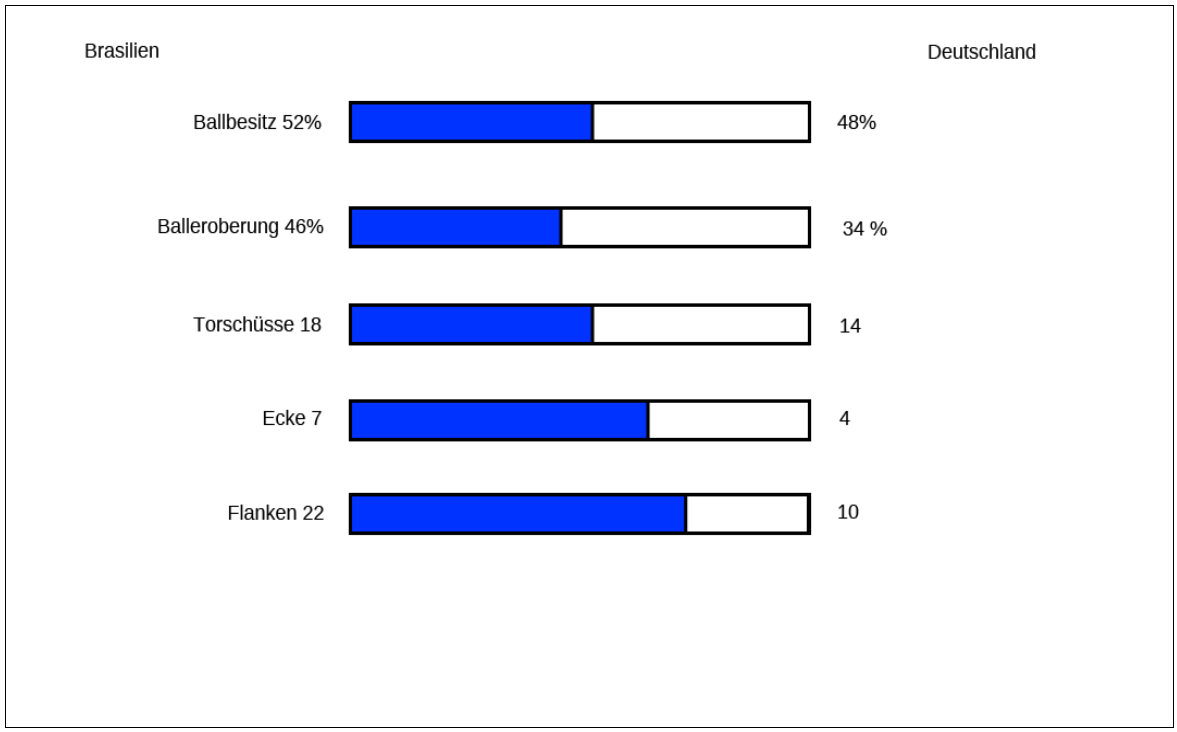

Abbildung 10:Brasilien-Deutschland

Diese Zahlen haben sich also als bedeutungslos erwiesen, allein die bloßen Zahlen erzeugen nur Informationen, aber kein Wissen, das ein Vorteil für die Mannschaft schaffen würde. Der Ballbesitz einer Mannschaft hat nur eine Korrelation mit dem Erfolg, wenn dieser Ballbesitz die meisten Zeit auf den gegnerische Halbfeld gespielt wird. Auch die von einem Spieler in einem Spiel gelaufenen Kilometer haben keinen Einfluss auf die Leistung eines Spielers. Wichtiger sind die Sprints (schnelle Reaktion, schnelles Laufen) von einem Spieler während einer wichtigen Situation im Spiel.

Die gesammelten Daten müssen folglich in einem Kontext analysiert werden, neben Statistikern und Informatikern müssen auch Sportwissenschafler einbezogen werden. Die sind nämlich in der Lage, das Spiel zu verstehen, und können so der Statistik in diesem Kontext helfen, damit die Zahlen der Statistik in Wissen oder Ergebnisse umgewandelt werden, die den Erfolg von einer Mannschaft unterstützen können.

3.1 Anforderungen an die Software

Die Software hat bestimmte Anforderungen zu erfüllen, die nachfolgend aufgelistet werden:

- Erstellung von einem Datenbank-Schema
- Importieren von Daten in die Datenbank.
- Anzeigen von Heatmaps, die die Dichte von Punkten auf einem Fußballfeld zeigen.
- Dichte soll als Farbe repräsentiert werden.

- Die Darstellung von Heatmaps soll nach Saison und nach Mannschaften geordnet möglich sein
- Es soll möglich sein, zwischen fünf, fünfzehn oder zwanzig Sekunden vor der Torschuss-Aktion zu wählen.

Die erste Anforderung ist die Erstellung von eine Datenbank mit den Daten, die von der Sporthochschule bereitgestellt werden. Diese Daten müssen strukturiert ausgelesen und in die Datenbank gespeichert werden. Die Daten werden dann für die Berechnung von einer Dichtefunktion, dem Abstand zwischen Punkten und der Berechnung von der Fläche vom Fußballfeld verwendet. Das Importieren von Daten ist ein wichtiger Bestandteil der Software, denn eine wichtige Voraussetzung für die Weiterverwendbarkeit von dieser Software ist, dass die neuen Daten aus der nächsten Bundesliga-Saison immer wieder importiert werden können, damit die Datenbank immer auf dem neuesten Stand ist.

Die Heatmaps sollen die Visualisierung von einer besonderen Häufung von Punkten intuitiv und schnell ermöglichen. Die Visualität soll durch Farben dargestellt werden; intensive Farbtöne wie Rot oder Orange sollen eine höhere Dichte von Ereignissen auf dem Fußballfeld repräsentieren. Diese Aktionen (Torschüsse) sollen nach dem Zeitpunkt unterteilt werden. Die Aktionszeiten sind in fünf, fünfzehn oder zwanzig Sekunden pro Saison aufgeteilt

Das Software soll die Darstellung von den Heatmaps nach Session oder Mannschaften zeigen können.

3.2 Analyse

Der erste Schritt der Analyse besteht darin, die Dateien vistrack-actions und Pos zu untersuchen, die von der Firma Deltrate AG bereitgestellt wurden. Die Datei vistrack-actions stellt eine XML-Datei

dar, die die Ereignisse und Daten über einzelne Spiele strukturiert enthält. Insgesamt beinhalten diese Dateien bis zu 2000 Daten für ein Spiel.

Zusätzlich zu der Dateien vistrack-actions werden auch die entsprechenden Pos-Dateien analysiert, die eine Text-Datei sind und die Koordinaten von jedem einzelnen Ereignis eines Spieles enthält.

Aus den aus der Analyse der beiden Dateien gewonnenen Informationen kann ein Entwurf für eine Datenbank erstellt werden. In der Datenbank werden nur die Informationen gespeichert, die für das Programm relevant sind.

3.2.1 Analyse der XML-Datei vistrac-actions

Die XML-Datei vistrack-actions beinhaltet alle erforderlichen Informationen von einem Spiel. Das XML-Dokument fängt mit dem Wurzel-Element <sport-content> an, aus dem das gesamte Dokument in einer logischen Struktur aufgebaut ist. Die Datei enthält verschiedene Informationen wie Pässe, Zweikämpfe, Fouls, Schüsse, Tore, gefährliche Torversuche, gelaufene Distanz, Anzahl an Sprints von einzelnen Spielern, gelbe und rote Karten, Uhrzeiten, Zeitstempel von Ereignissen, Mannschafts-Aufstellung, Spieldatum usw. Das Dokument kann in drei Bereiche aufgeteilt werden, in denen sich die wichtigen Informationen für das Programm befinden.

```
<team>
<team-metadata id="0_10" team-key="10" alignment="home" imp:uniform-color-
name="rot" imp:uniform-colorhex="#ed144e"><name full="FC Bayern München"
nickname="FC Bayern" />
                                                                    <player>
<player-metadata id="P_138374_28359" player-key="28359" positionregular="goalie"
position-event="goalie" status="bench" uniform-number="1">
    <name full="Manuel Neuer" nickname="M. Neuer" last="Neuer" />
</player-metadata>
</team>
```

Abbildung 12: Mannschafts-Informationen

Der erste Bereich bezieht sich über die Hintergrundinformationen des Spieles, wie Spieldatum, Id des Spieles (eindeutige Nummer, die ein Spiel identifiziert) und Spielliga (in welcher Liga wird das Spiel gespielt, z. B. Bundesliga oder Zweitliga). Der folgende Abschnitt aus der XML-Datei vistrack-actions zeigt den ersten Bereich.

```
<event-metadata id="E_130195" event-key="130195" event-status="post-event" start-date-time="20110819T203000+0200"  date-coverage-type="event" date-coverage-value="20110819T203000+0200"
</event-metadata>

<tournament-metadata tournament-name="Fussball Bundesliga"/>
```

Abbildung 11: Hintergrundinformationen eines Spiels

Das Attribut event-key ="130195" ist eine eindeutige Nummer für das Spiel, das am 19.8.2011 um 20:30 Uhr (start-date-time) stattgefunden hat. Das Attribut tournament-name="Fussball-Bundesliga" zeigt an, in welcher Liga das Spiel gespielt wurde.

Der zweite Bereich bezieht sich auf die Mannschaftsdaten. In diesem Teil finden sich Informationen über die Mannschaften und deren Eigenschaften, die nicht auf der Wettbewerbsleistung basieren. Daten wie Mannschaftsname, ID der Mannschaft, Mannschaftsuniform, Spielername, ID einzelner Spieler, Spielerrolle (-position) werden in dem Attribut <team> dargestellt, wie die folgende Abbildung zeigt

Aus dem oben erwähnten Abschnitt können Informationen wie team-key = 10, Mannschaftsname name-full="FC Bayern München", nickname="FC Bayern", Spieler-Daten wie name full="Manuel Neuer, Spieler-id player-key="28359", Spielerrolle position-regular="goalie"entnommen werden.
In dem dritten Bereich sind die wichtigsten Informationen zum Spielverlauf vorhanden. In dem Element <event-actions-soccer> sind untergeordnete spezifische Aktionen enthalten, die während des gesamten Fußballspiels vorgekommen sind, etwa Eckstoß, Freistoß, Spielfoul, Torversuch, erzielte Tore, Elfmeterfreistoß, Abseits, Pässe, Flanke usw.

Die folgenden Elemente aus der XML-Dateien beinhalten bestimmte Aktionen aus einem Spiel:

- action-soccer-score-attempt: wenn ein Spieler versucht, ein Tor ohne ein positives Ergebnis zu erzielen.
- action-soccer-score: wenn eine Mannschaft ein Tor erzielt,
- action-soccer-offside: zeichnet ein Abseits in einem Spiel auf
- action-soccer-foul: wenn ein Spieler ein Foul begeht
- action-soccer-penalty: eine Strafe, zu dem ein Foul führte, Elfmeterfreistoß.
- action-soccer-other: eine benutzerdefinierte Aktion (Eckstoß, Angriff, Flanken)

Abbildung 13: Aktionen Tabelle

Die wichtigsten Elemente für das Programm sind action-soccer-score-attempt, action-soccer-score und action-soccer-other. Jede Aktion hat eine eigene ID, die einzelne Aktionen identifiziert. Diese Aktionen sind für die Positionen der Ball auf dem Fußballfeld relevant, um die Zeit der Aktion zu bestimmen. Unter dem Element action-soccer-other sind auch die Attribute action-type, period-value, imp:timestamp und action-type zu finden, die bezeichnen, ob es sich um den Anfang oder das Ende der ersten oder zweiten Halbzeit handelt. Die Attribut period-value hat den Wert 1 oder 2, also erste oder zweite Halbzeit. Timestamp zeigt die genaue Uhrzeit, wann die erste und zweite Halbzeit anfing oder endete, wie das folgende Beispiel zeigt.

```
action-type="period-start" period-value="1" imp:timestamp="20:30:26.00"
action-type="period-end" period-value="1" imp:timestamp="21:16:29.00"
action-type="period-start" period-value="2" imp:timestamp="21:32:11.00"
action-type="period-end" period-value="2" imp:timestamp="22:17:13.00"
```

Abbildung 14: Halbzeiten eines Spiels

Die Elemente action-soccer-score-attemp und action-soccer-score signalisieren eine einzelne Aktion (erzieltes Tor oder Torversuch) auf dem Fußballfeld. Zu diesen Elementen gehören auch die entsprechenden timestamp-Attribute, die genau den Zeitpunkt angeben, an dem eine Aktion während des Fußballspiels stattgefunden hat. Mit diesen Informationen (Anfang und Ende von der ersten und zweiten Halbzeit und Aktionszeit) kann man genau definieren, in welcher Minute des Spieles ein Tor gefallen ist bzw. ein Torversuch gescheitert ist. Die berechnenden Spielminuten können dann später in Sekunden umgewandelt und in vier Gruppen aufgeteilt werden. Mit diesen vier Gruppen können die Aktionszeiten in genauer Aktionszeit, 5 Sekunden vor der Aktion, 10 Sekunden vor der Aktion und 20 Sekunden vor Aktion aufgestellt werden, die die Anforderung des Programms erfüllen. Diese vier Gruppen werden in der Datenbank gespeichert.

```
action-soccer-score id="-1130195189"
action-soccer-score-attempt id="-1130195346"
```

Abbildung 15: Aktionen

3.2.2 Analyse der Pos-Datei

In den Pos-Dateien sind verschiedene Informationen über die Positionen von Spielern, Ereignisse auf dem Fußballfeld, Geschwindigkeit und gelaufene Strecke von Spielern, Schiedsrichter und Ball zu jeder Datei vistrack-actions enthalten. Die Pos-Datei ist für ein Spiel von neunzig Minuten in 135.000 Frames aufgeteilt, dies bedeutet jeweils 25 Frames für eine Sekunde des Spiels. Die Frames werden in die erste und zweite Halbzeit aufgeteilt. In der ersten Spalte steht die Nummer

des Frames, sie fängt bei null an und geht bis zum Ende der ersten Halbzeit. Die zweite Halbzeit fängt wieder bei null an und geht bis zu Ende des Spieles. Um auf die Frames zu gelangen, berechnet man den Zeitstempel eines Ereignisses aus der Datei vistrack-actions in Sekunden und multipliziert mit 25, mit diesem Ergebnis und der Information, ob es die erste oder zweite Halbzeit ist, gelingt die proportionale Zuordnung der Frames zu seinen Ereignissen. Die zweite Spalte enthält die Spielminuten, die dritte Spalte zeigt, ob es die erste oder zweite Halbzeit ist. Die wichtigen Koordinaten stehen in dem vierten Abschnitt von den Frames. Dort befinden sich die Koordinaten *x*, *y* von den Ball für jede Sekunde auf dem Feld. Aus der Datei vistrack-actions bekommt man die Spielzeit, mit der man die Frames berechnet kann und dann erhält man aus den Frames der Pos-Datei die Koordinaten. Die Koordinaten *x*, *y* werden später in der Datenbank zusammen mit der betreffenden Aktion gespeichert.

```
62322,42,1;#0.9092,0.1083,11,1.07,1,2;
62324,42,1;#-0.9085,0.1077,11,1.07,1,2;
62325,42,1;#-0.9078,0.1070,11,1.07,1,2;
```

Abbildung 16: pos Datei

Die Abbildung ist ein Abschnitt von der Datei POS, die Daten sind durch Komma oder Semikolon getrennt, die erste Spalte bezieht auf die Frames, die zweite auf den Spielminuten, die Dritte auf die Halbzeit. Nach dem Hash-Zeichen stehen die Koordinaten *x* und *y*.

3.2.3. Analyse der Daten mit den Algorithmen DBSCAN und KernelDensity

Vor der Datenanalyse wurden die Daten von der Bundesliga 2011/2012 in das Programm Weka eingespielt. Die Daten wurden dann zuerst mit dem DBSCAN-Algorithmus getestet. In einem zweiten Schritt wurden die Daten mit dem Algorithmus KernelDensity von Weka bearbeitet. Die Ergebnisse von den beiden Untersuchungen wurden dann analysiert. Anhand dieser Ergebnisse wurde dann entschieden, welcher Algorithmus für die Berechnung der Dichte von den Positionen (x, y) vor den Torschüssen auf dem Fußballfeld am besten geeignet ist.

3.2.3.1 DBSCAN

Bei dem ersten Versuch mit den Algorithmus DBSCAN wurden die Positionen fünf Sekunden vor den Torschüssen verwendet. Es wurden 366 Positionen getestet. Die Punkte wurden auf eine (x, y)-Koordinate transformiert. Die y-Koordinate hat einen Wert von −1 bis 1 für die Höhe und die x-Koordinate von 0 bis 1 für die Länge des Fussballfelds. Die Punkte zeigten sich auf das ganze

Fußballfeld verteilt.

Es wurden eine Reihe von Tests vollzogen, mit zwei verschiedene Parametern. Der erste Parameter ist der Radius oder Umgebung von einem Punkt, der den höchsten Abstand von allen anderen, ihm zugeordneten Punkten angibt. Der zweite Parameter ist die Mindestanzahl an Punkten, die einen Cluster bilden.

Der erste Versucht wurde mit einem Radius von 0,9 und mindestens 6 Punkten unternommen. Die Ergebnisse zeigen, dass nur ein Cluster gebildet wurde. Die Cluster wird durch die blauen Punkte repräsentiert. Alle 336 Punkten wurden in einem einzelnen Cluster gruppiert, so dass keine neuen Erkenntnisse gewonnen wurden.

