Fachbereich Mathematik und Informatik
Institut für Informatik

Diplomarbeit

Verteilte Ereigniserkennung in Sensornetzen

Norman Dziengel - dziengel@inf.fu-berlin.de

24. Oktober 2007

Betreuer: Prof. Dr.-Ing. Jochen Schiller
Georg Wittenburg, M. Sc.

DANKSAGUNG

An erster Stelle möchte ich meinen Eltern den größtmöglichen Dank aussprechen. Ihr Vertrauen und die liebevolle Unterstützung während der gesamten Studienzeit konnte diese Arbeit überhaupt möglich werden lassen.

Herrn Prof. Dr.-Ing. Jochen Schiller möchte ich für sein Vertrauen und für die Ermöglichung dieser Arbeit an der Freien Universität Berlin danken.

Für die außerordentlich konstruktive und engagierte Betreuung in jedem Stadium dieser Arbeit möchte ich vielmals Herrn M. Sc. Georg Wittenburg danken.

Frau Dipl.-Hydrol. Marion Kühn möchte ich einen besonders hervorhebenden Dank aussprechen, da ihre Beta-Tests, Korrekturlesungen, Motivierungen und ihre geduldige Fürsorge entscheidend zur Fertigstellung dieser Arbeit beigetragen haben.

Desweiteren möchte ich mich für die Beantwortung vieler Fragen bei
Herrn Marco Hutta, Herrn Dipl.-Inform. Marco Block, Herrn Dr.-Ing. Achim Liers,
Herrn Dipl.-Inform. Marc Scherfenberg, Herrn Dipl.-Ing. (FH) Hans Peter Heitzmann,
Herrn M. Sc. Freddy López Villafuerte, Frau Dr. Dorith Sohr, Herrn Alexander Jäger,
Herrn Karsten Fiedler, Frau Dipl.-Inform. Kirsten Terfloth
und Herrn Dipl.-Inform. Enrico Köppe

sehr herzlich bedanken.

KURZFASSUNG

Drahtlose Sensornetze haben aufgrund ihrer verteilten Struktur die Möglichkeit, differenziert Informationen an unterschiedlichen Stellen der Umwelt zu erfassen. Die Erkennung von Ereignissen, die in ihrer Umgebung mehrere Parameter und Messpunkte verschiedentlich beeinflussen, wird mittels drahtlosen Sensornetzen möglich. Das vorgestellte System nutzt die Vorteile drahtloser Sensornetze, indem mehrere Sensorknoten das Problem der verteilten Ereigniserkennung gemeinsam lösen. Mit dieser Arbeit wird ein Erkennungssystem vorgestellt, das verteilt zu betrachtende Ereignisdaten auf Basis einer lokal auf den Sensorknoten durchgeführten Mustererkennung im Netz fusioniert und klassifiziert.

Eine dynamische Kalibrierung des Beschleunigungssensors und ein Mustertraining mit den Sensorknoten ermöglichen eine nachfolgende differenzierte Erkennung der gelernten Bewegungsmuster. Die erfassten Beschleunigungsdaten werden lokal auf den jeweiligen Sensorknoten komprimiert und mit den Daten benachbarter Sensorknoten fusioniert, um im Bezug auf das globale Ereignis ausgewertet zu werden. Verschiedene Fusionstechniken auf Ebene der Klassifizierung sowie der Mustermerkmale werden getrennt und miteinander verknüpft untersucht und bewertet.

Ein qualitativer Vergleich zwischen der hier erarbeiteten lokalen und verteilten Ereigniserkennung sowie einem bestehenden verteilten Erkennungsansatz wird gezogen. Die lokale Ereigniserkennung erreicht eine Korrektklassifikationsrate von 89,4 %. Basierend auf der Komponente der lokalen Erkennung erzielt das verteilte Erkennungssystem in den untersuchten Fusionsmethoden eine Korrektklassifikationsrate von 93,8 % bis 96,3 %. Die Erkennungsratensteigerungen werden auf den Einsatz der verteilten Erkennungsstrategie zurückgeführt.

Die energietechnisch kostenintensivste Operation in drahtlosen Sensornetzen ist das Versenden von Datenpaketen. Aufgrund dieser Problematik wird die Rentabilität der Datenübertragung innerhalb des Sensornetzes in Bezug auf die erbrachte Korrektklassifikationsrate in den Fusionsmethoden analysiert. Außerdem werden während der lokalen und verteilten Mustererkennung die Einflussstärken der Probanden auf die ermittelten Kennwerte untersucht und evaluiert.

INHALTSVERZEICHNIS

1 EINLEITUNG

„*Eine Idee muss Wirklichkeit werden können, oder sie ist nur eine eitle Seifenblase*“ – Berthold Auerbach

Ubiquitous Computing ist die Idee von immer weiter miniaturisierten Computern, die aus unserer direkten Wahrnehmung stückweise verschwinden, um dem Menschen, ohne ihn zu stören, unterstützend zur Seite zu stehen [Wei91] .

Mit der stetig voranschreitenden Verkleinerung der Computer werden Kleinstrechner für drahtlose Sensornetze (Wireless Sensor Networks) entwickelt und eingesetzt. Die Verkleinerung und der drahtlose Einsatz solcher Sensornetze führen zu einem Energieversorgungsproblem. Die Sensorknoten werden üblicherweise mit Batterien ausgestattet und sind dadurch an eine regelmäßige Wartung gebunden. Wartungsfreiheit ist durch den Einsatz von Solarzellen erreichbar, deren Einsatz wiederum zu einer größeren, weniger geeigneten Struktur der Sensorknoten führt. Die Reduktion des Energieverbrauches bei gleichzeitiger Erhaltung der Leistungsfähigkeit stellt eine große Herausforderung dar. Das Funkmodul hat während der Sende- und Empfangsoperationen die höchsten Energieansprüche aller Hardwarekomponenten des Sensors und stellt zugleich die Schnittstelle eines jeden Sensorknotens zu seinen Kommunikationspartnern dar. Die Kommunikation innerhalb des Sensornetzes ist zudem die Stärke, durch die es sich von anderen, nicht verkabelten Systemen abhebt. Sensornetze werden derzeit in verschiedenen Forschungsbereichen eingesetzt. Konkrete Einsätze können drahtlose Sensornetze bereits in der Logistik verzeichnen, bei der beispielsweise das Projekt der lückenlosen Güterverfolgung von [Sca071] eine Überwachung und Verfolgung von Containern ermöglicht. Projekte wie SLEWS (Sensor based landslide early warning system), beschrieben in [Sch07] , entwickeln ein Frühwarnsystem zur Erkennung verschiedenster Naturereignisse wie Hangabbrüche. Das Tsunamifrühwarnsystem GITEWS (German Indonesien Tsunami Early Warning System, [Rud06]), welches seit 2005 von dem GeoForschungsZentrum Potsdam aufgebaut wird, ermöglicht eine Reaktion innerhalb von fünf Minuten auf Seebeben, denen gefährliche Meereswogen folgen können. Das System basiert auf verschiedensten Sensordaten (Wettersensoren, Seismometer, Drucksensoren, Küstenpegelmesser), die in dem Warnzentrum Jakartas gesammelt und ausgewertet werden.

Drahtlose Sensornetze lassen Visionen entstehen, die Sensornetze als selbstständig handelnde und Entscheidungen treffende Systeme beschreiben. Systeme, die Schwelbrände in Wäldern rechtzeitig erkennen und schnelle Hilfe versprechen, sind noch zu teuer, um flächendeckend eingesetzt werden zu können, deren Realisierung ist aber denkbar. Sensornetze könnten nach der Vision von Endler [Endl05] Infektionskrankheiten und ihre Ausbreitung in Rinderherden mittels Sensoren (Temperatur etc.) erkennen und so einer Epidemie in der Rinderzucht oder anderen Tierarten vorbeugen. Nach Schoch et *al.* ist die in [Sch051] beschriebene Vision von intelligentem Abfall, der mit RFID-Chips ausgestattet ist, eine Möglichkeit die Verpackungsentsorgung besser zu motivieren. Auf einer Abfallsammelkarte soll bei korrekter Entsorgung direkt vom Sammelbehältnis ein Pfand gutgeschrieben werden. Visionen von intelligenten Schwarmsensoren, die zur Aufklärung in kontaminierten Landstrichen oder gefährlichen Kriegsgebieten

eingesetzt werden sollen, stecken bisher noch in den Anfängen der Forschung, siehe [Sen07], [Bok06] .

Der Einsatz von drahtlosen Sensornetzten soll die Vision des Ubiquitous Computing schrittweise Wirklichkeit werden lassen [Mat03] , [Mat07] . Drahtlose Sensornetze müssen auf Ereignisse der Umgebung direkt reagieren können, um sich von einfachen Datensammlern zu unterscheiden. Das bei einer Ereigniserkennung auftretende Datenvolumen ist derart umfangreich, dass eine vollständige Weiterleitung dieser Daten zu einer Basisstation im Dauereinsatz oft nicht praktikabel ist. Sensornetze, wie das erwähnte GITEWS, lösen dieses Problem mit umfangreicher Hardware, die sich mit Solarzellen selbst mit Energie versorgen. Sensorknoten streben jedoch an, kostengünstig, klein und unauffällig zu sein, um die Umwelt nicht durch ihre Existenz zu beeinflussen. Ziel muss es für die Zukunft sein, das Datenvolumen zu komprimieren und die Auswertungsqualität zu erhöhen. Ein drahtloses Sensornetz, bestehend aus kleinsten Hardwarekomponenten, das in der Lage ist, ohne eine externe Basisstation netzinterne Ereigniserkennung und Entscheidungsfindung zu ermöglichen, ist das Ziel dieser Arbeit. Sowohl die netzinterne Datenbearbeitung als auch die netzinterne Ereigniserkennung gelten als Kernaufgaben von drahtlosen Sensornetzen, die trotz ihrer augenscheinlichen Prägnanz kaum erforscht sind [Wit07] .

Eine Hauptaufgabe zur Lösung dieses Problems kann für viele Sensornetze zunächst die Erkennung von Ereignissen und Mustern in der Umgebung der Sensornetze sein. Eine netzinterne Auswertung und Klassifizierung von Ereignissen soll diese Ansprüche in der hier vorgestellten Arbeit erfüllen und stellt einen Schritt in Richtung eines intelligenten Sensornetzes dar.

1.1 Drahtlose Sensornetze

Drahtlose Sensornetze sind eine Konsequenz aus dem immerwährenden Fortschritt der Informationstechnologie. Insbesondere die Sensorknoten stellen als Komponente der Sensornetze den optimierten Kleinstrechner bei maximierter Lebensdauer und einem sehr guten Kosten-Nutzen-Verhältnis dar. Die Eigenschaften eines guten Sensornetzes stellen sich insbesondere durch ein wartungsfreies, selbstheilendes, über Funk kommunizierendes Netzwerk dar. Ein gutes Beispiel für ein solches Sensornetz stellt das aus den Forschungsarbeiten der Freien Universität Berlin hervorgegangene Sensornetz ScatterWeb [Sca071] dar. Verschiedenste Forschungszweige nutzen mittlerweile Sensornetze zur Erkenntnisgewinnung. Die einzelnen Komponenten des Sensornetzes sind die Sensorknoten. Sie haben zumeist die Aufgabe, mittels verschiedenster Sensoren Daten zu erfassen, zu sammeln, auszuwerten und gegebenenfalls per Funk weiterzuleiten.

Martinez hat mit [Mat05] ein verteiltes Sensornetz für die Gletscherdynamikforschung in Briksdalbreen (Norwegen) erstellt. Die dort eingesetzten Sensoren haben eine Größe von 14,8 x 6,8 cm und sind in ca. 70 m tiefe Bohrlöcher in den Gletscher eingelassen. Die Sensoren erfas-

sen Temperatur, Wasser-/Eisdruck, 3D-Ausrichtung und den elektrischen Leitungswiderstand zur Bestimmung der umgebenden Sedimente.

Juang et *al.* hat in der Arbeit [Jua02] für das Projekt ZebraNet ein drahtloses Netzwerk (30 Sensorknoten) zur Erforschung des Wildverhaltens in Kenia eingerichtet. Es werden verschiedene Protokolle unter dem Fokus der Energieeffizienz untersucht. Ziel aus Sicht der Biologen sind Erkenntnisse über Muster im Zugverhalten der Tiere unter Einfluss anderer Tierarten, des Wetters und der Menschen.

Der eigentliche Mehrwert eines drahtlosen Sensornetzes gegenüber der Erfassung mittels nichtvernetzter Messstationen ergibt sich erst durch die Möglichkeit, Daten aus vielen Sensoren aktiv schon während der Erfassung im Zusammenhang zu betrachten. Der Aspekt der Drahtlosigkeit gewährt einem Sensornetz seine Flexibilität und damit die Fähigkeit, Dynamiken der Umwelt besser zu erfassen. Mittels fusionierter Daten (Rohdaten, Merkmale, Klassen) können präzisere Erkenntnisse und andere Rückschlüsse über den vom Sensornetz abgedeckten Forschungsbereich gezogen werden, als es mit den jeweiligen Einzelsensoren möglich ist. Datenfusionen auf dedizierten Erfassungsstationen mit oftmals erweiterter Hardware ([Wer05] , [Bok06]) werden von einer neuen Generation von drahtlosen Sensornetzen abgelöst. Sensordaten eines Sensorclusters noch vor der Versendung zur zentralen Basisstation lokal auf Sensorknoten und damit innerhalb der Netztopologie zu fusionieren und auszuwerten, erlauben neue Qualitäten von Sensornetzen. Sensornetze sollen mit Datenfusionstechniken innerhalb des Sensornetzes deutlich effizienter agieren können. [Lli97] , [Rus07]

Sensornetzinterne Datenauswertungen sind in verschiedenen Einsatzgebieten vorstellbar. In der Medizin können Gesundheitszustände von Patienten mit Sensoren an verschiedenen Stellen des Körpers abgefasst werden. Die Funkverbindung der Sensorknoten verhindert, dass der Patient in seiner Bewegungsfreiheit eingeschränkt wird. Durch eine lokale Datenauswertung würde nur noch eine minimale Kommunikation zu einem externen System nötig sein. Zudem wären innerhalb des Sensornetzes bereits Notmaßnahmen wie eine selbstregelnde Medikamentenzufuhr denkbar. Brückenschwingungen ließen sich möglicherweise mit lokal auswertenden Sensorknoten soweit erfassen, dass eine Sicherheitsschrankenregelung von dem Brückensensornetz selber initiiert werden kann. Unerwünschte Zaunüberwindungen ließen sich möglicherweise über große Zaunanlagen hinweg überwachen. Die im Netz ausgewerteten Daten könnten selektiv Kameras aktivieren oder eine automatische Zielerfassung mittels Scheinwerfer im gewünschten Bereich ermöglichen. Frühwarnsysteme, die eine zeitnahe Evakuierung erlauben und mögliche Rettungswege bei Bränden ermitteln sowie gebäudeinterne Anzeigen steuern können, wären ebenfalls denkbar.

Die unterschiedlich möglichen Einsatzgebiete fordern von den Sensornetzen verschiedenste Kommunikationstechniken, Datenspeichermodelle und Netzwerkdynamiken. Eine taxonomische Einteilung drahtloser Sensornetze wird als Folge verschiedenster Ansprüche der Sensornetze von Tilak et *al.* in [Til02] erstellt.

Weitere Beispiele zu konkret umgesetzten Anwendungsbereichen werden im Kapitel 2 mit Bezug auf die hier vorliegende Arbeit besprochen.

1.2 Restriktionen in Sensornetzen

Drahtlose Sensornetze bilden auf der einen Seite ein sehr fortschrittliches Konzept, auf der anderen Seite muss mit Restriktionen gearbeitet werden, die die Umsetzung von typischen Konzepten der Informatik erschweren oder verhindern. Sensorknoten sind mit einem Prozessor, einem Funkmodul, im Verhältnis zu handelsüblichen PCs wenig Speicher und einer Energieversorgung (z. B. einer Batterie) ausgestattet.

Die einzige Möglichkeit, mit einem Sensorknoten zu kommunizieren, ist der Funkkontakt. Dies gilt für Wartungsarbeiten genauso wie für Sensorknoten, die untereinander in Kontakt treten müssen. Funksignale sind jedoch hinsichtlich des Energieverbrauches teuer, zudem ist die Reichweite der Funksignale beschränkt. Aufgrund beschränkter Energieressourcen und aufgrund der wegen der Größe des Sensorknotens beschränkten Fähigkeiten einzelner Bauteile kommt es zu komplexen Problemstellungen.

Sensorknoten lassen sich mit beliebigen Sensoren ausstatten. Die Sensorbauteile müssen jedoch exakt auf die Problembereiche der Sensornetze zugeschnitten sein. Die Größe der Sensorknoten, der Stromverbrauch, sowie deren Genauigkeit sollen es ermöglichen, ein Sensornetz über lange Zeit unbeaufsichtigt, wartungsfrei und mit minimalen Sensorknotenausfällen in Betrieb zu halten. Die Baugröße muss gesenkt werden, der Energieverbrauch soll so gering wie möglich gehalten werden, die Leistungsfähigkeit der Komponente soll so hoch wie möglich sein. „Besonders leistungsfähig“ bedeutet folglich in Sensornetzten, vor allem den Energieverbrauch bei wenigstens konstanter Leistung und Baugröße zu senken. Applikationen, die auf Sensorknoten entwickelt werden, stehen folglich mit ihrer Qualität und ihrer Fähigkeit, diese auch dauerhaft bereitstellen zu können, in ständiger Auseinandersetzung.

1.3 Ereignisse in Sensornetzen

> *„Die Zuweisung von physischen Objekten oder Ereignissen zu einer von mehreren vordefinierten Kategorien bzw. Klassen wird Mustererkennung genannt.“* (übersetzt nach [Dud01])

Duda definiert folglich die Zuordnung von Ereignissen und Objekten zu einer vordefinierten Kategorie oder Klasse als Mustererkennung. Eine Grundvoraussetzung, um eine verteilte Ereigniserkennung in Sensornetzen durchführen zu können, ist daher die Fähigkeit der einzelnen Sensorknoten, Muster oder Teilmuster des Ereignisses zu kennen und wiederzuerkennen. Abstrakt betrachtet sind verteilt zu erkennende Ereignisse definierte, zusammengesetzte Musterkombinationen. Deren Erkennung soll mittels verschiedener Sensoren und einer Vielzahl von Sensorknoten optimiert werden. Mustererkennung mit Einzelsensoren ist die Grundlage für eine solche verteilte Mustererkennung.

Ist ein Sensorknoten in der Lage, erfasste Daten auszuwerten, resultiert daraus der Vorteil, dass mit den Auswertungsinformationen weitergearbeitet werden kann. Die Rohdaten selber müssen nach der Auswertung nicht zwingend gespeichert und versendet werden. Die so entstehende Entlastung des Gesamtsystems macht Mustererkennung in Sensornetzen technisch interessant. Die so komprimierte Datenmenge minimiert den damit verbundenen Sendeenergieaufwand, wie in Kapitel 1.2 gefordert.

Ereignisse können in drahtlosen Sensornetzen jeglicher Art aus messbaren Sensordaten abgeleitet sein. Die Ereigniserkennung reicht dabei von einfachen Schwellenwertmessungen [Wer06] bis hin zu aufwendigen Erkennungssystemen [Bok06] . Komplexere Ereignisse können schwer mit nur einem Sensor sicher erfasst und differenziert werden. Solche Ereignisse sind z. B. Gesundheitszustände, Brückenschwingungen, Zaunüberwindung, Truppenaktivitäten oder Vulkanaktivitäten.

Zudem liegt die Überlegung nahe, dass eine verteilte Ereigniserkennung möglicherweise die Erkennungsgenauigkeit verbessert. Ereignisse definieren sich über charakteristische messbare Datenmuster, die in Teilen oder im Ganzen vordefinierten Kategorien zugeordnet sind. Eine Form dieser Ereignisse sind die in dieser Arbeit betrachteten Beschleunigungsmuster, die Sensoren in einem Sensornetz, z. B. durch Bewegung, erfahren.

1.4 Problemstellung und Lösungsansatz

Schon einfache Probleme stellen sich in Sensornetzen schnell als Herausforderung heraus und lassen im Voraus oft keine Aussagen über die Realisierbarkeit zu. Somit ist eine praktikable Lösung einer komplexen Aufgabenstellung wie der Ereigniserkennung stets durch die inhärenten Restriktionen des Sensornetzes gefährdet. Bisherige Forschungsansätze betrachteten die Datenaggregation in drahtlosen Sensornetzen bzw. Anfragen nach diesen Daten. Erst seit kurzem wird die Erkennung von Ereignissen in drahtlosen Sensornetzen betrachtet.

In dieser Arbeit wird das Erkennen von verteilten Ereignissen realisiert. In einem ersten Schritt wird dafür eine lokale, komplexe Datenauswertung und Interpretation entwickelt. In einem weiteren Schritt wird aufbauend auf der lokalen Erkennung das parallele Erkennen von demselben Ereignis mit mehreren Sensorknoten umgesetzt. Die verschiedenen Sensorknoten sollen Daten austauschen können, um das Ziel der gemeinsamen Erkennung zu erreichen. Die in Kapitel 1.2 besprochenen Restriktionen in Sensornetzen machen es erforderlich, bekannte Algorithmen zu diesem Zweck anpassen zu müssen. Im weiteren Verlauf dieser Arbeit werden zu der hier aufgezeigten Problemstellung verwandte Arbeiten in Kapitel 2 vorgestellt und mit dieser Arbeit verglichen. Kapitel 3 beschäftigt sich mit den Grundlagen dieser Arbeit. Dazu zählen die ScatterWeb Plattform, der auf dem Sensorknoten MSB 430 befindliche Beschleunigungssensor als Sensorikgrundlage und das Forschungsgebiet der Mustererkennung. Inspiriert durch das Forschungsgebiet der Mustererkennung werden Ereignisse, wie in Kapitel 1.3 definiert, als

Muster zu interpretieren sein. Die eingesetzten Systemkomponenten basieren daher auf typischen Mustererkennungsarchitekturen.

Nach Ripley definiert sich Mustererkennung als:

> *"Given some examples of complex signals and the correct decisions for them, make decisions automatically for a stream of future examples."* [Rip96]

Die Grundlage für eine verteilte Erkennung bildet das in Kapitel 4 beschriebene lokale Trainingssystem, welches dem dort vorgestellten Mustererkennungssystem die Basisdaten zur Erkennung von Mustern liefert. Zudem wird ein Kalibrierungssystem vorgestellt, welches es ermöglicht, Sensorknoten unter verschiedenen Umweltbedingungen einzusetzen. Es wird die Entwicklung eines dynamischen Trainingssystems beschrieben, welches ohne zusätzliche Hardware in der Lage ist, ein problemabhängiges Training durchzuführen. Zudem wird eine größtmögliche Flexibilität erhalten bleiben, die insbesondere die zu erkennende Klassenzahl sowie deren Merkmale betrifft.

Aufbauend auf dem Lösungsansatz für einzelne Sensorknoten wird in Kapitel 5 die verteilte Erkennung vorgestellt und untersucht. Es werden Methoden auf verschiedenen Ebenen des Systems bezüglich ihrer Eignung für eine verteilte Mustererkennung analysiert. In Kapitel 6 werden die gesammelten Erkenntnisse der lokalen und verteilten Erkennung evaluiert. Kapitel 7 fasst die Arbeit zusammen und gibt einen Ausblick für nachfolgende Arbeiten sowie eine abschließende Bewertung.

2 VERWANDTE ARBEITEN

Die verschiedenen Forschungsbereiche, in denen sich Sensornetze mittlerweile etabliert haben, beginnen sich über Militär, Gesundheitslehre, Naturforschung, Produktkontrolle und den Katastrophenschutz hinaus auszuweiten. Arbeiten, in denen bereits Mustererkennung direkt auf den Sensorknoten zur Ereigniserkennung führt, beginnen sich gegenwärtig in der aktuellen Forschung zu etablieren. Besonders verteilte Erkennungssysteme stellen eine große Herausforderung dar. Die bekannten Restriktionen in Sensornetzen (siehe Kapitel 1.2) stellen die verteilte, eingebettete Erkennung von Ereignissen bzw. Mustern vor große Probleme, wie die nachfolgend beschriebenen Projekte aufzeigen.

Bei der Betrachtung verwandter Arbeiten wird danach unterschieden, ob die Projekte die Mustererkennung im drahtlosen Sensornetz durchführen und ihre Entscheidungen entweder auf einer zentralen Station berechnen oder diese Entscheidung auf den Sensorknoten selbst, also dezentral, berechnet wird. Es wird eine Arbeit vorgestellt, die keine drahtlose Verbindung verwendet aber den Ansatz der verteilten Mustererkennung in eingebetteten Systemen untersucht und damit in dem Kapitel 2.1.2 als „Verkabelte Systeme“ zu finden ist.

2.1 Zentralisierte verteilte Entscheidungsfindung

2.1.1 Drahtlose Systeme

2.1.1.1 Verteiltes EKG

Ein Elektrokardiogramm (EKG) ist ein nichtinvasiver Analysevorgang, bei dem die dem mechanischen Herzschlag vorausgehende elektrische Aktivität untersucht wird. In [Jaf06] und [Jaf061] wird ein verteiltes EKG vorgestellt, das Herzströme an verschiedenen Körperstellen analysiert. Lokal in den jeweiligen Sensoren (Mica2 dot Motes) werden die Herzströme zum Teil nach einem klassischen Mustererkennungsmodell verarbeitet. Die Architektur des gesamten Systems soll nach Abbildung 2.1 ein vollständiges Mustererkennungssystem enthalten.

Zu den in den Sensorknoten implementierten Modellteilen gehören die Elemente der Datenvorverarbeitung inklusive Filterung, Herzschlagsegmentierung sowie die Merkmalsextraktion (23 Merkmale). Die Klassifizierung sollte nach einer Sammlung und Fusion der Daten in einem externen System erfolgen. Die Klassifizierung sowie der Fusionsschritt (Classifier Combiner) werden in dieser Arbeit jedoch nicht implementiert. Das lokale Erkennungssystem ist auf einem NesC-basierten Betriebssystem namens SOS implementiert. Aufgrund von Instabilitäten des SOS-Systems konnte kein vollständiger Vergleich der Energieeffizienz zu einer zweiten Implementierung des Erkennungssystems als Avrora Simulation auf einem PC von Jafari et *al.* gezogen werden.

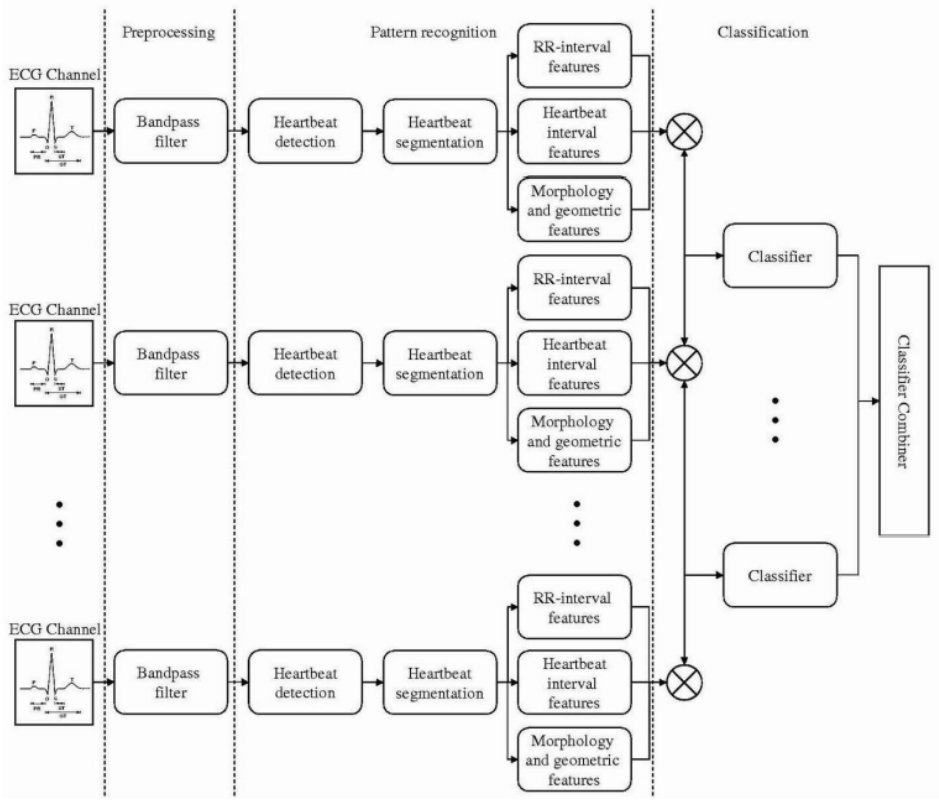

Abbildung 2.1 EKG Analyse-Schema [Jaf06] [Jaf061]

Im Gegensatz zur hier vorliegenden Arbeit ist der Fokus des verteilten EKG-Projektes die Verbesserung der Energieeinsparung in drahtlosen Sensornetzen, sowie die Berechnung der Merkmale zwischen zwei Herzschlägen. Das Team kann anhand der Simulation eine Energieeinsparung von 70 % prognostizieren.

2.1.1.2 Überwachung von Vulkanaktivitäten

Werner-Allen et *al.* hat an zwei hochaktiven Vulkanen Tungurahua [Wer05] und Reventador [Wer06] in Ecuador verteilte Sensornetze (Mica2 Motes, später TMote Sky) installiert, um Eruptionen der Vulkane mittels Ultraschallsensoren und Seismometern messen zu können. Auf Tungurahua wurden zunächst Daten und Erfahrungen gesammelt, um das spätere System auf dem Reventador korrekt zu kalibrieren und zu implementieren.

Für den Versuchsaufbau auf dem Reventador-Vulkan wurde ein verteiltes System mit einer Wahlfunktion geplant. Jeder Sensorknoten misst Eruptionswerte und sendet in die nähere Umgebung eine Anfrage, ob andere Sensorknoten ebenfalls eine Eruption ermittelt haben. Kommt die Wahlfunktion zu einer positiven Entscheidung, werden die bis dahin gepufferten Daten an die Basisstation gesendet und dort ausgewertet. Aufgrund technischer Probleme wurde die Wahlfunktion nicht implementiert. Als eingesetzte Lösung wurden Sensorschwellenwerte festgelegt, bei dessen Überschreitung die Sensoren eine Ereignismeldung zur Basis senden. Haben mehr als 30 % der Sensoren Ereignismeldungen an die Basis gesendet, beginnt die Basisstation mit der Datensammlung der auf den Sensorknoten gepufferten Messwerte. Damit reduziert sich

die Architektur auf ein Schwellenwerterkennungssystem, siehe Abbildung 2.2. Zusätzlich wurden die Sensorknoten mittels eines GPS-Senders synchronisiert.

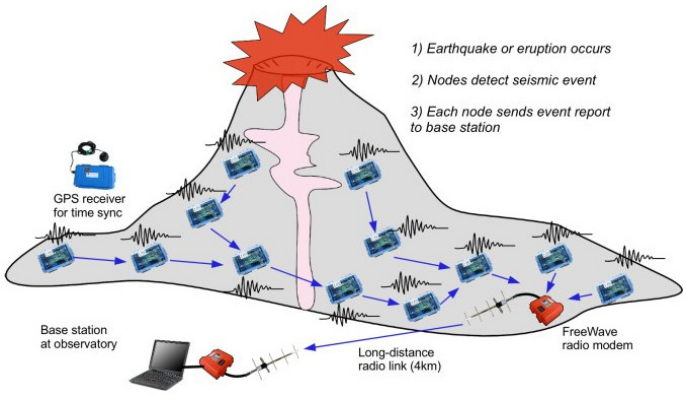

Abbildung 2.2 Vulkan-Sensorknotenarchitektur [Werm06]

Die verwendete Verteilungsarchitektur unterscheidet sich aufgrund der Übertragung aller Daten an die Zentraleinheit von der hier vorgestellten Arbeit. Die ausgewerteten Ergebnisse aus [Wer06] lassen auf starke Probleme des Gesamtsystems schließen. Diese Probleme werden zudem auf verschiedenen Ebenen auch in [Wer06] besprochen. Dazu zählen insbesondere Sensorknotenausfälle, Übertragungsprobleme in der Netzstruktur, algorithmische Probleme und Kalibrierungsprobleme der Seismometer.

2.1.2 Verkabelte Systeme

2.1.2.1 Active Floor

In [Hea01] werden Beschleunigungssensoren an den Eck- und Schnittpunkten von Bodenplatten platziert (Abbildung 2.3), um Personen zu verfolgen und sieben typische menschliche Bewegungen (kriechen, sitzen, hüpfen, gehen, aufstehen, springen, stehen) zu identifizieren.