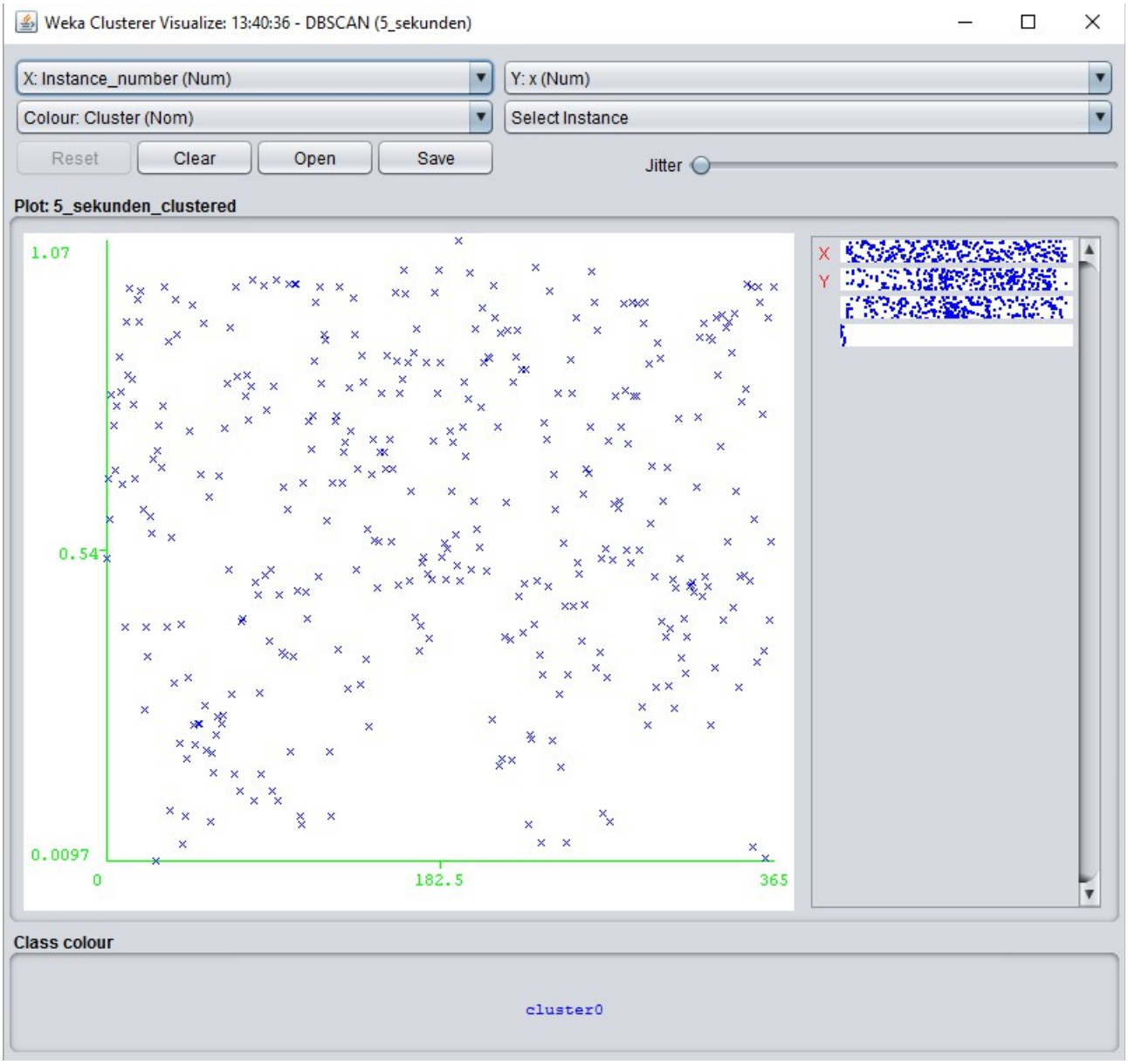

Abbildung 17: Versucht mit mindestens 50 und Radius 0,4

Es wurden unzählige Versuche vorgenommen, mit verschiedenen Kombinationen von Radien und Mindestpunktzahlen, die hier nicht dokumentiert werden. Die ersten brauchbaren Ergebnisse

wurden mit dem Radius 0,2 und einer Mindestpunktzahl von 70 gezeigt. Das beste Resultat war allerdings mit den Radius 0,2 und mindestens 90 Punkten gefunden worden.

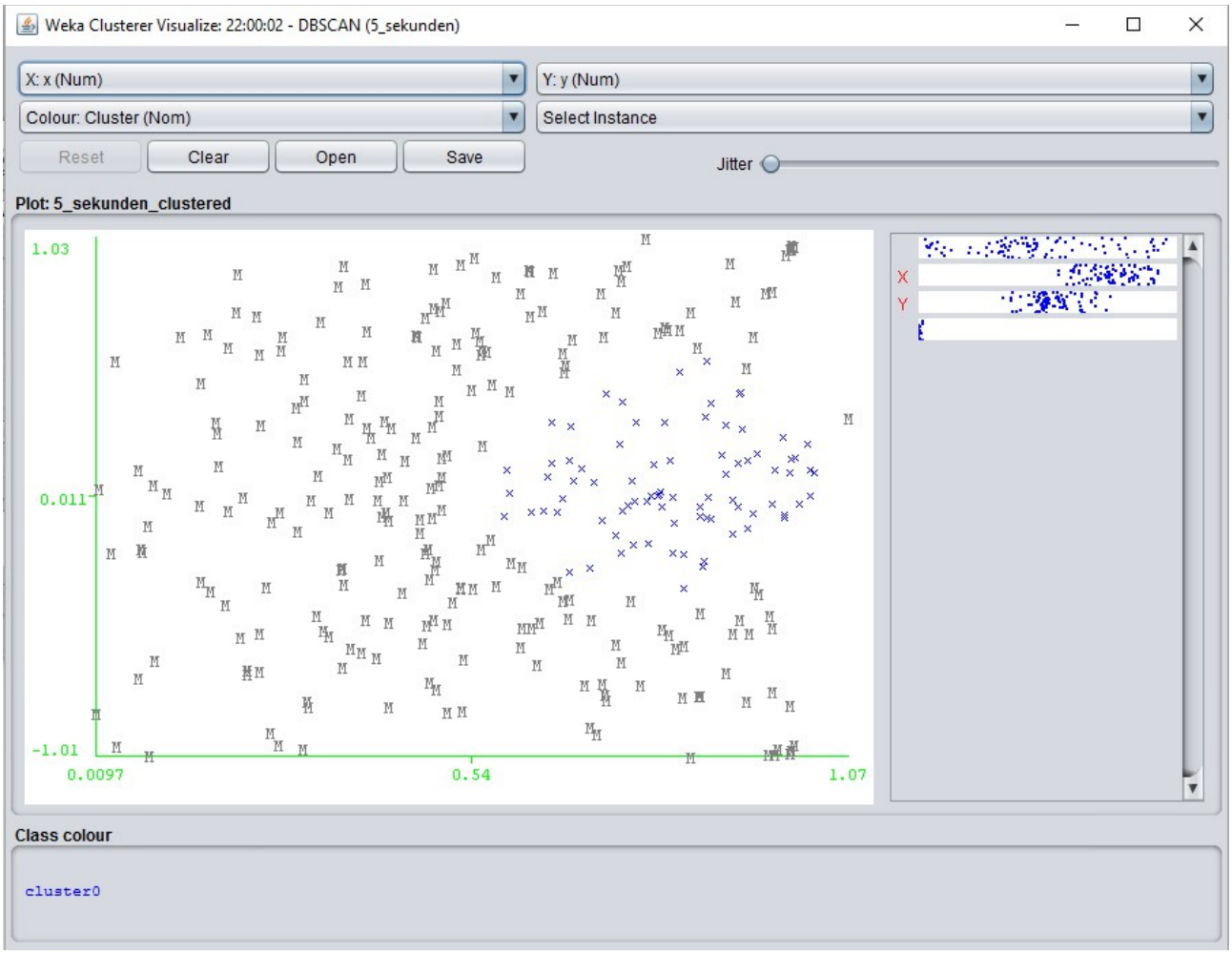

Abbildung 18:Versucht mit mindestens 90 und Radius 0,2

Die blauen Punkte repräsentieren ein Cluster. Die M-Punkte sind diejenigen Punkte, die nicht zu einem Cluster gehören. Der Bereich von 0.54 bis 1.07 auf der Grafik ist der Bereich auf dem Fußballfeld vor dem Tor, was in Bezug auf die Datei (die die fünf Sekunden vor den Torschüssen enthält) nachvollziehbar ist.

Die Dichteschätzungen durch DBSCAN reagieren sehr empfindlich auf den Radiuswert, die Dichte verändert sich sehr stark, je nachdem wie man den Radius auswählt. Die Verantwortung für die Festlegung von den beiden Parametern liegt bei dem Benutzer. Eine solche Parametereinstellung ist in der Regel nur sehr schwer zu leisten. Ein Problem dabei ist die Einschränkung von DBSCAN, dass dann, wenn Punkte sehr nah beieinanderliegen und *dadurch schwer abgrenzbar sind*, werden unterschiedliche Cluster als ein einzelnes Cluster interpretiert werden.
Auch bei Clustern mit unterschiedlichen Dichten zeigte DBSCAN Schwierigkeiten. Cluster können

für Rauschpunkte gehalten werden. Dieses Problem wird von verschieden Autoren erwähnt. In dem Buch *Introduction to data mining* von Tan, Steinbach und Kumar wird ein Beispiel vorgeführt:

> DBSCAN can have trouble with if the density of cluster varies widely, Consider Figure 8.24, which shows four clusters embedded in noise. The density of the clusters and noise regions is indicade by darkness. The noise around the pair of denser clusters, A and B, has the same density as cluster C and D. If the Eps threshold is low enough that DBSCAN finds C and D as cluster, then A and B and the points surrounding them will become a single if the Eps threshold is high enough that DBSCAN finds A and B as seperate clusters, and the points C and D the points surrounding them will also be marked as noise.[14]

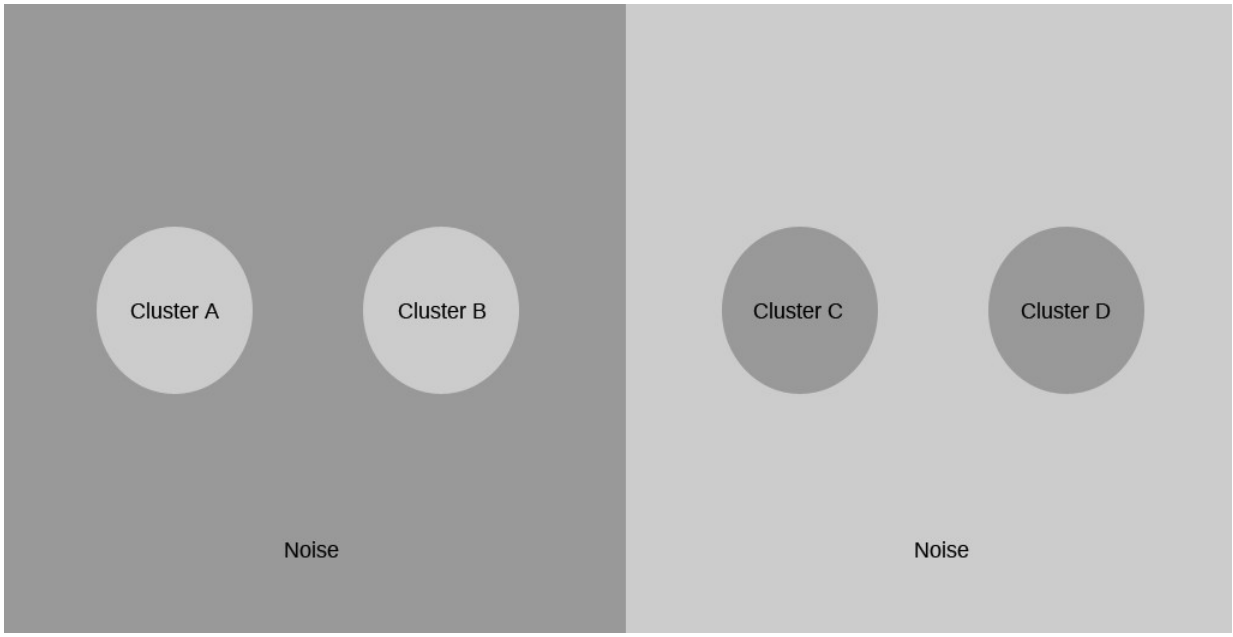

Abbildung 19:Cluster .[15]

3.2.3.2 Kerndichtschätzer (Kerneldensity)

Bei der Analyse des Algorithmus Kerneldensity wurden die Daten aus der Bundesliga 2011/2012 mit den Positionen (*x*,*y*) fünf Sekunden vor einen Torversuch verwendet. Die Analyse wurde mit dem Programm Weka durchgeführt, mit dem auch die Verteilung der Daten in einer Grafik dargestellt wird. Die Punkte sind wie auf einem Fußballfeld verteilt. Um die Häufigkeit von Punkten auf dem Fußballfeld festzustellen, werden die Positionen (*x*, *y*) getrennt. Für jede Dimension wird ein Histogramm mit Weka erstellt. Diese Histogramme zeigen getrennt die Häufigkeitsverteilung von den Koordinaten *x* und *y*. Mit den Grafiken von den Positionen *x*, *y* und den Histogrammen auf der (*x*, *y*)-Ebene ist es möglich, die Randverteilungen von diesen Positionen auf einer Fläche von 0 bis 1 (Breite) und von -1 bis 1 (Länge) in der Grafik anzuzeigen.

[14] Tan, Steinbach, Kumar (2014:529)

[15] Tan, Steinbach, Kumar (2014:530)

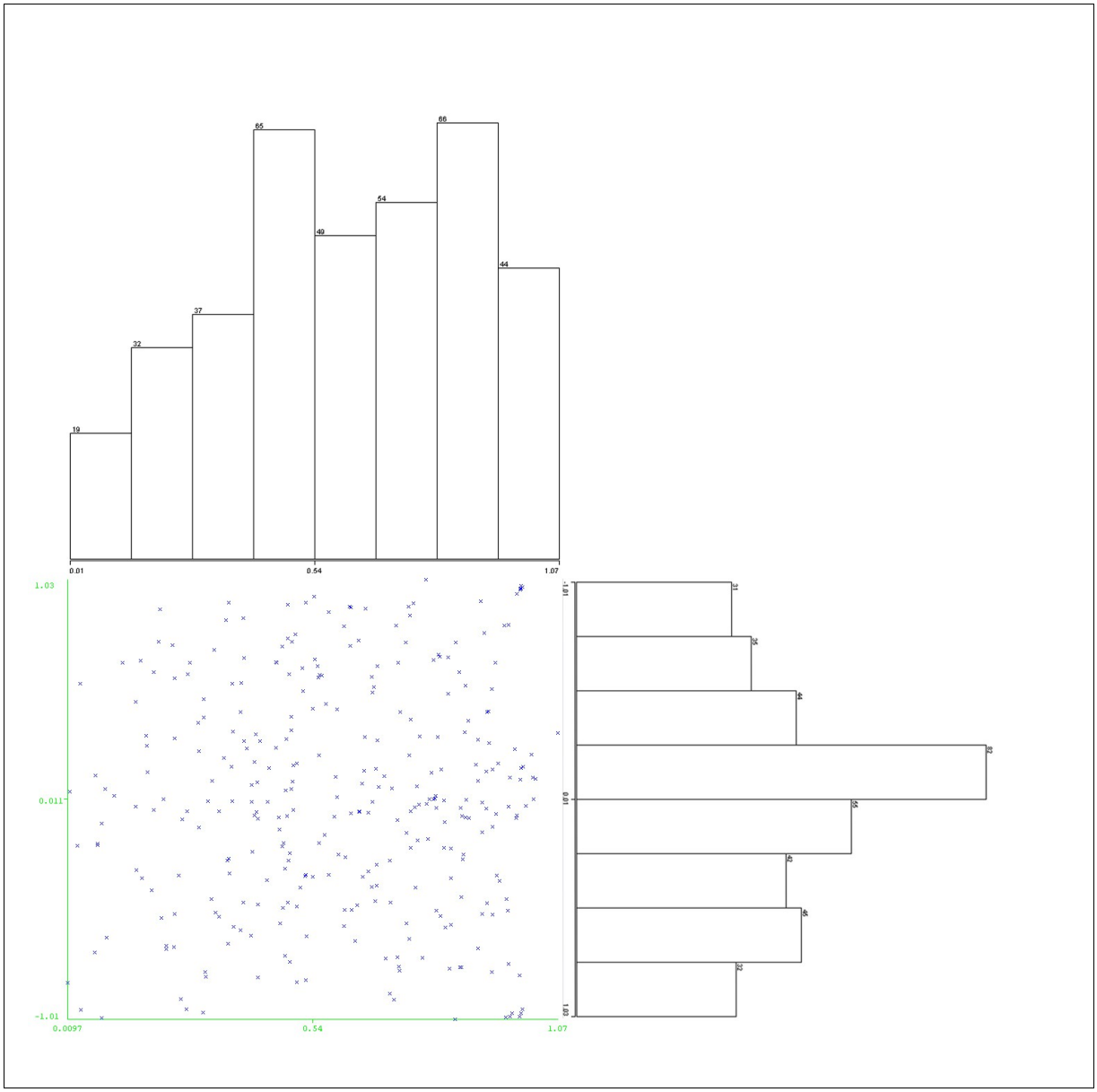

Abbildung 20:Grafik mit den Histogrammen

In der Abblildung 20 ist zu beobachten, dass sich in dem Histogramm für die *y*-Koordinate von den 336 Punkten auf der Grafik, 82 Punkte sich in dem Bereich (-0.243,0.011) befinden, und in dem Bereich (0.011, 0.266) sind es 55 Punkte.
Das Histogramm der *x*-Koordinate hat 66 Punkte in dem Bereich (0.808, 0.941), 65 Punkte in dem Bereich (0,409,0.542) und 54 Punkte in dem Bereich (0.675,0.808).