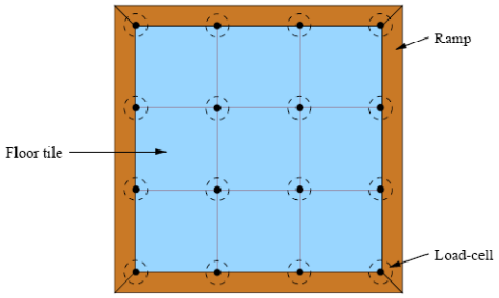

Abbildung 2.3 Draufsicht "Active Floor" [Hea01]

Die Sensoren sind per Kabel mit einer CORBA basierten Netzwerkstruktur verbunden, die ein typisches Mustererkennungssystem mit Datenvorverarbeitung, eine Merkmalsextraktion der Merkmale Mittelwert, Standardabweichung und Steigung (slope) sowie eine Klassifizierung beinhaltet. Für die Klassifizierung wird ein Hidden Markov Modell (HMM) verwendet, das mittels des Hidden Markov Model Toolkit (HTK) für Spracherkennung erstellt wird. Als

Grundvoraussetzung muss das Gewicht der Versuchsperson in der Kalibrierungsphase vom System erlernt werden.

Als Anwendung ist eine Interfacesteuerung für ein Computerspiel implementiert, die es ermöglicht, mittels der Bodenplatten einen Avatar zu steuern. Aufgrund der Auslagerung der gesamten Mustererkennung auf externe Hardware kann das System durch die so veränderten Randbedingungen nicht direkt mit dem eingebetteten System der hier vorgestellten Arbeit verglichen werden.

2.2 Dezentrale Verteilte Entscheidungsfindung

2.2.1 Drahtlose Systeme

2.2.1.1 DELTA

Das System DELTA [WälS07] erkennt und verfolgt Lichtquellen (Taschenlampen) in dunklen Umgebungen mittels der Lichtsensorik auf der ESB-Sensorplattform [Sca071] , dem Vorgängermodell der in der hier vorliegenden Arbeit verwendeten modularen Sensorplattform MSB [Sca071] . Das System erkennt Bewegungsmuster anhand des Lichtkegels der Taschenlampe.

Die Sensoren wählen einen sogenannten „Leader“ in Abhängigkeit vom Energiezustand der Sensorknoten und der Nähe des „Leaders“ zum Ereignis. Dem „Leader“ werden Daten im Falle eines Lichtereignisses der umliegenden Sensoren zugesendet, die dort gesammelt und ausgewertet werden (siehe Abbildung 2.4). Die Arbeit legt im Gegensatz zur vorliegenden Arbeit den Fokus auf die optimierte und deterministische Wahl des „Leaders“ und die optimierte Funkverbindung zwischen den Sensoren. DELTA ist dynamisch in der Lage, ereigniserfassende Sensorgruppen zu bilden.

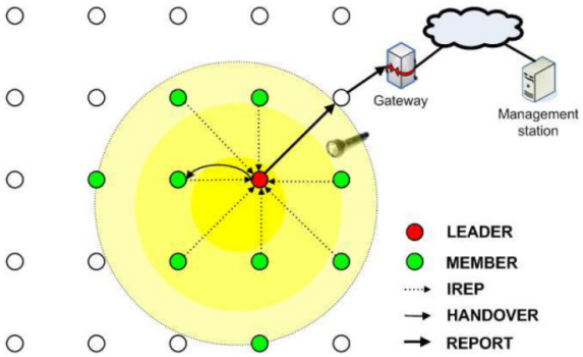

Abbildung 2.4 DELTA überwacht ein Ereignis und sendet die Daten an die Management-Station für weitere Bearbeitungsschritte [WälS07]

Aufgrund der gesamten Datenübermittlung von jedem Sensorknoten zum „Leader“ ist hier von einer Datenfusion auf Signalebene auszugehen. Konkrete Aussagen zur Datenfusion oder Merkmalsgewinnung können der Arbeit nicht entnommen werden.

Da für Lichtpegeländerungen in DELTA Schwellenwerte verwendet werden, ist zu vermuten, dass eine schwellenwertgestützte Ereigniserkennung implementiert wird, die nicht mit dem in der vorliegenden Arbeit eingesetzten, merkmalsgestützten Erkennung vergleichbar ist. Das in der vorliegenden Arbeit vorgestellte System verzichtet aus Effizienzgründen auf die in DELTA eingesetzte Rohdatenfusion.

2.2.1.2 Fence Monitoring

In [Wit07] wird das Projekt Fence Monitoring zur verteilten Ereigniserkennung mittels Beschleunigungssensoren am Beispiel der Zaunüberwachung vorgestellt. Das System arbeitet mit modularen Sensorboards (MSB) [Sca071] , die auch in der vorliegenden Arbeit verwendet werden. Die auf den Sensorknoten befindlichen Beschleunigungssensoren erfassen Zaunbewegungen und propagieren die daraus extrahierten Ereignisse an ihre Nachbarknoten. Auf das System wirken sechs Ereignisse (gegen den Zaun treten, gegen den Zaun lehnen, kurzes Schütteln, langes Schütteln, über den Zaun sehen mit kurzem „Anklettern“, über den Zaun klettern).

Ein Alarm soll ausgelöst werden, falls ein unberechtigtes Zaunüberwinden stattfindet, andere Ereignisse wurden per Annahme aus dem System ausgeschlossen. Zwei Intensitätsschwellenwerte dienen der Ereignisdefinition, womit der Speicheraufwand minimiert werden konnte. Fence Monitoring verwendet eine Softwarearchitektur bestehend aus vier Schichten (siehe Abbildung 2.6), bei der in der untersten Schicht die Rohdaten lokal auf den Sensorknoten aggregiert werden.

Abbildung 2.5 Versuchsaufbau Fence Monitoring [Witf07]

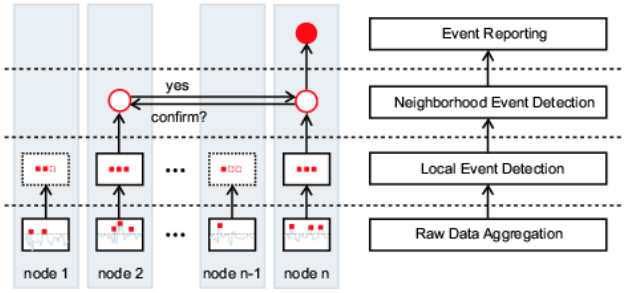

Abbildung 2.6 Softwareschichten des verteilten Ereigniserkennungsalgorithmus [Wit07]

In der zweiten Schicht wird eine Schwellenwertereignisermittlung durchgeführt. In der dritten Schicht werden die möglichen Ereignisdaten verteilt und zur Entscheidungsfusion an Nachbarknoten verteilt. Die Nachbarknoten bestätigen oder weisen das Ereignis zurück, woraufhin das Ergebnis in der vierten Schicht an die Basisstation propagiert wird. Die Autoren der Fence Monitoring-Arbeit untersuchen desweiteren, inwieweit sich n-hop-Netztechnologien in Simulationsumgebungen im Vergleich zu den gemessenen Ergebnissen verhalten. Von den Autoren konnte darauf geschlossen werden, dass die Ereignismeldung und das Routing für den Untersuchungsbereich des Fence Monitoring vernachlässigbare Problembereiche darstellen. Die vierte Schicht wurde daraufhin simuliert, da sie nicht wesentlich zur Evaluation der Erkennungsqualität von Ereignissen in drahtlosen Sensornetzen beiträgt.

Für die in dieser Arbeit vorgenommenen Untersuchungen kann auf diese Erkenntnis zurückgegriffen und die Konzentration auf die Erkennung von Ereignissen in 1-hop-Netzwerktopologien gelegt werden. Fence Monitoring vergleicht statistisch die lokale und verteilte Ereigniserkennung und erlaubt somit Vergleichsauswertungen zu der vorliegenden Arbeit. Erkennungssysteme basierend auf Schwellenwerten wie Fence Monitoring missachten möglicherweise wichtige Sensordaten und können daher keine stabilen Erkennungsraten unter realen Bedingungen garantieren. Die hier vorgestellte Arbeit unterscheidet Muster aufgrund ihrer Charakteristika und nicht durch Schwellenwerte.

2.2.1.3 Battlefield Surveillance

In [Bok06] haben die Autoren ein drahtloses Sensornetz erstellt, das die Erfassung und Auswertung von magnetischen und akustischen Signalen vornimmt, um z. B. Truppenbewegungen oder Fahrzeuge zu identifizieren. Die Sensorerfassung soll zusätzlich die Verfolgung innerhalb des von Sensoren abgedeckten Bereiches übernehmen. Die Drei-Schichten-Hybrid-Sensornetz-Architektur wird in Abbildung 2.7 gezeigt.

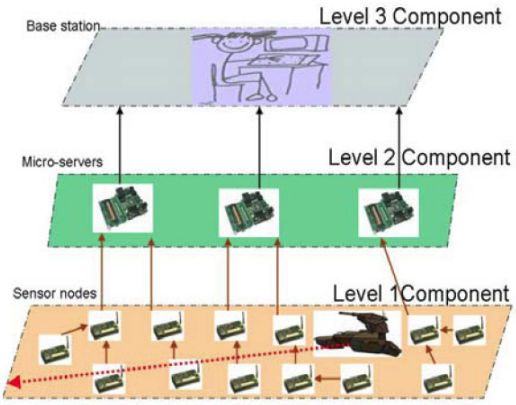

Abbildung 2.7 Hybride Sensornetz-Architektur (Pfeile zeigen den Datenfluss) [Bok06]

Der Ansatz dieses Projektes hebt sich von anderen insofern ab, dass keine Schwellenwertmustererkennung durchgeführt wird. Es wird ein Partikel-Filter-Algorithmus (Sequenzielle Monte-Carlo Methode), ein abgewandelter Bayes-Schätzer-Fusionsalgorithmus verwendet, um die Mustererkennung und die Lokalisierung des Objektes zu schätzen. Die Datenerfassung wird in Schicht 1 vollzogen, während die Datenfusion und Auswertung in Schicht 2 auf rechenstärkeren Komponenten, den sogenannten Micro-Servern, vorgenommen wird. Eine Übersicht über die resultierende Qualität des Systems wird bis auf ein Beispiel nicht gegeben. Es ist zu erwarten, dass die Lebensdauer des Systems stark beschränkt ist, da stets alle Daten der erfassenden Sensorknoten an die Micro-Server weitergeleitet werden müssen.

Das in der vorliegenden Arbeit verwendete System versucht hingegen die Aufgabe einer verteilten Mustererkennung zu lösen, ohne ein Hybridnetz mit ergänzender Hardware verwenden zu müssen. Das Verfolgen von Objekten wird in der vorliegenden Arbeit nicht untersucht, wodurch die Problematik der Sensorknotensynchronisierung nicht in Betracht gezogen werden muss.

2.2.1.4 SensIT

Forschungsarbeiten für das militärische Frühwarnsystem DARPA SensIT ([Bro03] , [LiD02] , [Dua07]) begannen im Jahre 2001. Hierbei werden Akustiksensoren zur Erkennung und Verfolgung von Ketten- und Radfahrzeugen verwendet. Es werden Fourier-Analysen (FFT) durchgeführt, um 50-dimensionale Akustikmerkmalsvektoren zu extrahieren, deren Extraktion nach der Überschreitung eines Schwellenwertes initiiert wird.

Die Sensorknoten sind in dynamische Cluster untergliedert. Diese virtuellen Sensorknotencluster werden aktiv, sobald ein Ereignis in diesem Cluster auftritt. Die Sensorknoten propagieren innerhalb ihrem Cluster die ausgewerteten Daten an einen Managerknoten. Der Managerknoten aktiviert weitere Cluster, falls diese für eine Pfadermittlung benötigt werden. An den Managerknoten werden bei SensIT jedoch die Klassifizierungsergebnisse des verwendeten Maximum-Likelihood-Klassifizierers propagiert und nicht, wie in anderen Projekten, die Rohdaten. Es kann von einer „Hard-decision"-Fusion auf der Klassifikationsebene gesprochen werden, die ebenfalls in der hier vorliegenden Arbeit eingesetzt werden soll. Im Fokus der Arbeit steht die Untersuchung, ob die Übermittlung von Merkmalsvektoren dem Klassifizierungsdatenaustausch vorzuziehen ist oder nicht. Die Arbeit kommt zu dem Schluss, dass aus kommunikationstechnischen und rechentechnischen Gründen die Fusion der Daten auf der Klassifikationsebene durchgeführt werden sollte.

Abbildung 2.8 University of Wisconsin SensIT Arbeitsgruppe bei 29 Palms, California 2001 unter Verwendung von "Collaborative signal processing" (CSP) [Dua07]

Aufgrund der im Vergleich zu der hier vorgelegten Arbeit deutlich höheren Merkmalsvektordimension ist diese Aussage nicht übertragbar. Die Klassifizierung unterscheidet bei SensIT zudem nur zwei Ereignistypen (Ketten- und Radfahrzeuge). Diese geringe Menge an Klassen stellt keinen adäquaten Vergleich zur vorliegenden Arbeit mit der Möglichkeit, die zu unterscheidende Klassenzahl skalieren zu können, dar. Desweiteren ist die hardwaretechnische Grundlage des SensIT-Projektes deutlich leistungsfähiger als die MSB-Plattform der hier vorgestellten Arbeit und lässt sich nach dem heutigen Stand der Technik laut [WälS07] mit einem PDA vergleichen. Derartige Hardwareunterschiede stellen SensIT nicht mehr mit dem hier vorgestellten Erkennungssystem auf eine vergleichbare Ebene.

2.3 Stand der Forschung

Nach dem aktuellen Stand der Forschung ist es nur dann möglich, Mustererkennung erfolgreich in verteilten Sensornetzen durchzuführen, wenn ausreichend starke Hardware in Form von Basisstationen oder Serversensoren zusätzlich zu den Sensorknoten eingesetzt wird. Systeme, deren Hardware tatsächlich nur auf die Sensortechnik wie in der hier vorliegenden Arbeit beschränkt ist, verwenden Schwellenwerte, um damit einfache Ereignisse zu definieren. Derartige Schwellenwerte können sich jedoch nur unzureichend von verschiedenen Ereignissen mit identischen Schwellenwerten abgrenzen, da mit dieser Form der reduzierten Erkennung wichtige Sensordaten missachtet werden. Somit lag bisher keine zufriedenstellende Lösung für die Erkennung von Ereignissen mit drahtlosen Sensornetzen vor.

Der Vorteil eines verteilten Sensornetzes, dass mehrere identische Komponenten aufgrund ihrer Redundanz in der Lage sind, ein Problem gemeinsam besser zu lösen als alleine, wird derzeit für die Ereigniserkennung kaum ausgenutzt.

Ein System, das die oben genannten Defizite behebt und Ereignisse nach einem spezifischen Training wiedererkennt und dafür die Vorteile eines verteilten Sensornetzes fördernd nutzt, wird in dieser Arbeit vorgestellt und untersucht.

3 GRUNDLAGEN

In Kapitel 3.1 werden die verwendeten Komponenten des Sensornetzes beschrieben. Kapitel 3.2 leistet eine Beschreibung der eingesetzten Sensorik mit deren Eigenschaften, die sich für diese Arbeit auf die Beschleunigung konzentriert. Für den Einsatz eines Ereigniserkennungssystems wird in Kapitel 3.3 ein allgemeines Mustererkennungsmodell als richtungsweisendes Konzept vorgestellt und ausgeführt. Kapitel 3.4 führt das Omnibusmodell als Fusionsmodell für die verteilte Mustererkennung ein.

3.1 ScatterWeb Plattform für drahtlose Sensornetze

Als Grundlage wird in dieser Arbeit eine neue Version der ScatterWeb Forschungshardware, wie sie in [Schi05] beschrieben wird, verwendet. Der Schwerpunkt der ScatterWeb Hardware liegt in der Modularität der Sensorknoten. Die Sensorknoten bestehen aus einem Basismodul MSB 430, dem Energieversorgungsgrundmodul MSB-T und einem optionalem Interface/Sensormodul MSB-S, wie in Abbildung 3.1 gezeigt. Das universitäre ScatterWeb-Forschungsprojekt ist unter http://cst.mi.fu-berlin.de/projects/ScatterWeb/ zu erreichen.

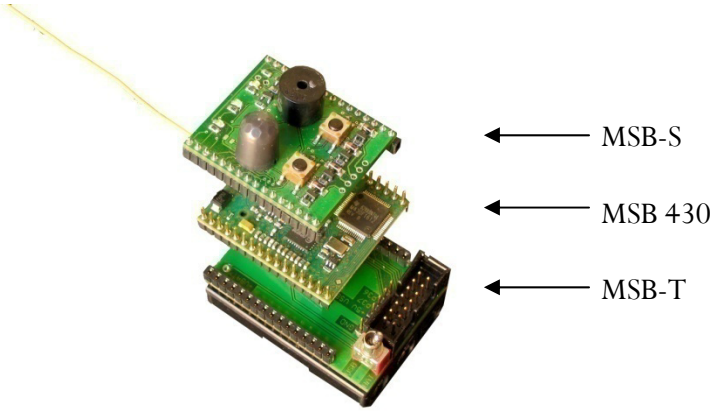

Abbildung 3.1 Die MSB-Sensorknotenmodule sind aufgrund des modularen Aufbaus äußerst flexibel einsetzbar

3.1.1 Hardware

- MSB 430 (Basismodul)
  Das MSB 430 verwendet den stromsparenden Texas Instruments MSP 430 F1612 16 Bit Prozessor. Für die Kommunikation wird der Chipcon CC1020 Funk Transceiver (Sender und Empfänger) mit 19,2 KBits/s (402-470 bzw. 804-940 MHz) verwendet.

Abbildung 3.2 MSB 430 [Sca071]

  Über den MSB 430 werden die Sensoren (analog und digital) angesteuert. Der MSB 430 verwendet einen 55 KByte großen Flash (dauerhafter Speicher) sowie einen 5 KByte großen RAM (flüchtiger Speicher). Ein Luftfeuchtesensor, Temperatursensor und eine LED befinden sich auf dem MSB 430. In dieser Arbeit wird der Beschleunigungssensor MMA726Q Verwendung finden, der in Kapitel 3.2.1 detailliert betrachtet wird.

- MSB-T (Energieversorgungsgrundmodul)
  Das MSB-T ist das Energieträgerboard und lässt sich mit drei 3V AAA Batterien bestücken. Desweiteren befindet sich eine USB-Schnittstelle auf dem MSB-T, die die Debuggschnittstelle darstellt und gleichzeitig ein Energielieferant sein kann. Eine JTAG-Schnittstelle zum Flashen bzw. Beschreiben des Sensorknotens ist ebenfalls auf dem MSB-T vorhanden.

- MSB-S (Interface/Sensormodul)
  Das MSB-S enthält zusätzlich ein Mikrophon, einen Beeper, einen Lichtsensor sowie drei LEDs und zwei frei programmierbare Knöpfe. Das MSB-S ist als optionales Board zu verstehen und wird in dieser Arbeit als Interface-Plattform verwendet, um in Testreihen ein schnelles, optisches und akustisches Feedback zu erhalten. Zusätzlich zu den genannten Sensoren sind weitere Sensoren wie z. B. GPS und Infrarot als Einsatzmöglichkeit denkbar.

In dieser Arbeit wird stets von einem „Sensorknoten“ oder dem „MSB“ gesprochen. Ein Sensorknoten oder MSB, bezieht es sich auf die hier vorliegende Arbeit, besteht immer aus allen drei zusammengesetzten Modulen (dem MSB 430, MSB-T mit 3 Batterien sowie dem MSB-S).

3.1.2 Software

Aufgrund des Vorhandenseins der ScatterWeb Systemsoftware (Firmware) und einem umfangreichen API (Application Programming Interface) ist es möglich, schnell erste Versuche mit den Sensorknoten durchzuführen. Die ScatterWeb Systemsoftware ist verantwortlich für Standardaufgaben, wie das Behandeln von Interrupts, Nachrichtenpaketen, Timer und den Speicherzugriff. Ein interner Watchdog schützt das System vor Endlosschleifen. Zeitgeber und die Zeit werden ebenfalls von der Systemsoftware zur Verfügung gestellt. Ebenso wie die vorliegende Systemsoftware in ANSI-C programmiert ist, muss die Nutzerapplikation in ANSI-C programmiert werden.

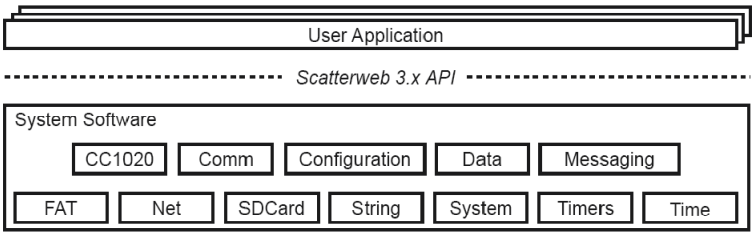

Abbildung 3.3 ScatterWeb 3.x Softwarearchitektur [Wit07]

Die Systemkomponenten und die Struktur der Software ist in Abbildung 3.3 dargestellt.

3.2 Beschleunigung

3.2.1 MMA7260 Beschleunigungssensor

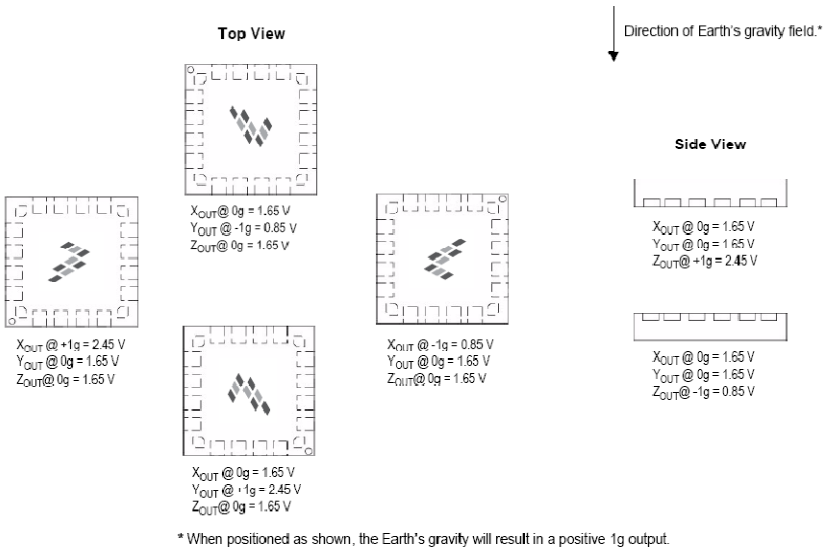

Abbildung 3.4 Auswirkung der Erdanziehungskraft auf die jeweilige Lage des MMA7260 [Fre07]

Der Sensorknoten MSB 430 ist mit dem Dreiachsenbeschleunigungssensor MMA7260 von Freescale Semiconductor, Inc. ausgestattet. Der Beschleunigungssensor liefert Spannungsdaten

im Bereich von 0 bis 3 Volt. Diese Spannungswerte werden mit einem 12-Bit Analog-Digital-Wandler digitalisiert und stellen eine lineare Abbildung der zugefügten Beschleunigung dar. Die Beschleunigung wird hierbei in der Einheit [g] angegeben und der maximal messbare Wert hängt von der Voreinstellung der Sensitivität ab. Die Sensitivität lässt sich während der Messung verändern und erlaubt Messungen der in Tabelle 3.1 angegebenen Beschleunigungsbereiche.

Tabelle 3.1 Spannungsveränderung pro g Sensibilitätseinstellung [Fre07]

g-Bereich	Sensibilität
1,5g	800 mV/g
2g	600 mV/g
4g	400 mV/g
6g	200 mV/g

Der Beschleunigungssensor selbst verwendet eine geschaltete Kondensatortechnik, siehe Abbildung 3.5 (rechts), um die verschiedenen Spannungsveränderungen zu ermitteln. Diese Technik lässt sich mittels eines abstrakten Balkenmodells, wie in Abbildung 3.5 (links) gezeigt, darstellen. Die beiden äußeren Balken sind fixiert. Die Abstände beider Balken zueinander stellen die maximal mögliche Messbandbreite dar. Der mittlere Balken ist flexibel und verändert seine Lage in Abhängigkeit von der zugefügten Beschleunigungsrichtung jeweils in Richtung einer der beiden fixierten Balken. Die Abstände zwischen den fixierten und dem flexiblem Balken geben Auskunft über die Intensität der Beschleunigung. Die Analogie zu der Kondensatortechnik stellt sich so dar, dass die Differenzen zwischen beiden Kondensatoren in Form von Spannungen ausgedrückt werden.

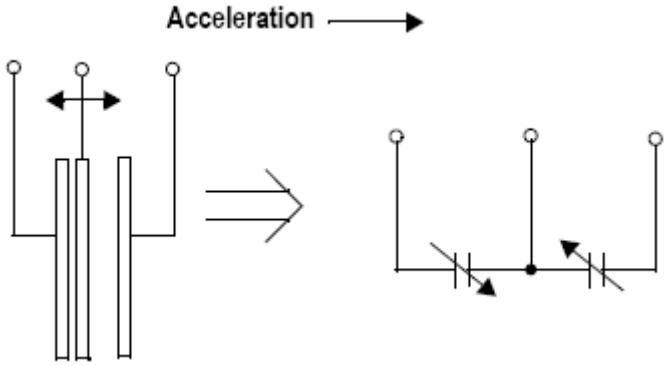

Abbildung 3.5 Vereinfachtes Messwandler Physik-Modell [Fre07]

Der gesamte Schaltkreis nutzt eine Temperaturkompensation und gewährt somit störungsfreie Messungen in einem Temperaturbereich von -40 C° bis +125 C°. Das vereinfachte Blockdiagramm in Abbildung 3.6 stellt einen Überblick der im Beschleunigungssensor integrierten Schaltungselemente dar. Die umgebende Schaltung taktet das System, filtert extreme Signalspitzen mittels eines Hardware-Tiefpass-Filters, verstärkt das Signal und sorgt für ein proportionales Verhältnis der Spannungsausgabe zur realen Beschleunigung. Die Ausgabe der Daten erfolgt

ratiometrisch, demzufolge wird die Versorgungsspannung auf 0 bis 100 % des möglichen Wertebereiches der Beschleunigung abgebildet. Die Proportionalität sorgt für eine lineare Abbildung. Die ermittelten Spannungswerte des Sensors werden nach dem oben beschriebenen Prinzip auf einer Werteskala von 0 bis 4095 abgebildet. Die Werteskala richtet sich nach der Auflösung des verwendeten 12-Bit Analog-Digital-Wandlers, der dem Beschleunigungssensor nachgeschaltet ist. Die Ruheposition, also der Mittelwert, wird demnach bei einem Wert von 2047 bzw. 2048 erwartet.

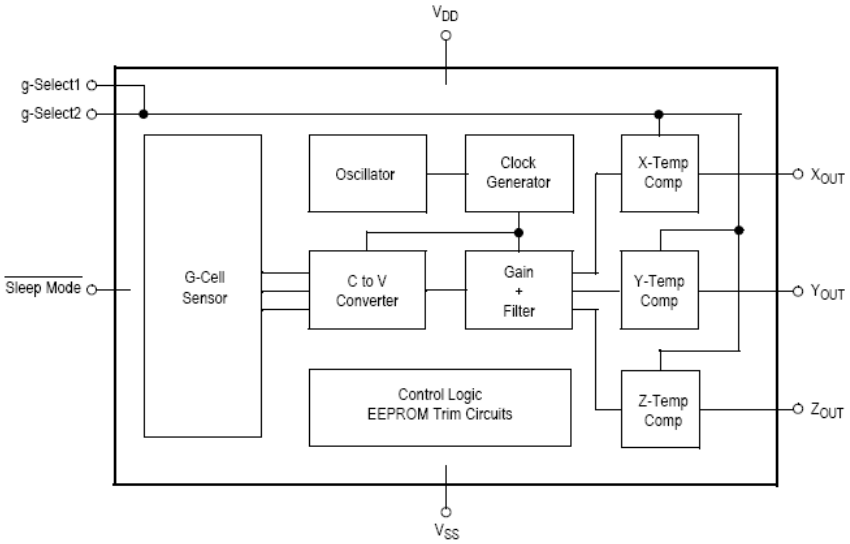

Abbildung 3.6 Vereinfachtes Blockdiagramm des Beschleunigungssensors [Fre07]

Aufgrund baulich bedingter Schwankungen, leichter Ausgangsschieflagen während der Kalibrierung und der Erdbeschleunigung kann der Erwartungswert des Sensors in Ruhestellung von 2048 nicht durch praktische Tests bestätigt werden. Es ist folglich nötig, für jeden Sensorknoten und jede Achse den individuellen Ruhepunkt zu ermitteln und die Sensibilität unter Berücksichtigung des jeweiligen Einsatzgebietes anzupassen. Die Umsetzung der Kalibrierung wird in Kapitel 4.3.1 und Kapitel 4.4.1 eingehend beschrieben.

Die Sensibilität lässt sich über die in Abbildung 3.6 gezeigten Leitungen *g-Select 1* und *g-Select 2* verändern. Die Sensibilitätseinstellungen entsprechen der maximal messbaren Beschleunigung. Sobald eine der Sensorachsen nicht mehr exakt horizontal zur Erdoberfläche ausgerichtet ist, wirkt die Erdanziehungskraft auf diese Achse. In der Praxis (Kalibrierung und Messung) darf demnach die Erdbeschleunigung nicht vernachlässigt werden. Sie wirkt sich insbesondere für Sensibilitätseinstellung von 1,5 g aus, da dort bereits die Erdbeschleunigung mit 1 g auf eine Achse wirken kann (siehe Abbildung 3.4). Die verbleibenden 0,5 g reichen für manche Anwendungsbereiche nicht mehr aus, um verwertbare Messungen für die von der Erdbeschleunigung beeinflusste Achse vornehmen zu können. Die Verwendung einer Sensibilität von 4g garantiert somit maximale Messungen von bis zu 3g in allen möglichen Ausrichtungslagen des Sensors.
Drehungen können dazu führen, dass Achsen, die während der Kalibrierung nicht von der Erdanziehung beeinflusst wurden, in den Bereich der Erdanziehungsachse gelangen. In Abbildung 3.4 werden die Auswirkungen der Erdanziehungseffekte auf alle Achsen in dem Moment dar-

gestellt, in dem eine Achse jeweils so ausgerichtet ist, dass der gesamte Bereich der Erdanziehungskraft auf die entsprechende Achse wirkt [Fre07] .

3.2.2 Verifikation der Beschleunigungsdaten

Um die Beschleunigungsdaten zu verifizieren, soll überprüft werden, ob sich eine geometrische Figur mittels der Positionsdaten als solche darstellen lässt. Positionsdaten aus Beschleunigungswerten erhält man mittels der zweifachen Integration. Diese Form der Verifikation zeigt, ob die erzeugten Beschleunigungsdaten richtig verarbeitet werden bzw. ob Schwächen im System zum Tragen kommen, die zunächst nicht ersichtlich sind. Ein gutes Verständnis der Funktionsweise des Beschleunigungssensors ist für eine praktische Arbeit wie diese nur zu erreichen, indem Erfahrungswerte gesammelt werden.

3.2.2.1 Motivation

Es wird untersucht, ob die Darstellung einer geometrischen Figur tatsächlich als solches möglich ist. Gesucht wird nun eine Datendarstellung, die das zukünftig unbekannte Muster so gut repräsentiert, dass man als Betrachter die Eigenschaften intuitiv gut erkennen kann. Desweiteren liegt ein Schwerpunkt auf der Verarbeitung der Daten lokal auf den Sensoren, was dazu führt, dass Datenkonvertierungen nur dann sinnvoll sind, wenn sie einen außergewöhnlichen Mehrwert bieten und zudem mit dem Sensorknoten technisch möglich sind.