Trotz der Überlappung und großen Anzahl von Punkten, wird es jetzt möglich, die Hotspots von Punkten visuell zu identifizieren. Eine große Konzentration von Punkten findet sich in den Bereichen (0.0808, 0.941) und (0.011,0.266) der *x*-Achse. Auf der *y*-Achse sind die meisten Punkte auf der linken Seite des Fussballfelds.
Die Algorithmus Kerneldensity wird so angewendet:

– auf das Fußballfeld wird ein Grid gelegt
– es werden gleichmäßig Fixpunkte auf dem Grid positioniert
– von den Fixpunkten wird nach allen Punkten in derem Einflussradius gesucht
– für jeden gefundenen Punkt auf den Radius, wird für diesen Punkt eine Kernfunktion berechnet
– die kumulierten Ergebnisse für diese Punkte bilden den Kern auf dem Grid

3.2.3 Zusammenfassung der Analyse

Nach der Analyse der Dateien vistrack-actions und Pos wurden die wichtigen Daten für das Programm herausgefiltert. Die Daten wurden so aufgestellt, dass sie die Anforderungen der Systemfunktionen unterstützen.

Die erste Datei, die analysiert wurde, ist die Datei vistrack-actions, anhand derer haben wir die eindeutige Identifikations-Nummer von einen Spiel. Mit dieser Identifikations-Nummer können wir alle anderen Informationen relativ zu einen Spiel bestimmen. Folgende Informationen wurden festgelegt: Spieldatum, Uhrzeit, Spielliga, die Mannschaften die gegeneinander spielen, die Spielzeit von dem Beginn zum Ende der ersten und zweiten Halbzeit.

Nach der Analyse wurden folgende Informationen für das Programm herausgefiltert:

- Id einer Mannschaft
- Mannschaftsname
- Id eines Spiels
- Datum des Spiels
- Liga
- Timestamp von dem Anfang der ersten Halbzeit
- Timestamp von dem Ende der ersten Halbzeit
- Timestamp von dem Anfang der zweiten Halbzeit
- Timestamp von dem Ende der zweiten Halbzeit
- Id einer Aktion
- Typ einer Aktion
- Halbzeiten eines Spiels

Die Pos-Datei enthält die (x, y) -Koordinaten auf dem Fußballfeld zu einem Ereigniss in einem Fußballspiel aus der vistrack-Datei. Aus dieser Datei Pos haben wir die Koordinaten x, y berechnet,

die 20, 10 und 5 Sekunden vor den Ereignissen eingenommen wurden. Für jede Position gibt es eine Positions-Id, die genau die jeweiligen Koordinaten identifiziert.

- Position_Id
- Koordinate *x*
- Koordinate *y*

Anhand der aus den vistrack- und Pos-Dateien herausgefilterten Informationen wurde das Datenbank-Schema aufgebaut.

3.2.4 Zusammenfassung der Analyse der Algorithmus

Der Algorithmus Kerneldensity hat bessere Ergebnisse als der DBSCAN gezeigt. Mit ihm konnten dem Kerneldensity wurden mehrere Bereiche identifiziert werden, wo sich die Punkte häuften.
Mit dem DBSCAN-Algorithmus war es nur möglich, *ein Cluster* zu finden, denn die unterschiedliche Dichte von Punkten auf der Grafik erschwert es dem Algorithmus, mehrere Cluster zu finden. DBSCAN reagierte sehr empfindlich auf die Wahl des Radius. Das Problem mit den verschiedenen Dichten bei DBSCAN wird auch in der Literatur erwähnt.[16]
Mit Kerneldensity wird über die ganze Fläche nach Punkten gesucht, was ein Vorteil für den Algorithmus ist, da die Punkte auf der Gesamtfläche verteilt sind.
Aus diesem Grund wird der Algorithmus Kerneldensity in dem Programm Dangerzone verwendet.

3.3 Entwurf des Datenbank-Schemas

Der Datenbank Entwurf ergibt sich aus der Anforderungsanalyse der Dateien vistrac-actions und pos. Durch die Informationen aus der Analyse konnten die Spezifikationen vom Datenbank-Schema beschrieben und festgelegt werden. Zu Speicherung der Daten wurden die folgenden Tabellen erstellt.

Tabelle team Diese Tabelle enthält die Mannschaftsnamen und Id-Nummern.

Attribute:

- **name** Typ: character
- i**d** (Primärschlüssel) Typ: integer

Anzahl:18 Mannschaften pro Session

Beziehungsbeschreibung: 2 Mannschaften sind an einem Spiel beteiligt, die Beteiligen-Tabellen

[16] Tan, Steinbach, kumar(2014:529)

sind teams und matchs.

Tabelle matchs Diese Tabelle enthält Informationen über die Spiele an sich. Das Attribut **id** ist der eindeutige Schlüssel aus der Datei vistrac-actions. Das Attribut **match_date** bezeichnet das Spieldatum, **tournament_name** bezeichnet die Liga, in der das Spiel stattfand. Das Attribut **first_half_begin** gibt die Uhrzeit an, zu der die erste Halbzeit begonnen hat. **first_half_end time** gibt die Uhrzeit an, zu der die erste Halbzeit beendet wurde. **second_half_begin** ist der Anfang der zweiten Halbzeit und **second_half_end** deren Ende. Die Attribute **team_id_1** und **team_id_2** sind Fremdschlüssel aus der Tabelle teams, sie repräsentieren die beiden Mannschaften, die gegeneinander spielen. **soccer_field** ist die Länge von dem Fußballfeld.

Attribute:

id (Primär Schlüssel) Typ: integer
match_date Typ:date
tournament_name Typ**:**varchar,
first_half_begin Typ time without time zone,
f**irst_half_end time** Typ time without time zone,
second_half_begin, Typ time without time zone,
second_half_end Typ:time without time zone,
team_id_1 (Fremdschlüssel) Typ:integer
team_id_2 (Fremdschlüssel) Typ:integer
soccer_field Typ: point

- **typ** Typ-varchar

Anzahl: 306 Spiele pro Saison.
In der 1. Liga sind es 34 Spieltage, und an jedem Tag gibt es 9 Begegnungen (bei 18 Vereinen). Die Zahl 34 ergibt sich daraus, dass jeder Verein zweimal gegen jeden anderen spielen muss, also jeder Verein 17 + 17 mal spielt. Als Summe ergibt sich $34 \times 9 = 306$ Spiele pro Saison.

Beziehungsbeschreibung: in einem Spiel werden Tore geschossen oder es wird wenigstens versucht, ein Tor zu schießen. Beteilige Tabellen sind matchs und actions.

Tabelle actions Diese Tabelle speichert Daten aus der Datei vistrac-actions. In dieser Tabelle findet man bestimmte Informationen über den gesamten Spielverlauf. Die Spalte **id** ist ein künstlicher Schlüssel, der eine Aktion im Spiel eindeutig identifiziert. **file_action_id** ist eine Ganzzahl aus der

Datei vistrac-actions, diese Zahl steht für eine Aktion innerhalb eines Fußballspiels. **match_id** steht für eine Aktion in dem jeweiligen Spiel. Die Spalte **Typ** enthält zwei Arten von Aktionen, „Tor versucht“ (action-soccer-score-attempt) oder „Tor erzielt“ (action-soccer-score), diese Spalte verbindet jede Aktion mit einem Spiel. **action_time** gibt die genaue Uhrzeit an, wann die Aktion geschehen ist, **half_time** bezieht sich darauf, ob die Aktion in der ersten oder zweiten Halbzeit passiert ist.

Attribute:

id (Primärschlüssel)	Typ: integer
match_id (Fremdschlüssel)	Typ: integer
file_action_id	Typ: integer
action_time	Typ: time
halftime	Typ: integer

Beziehungsbeschreibung: eine Aktion auf das Fußballfeld geschieht zu einem bestimmen Zeitpunkt während des Fußballspiels. Beteilige Tabellen sind action und secpoints.

Tabelle secpoints Diese Tabelle beinhaltet die Koordinaten auf dem Fußballfeld vor den Aktionen aus der Tabelle actions. Die Koordinaten *x* und *y* beziehen auf den Zeitpunkt 5, 10 und 20 Sekunden vor einem Torversuch oder einem erzielten Tor. Die Spalte **id** ist ein künstlicher Schlüssel für einen Punkt auf das Fußballfeld, **action_id** verbindet eine Aktion aus der Tabelle actions mit dem Tabelle secpoints. Die Spalte **position_id** hat die Aufgabe, eine Gruppe zu formieren. Diese Gruppe ist auf 1, 5, 10 oder 20 Sekunden aufgeteilt. Die Gruppe 1 steht für die anfängliche Aktion aus der Tabelle actions, die Gruppe 5 bedeutet 5 Sekunden vor der ursprünglichen Aktion von der Gruppe 1. Die Gruppen 10 und 20 beziehen sich jeweils auf 10 und 20 Sekunden vor der ursprünglichen Aktion.

i**d** (Primärschlüssel)	Typ: integer
action_id (Fremdschlüssel)	Typ: integer
position_id	Typ: integer
position_sec	Typ: geometric

Nach der Informationsstrukturierung wird die Erstellung eines Entity-Relationship-Modells vorgenommen. Die Erstellung vom Entity-Relationship-Modells erfolgte mit dem Werkzeug Erwin.

4. Realisierung

In diesem Kapitel wird nun die Realisierung des Konzepts aus dem Kapitel 3 beschrieben.

In dem Abschnitt 4.1, Systemmodellierung, wird die Funktionalität des Anwendungssystems erläutert und ausgewählte Aspekte des Programms werden als Diagramme gezeigt. Die internen Funktionalität des Systems werden auch dargelegt.
Die Modellierung und Erstellung der Datenbank wird im Abschnitt 4.2, Datenbank erstellen, ausführlich beschrieben. Zunächst wird sie mit PostgreSQL implementiert. Anschließend werden die Daten, die für die Aufgaben des Systems relevant sind, in ihre Tabellen eingefügt. Im Abschnitt 4.2.1, Import-Programm, wird in diesem Zusammenhang die Struktur der Anwendung dargestellt, nämlich wie das Programm die benötigen Daten aus den vistrac- und pos-Dateien liest und in der Datenbank PostgreSQL speichert. Auch die Vorbereitung der Daten für das Anwendungssystem wird dargestellt.
Im Abschnitt 4.3, Ermittlung des Frames, wird beschrieben, wie die Koordinaten aus der pos-Datei berechnet und gelesen werden können. Im Abschnitt 4.3.1, Saison, wird die Aufteilung der Saison in eine Hin- und Rückrunde geschildert.
Schließlich wird in dem Abschnitt 4.4, Benutzeroberfläche-GUI, die Benutzeroberfläche konkretisiert.

4.1 Systemmodellierung

Das Programm Dangerzone wurde so konzipiert, dass die Strukturen der Daten in der Datenbank dargestellt werden. Diese Entwurfsplanung ermöglicht es, auf einfache Weise Daten aus der Daten-Dank zu laden und Daten in der Datenbank zu speichern. Die Klassen Team, Macht, Action, Saison und Position stellen für eine spätere Weiterentwicklung des Programmes Schnittstellen bereit, die eine vereinfachte Bearbeitung der Daten ermöglichen.
Bei dem Importieren der Daten in die Datenbank werden die Daten in Listen mit der Struktur der jeweiligen Klassen geladen. Die Ladung der Klassen geschieht in der Klasse Readfiler, mit der die Dateien vistrac-actions und pos gelesen werden. Für jede Tabelle ist eine Klasse zuständig, die *insert statements* werden in dieser Klasse ausgeführt.
In der Klasse Util werden die SQL Abfragen mit demn Werten der Saison, Zeitpunkte, Radiuswert und Mannschaften durchgeführt und die Daten werden aus der Datenbank geholt. Die Klasse Init bekommt die Daten, berechnet die Dichte und sortiert diese nach der Große. Die Klasse Heatmap wird dann aus der Klasse Init aufgerufen. Sie erzeugt die Heatmap als Bild und speichert das Bild auf der Festplatte. Die Gui Klasse von Dangerzone lädet das Heatmap-Bild hoch und stellt auf der Oberfläche dar.
Die Programm Dangerzone wurde in acht Paketen geteilt. Jedes Package ist verantwortlich für verschiedene Aufgaben in dem Programm.

-Package db

Die Klasse Database ermöglicht die Verbindung mit der Datenbank.

-Package importieren

Das Paket Importieren ist zuständig für das Lesen der Dateien und die Speicherung den Daten in der Datenbank

-Package util

Nachdem die Dateien vistrac-actions und pos gelesen wurden, werden die Datentypen der Attribute umgewandelt. In derselben Klasse wird auch die Spielminutenzeit in Sekunden umgerechnet. Dies wird dann später benutzt, um die Spielpositionen fünf, zehn und zwanzig Sekunden vor einer Aktion zu berechnen.

-Package data

In diesem Pakete werden die Daten hochgeladen, die später in der Datenbank gespeichert werden. Die Klassen Team, Macht,Action,Saison Positision, Machttime stellen die Tabellen in der Datenbank dar.

- Package init

In der Klasse Init werden dann die Fixpoints auf dem Fußballfeld verteilt, Eine Anpassung auf dem Bild mit dem Fußballfeld in Bezug auf die reale Große eines Fuballfeldes wird vorgenohmmen. Die Kerdichtschätzer wird berechnet und auch nach den Dichten sortiert.

-Package heatmap

Die Klasse Heatmap ist zuständig für die Erstellung von Heatmaps.

-Package gui

Dieses Paket enthält Klassen für die Oberfläche des Programm, die Import-Frame und die Fehlermeldung-Frame.

-Package main

Die Klasse main ist verantwortlich für die Ausführung des Programmes.

Um einen besseren Überblick zu sorgen, wurden zwei Klassen Diagramm gezeichnet. Die erste bezieht sich auf die Erstellung eines Heatmaps. Die Zweite ist ein Klassendiagramm für das Lesen von Dateien und die Speicherung von Daten in der Datenbank.

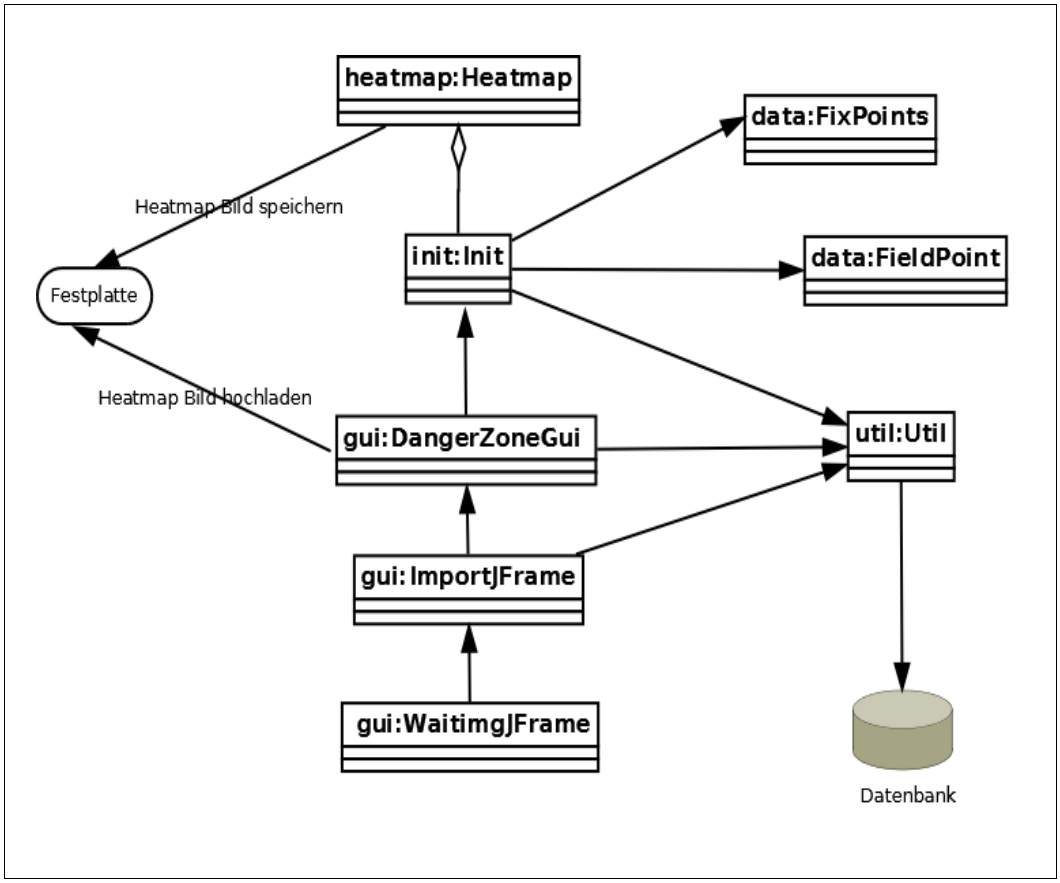

Abbildung 21:Heatmap- Klassendiagramm

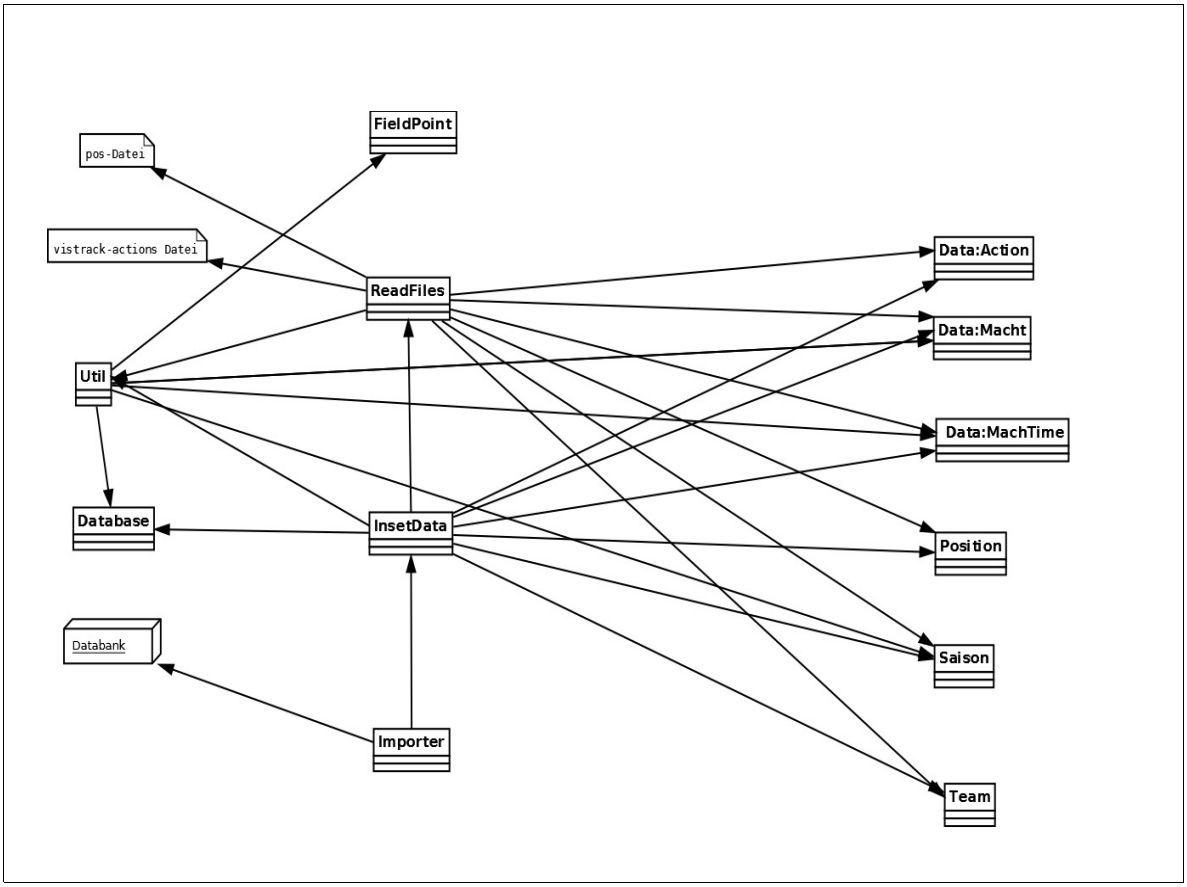

Abbildung 22:Import-Klassen Diagramm

4.2 Datenbank erstellen

Als Konsequenz des Abschnittes 3.3, Entwurf des Datenbank-Schemas, wurde die Datenbank mit der SQL-DDL (Data Definition Language) implementiert, daraus ergibt sich die in der Abbildung (20) dargestellte relationale Datenbank. Dieses Datenbankschema wurde aus dem oben genannten Abschnitt 3.3, Entwurf des Datenbank-Schemas, übernommen.