Für die Verifikation der Beschleunigungsdaten ist es hilfreich, diese von bekannten, einfachen geometrischen Formen wie Kreis, Quadrat oder Dreieck zu sammeln. Beschleunigungsdaten dieser geometrischen Figuren lassen sich in ihrer zweiten mathematischen Integration darstellen, wobei die Position des Beschleunigungssensors ermittelt wird. Eine direkte grafische Darstellung der Beschleunigungsdaten ist intuitiv schwer auswertbar und reicht damit zunächst nicht aus, um dem Sensorknoten zugefügte Bewegungsmuster zu verifizieren. Die erste mathematische Integration der Beschleunigungsdaten liefert die aus den zugeführten Beschleunigungen resultierenden Geschwindigkeiten $\vec{v}$. Diese Geschwindigkeiten geben einen besseren Einblick auf die Qualität der Beschleunigungsvektoren $\vec{a}$ (siehe Gleichung 1). Die Integration der Geschwindigkeitswerte liefert die Position des Sensors über den Entfernungsvektor $\vec{s}$ und stellt somit einen optimalen visuellen Abgleich mit der Wirklichkeit dar (siehe Gleichung 2). In Gleichung 3 werden die soeben angesprochenen Gleichungen kombiniert dargestellt.

$$\vec{v} = \int (\vec{a}) dt \tag{1}$$

$$\vec{s} = \int (\vec{v}) dt \tag{2}$$

$$\vec{s} = \int (\int (\vec{a}) dt) dt \tag{3}$$

Insbesondere bei einfachen geometrischen Figuren zeigt die resultierende Darstellung die durchgeführte Bewegung als intuitives Bild. Ein solch intuitives Bild hilft, den Beschleunigungs-

sensor zu verstehen und die Definition zu bestätigen, dass die zweite Integration der Beschleunigungsdaten auch in der Praxis das Bewegungsmuster repräsentiert. Auf diese Weise ist die korrekte Behandlung der Sensordaten zu überprüfen.

Eine mathematische Funktion liegt den zu integrierenden Daten nicht zugrunde. Es wird mit einer Folge von Werten gearbeitet, einer sogenannten Zeitreihe. Für die Integration stellt das ein Problem dar, da sie in ihrer ursprünglichen Form nur mit Funktionen arbeiten kann, siehe Gleichung 3. In [Sei07] wird empfohlen, ein Approximationsverfahren für die Integration zu verwenden. In der vorliegenden Arbeit wird die Euler-Approximation [Col51] eingesetzt, deren Genauigkeit davon abhängt, wie klein die Messintervalle sind, da angenommen wird, dass die Beschleunigung innerhalb eines Intervalls konstant bleibt.

3.2.2.2 Durchführung

Für die Visualisierung eines Musters wird die geometrische Figur „Kreis“ ausgewählt. Um diese Darstellung zu demonstrieren, wird der Sensorknoten über eine gezeichnete Kreisvorlage gleichförmig im Kreis bewegt. Die Kreisvorlage hat einen Durchmesser von ca. 32 cm. Der Sensorknoten selbst ist auf einem widerstandsarmen Gleiter montiert, der zudem seitliche Kippbewegungen verhindert.

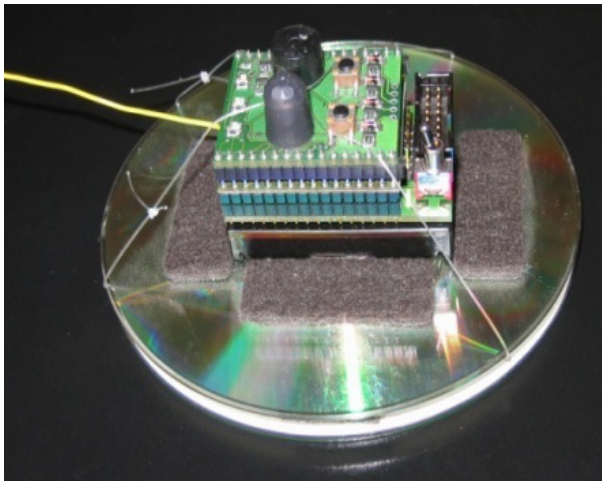

Abbildung 3.7 Sensorknoten auf Gleitvorrichtung montiert

Um die Auswertung der Bewegung zu vereinfachen, wird von der Auswertung der Z-Achse abgesehen. Für die Kreisbewegung sind nur die zwei Achsen X und Y relevant. Das im Verlauf der Arbeit zu entwickelnde Mustererkennungssystem wird jedoch alle drei Achsen verwenden können.

Die nachfolgenden Abbildungen visualisieren die vorangegangenen Überlegungen. Es werden hier die Beschleunigungsdaten und ihre mathematischen Integrationen erläutert.

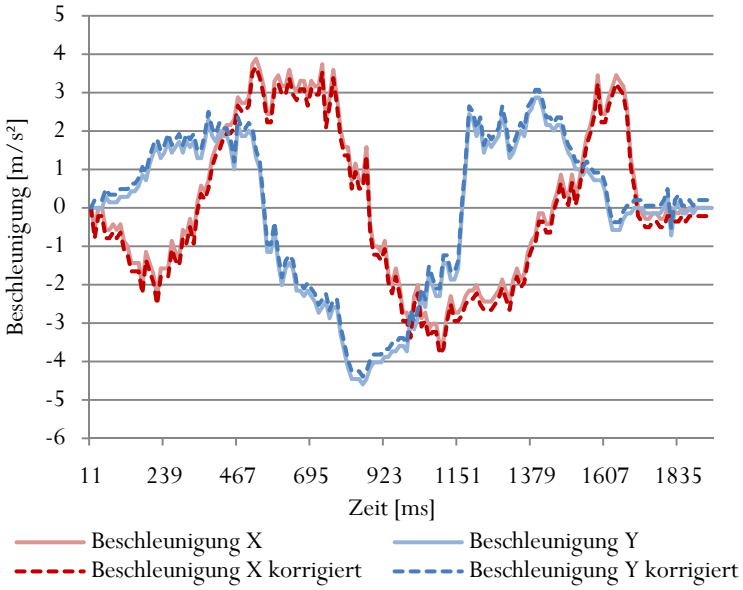

Abbildung 3.8 Beschleunigungsdaten vor und nach Offsetkorrektur

Abbildung 3.8 zeigt die Beschleunigungsrohdaten, die der Beschleunigungssensor liefert. Der computertechnische Aufwand beschränkt sich bisher auf die Vorverarbeitung der Daten. Die Form der Kurven hängt stark von der Gleichförmigkeit der Bewegung ab. Die Charakteristik der Kurven ist jedoch erkennbar. Die Rohdaten sind bereits so aufgearbeitet, dass sie einen korrekten Nullpunkt aufweisen und der üblichen physikalischen Einheit für die Beschleunigung $[^{m}/_{s^2}]$ genügen.

Die gestrichelten Kurven stellen in Abbildung 3.8 bereits eine nachkorrigierte Fassung der Daten dar. Der Zweck der Korrektur wird jedoch erst in Abbildung 3.11 ersichtlich. Diese minimale Korrektur stellt die Auswirkung eines veränderten Offsets und damit des Nullpunktes dar. In Abbildung 3.9 und Abbildung 3.10 werden die Auswirkungen dieser Veränderung ebenfalls durch gestrichelte Kurven aufgezeigt. Die Offsetverschiebung bewirkt, wie in Abbildung 3.11 zu erkennen, die korrekte Kreisdarstellung und wird im folgenden Text begründet.

3.2.2.3 Auswertung

Wie in Kapitel 3.2.2 besprochen, wird die erste und zweite Integration der Beschleunigungsdaten grafisch in Abbildung 3.9 und Abbildung 3.10 veranschaulicht. Bei Verwendung der Integralrechnung fallen die fehlenden Messdaten und eine nicht perfekte Kalibrierung anhand der auseinanderlaufenden Kurven auf. Das Problem der in der Approximation als konstant angenommenen Daten wirkt sich bei der Berechnung der Geschwindigkeit so aus, dass die Endgeschwindigkeit noch 0,25 $^{m}/_{s^2}$ beträgt, obwohl der Sensorknoten in der Realität still steht. Zudem verstärkt sich der Fehler mit der zweiten Integration soweit, dass sich das visualisierte Resultat der zweiten Integration nicht als Kreis darstellt. Darüber hinaus verstärkt sich der Fehler mit andauernder Zeit. Je kürzer also das Muster ist, umso weniger wirken sich die Fehler aus. Das hier vorgestellte Muster hat eine Dauer von ca. 1,5 Sekunden. Es ist davon auszugehen, dass zukünftig betrachtete Muster deutlich länger dauern können und dieser Fehler die Darstellungen noch stärker verfälschen wird, je länger die Muster dauern. Mittels einer leichten Verschiebung des Offsets, also des eigentlichen Nullpunktes beider Achsen, ist es möglich, den Fehler nahezu komplett auszugleichen. Die Veränderung der Originaldaten bleibt dabei praktisch unsichtbar. Die Resultate der ersten und zweiten Integration dagegen sind erheblich verändert. Die korrigierten Daten sind mittels der gestrichelten Linien in allen Abbildungen des Kapitels 3.2.2 dargestellt.

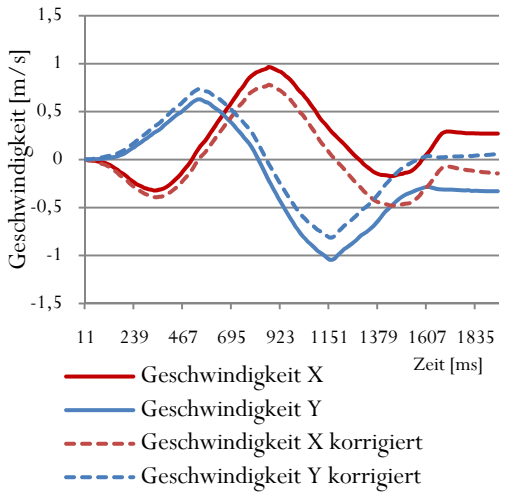

Abbildung 3.9 Erste Integration: Geschwindigkeitsdaten vor und nach Offsetkorrektur

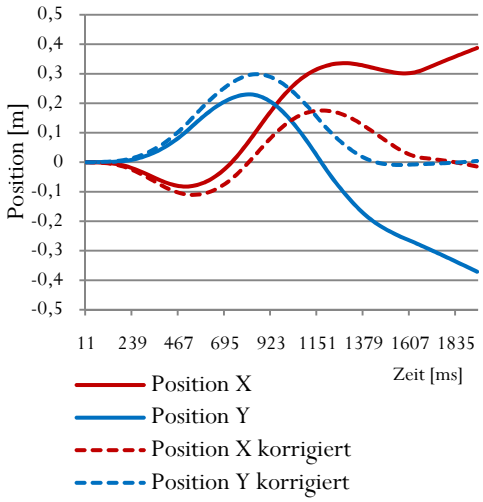

Abbildung 3.10 Zweite Integration: Positionsdaten vor und nach Offsetkorrektur

Die Abbildung 3.9 und Abbildung 3.10 stellen die errechneten Geschwindigkeiten des Musters und die zurückgelegten Strecken (Positionen) dar. Abbildung 3.11 kombiniert die Daten der Abbildung 3.10 in einer gemeinsamen Darstellung auf zwei Achsen, um die Form des Musters zu visualisieren. Ein Kreis ist aus den X-Y-Positionsdaten (durchgezogene dunkelgrüne Linie) nicht erkennbar aber vorstellbar. Die verzerrte Darstellung des Kreises resultiert aus zwei Fehlerquellen, der Kalibrierung des Systems und dem nicht kontinuierlichen Datenstrom, auf dem integriert wird. Eine 100 %ig perfekte Kalibrierung ist aufgrund der in Abschnitt 3.2.1 beschriebenen hardwarebedingten Ungenauigkeiten nicht möglich.

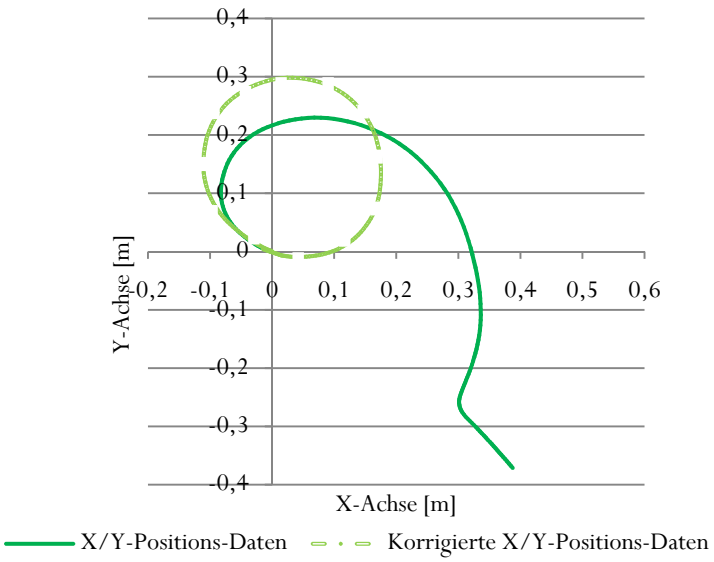

Abbildung 3.11 Ergebnis der gemeinsamen Achsendarstellung von X/Y: Kreisdarstellung vor und nach Offsetkorrektur (Durchmesser ca. 32 cm)

Selbst eine zeitlich hochauflösende Messreihe ist eine nicht-kontinuierliche Messreihe. Dadurch enthalten die Messungen des Sensors nicht alle Daten, die nötig wären, um einen korrekten Kreis aus den Daten zu extrahieren. Dies bedeutet, dass sich Lücken in der Messreihe ergeben, an denen keine Beschleunigungswerte gemessen werden. Trotz der Tatsache, dass die Daten in dieser Arbeit schneller als nach der Empfehlung des Nyquist-Shannon Abtasttheorems erfasst werden, sind sie verlustbehaftet. Das Abtasttheorem besagt: Tastet man das analoge Signal mit mindestens der doppelten maximalen Signalfrequenz ab, ist mit der Abtastung kein Informationsverlust verbunden, man kann das ursprüngliche analoge Signal wieder herstellen. Das Nyquist-Shannon Abtasttheorem basiert auf gleichförmigen Schwingungen, wie einer Sinusschwingung. Eine Sinusschwingung wird jedoch bei Bewegungsmustern nicht erzeugt, auf dessen Basis (Grenzfrequenz des Tiefpassfilters ca. 15 Hz) eine Abtastfrequenz von mehr als 30 Hz nach dem Nyquist-Shannon-Abtasttheorem gewählt werden müsste. Die gewählte Samplingfrequenz wird bei 100 Hz festgelegt, um möglichst detaillierte Informationen über den Signalverlauf zu erhalten. Damit wird bereits eine 10-fach höhere Frequenz und Genauigkeit verwendet als in der Arbeit von Wittenburg [Wit07] , die generell einen guten Vergleich zu dieser Arbeit darstellt. Eine höhere Abtastfrequenz bedeutet zugleich, mehr Daten bei konstantem Speicher auf dem Sensor puffern zu müssen. In Kapitel 3.2.2 wird die Samplingfrequenz für die konkrete Mustererhebung auf 50 Hz gesenkt.

3.2.2.4 Schlussfolgerung

Die erwartete Darstellung, in diesem Fall ein Kreis, wird durch eine minimale Offsetänderung erzielt. Die Charakteristik der Daten aus der zweiten Integration ist bei geringfügigen Fehlern in den Ausgangsdaten bereits soweit verändert, dass das Muster „Kreis“ nicht mehr als solche Form erkannt wird. Vielmehr können leichte Veränderungen der Ausgangsdaten vollkommen verschiedene geometrische Musterformen hervorbringen, wie Abbildung 3.11 zeigt.

Die Beschleunigungsdaten sind trotz der leichten Abweichungen plausibel, da die Positionswerte inklusive der Größenordnung des Durchmessers von ca. 32 cm der Kreisvorlage nahezu entsprechen. Der Vorteil einer intuitiven Darstellung des Musters als Bild stellt sich als hilfreich heraus, um zu verstehen, welche Auswirkungen eine schon leicht fehlerhafte Kalibrierung haben kann. Die gewonnenen Erfahrungswerte durch die Verifikation der Beschleunigungsdaten bringen zusätzlich Klarheit über das Themenfeld der Beschleunigungssensorik. Die Bedeutung der Offsetberechnung während der Kalibrierung muss besonders betont werden. Der Einsatz einer visualisierten Musterdarstellung für die Mustererkennung schließt sich jedoch aus, da der Berechnungsaufwand für die nötigen Integrationen sehr umfangreich ist und der entstehende Mehrwert fraglich ist. Die Visualisierungen lassen sich nicht wiederholt und automatisch in gleicher Qualität erzeugen, was an den schwankenden Offsetkorrekturen liegt, die abhängig von der Kalibrierung und dem erzeugten Muster sind. Zudem ist nicht auszuschließen, dass weitere bisher nicht untersuchte Faktoren, wie Temperatur und Ladestand der Batterien, einen Einfluss haben. Die Verwendung der Beschleunigungsdaten hingegen bietet sich aufgrund ihrer immer wiederkehrenden sehr ähnlichen Strukturen als Grundlage für eine Merkmalsextraktion an.

3.3 Mustererkennung

"The science that concerns the description or classification (recognition) of measurements" [Sch92]

Mustererkennung ist ein aktiver Forschungsbereich. Der wirtschaftliche Bedarf an Möglichkeiten einer automatischen Klassifizierung von Objekten zeigt die enorme Signifikanz der Mustererkennung. Eine Klassifizierung findet bereits in vielen verschiedenen Bereichen Anwendung, wie z. B. in der Objekterkennung (Geldstücke, Banknoten), Spracheerkennung, Bildererkennung (Röntgenbilder, Magnet-Resonanz-Tomographie-Bilder), Bewegungserkennung (Gebäudeüberwachung, Sportanalysen), Handschriftenerkennung, Topographische Strukturanalysen, DNA-Sequenzerkenung, Biometrikerkennung (Personenidentifikation, Iris-Scan, Fingerabdruck, Gesichtszüge), Elektrokardiogrammanalyse und in vielen neu aufkommenden Anwendungsbereichen. Typischerweise beobachtet man in der Mustererkennung spezielle Parameter der Muster und schafft Möglichkeiten, diese algorithmisch in bekannte Klassen zu klassifizieren. Letztendlich wird die Klassifizierungsentscheidung unter Zuhilfenahme von vorher extrahierten Informationen aus Trainingsdaten durchgeführt. [RPo06]

Alle Mustererkennungsmodelle verwenden Standardschritte, um die Mustererkennung durchzuführen. Es wird im Anschluss detaillierter auf die Funktion und die Möglichkeiten der einzelnen Standardschritte aus typischen Mustererkennungssystemen eingegangen. Einen Überblick zur Mustererkennung beschreibt Liu in [Liup06] .

3.3.1 Terminologie und Einführung

Ziel der Mustererkennung ist es, eine geeignete Transformation eines gegebenen Musters in eine zugehörige Klasse zu finden. Diese komplexe Aufgabe wird zunächst zerlegt, um die Lösung der Teilprobleme zu vereinfachen. Es gibt bisher keine optimale Routine für die Lösung

eines beliebigen Mustererkennungsproblems. Das Modell in Abbildung 3.12 ist erfolgreich getestet und vielfach eingesetzt. Dieses und ähnliche Standardmodelle für Mustererkennungsprobleme werden sehr oft in der Literatur als Grundlage verwendet, um darauf geeignete Mustererkennungssysteme aufzusetzen. [Nie03] , [Dic98] , [Dud01] Das in Abbildung 3.12 gezeigte Standardmodell ist für den jeweiligen Einsatz zu konkretisieren. So wird auch im Rahmen dieser Arbeit das Standardmodell übernommen und den gegebenen technischen Umständen angepasst.

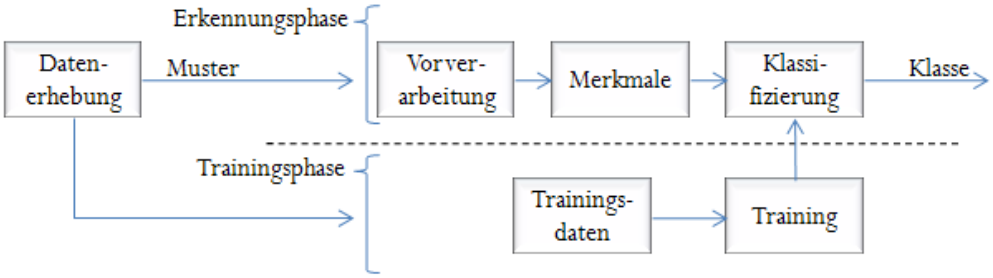

Abbildung 3.12 Die Hauptmodule des Standardmodells der Mustererkennung, basierend auf [Nie03]

Wie in Abbildung 3.12 dargestellt, wird ein zu klassifizierendes Signal zunächst durch einen oder mehrere Sensoren aufgenommen (**Aufnahme** bzw. **Datenerhebung**) und digital umgewandelt. Diesen Vorgang übernimmt der Beschleunigungssensor hier selbständig, wie in Kapitel 3.2.1 erwähnt. Die auf die Digitalisierung einflussnehmenden Parameter sind insbesondere das verwendete Samplingintervall und die eingestellte Sensibilität des Beschleunigungssensors. Die digitalisierten Daten werden im Folgenden **Rohdaten** genannt. Während der Vorverarbeitung werden die Signaleigenschaften unter dem Gesichtspunkt der Analysierbarkeit und Performanz des Gesamtsystems verbessert. Die so verbesserten Daten werden **vorverarbeitete Rohdaten** genannt Die daran anknüpfende Merkmalsextraktion basiert auf der Annahme nachfolgend erläuterter Postulate und filtert die Kerneigenschaften der **Muster** heraus. Muster können als Abfolgen von **Merkmalen** beschrieben werden, deren zugehörige **Klasse** zunächst unbekannt ist. Diese so genannten Merkmale werden für jede Klasse in einem **Merkmalsvektor** zusammengefasst. Die Merkmale werden entsprechend klassifiziert und bilden das Muster somit eindeutig auf eine Klasse ab. Klassen beschreiben eine Mustergruppe, wobei zunächst kein Ausschluss von Mustern vorgenommen wird, sondern nur die konkreten Eigenschaften der Muster enthalten sind. Die Klassifikationsinstanz muss die **Klassengrenzen** kennen, um in der Lage zu sein, eine geeignete Klassifikation vornehmen zu können (siehe Abbildung 3.13). Alle Informationen wie Klassengrenzen und Merkmale der jeweiligen Klassen, sind aus den gesammelten **Stichproben** zu extrahieren. Während des **Trainings** werden die Stichproben der möglichen, aufkommenden Muster erfasst und dem System übergeben. Das Training bildet die Grundlage jeder Mustererkennung und ist mitverantwortlich für deren Qualität.

Die im vorigen Abschnitt angesprochenen Merkmale definieren die Mustercharakteristik. Hierbei entsteht oft die Annahme, dass eine größere Anzahl von Merkmalen auch mehr Informationen bieten und damit eine verbesserte Klassifizierung erlauben. Eine extrem hohe Dimensionalität birgt jedoch auch Risiken. Die inhärente Komplexität solch hochdimensionaler Räume wird in der Literatur "**Fluch der Dimensionalität**" genannt.

> "*The fundamental reason for the curse of dimensionality is that high-dimensional functions have the potential to be much more complicated than low-dimensional ones, and that those complications are harder to discern. The only way to beat the curse is to incorporate knowledge about the data that is correct.*" [Dud01]

Für den Menschen ist es schwer, ein Verständnis für hohe Dimensionen zu entwickeln. Demzufolge sind hohe Dimensionen nur dann zu handhaben, wenn bekanntes Wissen über die Korrektheit von Teilergebnissen vorliegt.

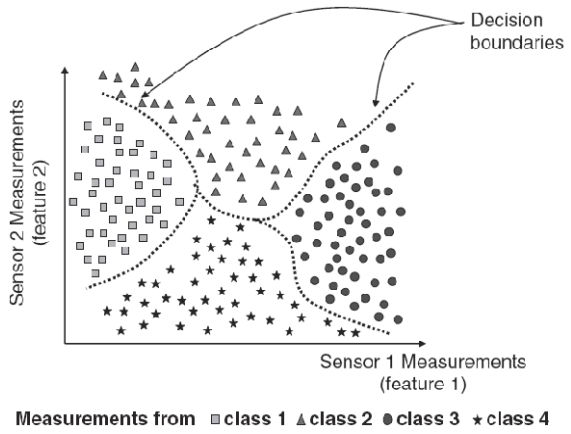

Abbildung 3.13 Beispiel für optimistische Klassengrenzen und 2D-Vektoren [RPo06]

In Abbildung 3.13 ist der Idealfall repräsentiert, in dem Klassengrenzen klar definiert sind. Hier werden zwei Merkmale für vier Klassen verwendet. Diese geringe Anzahl von Merkmalen reicht oft nicht aus, um Klassen zu unterscheiden, wie Abbildung 3.14 verdeutlicht. Eine starke Erhöhung der Merkmalsanzahl kann sich sogar ungünstig auswirken, da die Klassen aufgrund des Fluches der Dimensionalität möglicherweise nicht mehr optimal unterschieden werden können.

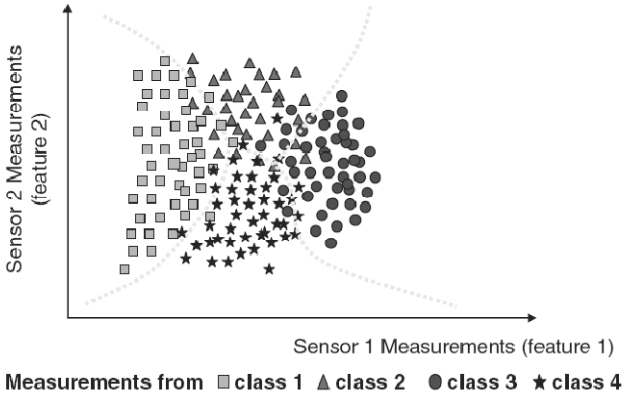

Abbildung 3.14 Abgrenzungsproblem ist nicht einfach lösbar, da sich Klassengrenzen überschneiden [RPo06]

Entscheidend ist, dass nur soviel wie nötig der nach Postulat 3 (siehe Kapitel 3.3.2) definierten Merkmale verwendet werden.

3.3.2 Postulate

Um trotz hochdimensionaler Merkmalsvektoren gute Erkennungsraten zu erzielen, ist es nötig, ausreichend viele und optimale Trainingsdaten zu erzeugen. Trainingsdaten bzw. Stichproben sollten so repräsentativ wie möglich für die jeweilige Klasse sein. Wird die Klasse nicht ausreichend durch die Trainingsdaten repräsentiert, kann dies eine Begründung dafür sein, dass die Klassifizierung nicht gut funktioniert. Eine nicht ausreichende Repräsentation entsteht beispielsweise durch stark verrauschte Daten. Eine zu geringe Trainingsmenge, aufgrund derer einige Sonderfälle nicht in den Trainingsdaten vorhanden sind (von denen aber erwartet wird, dass sie richtig klassifiziert werden), kann ebenfalls zu unbefriedigenden Erkennungsraten führen.

Beide Anforderungen, Rauschfreiheit und ausreichende Präzision der Trainingsdaten, stören sich bei deren Erfüllung gegenseitig und erfordern einen Kompromiss in der Trainingsmenge. Eine zu große Trainingsmenge lässt die Daten verrauschen und eine zu geringe Trainingsmenge ist zu unpräzise. Beide Anforderungen sind für reale Probleme daher kaum zu erfüllen. Ein System gilt als **übertrainiert**, wenn sehr viele Sonderfälle trainiert wurden und so keine wirklich sauberen Klassengrenzen gezogen werden können. Oft sind Sonderfälle direkt mit Rauschen in Verbindung zu bringen.

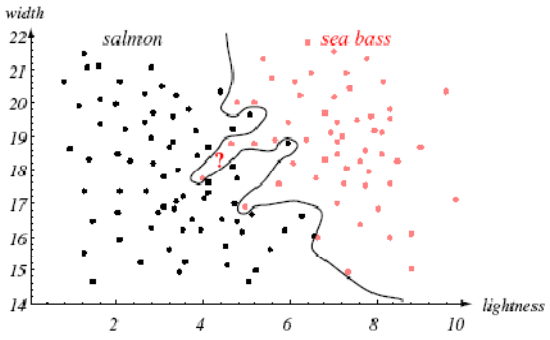

Abbildung 3.15 Übertrainiertes System mit einer schlechten Performanz. [Dud01]

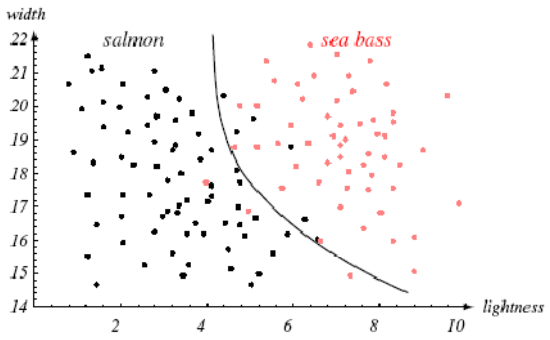

Abbildung 3.16 Hohe Genauigkeit der Klassifizierung und hohe Performanz mittels generalisierter Klassengrenze. [Dud01]

Wenn die Rohdaten Rauschen enthalten und die Klassengrenzen so spezifisch definiert sind wie in Abbildung 3.15, liegt das Problem vor, dass das System extrem komplexe Klassengrenzen verwalten muss. Die Klassifizierungsqualität wird durch diese Klassengrenze gut aber nicht 100 %ig genau sein. Zudem wird die Leistungsfähigkeit des Systems durch komplexe Klassengrenzen dramatisch reduziert. Trotz dieser präzisierten Klassengrenze wird es Ausreißer geben, die falsch klassifiziert werden, wie das rote Fragezeichen in Abbildung 3.15 andeutet. Da es sehr schwer ist, alle Möglichkeiten zu trainieren, können derartige Ausreißer in verrauschten Syste-

men immer vorkommen. Einen guten Kompromiss zwischen Performance und Genauigkeit der Klassifizierung stellt Abbildung 3.16 dar.

Die folgenden wichtigen Mustererkennungspostulate lassen sich aus dem vorher beschriebenen Inhalt ableiten und bilden die theoretische Grundlage nahezu aller Mustererkennungssysteme [Dud01] , [Nie03] :

- Postulat 1: Information
  Innerhalb einer Problemdomäne wird angenommen, dass es möglich ist, repräsentative Informationen aus Trainingsdaten zu erhalten.

- Postulat 2: Eigenschaftsvektoren
  Es ist möglich, für jede Klasse einen charakteristischen Merkmalsvektor zu berechnen. Die Dimensionalität des Vektors sollte so klein wie möglich sein, um nicht durch den „Fluch der Dimensionalität“ auf schwer lösbare Probleme zu treffen. Ein Klassifizierer ist nur so gut wie die Merkmale des Erkennungssystems.

- Postulat 3: Kompaktheit
  Die Merkmalsabstände eines Merkmals sollten innerhalb einer Klasse möglichst gering sein. Wohingegen die Abstände zu anderen Klassen möglichst groß sein sollten.

- Postulat 4: A priori Information
  Komplexe Muster können in kleine, voneinander abhängige Teile zerlegt werden.

- Postulat 5: Struktur
  Komplexe Muster haben eine spezifische Struktur. Das bedeutet, dass nicht aus jeder beliebigen Kombination von Musterteilen neue Muster entstehen.

- Postulat 6: Ähnlichkeit
  Zwei Muster sind dann ähnlich, wenn die Merkmale der verglichenen Muster im Merkmalsraum eine Position einnehmen, die näher beieinander liegt als die zu anderen Mustern.

3.3.3 Allgemeines Mustererkennungsmodell

In diesem Kapitel wird auf die einzelnen Komponenten des in Abbildung 3.12 dargestellten Modells eingegangen und einen Einblick in deren Struktur geben. Eine umfassende Darstellung ist jedoch nicht möglich und wird bereits mehrfach in einschlägigen Werken erstellt. [RPo06] , [Dud01] , [Nie03]

3.3.3.1 Datenerhebung

In der Datenerhebung werden die Signaldaten in Rohdaten konvertiert, indem sie von einem analogen Signal in ein digitales Signal umgewandelt werden. Für die Datenerhebung sind Sensoren (Filmkamera, Bildscanner, Mikrofon, Beschleunigungssensor, etc.) verantwortlich und stellen damit die erste Instanz der Mustererkennung dar.

Die Datenerhebung findet im gesamten System zu zwei verschiedenen Zwecken statt. Im ersten Fall werden die Trainingsdaten erfasst. Im zweiten Fall liegt die Erkennungsphase vor, bei der die unbekannten zu erkennenden Muster mit den zuvor erfassten Trainingsdaten verglichen werden. Das eigentliche Mustererkennungssystem arbeitet nach der Trainingsphase in der Erkennungsphase.