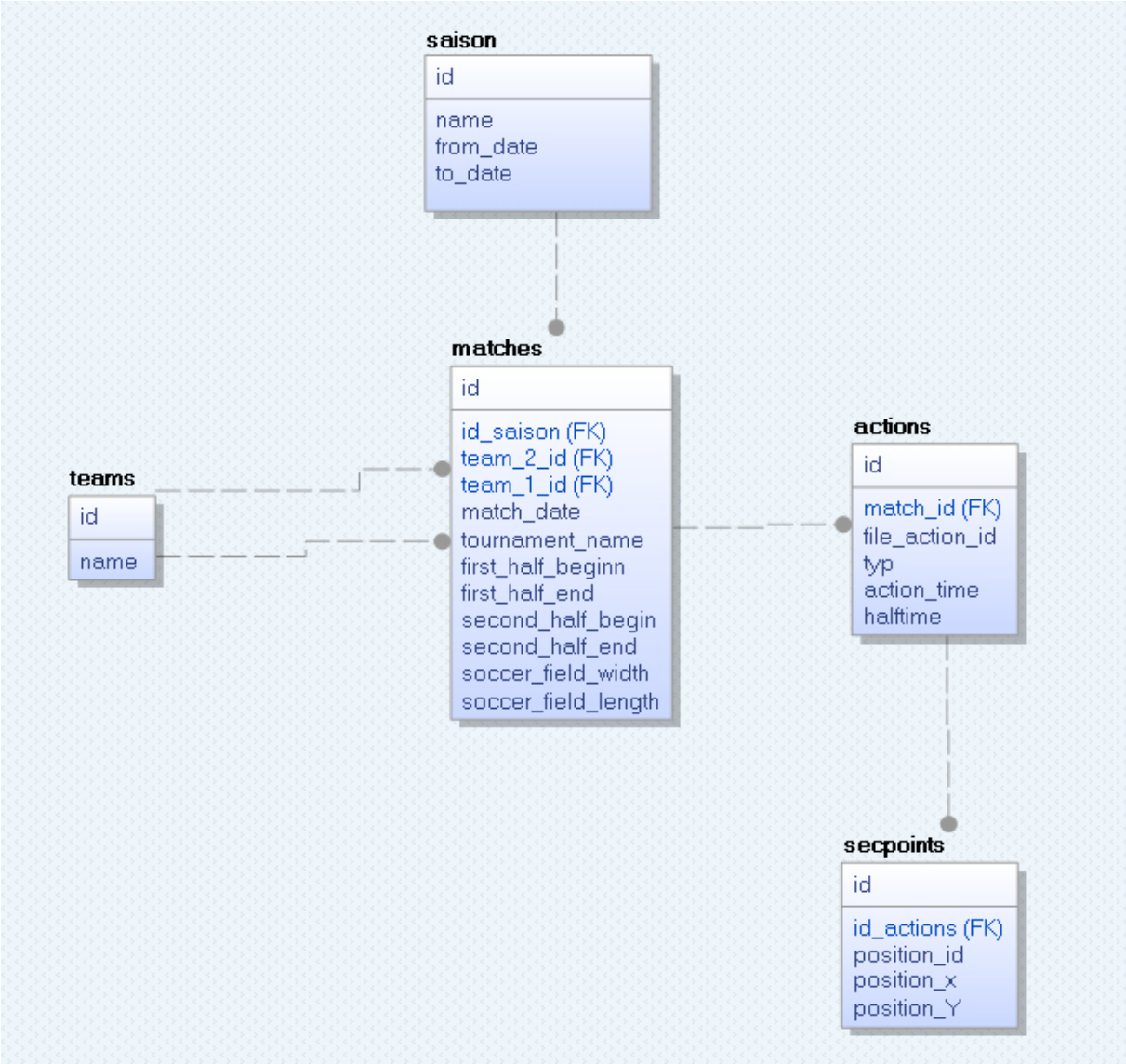

Abbildung 23:Datenbank-Schema

4.2.1 Import-Programm

Der Ausgangspunkt für das Anwendungssystem ist das Importieren der Daten in die Datenbank. Zuständig für diese Aufgabe ist das Paket Importieren. In diesem Paket befinden sich die drei Klassen Readfile, Importer und InsertData. Die Klasse Importer liest die Adresse der Dateien vistrac-actions und pos, nachher wird eine Liste erzeugt mit den Adressen der beiden Dateien. Die Liste wird dann der Klasse InsertData übergeben. Diese eröffnet dann die Verbindung mit der Datenbank. Sie erhält auch die Listen mit den Daten, die von der Klasse Readfile gelesen wurden, und ist auch für die Ausführung der *insert statements* zuständig.

In der Klasse ReadFile werden die XML-Dateien gelesen und durch den Parser in ein Dokumenten-Objekt umgewandelt, das die Hierarchie der XML-Datei darstellt. Ein solches Dokumenten-Objekt vereinfacht die Verwendung und Abarbeitung von XML-Strukturen. In dem Abschnitt 3.2.3,

Zusammenfassung der Analyse, wurden die Daten anhand ihrer jeweiligen Attribute aus der XML-Datei ausgesucht. Die Daten werden dann in Listen aufgestellt. Jede von diesen Listen hat bestimmte Typen. Diese Klassen haben dieselbe Struktur wie die Datenbank, wie Action, Team, Saison, Match und Position, letztere steht für die Koordinaten von Aktionen für die Tabelle Secpoints und wird mit einem Bufferreader gelesen.

Um die Daten zu importieren bietet das Programm in der Menüleiste das Menü Datei an, wo der Benutzer die Möglichkeit hat, die Dateien zu importieren oder das Programm zu beenden. Mit dem jButton *import* kommt man auf ein Import-Fenster, wie in der Abbildung (20) gezeigt wird. Mit diesem Fenster wird der Benutzer aufgefordert, die actions- und pos-Dateien auszuwählen. Mit dem jButton *Browse* gelangt der Benutzer auf die Datei-Verzeichnisse, wo er die Ordner von dem jeweiligen Spiel aussuchen kann. Die ausgesuchten Dateien werden dann mit Namen und Pfad in einer Tabelle angezeigt.

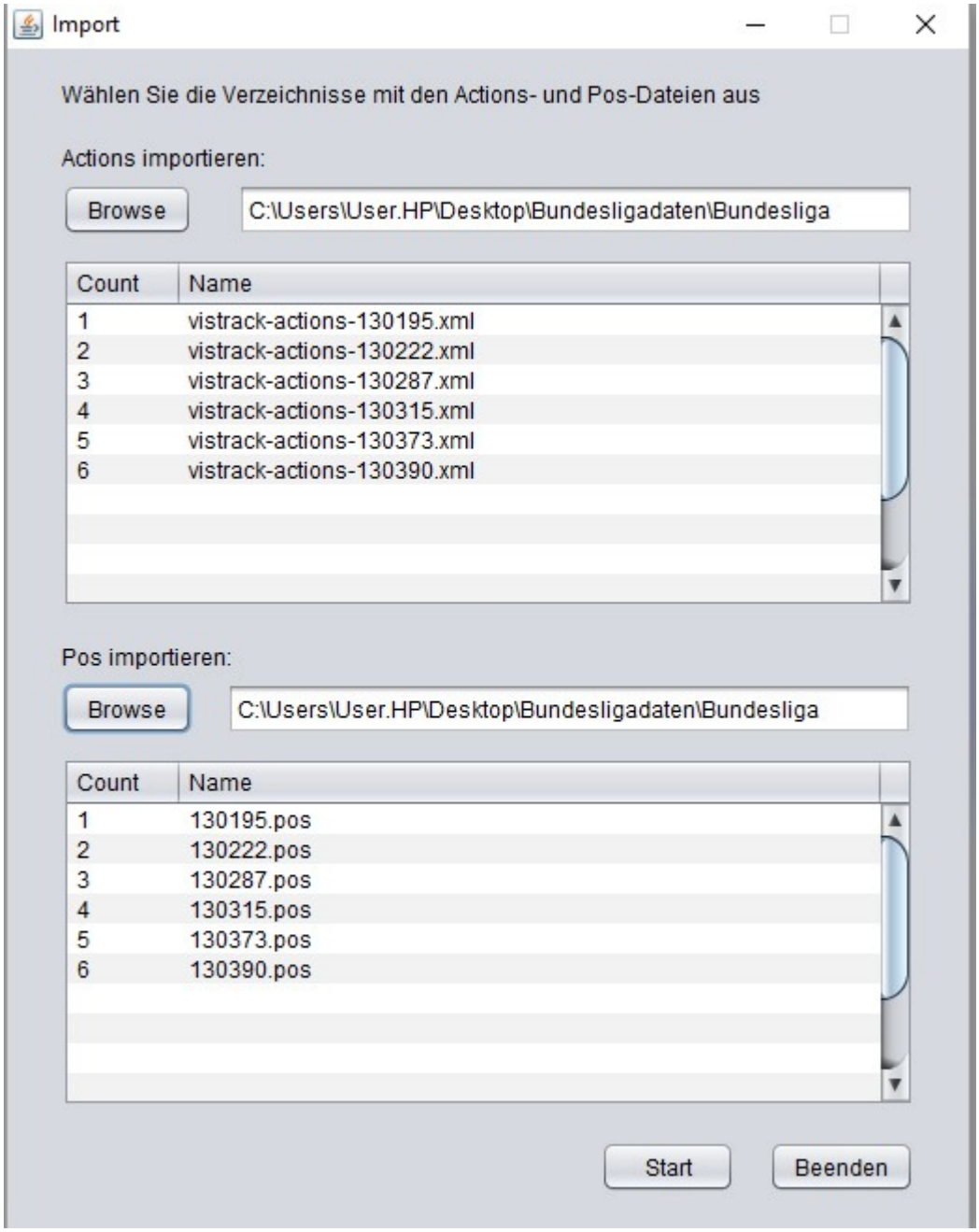

Abbildung 24:Import- Fenster

Pro Saison hat die Bundeliga 34 Spieltage, bei 18 Mannschaften sind es 306 Spiele. Das heißt pro Saison werden 306 Ordner für jedes Bundesligaspiel angelegt. Die Import-Funktion wurde so konzipiert, dass, wenn man einen Ordner öffnet, das Programm die beiden wichtigen Dateien, die pos-Datei und vistrac-actions selektiert. Es wäre auch möglich, alle Bundesligaordner in einen Zielordner zu kopieren und aus diesem Ordner nur die beiden Dateien zu importieren. So könnte man die Arbeit des Benutzers, um alle 306 Ordner anzuklicken, reduzieren.

4.3 Ermittlung der Frames

Das Programm bietet dem Benutzer bei der Datenvisualisierung die Möglichkeit an, ein Filter einzusetzen, welches dann die Koordinaten selektiert, abhängig davon, ab welchem Zeitpunkt die Offensivspielzüge vor einem Angriff visualisiert werden sollen. Aus diesem Grund wird schon beim Import der Daten, der Zeitpunkt von 5, 10 oder 20 Sekunden vor den Offensivspielzügen in der Datenbank gespeichert.

Bei der Berechnung eines einzelnen Zeitpunktes von einem Offensivspielzug wird die Zeit des Spielbeginns zusammen mit dem Anfang der zweiten Halbzeit und ob der Angriff in der ersten oder zweiten Halbzeit stattgefunden hat, aus der Datenbank gelesen. Diese Daten werden in einer Liste zusammengeführt. Um genau zu bestimmen, auf welcher Position der Angriff auf dem Fußballfeld stattgefunden hat, müssen aus der pos-Datei die Frames ausgelesen werden.

Um den genauen Zeitpunkt eines Angriffs zu bestimmen, wird zuerst festgelegt, ob dieser Angriff in der ersten oder zweiten Halbzeit stattfand. In einem zweiten Schritt wird der Zeitstempel des Angriffs um die Startzeit des Spiels oder die Startzeit der zweiten Halbzeit subtrahiert. Die Ergebnisse einzelner Aktionen werden dann mit 25 multipliziert. Die Zahl 25 entspricht der Aufteilung der pos-Datei, in den eine Sekunde aus 25 Frames besteht.

Die Frames werden bezüglich der Ergebnisse der Multiplikation und der Halbzeit des Spieles ausgesucht. Die Offensivspielzüge 5, 10 und 20 Sekunden vor einem Angriff werden ab dem Zeitpunkt des Angriffs berechnet. Bei jedem Zeitpunkt eines Angriffs werden dann entsprechend 5, 10 und 20 Sekunden abgezogen. Die Ergebnisse werden dann auch mit 25 multipliziert, damit die passenden Frames gefunden werden können. Mit beiden Daten werden die Koordinaten x und y von dem Angriff aus der Pos-Datei entnommen.

Wenn die Frames für 5, 10 und 20 Sekunden vor der Aktion gefunden sind, werden dann diese in die Datenbank in der Tabelle secpoints gespeichert.

Die untere Abbildung 25 ist ein Ausschnitt aus der pos-Datei, in Rot markiert sind die Frames, Spielzeit und Halbzeit.

```
43495,74,2,2013-04-13T17:01:02.888;#22,-0.7562,-0.0985,0.47;26,-0.0901,0.2220,0.44;5,-0.0957,-0.4197,0.72;17,
43496,74,2,2013-04-13T17:01:02.919;#22,-0.7563,-0.0977,0.48;26,-0.0900,0.2210,0.45;5,-0.0951,-0.4191,0.73;17,
43497,74,2,2013-04-13T17:01:02.966;#22,-0.7565,-0.0968,0.49;26,-0.0899,0.2200,0.47;5,-0.0946,-0.4184,0.73;17,
43498,74,2,2013-04-13T17:01:02.997;#22,-0.7566,-0.0960,0.51;26,-0.0897,0.2190,0.48;5,-0.0940,-0.4178,0.74;17,
43499,74,2,2013-04-13T17:01:03.044;#22,-0.7567,-0.0952,0.51;26,-0.0896,0.2181,0.50;5,-0.0934,-0.4171,0.75;17,
43500,75,2,2013-04-13T17:01:03.091;#22,-0.7568,-0.0944,0.52;26,-0.0894,0.2171,0.52;5,-0.0929,-0.4164,0.75;17,
43501,75,2,2013-04-13T17:01:03.122;#22,-0.7569,-0.0936,0.53;26,-0.0891,0.2161,0.53;5,-0.0924,-0.4157,0.76;17,
43502,75,2,2013-04-13T17:01:03.169;#22,-0.7569,-0.0928,0.54;26,-0.0889,0.2152,0.55;5,-0.0918,-0.4148,0.77;17,
43503,75,2,2013-04-13T17:01:03.200;#22,-0.7570,-0.0919,0.55;26,-0.0886,0.2142,0.57;5,-0.0913,-0.4140,0.78;17,
43504,75,2,2013-04-13T17:01:03.247;#22,-0.7569,-0.0912,0.56;26,-0.0882,0.2132,0.59;5,-0.0908,-0.4130,0.79;17,
43505,75,2,2013-04-13T17:01:03.278;#22,-0.7569,-0.0903,0.57;26,-0.0879,0.2123,0.61;5,-0.0903,-0.4120,0.80;17,
43506,75,2,2013-04-13T17:01:03.325;#22,-0.7569,-0.0894,0.58;26,-0.0876,0.2114,0.63;5,-0.0898,-0.4110,0.82;17,
43507,75,2,2013-04-13T17:01:03.357;#22,-0.7568,-0.0883,0.60;26,-0.0872,0.2106,0.65;5,-0.0893,-0.4099,0.83;17,
43508,75,2,2013-04-13T17:01:03.403;#22,-0.7568,-0.0872,0.62;26,-0.0868,0.2097,0.66;5,-0.0887,-0.4089,0.85;17,
43509,75,2,2013-04-13T17:01:03.450;#22,-0.7567,-0.0861,0.64;26,-0.0864,0.2089,0.68;5,-0.0882,-0.4079,0.86;17,
43510,75,2,2013-04-13T17:01:03.482;#22,-0.7566,-0.0850,0.65;26,-0.0860,0.2082,0.70;5,-0.0877,-0.4069,0.88;17,
43511,75,2,2013-04-13T17:01:03.528;#22,-0.7565,-0.0839,0.67;26,-0.0855,0.2074,0.72;5,-0.0871,-0.4059,0.89;17,
43512,75,2,2013-04-13T17:01:03.560;#22,-0.7564,-0.0829,0.68;26,-0.0851,0.2067,0.74;5,-0.0866,-0.4050,0.90;17,
43513,75,2,2013-04-13T17:01:03.607;#22,-0.7562,-0.0821,0.68;26,-0.0846,0.2067,0.75;5,-0.0860,-0.4042,0.91;17,
43514,75,2,2013-04-13T17:01:03.638;#22,-0.7561,-0.0811,0.70;26,-0.0840,0.2072,0.75;5,-0.0854,-0.4032,0.93;17,
43515,75,2,2013-04-13T17:01:03.685;#22,-0.7559,-0.0800,0.71;26,-0.0835,0.2082,0.76;5,-0.0849,-0.4023,0.94;17,
43516,75,2,2013-04-13T17:01:03.732;#22,-0.7556,-0.0793,0.71;26,-0.0829,0.2085,0.76;5,-0.0844,-0.4012,0.96;17,
43517,75,2,2013-04-13T17:01:03.763;#22,-0.7554,-0.0782,0.73;26,-0.0822,0.2080,0.78;5,-0.0839,-0.4002,0.97;17,
43518,75,2,2013-04-13T17:01:03.810;#22,-0.7553,-0.0770,0.75;26,-0.0815,0.2073,0.81;5,-0.0834,-0.3993,0.98;17,
43519,75,2,2013-04-13T17:01:03.841;#22,-0.7550,-0.0761,0.76;26,-0.0808,0.2069,0.84;5,-0.0831,-0.3984,0.98;17,
```