3.3.3.2 Trainingsphase

In der Trainingsphase werden Trainingsdaten gesammelt, sodass die daraus extrahierten Merkmale so repräsentativ wie möglich für die erwarteten Muster sind. Während des Trainings wird das System soweit trainiert, dass es in der Lage ist, die trainierten Muster später wiederzuerkennen. Problematisch ist die Frage nach der Menge der Trainingsdaten. Einige Quellen raten zu einer konkreten Trainingmenge, wie z. B. für jede Klasse zehnmal soviele Trainingssätze wie Merkmale zu erzeugen [RPo06] . Weitere Quellen raten zu einer problemabhängigen Trainingsdatenerfassung [Nie03] . In einem typischen Mustererkennungssystem sind die Klassen der Trainingsdaten für das nachfolgende Training vorher bekannt. Dies bedeutet, dass es eine Instanz gibt, die dem System während des Trainings sagt, zu welcher Klasse das trainierte Muster gehört. Diese Art des Trainierens benötigt das Wissen des Trainers und wird **überwachtes Training** genannt. **Nicht überwachtes Training** dagegen benötigt kein Wissen über die Klassenzugehörigkeit. Die Anzahl der Klassen muss jedoch bekannt sein. So sucht sich das Trainingssystem aus den vollständigen Trainingsdaten Merkmale heraus und generiert aus den Daten *n* entsprechende Klassen. Der Nachteil dieser Trainingsart ist der nötige Speicherplatz, der vorliegen muss, um alle Trainingsdaten erfassen und vorhalten zu können. Außerdem kann es durch Merkmale, die klassenübergreifend starke Ähnlichkeiten beinhalten, dazu führen, dass das System falsche Klassen generiert. Dies bedeutet, dass später fehlerhafte Klassifizierungen durchgeführt werden. Hierbei ist die Qualität der Trainingsdaten umso wichtiger. Im Anwendungsfall von Bao in [Bao04] ist es von Vorteil, personalisierte Trainingsdaten zu verwenden, da in der Arbeit von Bao personenbezogene Bewegungsmuster trainiert werden. Sollen aber allgemein gültige Muster erkannt werden, ist ein personalisiertes Training hinderlich. Eine selten verwendete Trainingsart ist das **Reinforcement Training**. Es basiert auf einer langen Lernphase, in der der Benutzer dem System auf ein Klassifizierungsergebnis ein „Richtig“ oder „Falsch“ übergibt. Kritiker merken an, dass aufgrund des unspezifischen Feedbacks solche Systeme nur schlecht lernen können [Dud01] . Auf diese Art des Lernens soll in dieser Arbeit nicht weiter eingegangen werden.

3.3.3.3 Vorverarbeitung

Grundsätzlich werden in der Vorverarbeitung Muster in andere Muster konvertiert. Die Kerninformationen der Muster müssen jedoch erhalten bleiben, da sonst das Muster in ein neues unterschiedliches Muster konvertiert wird. Zunächst sollen die Rohdaten in ihrer Qualität verbessert werden. Qualität bedeutet in diesem Zusammenhang, dass die nachfolgenden Arbeitsschritte mit vorverarbeiteten Rohdaten schneller und korrekter abgearbeitet werden können. Während der Datenvorverarbeitung können die Rohdaten durch diverse Filter (z. B. verschiedene Hoch-/und Tiefpassfilter) optimiert werden. Es werden damit bereits unerwünschte Datenbereiche gelöscht, jedoch nicht in dem Maße wie dies in der späteren Merkmalsextraktion geschieht.

Die Vorverarbeitung sorgt neben der Signalglättung und Signalverstärkung für die Mustersegmentierung und kann bereits die Normierung der Daten auf einen gemeinsamen einheitlichen Wertebereich vornehmen. Der Sensorknoten verwendet bereits signalvorverarbeitende Hardware (Hardware-Tiefpassfilter), wie in Kapitel 3.2.1 beschrieben. Folgende Elemente werden oft nicht einzeln erwähnt, lassen sich aber gut aus der Vorverarbeitung extrahieren.

Segmentierung

Die Segmentierung separiert Muster voneinander zu einzelnen abgeschlossenen Mustern und ist eines der komplexesten Probleme der Mustererkennung [Dud01] . Insbesondere in der Bildverarbeitung muss bei der Segmentierung ein Muster von dessen Hintergrund getrennt oder dessen Konturen extrahiert werden. Während der Segmentierung können Muster aus einem kontinuierlichen Signalstrom entnommen werden, oder zerstörte Muster werden bei einer Überlappung voneinander separiert. Sogenannte Untermengen von Mustern können erkannt werden. Ein Beispiel für solche Probleme ist die Spracherkennung und die zu erkennenden Konsonanten. Diese drücken sich zunächst nur durch Stille aus, da sie keine Stimmdynamik haben und somit nur als Vokalübergänge ersichtlich sind. Stille wird aber auch zwischen zwei Sätzen oder Worten zu erwarten sein. Klare Satz- und Wortgrenzen haben also in den Mustern weitere Bedeutungen, was erschwerend auf die Mustersegmentierung wirkt.

Normierung

Bevor Daten der Normierung (*engl. standardize*) unterzogen werden, muss der damit mögliche Informationsverlust erörtert werden. Die Information der Bildgröße oder der Dauer einer Bewegung können verloren gehen. Es kommt hier auf die Aufgabenstellung der Mustererkennung an. Soll die Zeit oder die Bildgröße eine Rolle spielen, müssen die Eckdaten vor der Normierung als Merkmal gesichert werden. In der Normierung wird der Wertebereich auf einen festgelegten Wert beschränkt oder erweitert, damit werden Muster vergleichbar gestaltet. In der Bilderkennung wird oft die Bildgröße aller zu klassifizierenden Bilder normiert, da es wesentlich einfacher ist, Bildinhalte zu vergleichen, deren Randparameter (z. B. Seitenverhältnis oder Auflösung) identisch sind.

3.3.3.4 Merkmale

Merkmale stellen sich in den Rohdaten zunächst als Pixel oder als Messwerte dar. Jedes Pixel oder jeder Messwert ist demnach zunächst ein Merkmal. Die hohe Anzahl von Merkmalen muss komprimiert werden, um ein Mustererkennungssystem nicht dazu zu zwingen, jedes Pixel aller Trainingsbilder mit einem unbekannten Bild vergleichen zu müssen. Demzufolge sollte es verhindert werden, die nicht komprimierten Trainingsdaten verwenden und speichern zu müssen, um sie während der Erkennung schnell zur Verfügung gestellt zu bekommen. Einige Merkmale sind in Tabelle 3.2 aufgeführt.

Merkmalsextraktion

Die Merkmalsextraktion erzeugt aus den vorverarbeiteten Rohdaten charakteristische Merkmale in der Form, dass diese Merkmale die Klassen so exakt wie möglich beschreiben. Die Merkmale müssen innerhalb einer Klasse einander ähneln und müssen sich gegenüber anderen Klassen unterscheiden. Diese wichtige Eigenschaft (siehe Postulat 3 in Kapitel 3.3.2) garantiert, dass sich die Muster anhand der gewählten Merkmale voneinander unterscheiden. Merkmale dieser Art sind unabdinglich, da sie die zentralen musterbeschreibenden Informationen beinhalten und somit die Klasse repräsentieren. Alle anderen Sensordaten werden ab diesem Moment ignoriert. Das bedeutet, dass große Mengen von vorverarbeiteten Rohdaten auf wenige Merkmale reduziert werden.

Tabelle 3.2 Einige Merkmale [RPo06] , [Nie03]

Kernmerkmale:	Statistische Merkmale:
• Größe • Dauer • Gewicht • Länge • Höhe • Fläche • Durchschnittliche Helligkeit/Lautstärke/Amplitude	• Mittelwert • Varianz • Histogramme • Steigung • Diskrete/Schnelle Fourier Transformationen (FFT) • Wavelet Transformationen

Man spricht in diesem Zusammenhang auch von Dimensionsreduktion oder Datenkomprimierung. Der Begriff der Dimensionsreduktion basiert darauf, dass jedes Element des Musters (Pixel, Sample etc.) in der ursprünglichen Form ein Merkmal repräsentiert hat und jedes Merkmal als eine Dimension verstanden wird. Merkmale müssen teilweise invariant gegenüber problemabhängig relevanten Transformationen, wie z. B. Rotation, Skalierung, Farbe und Deformierung, sein. Ein Merkmalsvektor sollte so klein wie möglich und so beschreibend wie möglich sein. Muster werden als Punkte, in dem durch die Merkmale aufgespannten Vektorraum, dargestellt. Das beschränkt das intuitive menschliche Verständnis von Merkmalsvektoren jedoch auf wenige Dimensionen. Der Merkmalsvektor $\underline{x}$ ist wie folgt aus n Merkmalen x_i definiert:

$$\underline{x} = (x_1, x_2, .. x_n) \tag{4}$$

Merkmalsauswahl

Da der Speicherbedarf und der Rechenaufwand mit jedem Merkmal zunimmt und die Gefahr besteht, einen zu großen Dimensionsraum aufzuspannen, ist es nötig, die Merkmalsmenge zu reduzieren und zu optimieren.

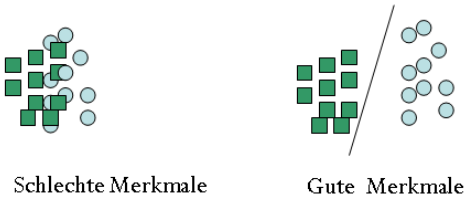

Abbildung 3.17 Gute Merkmale sollten die Klassen deutlich voneinander trennen

Die Merkmalsauswahl gilt als die schwierigste Problematik in der Mustererkennung. Die Merkmalsextraktion extrahiert unter Umständen zu viele vermeintlich gute Merkmale. Ein gut balancierter Merkmalsvektor, der Postulat 3 genügt, ist erforderlich. Merkmale werden manuell oder technisch selektiert.

Eine Teilaufgabe für die Entscheidung für oder gegen ein Merkmal ist eine Signifikanzprüfung. Es muss für die Merkmale festgestellt werden, ob sie signifikant sind oder nicht. Merkmale können mittels Kreuzvalidierung (engl. *cross validation*) auf ihre Signifikanz hin geprüft werden oder sie werden direkt mit einem Klassifikationsverfahren getestet. Für die Mustererkennung werden bei einer Kreuzvalidierung die Trainingsdaten in zwei disjunkte Teilmengen zerlegt. Daraufhin werden alle Merkmale sowohl für die disjunkten Trainingsmengen als auch für die ursprüngliche gesamte Trainingsmenge berechnet. Im nächsten Schritt werden die Merkmale eines Merkmaltyps verglichen und deren Differenzen gesichert. Je kleiner die Differenzen der Merkmale untereinander sind, umso geeigneter ist ein Merkmal für die Klassifizierung.

Bei jeder Änderung der gewählten Merkmale verändern sich jedoch die Signifikanzen der Merkmale untereinander. Dazu ist die einfachste technische Lösung mit bekannten Testdatenreihen und einem Brute-Force-Algorithmus anzuwenden. Alle möglichen Untermengen der Merkmale werden dabei gegen das System getestet. Die Merkmalsvektoren mit der höchsten Erkennungsrate sind daraufhin zu wählen. Diese jedoch sehr rechenintensive Technik lässt sich mit Sensorknoten kaum realisieren.

Andere Algorithmen wie depth-first-search, breath-first-search, branch-and-bound-search sowie hill-climb-search stellen weniger rechenaufwendige Varianten zu dem Brute-Force-Algorithmus dar. Da jedoch die Merkmalsauswahl stark von der Musterproblematik abhängt und diverse statistische Abhängigkeiten zwischen den Merkmalen existieren, ist es nicht möglich, immer einen optimalen Merkmalsvektor algorithmisch zu finden. Mit Hilfe einer in [Nie03] beschriebenen Approximation ist es möglich, einen Merkmalsvektor zu finden, der besser ist als ein Zufallsmerkmalsvektor. Ein Problem stellt die Bewertungseinheit der Merkma-

le dar. Diese Einheit muss vor der Merkmalsauswahl berechnet werden und stellt die Basis für den Qualitätsvergleich der Merkmale dar. Diese sehr rechenintensiven und komplexen Berechnungen lassen sich jedoch durch sequentielle Algorithmen (z. B. „plus-l-take-away-r"-Algorithmus) vereinfachen, die zumindest eine annähernd optimale Lösung erzeugen. Diese Algorithmen sind weiterhin abhängig von der Bewertung der Merkmale [Nie03] .

Das Prinzip der sequentiellen Algorithmen, wie z. B. „plus-l-take-away-r", sei hier zusammengefasst vorgestellt: Ein Merkmalsvektor wird sukzessiv um signifikante Merkmale ergänzt bzw. um nicht signifikante Merkmale reduziert oder es bleiben Merkmale im Vektor enthalten, die als besonders signifikant gelten. Die in dem Algorithmus erforderliche Signifikanzprüfung erfordert das Gegentesten der Vektoren gegen die Trainingsdaten in ihrer Gesamtheit. Jedes Hinzunehmen von Merkmalen beeinflusst dabei erneut die Signifikanz zuvor gewählter oder entfernter Merkmale. Demnach muss die Möglichkeit gegeben sein, auch bereits getroffene Entscheidungen zu revidieren. Grundsätzlich werden Verfahren der sequentiellen Vorwärtsselektion (SFS), beginnend mit einem leeren Merkmalsvektor, mit der sequentiellen Rückwärtsselektion (SBS), beginnend mit einem gefüllten Merkmalsvektor, kombiniert oder separat angewendet. Eine ausführliche Betrachtung verschiedener Algorithmen zur Merkmalsauswahl nimmt Schütze in [Sch05] vor. Diese Algorithmen setzen jedoch das Vorhandensein von der kompletten Trainingsmenge auf den Erkennungssystemen voraus, was aus speichertechnischen Gründen nicht für die Sensorknoten geleistet werden kann. Ein externes Trainingssystem erscheint hier als geeignet, siehe Kapitel 7.1.

Ein weiterer Punkt, der eine automatische Merkmalsauswahl aus einer großen Kollektion von Merkmalen erschwert, besteht darin, dass nicht alle Merkmale gleichermaßen für spezielle Mustertypen geeignet sind. Die Intensität einer Farbe in einem Bild macht beispielsweise bei der Mustersuche nach Formen keinen Sinn. So müssen Merkmale immer problemspezifisch erstellt und ausgewählt werden. Die Entscheidung, welche Merkmalstypen für ein spezifisches Erkennungsproblem von Nutzen sind, wird demnach manuell zu treffen sein, da nur der Anwender den speziellen Unterscheidungsfall kennt. Die automatische Merkmalsauswahl macht folglich erst dann Sinn, wenn ein stets wiederkehrendes gleiches Mustererkennungsproblem behandelt wird und für dieses Problem bereits Merkmalstypen gewählt wurden. [Dud01] , [Nie03]

3.3.3.5 Klassifizierung

Die Klassifizierung verwendet die vorherigen Ergebnisse, um einen neuen, unbekannten Merkmalsvektor einer Klasse zuzuordnen. Durch diesen letzten Schritt wird die Mustererkennung abgeschlossen. Ein Klassifizierer, der den Merkmalsvektor $\underline{x}$ auf eine Klasse k abbildet, wobei k Element aller zugelassenen Klassen K sein muss, wird wie folgt in Gleichung 5 definiert:

$$\underline{x} \rightarrow k\,, \qquad k \in K \tag{5}$$

Statistische Klassifizierung

In der allgemeinen, statistischen Klassifizierung wird vorausgesetzt, dass die jeweiligen Klassen einen zusammenhängenden und geschlossenen Bereich im Merkmalsraum einnehmen. Dabei müssen alle Merkmale Element des Merkmalraumes sein. Es wird hier der einfache Fall angenommen, dass nur zwei Klassen zu betrachten sind. Zwei Klassen C_1 und C_2 werden durch sogenannte Trennfunktionen im hier beispielhaften zweidimensionalen Raum R^2 in Gleichung 6 getrennt.

$$d(\underline{x}) = w_0 + w_1 x_1 + w_2 x_2 \tag{6}$$

Die Koeffizienten w_i (wobei $i > 0$ gilt) sind die jeweiligen zu ermittelnden Gewichte der Merkmale. w_0 entspricht einem Funktionsschwellenwert. Um eine gültige Klassifizierung vornehmen zu können, muss $-w_0$ kleiner oder größer sein als die Summe aller $w_i x_i$. Die Klassifikation mit Hilfe der Trennfunktion (auch Diskriminanzfunktion) geschieht mittels der Klassifikationsregel in Gleichung 7.

$$\underline{x} \in \begin{cases} C_1, & d(\underline{x}) > 0 \\ C_2, & d(\underline{x}) < 0 \end{cases} \tag{7}$$

Für den Fall, dass das Muster direkt auf der Trenngeraden liegt ($d(\underline{x}) = 0$), wird das Muster entweder nicht klassifiziert oder es wird eine Zufallsentscheidung getroffen. Die Trennfunktion kann für beliebig große Merkmalsräume erweitert werden.