Abbildung 25:Pos Datei

	id [PK] serial	position_id integer	action_id integer	position_secs geometry(Point)
1	1	1	1	0101000000E6AE25E4839EEFBFDC68006F8104C5BF
2	2	5	1	0101000000BF0E9C33A2B4E1BF61C3D32B6519E43F
3	3	10	1	0101000000D044D8F0F44AE1BFA54E401361C3ED3F
4	4	20	1	01010000004DF38E537424973F96438B6CE7FBED3F
5	5	1	2	01010000009B559FABADD8EB3F3E7958A835CDBB3F
6	6	5	2	0101000000039D6C56D3480DF3F6666666666666CEBF
7	7	10	2	0101000000035EF38454772B9BF5C8FC2F5285CBF3F
8	8	20	2	0101000000E561A1D634EFC03F166A4DF38E53A43F
9	9	1	3	01010000005C2041F163CCEF3FA54E401361C3CB3F
10	10	5	3	0101000000DBF97E6ABC74E13F95D40968226CEE3F
11	11	10	3	0101000000A1D634EF3845D73F5F07CE1951DAF03F
12	12	20	3	01010000000A68226C787AE93F894160E5D022DF3F
13	13	1	4	0101000000211FF46C567DE0BF4C37894160E5C83F
14	14	5	4	01010000005F07CE1951DAE7BF89D2DEE00B93E73F
15	15	10	4	0101000000D7A3703D0AD7B33FD1915CFE43FABDBF
16	16	20	4	01010000008FC2F5285C8FD63F7B14AE47E17A84BF
17	17	1	5	0101000000B1BFEC9E3C2CF03F80B74082E2C7C0BF
18	18	5	5	0101000000FA7E6ABC7493E43FE9482EFF21FDA6BF
19	19	10	5	0101000000F163CC5D4BC8DB3FB537F8C264AAE0BF
20	20	20	5	0101000000BA6B09F9A067DB3FC217265305A3E0BF
21	21	1	6	01010000009A9999999999F1BFD42B6519E258C73F
22	22	5	6	0101000000AEB6627FD93DE7BFF54A598638D6B5BF
23	23	10	6	01010000002D211FF46C56D1BF7D3F355EBA49E23F
24	24	20	6	01010000002AA913D044D8D43F3B70CE88D2DEE63F
25	25	1	7	0101000000C1CAA145B6F3EB3F1E166A4DF38EB33F
26	26	5	7	0101000000C286A757CA32E23F5F07CE1951DAC3BF
27	27	10	7	01010000008126C286A757E23F736891ED7C3FD53F
28	28	20	7	0101000000E0BE0E9C33A2EA3FBB270F0BB5A6F03F

Abbildung 26:Tabelle secpoints

4.3.1 Saison

Die Saison der Bundesliga fängt in August an und geht zunächst bis in den Dezember. In der Hinrunde werden 17 Spiele gespielt. Von Januar bis Juni werden noch 17 Spiele in der Rückrunde gespielt. Die genauen Tagesdaten sind von Jahr zu Jahr unterschiedlich. Um die Saison eines jeden Jahres zu bestimmen, wird das Spieldatum aus der vistrac-actions-Datei ausgelesen. Abhängig davon, in welchem Monat das Spiel stattgefunden hat, wird das Spiel dann zu einer Saison hinzufügt und festgestellt, ob es zur Hin- oder Rückrunde gehört. Nachdem jedes Spiel einer Saison zugeordnet wurde, werden die Saison und das Datum der Hin- und Rückrunde in die Klasse Saison geladen und dann in der Datenbank gespeichert.

4.3.2 Anpassung des Fußballfelds

Die Koordinaten in der pos-Datei, in der alle Positionen eines Fußballspiels gespeichert sind, sind in vier Quadranten von [-1,-1] bis [1,1] aufgeteilt. Die Koordinaten (x, y) aus der pos-Datei werden nun auf einen Quadranten umgewandelt. Das normale Fußballfeld in der Bundesliga misst 105 m × 68 m. Zuerst werden die Koordinaten aus der pos-Datei auf die Maße des Fußballfeldes übertragen. Die Position der Koordinaten beschränkt sich dabei auf eine Hälfte des Fußballfeldes, da sie nur kurz vor dem Zeitpunkt eines Angriffes betreten werden. Die Höhe und Breite des Feldes werden also durch 2 geteilt und mit den Koordinaten (x, y) multipliziert. Mit der Multiplikation kann man die tatsächlichen Koordinaten in Metern auf dem Feld erhalten.

Um die Koordinaten in nur einen Quadranten zu bringen, werden die Punkte x, wenn sie negativ sind, mit –1 multipliziert. Wenn der Punkt y positiv ist, addiert man 34, sonst subtrahiert man 34 von ihm. Der Wert 34 ergibt sich aus der Division von 68 durch 2, da die Achse y von 0 bis 68 verläuft.

DieUmwandlung von Koordinaten aus der pos-Datei in die Maße des Fußballfelds erlaubt dem Benutzer eine bessere Vorstellung, wo sich die Punkte auf dem Feld befinden.

4.4 Fixpunkte

Um den Kerndichteschätzer zu berechnen, wurde er zuerst auf den ganzen Fußballfeld-Punkten verteilt. Jeder Punkt hat einen Abstand von 3,5 Metern zum nächsten Punkt in der Breite und von 3,4 Metern in der Länge. Es sind 14 Punkte in der Breite und 19 Punkte in der Länge, das macht insgesamt 266 Punkte. Die Punkte decken die gesamte Fläche des Fußballfelds ab. Sie werden dann als Referenz verwendet, wo sich mit größter Wahrscheinlichkeit eine Dichte von Punkten oder Hotspot befinden.

Diese Punkte werden *fixed points* oder Fixpunkte genannt und für jeden Punkt wird eine Funktion

zur Schätzung der Dichte berechnet. Die Werte liegen zwischen 0 und 1, und je größer der Dichte ist, desto mehr tendieren die Werte zu 1.

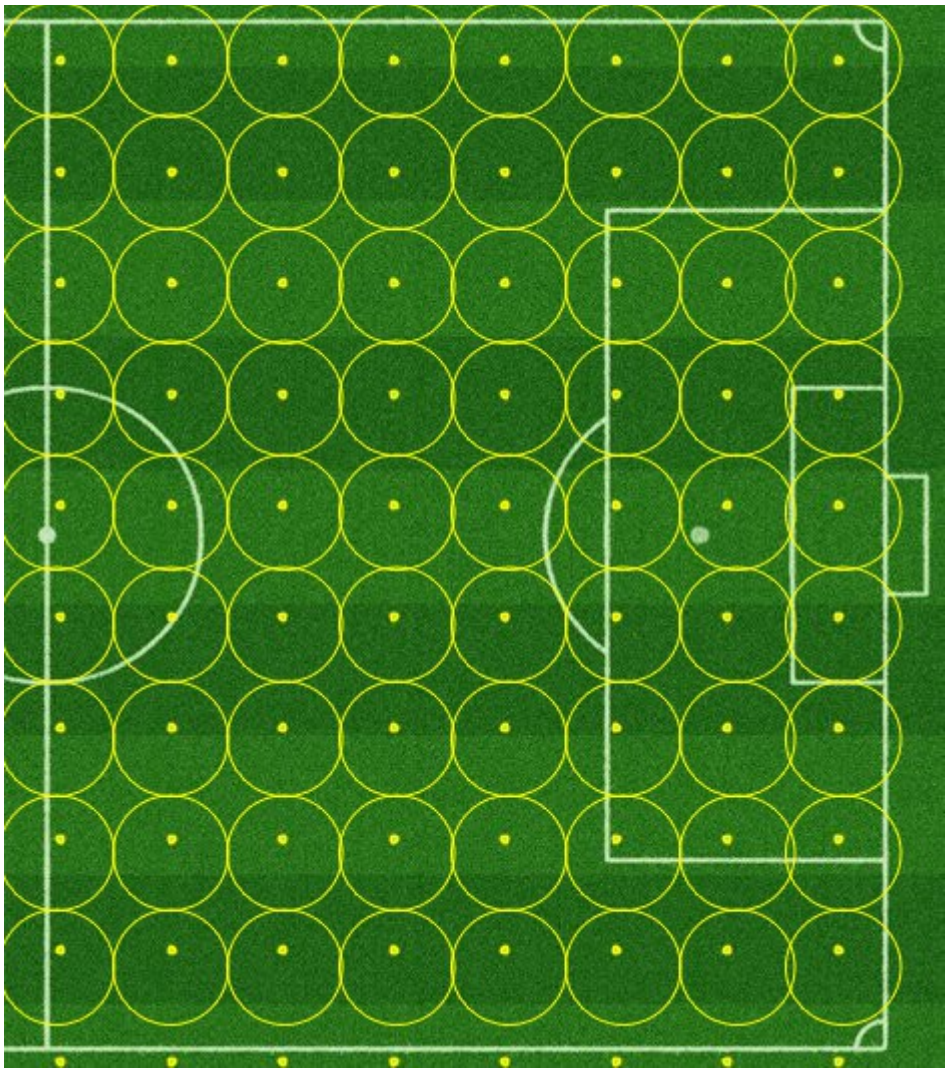

Abbildung 27:Fulballfeld mit Fixpoints

```
public List<FixedPoint> getFixedPoints(List<Point> pointFields) {

            List<FixedPoint> fPoints = new ArrayList<FixedPoint>();

            Double x = 3.5;
            Double sumX = 0.0;
            Double y = 3.4;
            Double sumY = 0.0;
            for (int width = 0; width < 14; width++) {
                  sumX += x;
                  sumY = 0.0;
                  for (int heigth = 0; heigth < 19; heigth++) {
                        sumY += y;
                        FixedPoint fp = new FixedPoint(sumX,  sumY);
                        fPoints.add(fp);
                  }
            }
```

Abbildung 28:Fußballfeld mit Fixpoint implementierung

4.4.1 Kerndichteschätzer

Jeder von diesen Fixpunkten hat einen bestimmten Radius, der vom Benutzer ausgewählt wird. Die Koordinaten werden entsprechend der Eingabe des Benutzers aus der Datenbank geladen. Der Abstand zwischen den Fixpunkten und den Koordinaten wird durch die euklidische Distanz berechnet. Die euklidische Distanz ist gemäß dem Satz von Pythagoras der Abstand von zwei Punkten in einem kartesischen Koordinatensystem.

$$dist(\rightarrow x, \rightarrow y) = \sqrt{(x_1^2 - y_1^2) + (x_2^2 - y_2^2)}$$

Mit dem vom Benutzer ausgewählten Radius wird dann überprüft, ob die einzelne Koordinate zu einem Fixpunkt gehört. Für jeden Fixpunkt wird so eine Liste von Koordinaten erzeugt, die sich innerhalb des Radius von diesem Fixpunkt befinden.
Wie in dem Abschnitt Kerndichteschätzer erläutert wurde, wird für jeden Fixpunkt eine Dichte in Bezug auf die Koordinaten berechnet. Diese Berechnung der Dichte wird in folgender Abbildung gezeigt.

```
     public Double calculateDensity(FixedPoint fixedPoint) {
           Double sum = 0.0;
           for (FieldPoint point : fixedPoint.getPoints()) {
           //sum += getGaussKernel(point.getDistanceToFiexedPoint()) *
(point.getDistanceToFiexedPoint()/radius);
           sum += getGaussKernel(point.getDistanceToFixedPoint()/radius);
           }
           sum = sum /radius*(fixedPoint.getPoints().size());

           return sum;
           }
           private Double getGaussKernel(Double distance) {
           if (distance == null) {
           return null;
           }
           return ((1/Math.pow(2*Math.PI,2))) * _Math.exp((-
(1/2*radius)*distance*distance));
```

Abbildung 29:Kerndichteschätzer implementierung

Nach der Berechnung werden dann die Fixpunkte nach der Dichte sortiert. Um eine gute Darstellung von den Hotspots zu erstellen, werden dann die ersten fünf Fixpunkte selektiert. Die

fünf Fixpunkte mit größter Dichte wurden ausgesucht, um eine Überflutung von Punkten auf dem Fußballfeld zu vermeiden, die fast keine Aussage über die tatsächliche Anhäufung von Punkten oder Dichte von Punkten auf einer bestimmten Fläche des Fußballfelds liefert. Nachdem die Fixpunkte ausgesucht wurden, werden die Koordinaten der Punkte, die zu einem Fixpunkt gehören, einer Liste hinzugefügt. Diese Liste wird dann an die Klasse Heatmap übergeben, wo die Visualisierung von den Heatmaps entsteht.

4.5 Aufbau der Heatmap

Zum Aufbau von der Heatmap werden die Koordinaten von den Punkten aus den fünf Fixpunkten mit den größten Dichteschätzern ausgesucht. Wie bereits im Konzept erwähnt wurde, repräsentieren diese Punkte, wo sich auf dem Fußballfeld die sog. Hotpots, d. h. eine größere Häufung von Punkten befinden. Diese List von Punkten wird dann an die Klasse Heatmap übergeben, wo die Visualisierung der Hotspots mit einer Farbskala erzeugt wird.
Aus der übergebenen Liste wird eine Struktur erzeugt, um die Punkte zu gewichten. Mit einer Hash-Tabelle wird berechnet, wie oft ein Punkt in der Datenmenge vorkommt. Die Anzahl wird dividiert durch die maximale Anzahl. Je höher die Häufigkeit eines Punktes ist, desto größer ist die Intensität auf der Oberfläche. Um das Verhältnis zwischen Transparenz und Intensität zu setzen, wird die Anzahl des Vorkommens von jedem Punkt mit der Zahl 0,3 multipliziert.
Der Wert 0,3 wurde nach der Regel von dem Porter–Duff-Algorithmus festgelegt, er ist eine Methode, um zwei Bilder zu überlappen, indem eine Transparenz zwischen den Bildern auszuwählen ist. Aus den Farben A und B wird die neue Farbe C berechnet, wobei die Transparenz durch $\alpha a,\ \alpha b, \alpha c$ dargestellt wird.

Der Multiplikator zwischen der Transparenz und der Intensität wird in der Variable *opaque* gespeichert. Dieser Wert wird für jeden Punkt berechnet und gibt das Gewicht von jedem *Wärme*-Punkt (hotspot) wieder.

Es gibt verschiedene Verfahren, wie man die Intensität (Gewicht) eines Pixels zu berechnen hat. Dadurch wird festgelegt, wo die Intensität am stärksten ist und wie die Intensität von naheliegenden Pixeln abfällt. In diese Arbeit wurden zwei Verfahren ausprobiert.
Das erste Verfahren war der Gauß-Filter. Er wird in der Bildbearbeitung zur Glättung von Bilder verwendet, mit Glättung ist hier in dieser Arbeit eine Kontraständerung in der Pixelfarbe gemeint. In Grunde genommen ist ein Filter eine Matrix von Koeffizienten, die eine direkte Manipulation der Pixelwerte in einem Bildbereich ermöglichen. Es gibt beliebig große Filter von 3×3, 5×5 etc. und mit verschieden Filterkoeffizienten, abhängig von dem Effekt, der realisiert werden sollte. Bei dem

Gauß-Filter hat der mittlere Koeffizient den maximalen Wert, mit steigender Distanz zum Mittelpunkt werden die restlichen Koeffizienten ihren Wert fortlaufend reduzieren. Die Gewichtung oder Skalierung ist die Summe aller Koeffizienten in der Filtermatrix. Ziel der Gewichtung ist, dass die Ergebnisse der Pixel des Ausgangsbilds in dem ursprünglichen Wertebereich bleiben.

$I'(u,v)$ ist das Ausgangsbild.

$H(i,j,)$ ist die Filtermatrix.

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

Abbildung 30:Gauß-Filter

Die Filtermatrix wird über dem Zielpixel positioniert. Die Elemente der Filtermatrix werden mit dem Zielpixel multipliziert und aufsummiert. Das Ergebnis wird dann auf der Position auf dem Ausgangsbild fixiert.

Beim Lauf über die Dimensionen von dem Eingangsbild wird die Filtermatrix mit den korrespondierenden Pixeln des Eingangsbilds multipliziert.

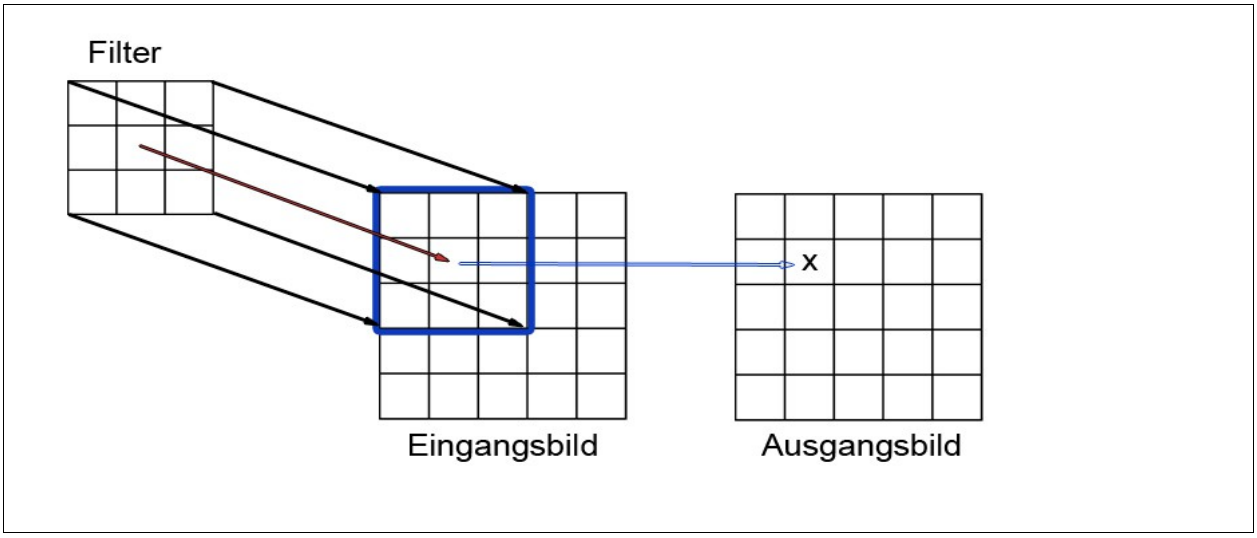

Abbildung 31: Gauß-Filter Abbildung Ausgangsbild

Mit dem Plug-in ij.plugin.filter.GaussianBlur aus der Bibliothek von ImageJ wurde auf jedem Punkt ein Gauß-Filter implementiert, mit dem Ziel, auf jeden Punkt einen Radial-Gradienten zu simulieren, um den Farbenverlauf zu gewichten. Bei dem Versuch diese Bibliothek zu verwenden, fiel die Laufzeit relativ langsam aus. Auch die Randbehandlung musste bearbeitet werden, da sich ein Teil des Filters dann außerhalb der Eingangsbilds befindet.

Bei dem zweiten Verfahren wurde anstatt der Gaußfunktion ein radialer Gradient wie in der Abbildung 32 eingesetzt. Ein radialer Gradient ist ein radialer Farbverlauf, bei dem der Mittelpunkt des Kreises die maximale Farbintensität erhält. Diese Intensität wird nach außen verbreitet und verliert mit dem Farbverlauf ihre Werte. Auf jedem Koordinatenpunkt wird der radiale Gradient platziert. Danach werden dann die Farben verwendet.

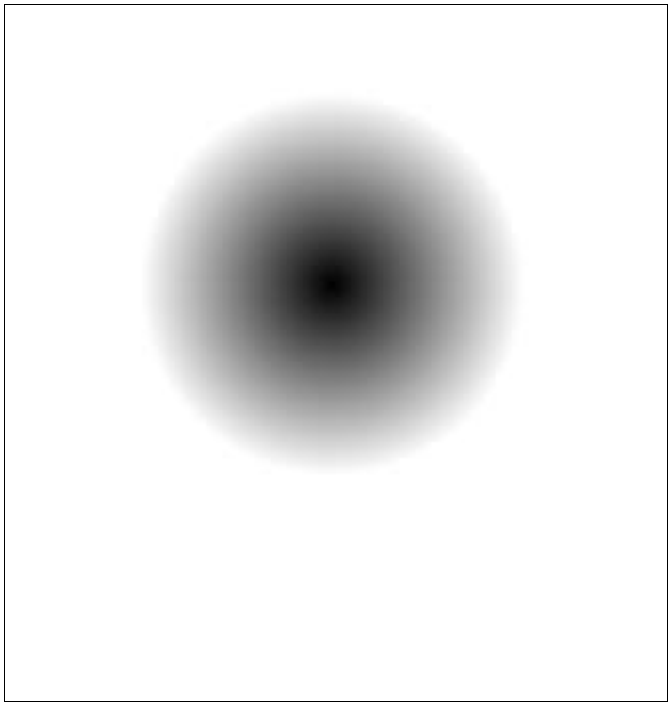

Abbildung 32:Radialgradient

Bei diesem Verfahren wird zuerst eine Intensitätsmaske erstellt. Eine Intensitätsmaske ist ein Schwarz-Weiß-Bild, in dem die Graustufen die Intensität von jedem Pixel repräsentieren.
Aus der Klasse Data wird eine Liste mit allen Koordinaten von den Punkten aus den ersten fünf Fixpunkts übergeben. Aus dieser Liste wird eine Hashtabelle erzeugt und den Punkten, die gleiche Koordinaten haben, wird eine höherer Wert zugewiesen. Dieser Wert zeigt die Intensität, oder wie oft die gleichen Koordinaten auf dem Fußballfeld vorgekommen sind. Je häufiger ein Punkt vorkommt, desto größer ist deshalb die Intensität der Farbe des Pixels.

Zuerst wird ein weißes Bild von der Größe des Fußballfeldes erzeugt. Auf jeden Punktkoordinaten wird das Radiale-Gradienten-Bild mit seiner zugehörigen Transparenz auf dem weißen Fußballfeld-Bild drauflegt.
Die Methode getInstance (int rule, float alpha) aus der Klasse AlphaComposite der Bibliothek von Java2D ermöglicht es, ein Bild mit einem anderen Bild unter bestimmen Regeln zu Transparenzen zusammenzustellen. In der Methode getInstance werden zwei Parameter angegeben. Der erste Parameter stellt die Regel vom Transparenzeffekt dar und im Programm wird die Regel *AlphaComposite.SRC_OVER* verwendet, die ein Quellenbild über ein Zielbild platziert. Der zweite Parameter enthält den Tranparenzwert, der im Programm für jeden einzelnen Point berechnet wird. Die Tranparenzwerte gehen von 0 bis 1, indem die 1 für undurchsichtig und 0 für durchsichtig steht. Diese Transparenz wird daraus berechnet, wie oft eine Koordinate vorkommt und dem Parameter 0.3 alpha. Die Methode drawImage fügt dann die radialen Gradienten auf dem weißen Fußballfeld zusammen.

```
   private void addImage(final BufferedImage buff1, final BufferedImage
weißbild,
            final float opaque, final int x, final int y) {


        final Graphics2D g2d = weißbild.createGraphics();

        // AlphaComposite Klasse implementiert grundlegende Alpha-
Compositing-Regeln zum Kombinieren von Quell- und Zielfarben, um Misch-
und Transparenzeffekte mit Grafiken und Bildern zu erzielen.
        // AlphaComposite.SRC_OVER : transparenzeffekt wird über das
output-bild angewendet.

g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
opaque));


        g2d.drawImage(ausgabebild, x, y, null);
        g2d.dispose();
```

Abbildung 33:Methode drawImage

Die methode drawImage fügt zussamen die Radialgradient auf dem Weisses Fussballfeld.

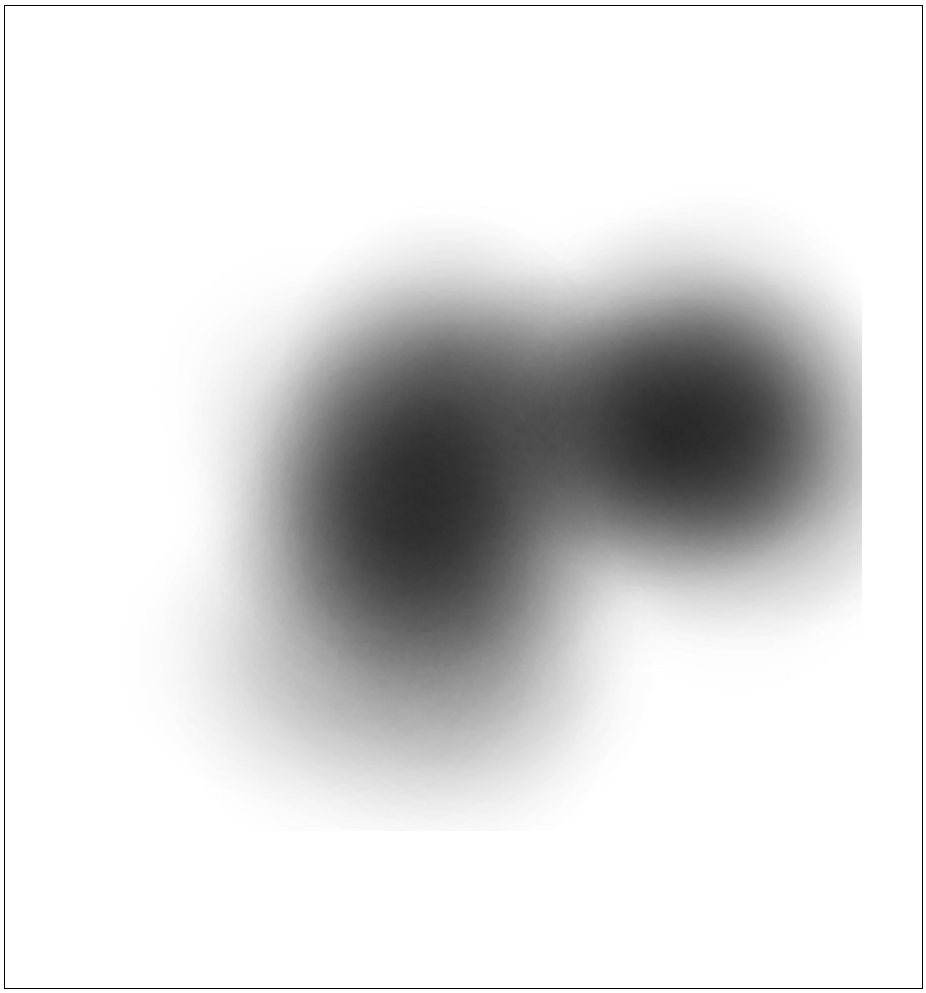

Abbildung 34:RadialeGradient auf den Weisses Fußballfeld

Der nächste Schritt besteht darin, eine Intensitätsmaske zu erstellen. Eine Intensitätsmaske ist grundsätzlich ein Graustufenbild (mit 256 Graustufen), das wir verwenden, um die Intensität jedes Pixels in unserem endgültigen Bild zu markieren. Jeder Farbton im weißen bis schwarzen Spektrum wird in unserer Farbverlaufspalette direkt auf eine tatsächliche Farbe abgebildet.

```
private BufferedImage negateImage(final BufferedImage img) {
        final int width = img.getWidth();
        final int height = img.getHeight();
        for (int x = 0; x < width; x++) {
            for (int y = 0; y < height; y++) {


                final int rGB = img.getRGB(x, y);
                final int r = Math.abs(((rGB >>> 16) & 0xff) -
255);

                final int g = Math.abs(((rGB >>> 8) & 0xff) - 255);

                final int b = Math.abs((rGB & 0xff) - 255);
                img.setRGB(x, y, (r << 16) | (g << 8) | b);
            }
        }
        return img;
    }
```

Abbildung 35: Intensitätsmaske Implementierung

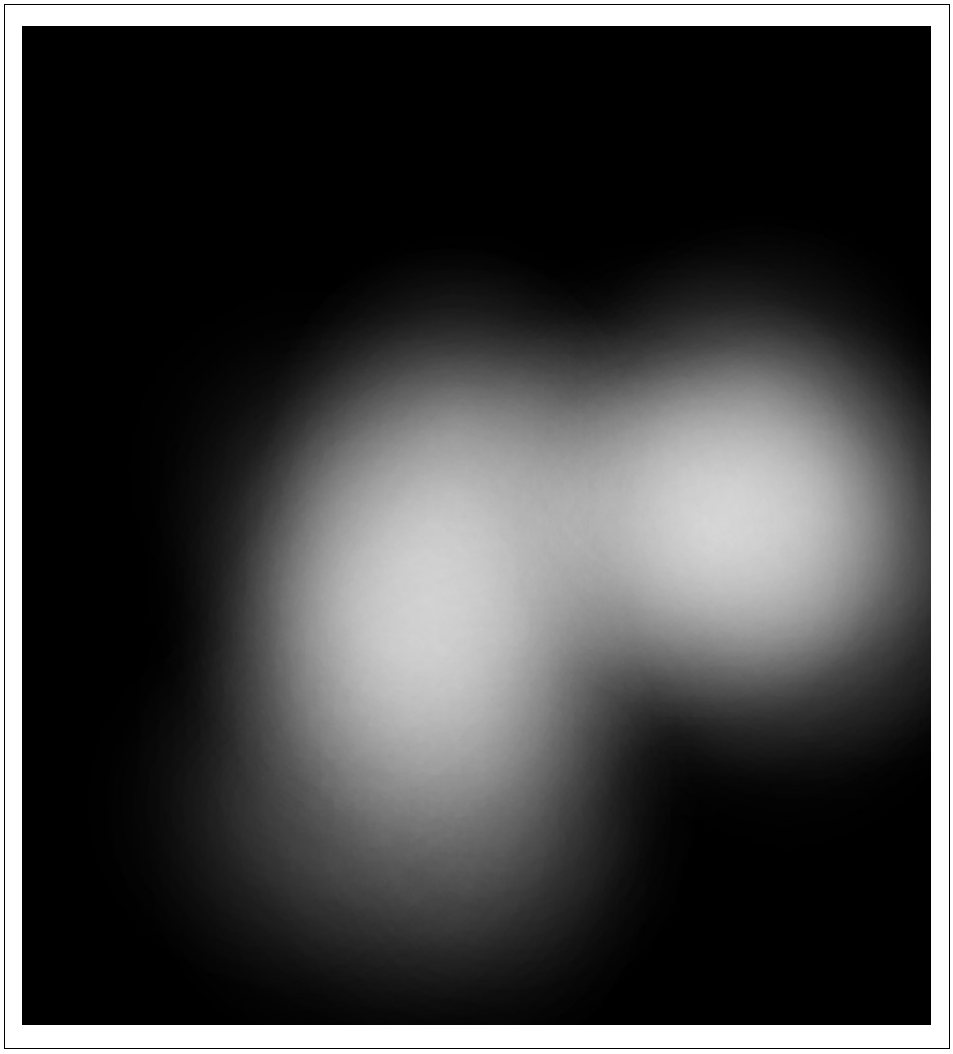

Abbildung 36: Intensitätsmaske

Java ermöglicht es mit der Klasse BufferedImage, Pixel zu lesen und zu schreiben. Dabei wird der Standard RBG benutzt. Der Standard RBG besteht aus der Kombination von den drei Farbkanälen Rot (R), Grün (G) und Blau (B), mit dem allen anderen Farben entstehen können. Jeder Farbkanal hat den maximalen Wert von 255. Die weiße Farbe hat den Wert 255 in den die drei Farbkanälen, Schwarz hat im Gegensatz dazu den Wert 0 in alle drei Farbkanälen.

Um ein negatives Bild zu erzeugen, wird die aktuelle Farbe des Pixels minus 255 (mod 256) berechnet. Mit dieser Berechnung werden die hellen Farben zu dunklen Farben und die dunklen zu hellen Farben. Die Berechnung ist in der folgenden Abbildung 28 dargestellt. Durch die Subtraktion von 255 von jeder Kanalfarbe werden die Farben invertiert.

4.6 Farben

Um die Buntfarben zu erzeugen, werden die drei Farbkanäle aus dem negativen Bild jedes Pixels gelesen. Das Graustufenbild (mit 256 Graustufen) wird verwendet, um die Intensität jedes Pixels zu markieren. Die drei Werte der Farbkanäle Rot, Grün und Blau werden miteinander multipliziert. Nach der Multiplikation wird das Ergebnis durch 16 581 375 geteilt. Diese Zahl bezieht sich auf die Farbe Weiß (255*255*255). Auf dem negativen Bild ist es die Farbe Weiß, die die Stellen markiert, wo sich die Hotspots befinden. Jeder Farbton von dem weißen bis zum schwarzen Spektrum wird in der Farbverlaufspalette direkt auf eine tatsächliche Farbe abgebildet.

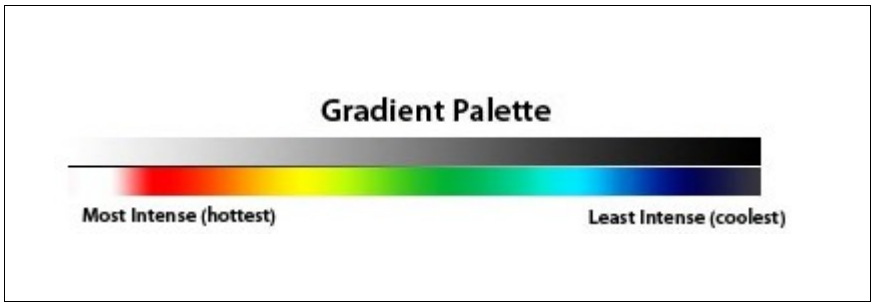

Abbildung 37:Farbenverlaufpalette

In der Variablen `colorGradient` wird die Farbverlaufspalette gespeichert. Die Farbverlaufspalette hat die Höhe von 499 Pixel. Jeder Pixel korrespondiert mit einem Farbton von Dunkelblau bis Dunkelrot.

Die Variable `multiplier` hat Werte zwischen 0 und 1. Dieser Wert entsteht durch die Division des Grautons aus dem Schwarz-Weiß-Bild durch die Zahl 16581375. Mit der anschließenden Multiplikation der beiden Variablen `colorGradient` und `multiplier` wird ein Farbwert zwischen 0 bis 499 erzeugt. Diese Werten zeigen auf die Skala von der Farbverlaufspalette, wo die Pixelfarben stehen.

```
private void remap(final BufferedImage heatMapBW) {
 final BufferedImage colorGradiant = loadImage(SPECTRUMPIC);

       final int width = heatMapBW.getWidth();
       final int height = heatMapBW.getHeight();
       final int gradientHight = colorGradiant.getHeight() - 1;
       for (int i = 0; i < width; i++) {
           for (int j = 0; j < height; j++) {


               final int rGB = heatMapBW.getRGB(i, j);


               float multiplier = rGB & 0xff;
               multiplier *= ((rGB >>> 8)) & 0xff;
               multiplier *= (rGB >>> 16) & 0xff;
               multiplier /= 16581375;

               final int y = (int) (multiplier *gradientHight);



               final int mapedRGB = colorGradiant.getRGB(0, y);

               heatMapBW.setRGB(i, j, mapedRGB);
           }
       }
   }
```

Abbildung 38:Farbenberrechnung

Für jedes Pixel des Schwarz-Weiß-Bildes wird die aktuelle Farbe verändert. Je weißer ein Pixel ist, desto höher wird die Intensität der Pixelfarbe sein. Der Mittelpunkt des Kreises des radialen Gradienten hat genau die Koordinaten eines Punktes und zeigt, wo die Punktdichte konzentriert auftritt. Der Gradientenstrahl des radialen Gradienten wird nach außen gleichmäßig skaliert. Je größer der Abstand von dem Mittelpunkt des Kreises wird, desto niedriger ist die Intensität der Farben des Pixels und die Dichte von Punkten wird niedriger sein. Wenn Nachbarpunkte auf eine Flächeneinheit treffen, haben die Pixel die gleichmäßigen Werte von den Grautönen. Die Grautöne werden dann in eine ähnliche Farbe umgerechnet. Dies ermöglicht, dass der Gradientenstrahl verschiedene Formen annehmen kann, abhängig davon, wie der radiale Gradient beziehungsweise die Punkte verteilt sind.

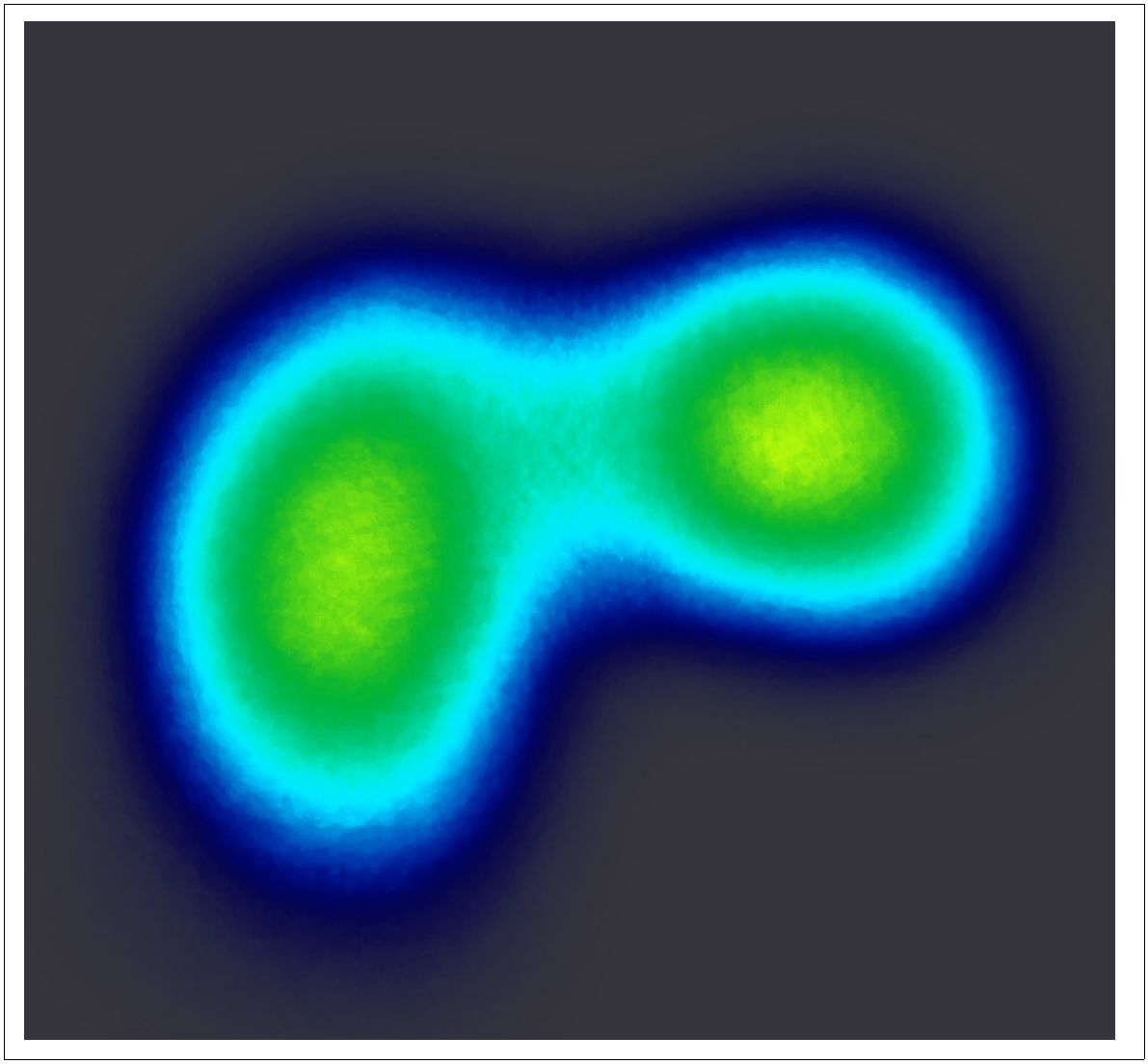

Abbildung 39:Gradientenstrahl

Der nächste Schritt ist jetzt, das Fußballfeldbild unter die Zeichnung der farbigen Hotspots zu legen. In der Methode setComposite() werden dann das Bild von den Fußballfeld mit dem der Hotspots kombiniert. Zwischen die beiden Bilder wird ein Transparenzbild gelegt. Dies soll den Kontrast zwischen den beiden Bildern deutlicher hervorheben. Das Bild wird dann auf der Festplatte gespeichert.

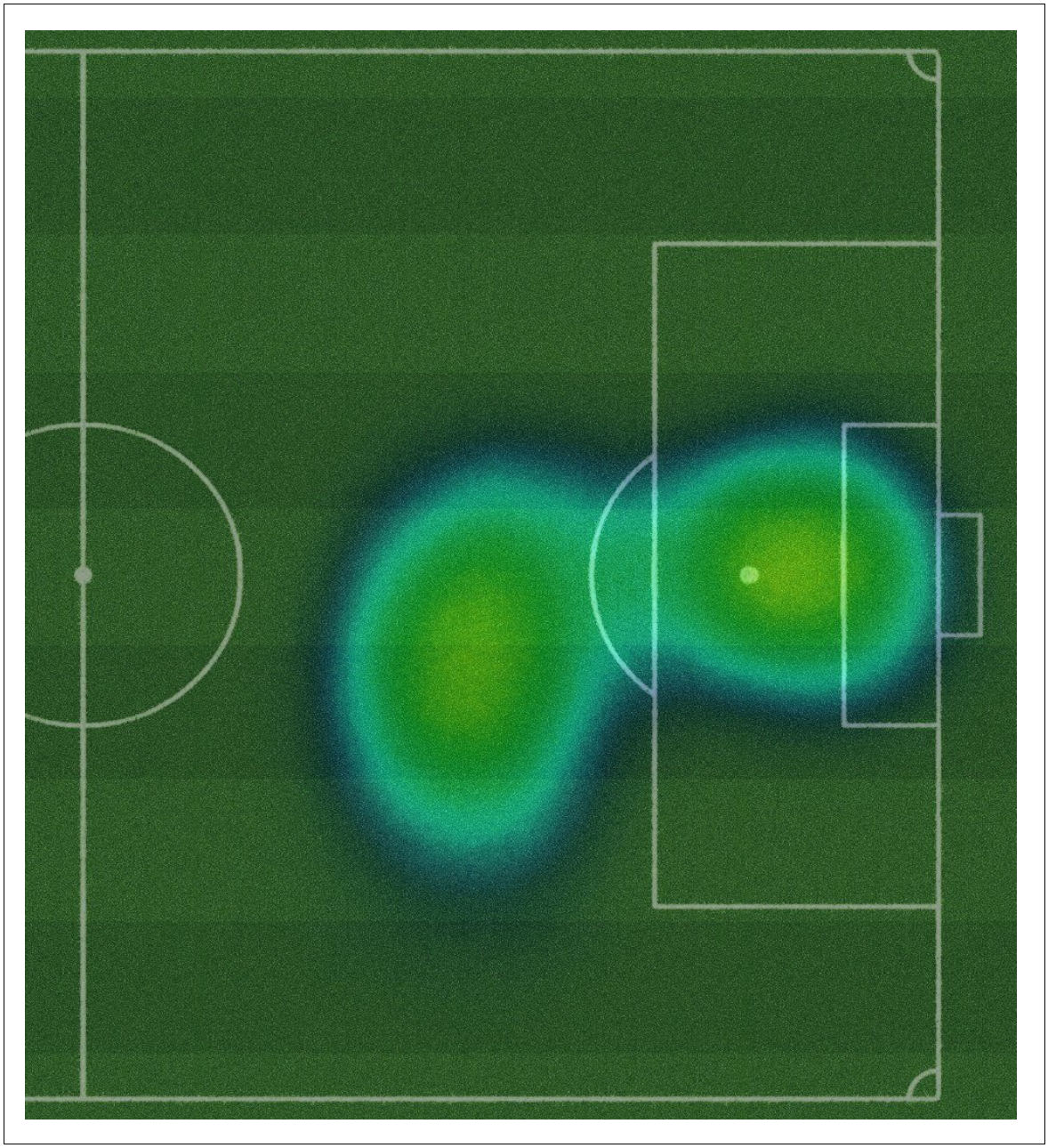

Abbildung 40: Heatmap mit 5 Sekunden

4.7 Benutzeroberfläche

Die Bedienkonzept der Anwendung ist ein fensterbasierter Desktop (siehe Abbildung 20). Die Benutzeroberfläche wird durch ein Fenster visualisiert. Dieses Fenster kann in zwei Teile aufgeteilt werden. Der rechte, größere Teil zeigt ein Fußballfeld, auf dem die Heatmaps gezeigt werden. Der linke Teil wird von dem Benutzer kontrolliert. Oben links auf der Taskleiste befinden sich der jButton „Datei“ mit dem der Benutzer das Programm beenden oder die Daten in die Datenbank importieren kann.

Unter dem jButton „Datei“ wird den Benutzer aufgefordert, die Saison auszusuchen, auf der die Heatmaps basieren sollen. Mit dem jBotton „Saison auswählen“, bekommt man eine Auswahl von den Fußballsaisons der Bundeliga gezeigt, die in der Datenbank zur Verfügung stehen. Durch den jRadiusbutton „Sekunden vor Torschuss“ wird der Zeitpunkt für die Aktion vor den Torschuss-Gelegenheiten festgelegt. Der Benutzer hat die Möglichkeit von fünf, zehn oder zwanzig Sekunden.

Die jLabel „Radius“ ist der Suchradius von einem Punkt auf dem Fußballfeld, in diesem Punkt werden mehrere Beobachtungen oder Positionen des Balls zu einem gewissen Zeitpunkt darstellt. Größere Werte des Suchradiusparameters führen zu einer größeren Suchaktion auf dem Fußballfeld. Kleinere Werte erzeugen eine präzisere Beobachtung auf einem Teil der Fläche des Fußballfelds.

Die Benutzeroberfläche wurde mit der Entwicklungsumgebung NetBeans entwickelt. Netbeans lässt sich problemlos in ein Eclipse-Projekt übertragen. Durch den Drag-and-drop-Editor ermöglicht es Netbeans, Komponenten in Form von Containern und Text hinzuzufügen, um eine grafische GUI zu gestalten.

Die Benutzeroberfläche wurde zuerst auf einem jFrame als allgemeinem Fenster der Anwendung erzeugt. Dem jFrame werden dann zwei jPanels hinzufügt, auf das erste werden die Interaktions-Komponenten wie jButton, jRradiosButton, jCombobox, jMenus gestellt. Auf dem zweiten jPanel wird das Fußballfeld in Form eines Bildes dargestellt.

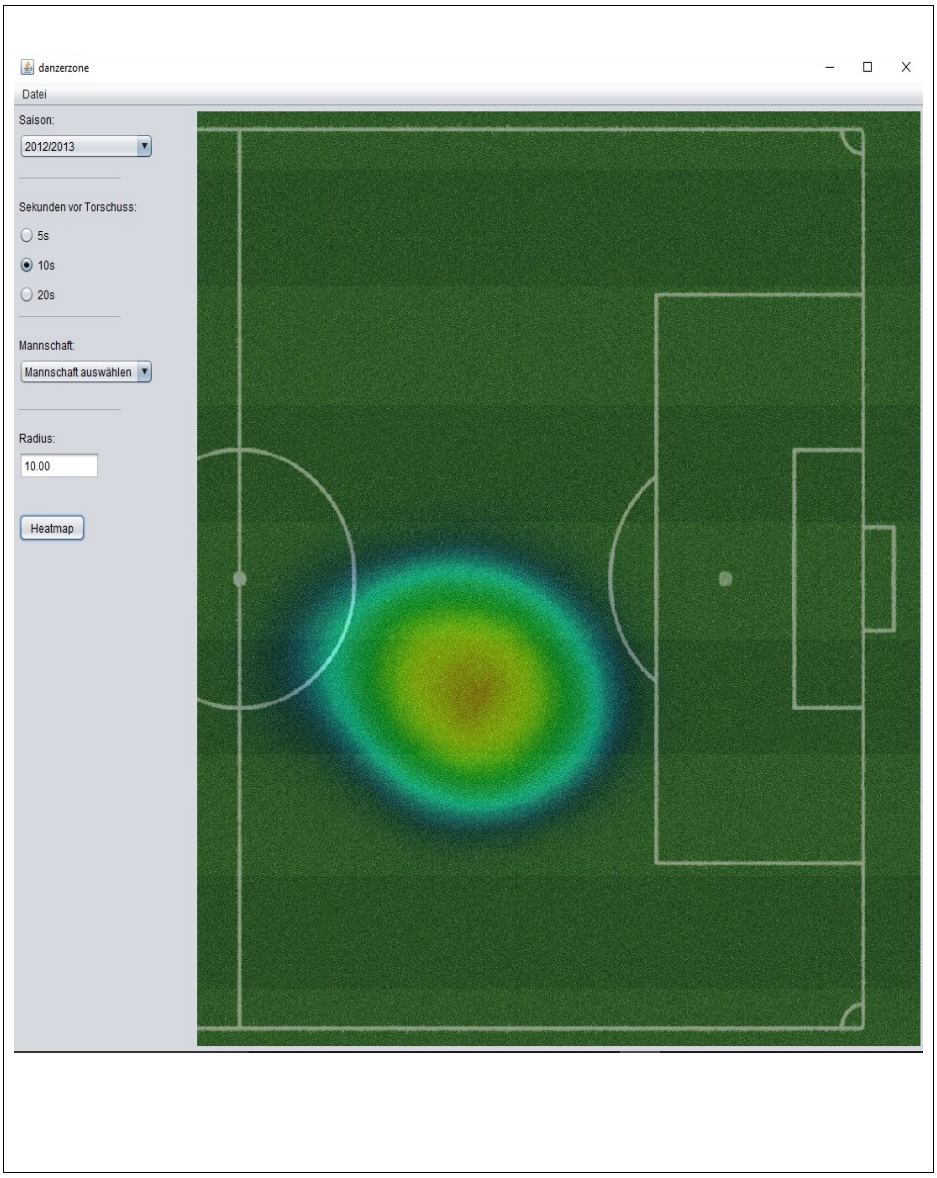

Abbildung 41:Oberfläche

4.8 Vergleich zwischen Wecka und Danzerzone-Programm

Bei dem Versuch, das Software-Tool Weka mit dem Algorithmus DBSCAN anzuwenden, wurde nur ein Clusterergebnis erzielt. Das Ergebnis hing stark davon ab, mit welchem Wert die Parameter Mindest-Punkte (MinPts) und Umgebung (ε) eingesetzt wurden. DBSCAN erreichte sehr gute Ergebnisse, wenn die Cluster sehr klar getrennt waren. Bei Clustern mit verschiedenen Dichten, wo die Punkte beieinanderliegen, traten Probleme auf. DBSCAN arbeitet mit dem Abstand zwischen einem Punkt und den *k*-nächsten Nachbarpunkten. Die *k*-nächsten Nachbarn werden über die Umgebung (ε) berechnet. Die Punkte werden durch eine Kette von über Dichten erreichbare Punkte verbunden. Diese Punkte werden als abhängig von der Dichte betrachtet und können verschiedene Ketten bilden. Diese Ketten sind ein Bestandteil von Clustern und können miteinander verbunden sein. Wenn die Umgebung zu groß gewählt wird, können in diesem Fall Schwierigkeiten auftreten. Rauschpunkte werden dann als Cluster gehalten und die Auswahl der Umgebung (ε) wird besonders schwierig. DBSCAN wird in verschiedene Bereichen eingesetzt. In der Medizin werden die Zusammenhänge von Krankheiten oder Diagnose von genetischen Erkrankungen analysiert. In dem Bereich Produktion wird mit DBSCAN ein Überblick über die gefertigten Produkte und wiederverwendbare Teile verschafft. In diesen Fällen ist im Voraus bekannt, wie sich die Ähnlichkeit oder Distanz zwischen den Objekten verhält. Die Verteilung oder Dichte von dem Objekt kann vermutet werden. Unter dieser Voraussetzung können die Parameter Umgebung (ε) und Mindestpunkte (MinPts) exakt eingestellt werden.

Mit Weka haben die Positionen vor den Torschüssen aus der Bundesliga 2011/2012 gezeigt, dass die Punkte auf dem Fußballfeld nahe beieinanderliegen. Abhängig davon, ob die Daten 5, 10 oder 20 Sekunden eingespielt werden, sind die Punkte mit unterschiedlichen Dichten auf dem Fußballfeld verteilt. Die Software Dangerzone wurde für mehrere Saisons der Bundesliga konzipiert. Ein Muster der Verteilung der Dichte von verschiedenen Saisons ist nicht im Voraus zu schätzen. Die Ergebnisse mit DBSCAN haben zu allen drei Zeitpunkten nur einen einzigen Cluster gezeichnet. Dieser Cluster hat die Punkte in eine einzelne Gruppe eingeteilt, ohne genau festzustellen, in welchem Bereich des Clusters sich eine größere Konzentration von Punkten befand. Mit dem Cluster wird also nur eine ungenaue Vorstellung von der Fläche gebildet, wo sich eine Häufung von Punkten befindet. Trotz mehrerer Versuche mit der Parametereinstellung und mit DBSCAN war nicht möglich, mehrere Cluster zu bilden. Wie die Punkte auf dem Fußballfeld verteilt sind, beeinflusst den Algorithmus DBSCAN bei der Formation von Clustern nicht.

Die Kerndichteschätzung (KDE) hat sich bei dem Programm Dangerzone besonders effizient gezeigt. Ein Vorteil der Kerndichteschätzung (KDE) verglichen mit DBSCAN sind die Fixpunkts, die auf dem Fußballfeld verteilt sind. Die Fixpunkte geben die Möglichkeit genau zu wissen, auf welcher Fläche des Fußballfelds sich die Hotspots von Punkten befinden. Ein anderer Vorteil ist,

dass für jeden dieser Fixpunkte Werte berechnet werden können. Diese Werte werden nach der Dichte sortiert. Die ersten 5 Fixpunkte nach der Sortierung repräsentieren die Fläche mit der größeren Dichte. Die Berechnung der Kerndichteschätzung (KDE) bezieht sich auf die Distanz von den Punkten zu den Fixpunkten und die Anzahl von Punkten, die sich in dem Bereich des Radius des Fixpunkts aufhalten.

Für die Anforderungen des Software Dangerzone ist dieser Algorithmus (KDE) sehr geeignet. In Vordergrund der Aufgabestellung steht die Identifikation der Fläche auf dem Fußballfeld mit der größten Dichte von Torschüssen und erzielten Toren. Mit der Implementation von Fixpunkten wurde die Möglichkeit gegeben, genau diese Fläche zu visualisieren. Auch die genauen Koordinaten dieser Flächen konnten gezeigt werden. DBSCAN hingegen geht über alle Punkte durch. Es gibt in dem Algorithmus DBSCAN keinen Bezug zur Fläche, wo die Cluster gebildet werden. Diese Eigenschaft hat sich bei der Festlegung der Fläche mit der größten Dichte als Nachteil erwiesen.

Bei dem gleichen Problem beschäftigen sich auch geographische Informationssysteme wie zum Beispiel Arc**GIS**, das Software-Tool zur Bearbeitung, Anzeige und Analyse von räumlichen Informationen. Unter den verschiedenen Funktionen wird auch die Toolset-Dichte angeboten. Diese Funktion wird für die Ermittlung der Dichte, z. B. der Kriminalität in einer Stadt, von Verkehrsunfällen oder Waldbränden in einer Region verwendet. In all diesen Fällen wird ebenfalls die Kerndichteschätzung verwendet, um die Fläche zu identifizieren, wo die Hotspots auftauchen.

5. Fazit

In diesem Kapitel werden die aus dieser Bachelorarbeit gewonnenen Erkenntnisse zusammengefasst und bewertet.

Der erste Teil der Arbeit bestand darin, die Konzeption der Software zu entwickeln und die Analyse und Auswertung die freigestellten Dateien der Bundesliga Saison 2011/2012 vorzunehmen. Die Analyse der Dateien war sehr hilfreich, um die Datenbank PostgreSQL aufzubauen. Mit Hilfe dieser Analyse wurde die Struktur der Tabellen und die Beziehungen zwischen den Tabellen erkannt. Bei dem Algorithmus DBSCAN wurden die Versuche mit dem Software-Tool WEKA vorgekommen. Die Versuche haben sich als wichtig erwiesen und gezeigt, für welche Art von Aufgaben DBSCAN geeignet ist. Die Koordinaten sind auf dem Fußballfeld nahe beieinander verteilt. DBSCAN hat eine Schwierigkeit Cluster zu bilden, wenn die Punkte in dieser Form aufgeteilt sind. Der Kerndichte-schätzer ist die geeignete Option für die Lösung von dieser Art von Problemen. Bei kommerzieller Software, die ebenfalls vor genau diesem Problem stehen, wird auch der Kerndichteschätzer ver-wendet.

In dem zweiten Teil wurden die Konzepte der Software implementiert. Als ersten wurden die Daten-bank PostgreSQL installiert. Darauf folgte die Erstellung der Tabellen. Um die Daten in der Datenbank zu speichern, wurden die Daten gelesen und Klassen geladen. Die Klassen übernehmen die Struktur der Tabelle aus der Datenbank, damit könnten sie auch bei einer Erweiterung des Programms Dangerzone benutzt werden. Die Implementierung von den Fixpunkten auf dem Fußballfeld erwies sich als sinnvoll. Die Sortierung von den ersten fünf Fixpunkten verhindert eine Überlagerung von Punkten auf dem Fußballfeld und ermöglicht eine bessere Visualisierung der relevanten Fläche, wo sich die Hotspots befinden. Für die Kontraständerung in der Pixelfarbe oder Glättung wurde die Methode der Radialgradiente verwendet, anstatt der Berechnung mit dem Gauß-Filter. Die Methode mit dem Radialgradienten hat sehr gute Ergebnisse gezeigt. Die Farben definieren genau, in welchem Bereich des Fußballfelds sich die Hotspots aufhalten. Der Farbenverlauf gibt eine gute Vorstellung, wie die Punkte verteilt sind. Der Berechnung des Gauß-Filters zeigt sich dagegen sehr langsam bei der Ausführung des Programms, denn für jeden Punkt oder Koordinate wurde ein Filter berechnet, was die Laufzeit beeinträchtigt. Die Größe des Suchradius beeinflusst das Ergebnis der Suche nach Punkten. Je größer der Suchradios, desto wahrscheinlicher ist, dass die Fixpunkte in seiner Nachbarschaft mehrere Punkte findet. Damit können mehrere Details angezeigt werden. Die Benutzeroberfläche wurden mit JavaBean programmiert, was die Entwicklung der GUI vereinfachte. Die Laufzeit, die zuerst mit DBSCAN ein Problem war, das zu überdenken war, konnten mit dem Kerndichteschätzer gelöst werden. Alle Anforderungen an die Software Dangerzone wurden erfüllt.

Zusammenfassend kann gesagt werden, dass der Kerndichtschätzer die beste Lösung für diese

Aufgabe ist. Eine Analyse des Verhaltens der Daten ist entscheidend für die Planungssoftware und die Ergebnisse. DBSCAN ist ein weit verbreiteter Algorithmus zum Beispiel in der Biologie, der medizinischen Forschung und Produktionsfertigung. In dem Vordergrund stehen die Objekte und die Ähnlichkeiten zwischen diesen Objekten. Ein direkter Bezug zur Fläche, wo sich die Cluster bilden, wird nicht betrachtet. Obwohl bei den Versuchen mit WEKA nur ein einziger Cluster entstand, verbreiterte sich dieser auf über dreißig Meter auf dem Fußballfeld. Mit diesen Ergebnissen konnte die genaue Position von den Punkt-Dichten oder Hotspots nicht festgestellt werden. Ein anderes Problem war die Sensibilität des Suchradius. Es mussten mehrere Versuche durchgeführt werden, bis ein zufriedenstellendes Ergebnis erzielt wurde. Es besteht also die Möglichkeit, dass der Benutzer nicht in einer angemessenen Zeit ein gutes Ergebnis erzielt, ohne dass mehrere Versuche mit dem Suchradius benötigt werden, um einen Cluster zu bilden.

Letztlich bietet das Programm auch die Option der Weiterentwicklung. Die Ausgabe der Koordinaten auf dem Fußballfeld der Fixpunkte, die Darstellung der Fixpunkte, die Zeichnung des Suchradios und das Ausdrucken der Heatmaps sind Beispiele zu diesen Weiterentwicklungsmöglichkeiten.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AWT- Abstract Window Toolki

DBSCAN *Density-Based Spatial Clustering of Applications with Noise*

DOM-Document Object Model

JDK *Java Development Kit*

JRE *Java Runtime Environment, kurz*

SQL *Structured Query Language*

KDD *Knowledge discovery in databases*

KDE *kernel density estimation,*

Weka *Waikato Environment for Knowledge Analysis*

XML *Extensible Markup Languag*

Literaturverzeichnis

Aaron E. Walsh/Doug Gehringer (2002)

Aaron E. Walsh/Doug Gehringer: Java 3D- APIJump-Start-1 Auflage – Verlage Prentice Hall PTR-Upper Saddle River, New Jersey-2002 .

Alexander Engelhard (2013)

Alexander Engelhard; Histogrammm, http://www.crashkurs-statistik.de/histogramme/ Dowload am (02.07.2016).um15:10 Uhr.

AlfonsKemper/ Andre Eickler (2013)

AlfonsKemper/ Andre Eickler: Datenbanksysteme Eine Einführung -9Auflage Oldenbourg Verlag München 2013.

Anderson medeiros(2014)

Anderson medeiros; Introdução aos Mapas de Kernel, http://andersonmedeiros.com/mapas-de-kernel-parte-1/, Dowload am (08.06.2016).um 20:30 Uhr.

Augst,Siege()

Augst,Siege, Institut für Wissensbasierte Systeme und Bildverstehen, TU München; Positionsdynamische Modellierung zur Situations- und Spieleridentifikation im Fußball; http://www.sport.uni-augsburg.de/.../Positionsdynamische_Modellierung_zur_Situations-_und_Spieleriden.pdf, Dowload am (08.06.2016) um 16:40 Uhr.

Burger/Burger ()2006

Wilhelm Burger/Mark James Burger: Digitale Bildverarbeitung – Eine Einführung mit Java und ImageJ.-2 Auflage Verlag Springer Berlin Heildelberg-2005-2006.

Cristian Ullenboom (2014)

Cristian Ullenboom: Java SE Standard- Bibliothek- 2 Aauflage Velage Galileo Comuting 2014.
Dylan Vester (15.10.2015)

Dylan Vester, dylanvester, Creating Heat Maps with .NET, http://dylanvester.com/2015/10/creating-heat-maps-with-net-20-c-sharp/, Dowload am (07.02.2016) um 15:30Uhr.

Eduardo Wildt- Graziani (2015)

Eduardo Wildt-Graziani: Statistiktiktool für die E-Learning-Plattform edb der FH Köln,-1 Auflage-Verlag-AkamikerVerlag Saarbrücken 2015.

Fahrmeir/künstler/Pigeot/Tutz (2010)

Fahrmeir/künstler/Pigeot/Tutz: Statistik:DerWeg zur Datenanalyse-7 Auflage-Verlage Springer 2010.

Frank Klawonn (2010)

Frank Klawonn: Grundkurs Computergrafik mit Java 3. Auflage Verlage Vieweg+Teubner Wiesbaden -2010.

Hans-Jochen Bartsch (2015)

Hans-Jochen Bartsch-Taschenbuch: Mathematischer Formeln,-19Auflage- Verlag Fachbuchverlag Leizig im Carl Hanser -2001.

Helmut Balzer (2001)

Helmut Balzer: Lehrbuch der Software-Technik 2 Auflage Verlage Spektrum 2001.

Heide Faeskorn/Woyke (2007)

Heide Faeskorn-Woyke: Datenbanksystem Theori und Praxis mit SQL2003, Oracle und MySQL 1.Auflage, Verlage: Pearson Studium, München 2007

Herbert Büning/Götz Trenkler (1994)

Herbert Büning/Götz Trenkler: Nichtparametrische statische Methoden 2-Auflage Verlage walter de Gruyter NewYork 1994

Iam Witten (o.J)

o.V, the University of Waikato,Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/, Dowload am (12.12.2015) um18:00 Uhr.

Jawei/Micheline Kamber / Jian Pei (2012)

Data Mining Concepts and Techniques -3 Auflage Verlage Elsevier 2012.

João Medeiros (23.01.2014)

João Medeiros :Wired, The winning formula: data analytics has become the latest tool keeping

football teams one step ahead, http://www.wired.co.uk/article/the-winning-formula, Dowload am (16.03.2016) um 20:30 Uhr

Katrin Ahr (06.08.2012)

Katrin Ahr.: HD-Spielertracking in der Bundesliga, http://www.elektroniknet.de/automation/sonstiges/artikel/90403/0/ Dowload am (07.02.2016) um 15:30 Uhr.

Leen Ammerraal (1998)

Leen Ammerraal: Comuter Graphics for Java Programmers 1. Auflage -Verlag John Wiley & Sons -New York- 1998.

Lutz Emmel (2001)

Lutz Emmel: Clixx 3D Der Grundkurs 1 Auflage Verlag Harri Deutsch 2001

Marcus Josiger/Kathrin Kirchner (2016)

Marcus Josiger/Kathrin Kirchner -Moderne .:Clusteralgorithmen- eine vergleichende Analyse auf zweidimensionalen Daten, http://www.kde.cs.uni-kassel.de/ws/LLWA03/fgml/final/Kirchner.pdf Dowload am (05.03.2016) um 17:30 Uhr.

Markus von Rimscha (2008)

Markus von Rimscha: Algorithmen Kompakt und verständlich 2Auflage Verlag Viewege+Teubner 2008.

o.V.:(2010)

o.V:.-Institut für empirische Designforschung -Der Fußball in Zeit und-Raum, https://designforschung.wordpress.com/2010/07/08/der-fussball-in-zeit-und-raum/, Dowload am (24.03.2016) um 14:30 Uhr.

o.V. (26.03.2013)

o,V, Geographic Information Systems, http://gis.stackexchange.com/questions/55424/heatmap-algorithm-to-visualise-point-diversity, Dowload am (20.04.2016) um 12.30 Uhr.

o.V. (14.10.2012)

o.V., Software-talk, Java Heatmap (Example Code), http://software-talk.org/blog/2012/01/java-heatmap-example/ Dowload am (9.09.2016) um 13:30 Uhr.

o.V(o.J)

o.V, Tu-chemnitz, 10 Grafiken mit Java2D; https://www.tu-chemnitz.de/wirtschaft/wi1/lehre/2004_ss/wi_pr3/java2/java2d.htm, Dowload am (05.01.2016) um 18:20 Uhr.

o.V(o.J)

o.V, ArcGis online, Kerndichte (Spatial Analyst)

, http://resources.arcgis.com/de/help/main/10.2/index.html#/na/009z0000000s000000/, Dowload am (05.01.2016) um 14:00 Uhr.

o.V(o.J)

o.V, FileFilter, http://java-tutorial.org/filefilter.html, Dowload am (08.06.2016) um 14:10 Uhr.

Peter Andreas Möller(2014)

Peter Andreas Möller, Workshop PostgreSQL 9, http://workshop-post gresql.de/, Dowload (08.06.2016).

Rainer Leonhart (2009)

Rainer Leonhart: Lehrbuch Statistik Einstieg und Vertienfung 2 Auflage Verlage Huber 2009.

Rinne (2008)

Rinne: Taschenbuch der Statistik-4 Auflage Verlag WissenschaflicherVerlag Harri Deutsche GmbH-Frankfurt am Mai-2008

Rodolfo Maduro Almeida (2016)

Rodolfo Maduro Almeida QGIS http://qgisnapratica.blogspot.de/2015/12/analise-de-densidade-de-eventos.html (18.12.2016) um 17.18

Stefan Brunn (10.06.2010)

Stefan Brunn.:Elf Dateien müsst Ihr sein, http://www.heise.de/tr/artikel/Elf-Dateien-muesst-Ihr-sein-1017763.html, Dowload am (20.02.2016) um 17:40 Uhr.

Tan /Steinbach /Kumar (2014)

Tan /Steinbach /Kumar: Introduction to Data Mining-1 Auflage Verlage person 2014.

WendyL. Martinez/ AngelR. Martinez (2008)

WendyL. Martinez/ AngelR: Martinez-Computational Statistics Handbook with Matlab-2-Auflage-

Verlag Chapman & Hall/CRC- London/New York-2008.

Wolf Dieter Rase(17.03.2016)

Wolf Dieter Rase.:Kartographische Oberflächen: Interpolation, Analyse, Visualisierung,1 Auflafe, Verlage BoD-Books on demand, Norderstedt (17.03.2016).

Zach Slaton (26.08.2012)

Zach Slaton , Forbes, The Analyst Behind Manchester City's Rapid, Risehttp://www.forbes.com/.../2012/08/16/the-analyst-behind-manchester-citys-player-investments-part-1/2/#2935703416d3, Dowload am (26.08.2016) um 14:10Uhr.

Erklärung über die selbständige Abfassung der Arbeit

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Köln, den 27.12.2016 _____

(Alexandre Wildt Graziani)