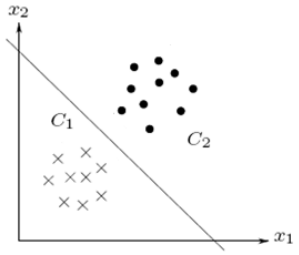

Abbildung 3.18 Lineare Trennfunktion $d(\underline{x})$ trennt C_1 und C_2 [JXi06]

Trennfunktionen können deutlich komplexer als das hier aufgezeigte Beispiel aus Abbildung 3.18 sein und werden vertieft in [Dud01] behandelt.

k-Nearest-Neighbour-Algorithmus (kNN)

Die Klassifikation mit Metriken basiert auf der Möglichkeit, eine Distanzfunktion verwenden zu können, die in der Lage ist, einen vergleichbaren Abstand zwischen zwei Mustern zu ermitteln. Diese Funktion muss für alle vorkommenden Muster definiert sein. Ein Beispiel dieses Klassifizierers ist der kNN. Besonders hervorzuheben sind die Einfachheit und Effizienz dieser Algorithmen. kNN ist ein verbreiteter Algorithmus, da er intuitiv verständlich ist und sehr gute Klassifizierungsergebnisse liefert. Zudem ist die Performanz im Verhältnis zu anderen Systemen bei dem kNN sehr gut. Um das Prinzip der Metriken nutzen zu können, wird jedes Muster-

merkmal als gleichwertig betrachtet und als eine Dimension definiert. Das bedeutet, dass Muster (siehe Abbildung 3.19), welche über zwei Merkmale definiert werden, durch einen zweidimensionalen Merkmalsvektor dargestellt werden.

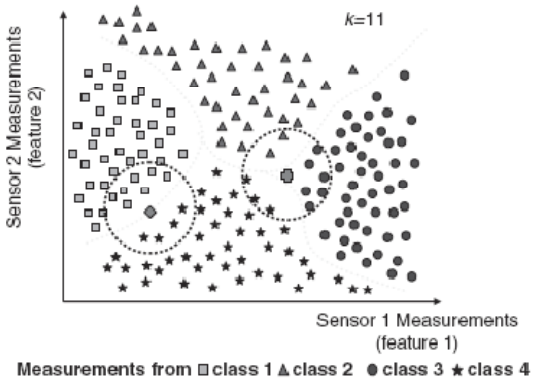

Abbildung 3.19 KNN-Klassifizierung illustriert: Kreuz=3, Raute=1 [RPo06]

Der kNN verwendet zumeist den Euklidischen Abstand als Distanzfunktion, wobei die Distanzfunktionen substituierbar sind. Dies bedeutet, dass der Abstand von allen Trainingsmerkmalsvektoren zu dem zu klassifizierenden Merkmalsvektor berechnet werden muss. Unter den *k* kürzesten Distanzen wird dann eine Mehrheitsentscheidung gefällt. Es werden unter den *k* nächsten Elementen alle Elemente nach ihrer Klasse gruppiert. Die größte Gruppe gibt Auskunft darüber, dass der unbekannte Vektor entsprechend dieser Klasse zuzuordnen ist. Genau diese Zugehörigkeit wird in Abbildung 3.19 dargestellt.

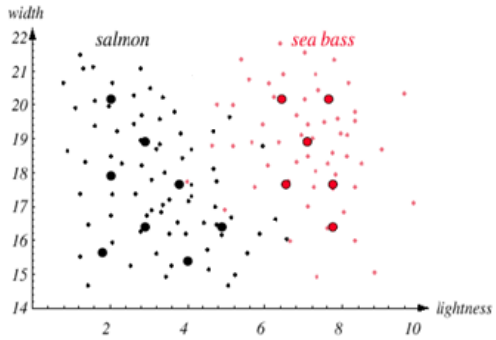

Abbildung 3.20 Verringerung der Trainingsmenge durch Bildung von Repräsentanten (dicke Punkte)verändert nach [Dud01]

Die nächsten Nachbarn sind innerhalb der gestrichelten Kreise zu lokalisieren. Der zu klassifizierende Merkmalsvektor wird der Klasse mit den meisten Anteilen zugewiesen. Der Nachteil der kNN-Algorithmen besteht jedoch darin, dass sie stets alle Trainingsdaten benötigen, um ein Ergebnis liefern zu können. Als Lösung bieten sich Punkteverdichtungen an, wobei überflüssige Elemente aus den Trainingsdaten gelöscht werden, wenn für diese ein Repräsentant existiert, berechnet oder gewählt wurde. Somit bleiben nur noch wenige Repräsentanten übrig, die die Laufzeiten deutlich verbessern und den Speicherbedarf verringern (siehe Abbildung 3.20).

k-Means-Algorithmus

Der k-Means-Algorithmus clustert Objekte in verschiedene Partitionen, sogenannte Voronoizellen. Ähnlich zur Ermittlung von Merkmalsrepräsentanten des kNN reduziert der k-Means-Algorithmus die Trainingsdatenmenge. Die Trainingsmenge wird reduziert, indem aus allen Trainingsdaten *k* Mittelwerte gebildet werden, die als Klassenzentren fungieren. Üblicherweise arbeitet der Algorithmus mit nicht klassifizierten Daten, die mittels eines iterativen Algorithmus die jeweiligen Zentren bilden. Der Algorithmus wählt zunächst zufällig für jede zu findende Klasse gleich viele Trainingsdaten aus.

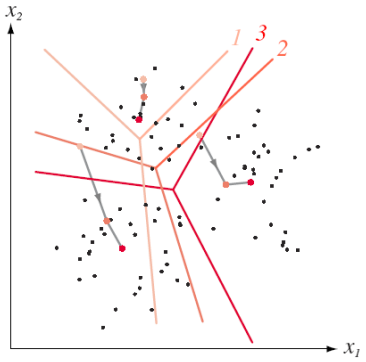

Abbildung 3.21 k-Means bildet hier in drei beispielhaften Iterationen selbständig Voronoi-Cluster für eine 2D-Menge [Dud01]

Daraufhin wiederholt sich der nachfolgende Algorithmus ab Schritt 2 bis keine relevanten Änderungen der Zentrumspositionen (entspricht den Mittelwerten der Voronoizellen) mehr auftreten:

k-Means-Algorithmus:

1. Initialisierung des Systems mit einer beliebigen Zuordnung der Trainingsdaten zu *n* Klassen und beliebige Festlegung der Klassenzentren.
2. Berechnung aller Abstände von allen Trainingsvektoren zu allen Klassenzentren.
3. Zuordnen der Trainingsvektoren zu dem am nächsten liegenden Klassenzentrum.
4. Neuberechnung der Klassenzentren. Sichern der alten Klassenzentren.
5. Wiederholung ab Schritt 2, wenn relevante Änderungen nach der Neuberechnung der Klassenzentren vorliegen, sonst Beendigung des Algorithmus.

Um die neuen Repräsentanten bilden sich Ebenen, die eindeutig nach einer Abstandsfunktion (z. B. dem Euklidischen Abstand) jedem Mittelpunkt alle ihm nächsten Punkte zuordnen, siehe Abbildung 3.21. Die Zuglinien der durch den k-Means ermittelten Mittelwerte zeigen in drei Iterationen, wie eine zweidimensionale Datenmenge nicht überwacht zerlegt wird. Das Voronoi-Mosaik bildet die Klassengrenzen. Die berechneten Mittelwerte entsprechen den „Zentren“ der Voronoizellen.

Entscheidungsbäume

Entscheidungsbäume finden insbesondere in Problemkreisen Anwendung, in denen diskrete, nicht vergleichbare Objekte auftreten, wie z. B. der Medizin, Biologie oder Klimaforschung. Blätter repräsentieren exakt eine Klasse. Die Baumknoten stellen Tests dar, deren eindeutigen Ergebnisse jeweils eine Kante zum nächsten Knoten darstellen.

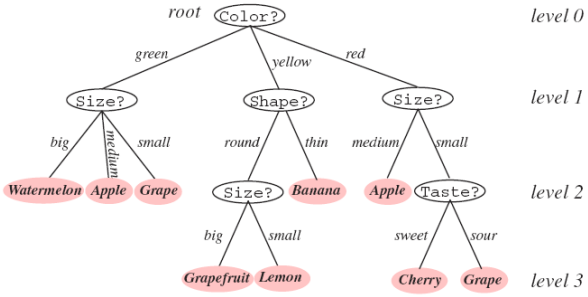

Abbildung 3.22 Entscheidungsbaum für Früchteklassifikation [Dud01]

Überdies kann jeder Entscheidungsbaum, vorausgesetzt, dass jeder Test binär antwortet, in einen binären Entscheidungsbaum überführt werden (siehe Abbildung 3.23). An Übersichtlichkeit wird eingebüßt, dafür hat der Baum an universell gültiger Ausdruckskraft gewonnen und ermöglicht so eine leichte Umsetzbarkeit in Binärsysteme. Die Vorteile liegen in dem selbstdokumentierenden Regelsystem, das von oben nach unten strikt abgearbeitet wird, in der Geschwindigkeit, mit der Entscheidungsbäume abgearbeitet werden können. Zudem können derartige Bäume gut mit Hilfe von Experten erstellt werden, insbesondere dann, wenn die Trainingsdatenmenge gering ist.

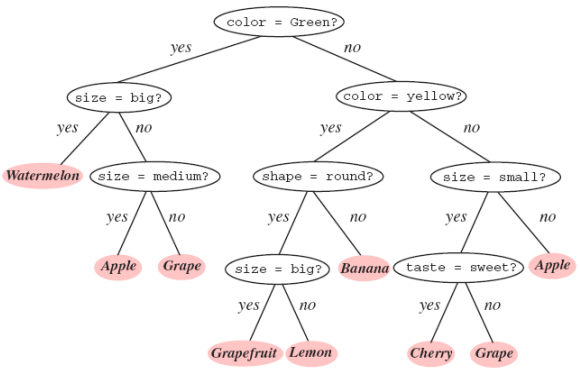

Abbildung 3.23 Binärer Entscheidungsbaum aus Abbildung 3.22 erzeugt [Dud01]

Ein Algorithmus zur automatischen Erstellung eines Entscheidungsbaums aus Trainingsdaten ist z. B. CART (Classification and Regression Trees) [Dud01] . CART bewertet die Attribute des Baumes nach ihrer Stärke. Die Stärke der Attribute ist die Fähigkeit, mit der jedes Attribut in der Lage ist, die möglichen Zielklassen, die für das unbekannte Muster in Frage kommen, weiter einzuschränken. Je mehr Zielklassen durch ein Attribut ausgeschlossen werden können, umso stärker ist es. Sind die Attribute besonders gut in der Lage zu klassifizieren, wandern sie in der Baumhierarchie nach oben. Das heißt, die Entscheidung, die dem potentiellen Betrachter

die größte Unsicherheit nimmt, zu welcher Klasse der Datensatz zuzuordnen ist, definiert die Wurzel des Baumes.

Zurückweisung von nicht eindeutigen Mustern

Es kann vorkommen, dass ein Muster zurückgewiesen werden muss, da es keiner Klasse zugeordnet werden kann. Eine Fehlklassifizierung eines Musters kann aber teurer sein als eine Zurückweisung (Beispiel: Geldzählautomat). Typischerweise werden Klassen definiert, die als Rückweisungsklasse fungieren. Außerdem definieren sogenannte Klassengrenzen klar die maximalen Bereiche der Klassen. Beispielsweise müsste bei dem kNN-Algorithmus eine maximale Distanz von den Referenzvektoren zu neuen, zu klassifizierenden Vektoren festgelegt werden, die ein Merkmalsvektor haben darf, um klassifiziert werden zu dürfen. In Abbildung 3.24 ist ein solcher Fall exemplarisch hervorgehoben. Es wird das Problem der zu fällenden Entscheidung bezüglich der Zuordnung eines Merkmalvektors zu einer Klasse oder einer Zurückweisung verdeutlicht.

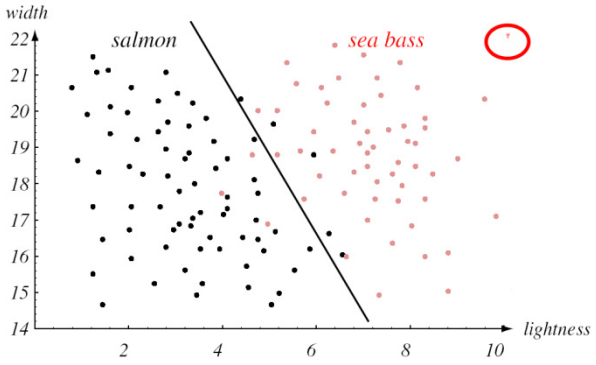

Abbildung 3.24 Ein Außenseiter (rot umrandet) muss korrekt zugeordnet oder zurückgewiesen werden, geändert nach [Dud01]

Ein interessanter Ansatz zur Musterzurückweisung ist die Verwendung von mehreren Klassifizierern in [Mat06] . Die Arbeit vergleicht die Ergebnisse mehrerer Klassifizierer miteinander. Es wird eine Entscheidung getroffen, die von der Mehrheit der Klassifizierungen abhängt und dann die Möglichkeit bietet, einen weiteren angepassten Klassifizierungsprozess zu starten.

Neuronale Netze

In [Bao04] wird beschrieben, dass der Großteil der Mustererkennungsalgorithmen auf Basis der Statistik aufgebaut wird. Alle anderen Algorithmen ordnen sich zumeist dem Bereich der neuronalen Netze zu. Nach [Bao04] ist es möglich, eine direkte Abbildung von Mustererkennungssystemen der neuronalen Netze auf statistische Lösungen zu entwickeln. Neuronale Netze beschreiben einen großen Forschungsbereich, der in der Mustererkennung z. B. durch Ferber in [Fer99] gut beschrieben wird. In dieser Arbeit wird nicht weiter auf neuronale Netze eingegangen.

3.3.3.6 Evaluation der Modell-Komponenten

Jede Komponente des Standardmodells trägt im System dazu bei, das ursprünglich unbekannte Muster zu klassifizieren und dabei die zugrundeliegenden Daten zu verarbeiten. Es ist daher interessant zu wissen, wie gut die einzelnen Komponenten arbeiten. Eine Evaluation der Elemente ist in sofern informativ, dass der Erfolg der Klassifizierung letztendlich von jedem einzelnen Element abhängt. Die Komponenten des Modells beeinflussen sich jedoch gegenseitig und lassen sich nur schwer gesondert betrachten und bewerten. Aus diesem Grund ist eine Evaluation der einzelnen Komponenten kaum möglich. Ein wichtiges Kriterium ist die subjektive Evaluation der Qualität jeder einzelnen Datentransformation, indem die jeweiligen Daten visuell, akustisch oder statistisch geprüft werden. Derzeit ist nicht bekannt, wie einzelne Komponenten evaluiert werden können, ohne dass dies für das gesamte System vorgenommen werden muss. Damit schwindet die Aussagekraft der Evaluierungsergebnisse einer Komponente, da sie nicht mehr alleine für die Ergebnisse verantwortlich sind. Der Aufwand für eine tatsächlich aussagekräftige Evaluation der einzelnen Komponente ist somit nicht abschätzbar [Nie03] .

3.4 Modell der verteilten Mustererkennung

> *„Ein wesentlicher Nachteil von Einzelsensoren ist nämlich die Unfähigkeit, die inhärente Unsicherheit bei der Interpretation des Sensorsignals […] zu beheben. Mit der Fusion von Informationen aus verschiedenen Quellen können die Wahrnehmungsfähigkeiten erweitert und die Glaubwürdigkeit der Sensoraussage erhöht werden.“* [Rus07] .

Anhand eines Fusionsmodells sollen in diesem Kapitel die Idee der Datenfusion erläutert und mögliche Herangehensweisen zu Datenfusionen angedeutet werden.

3.4.1 Abstraktionsebenen der Datenfusion

Datenfusionen können in verschiedenen Ebenen eines Mustererkennungssystems vorgenommen werden. Typischerweise ergeben sich drei Abstraktionsebenen, auf denen agiert werden kann. Üblicherweise werden Probleme zumeist innerhalb einer Abstraktionsebene gelöst. Es ist jedoch durchaus möglich, ebenenübergreifend Datenfusionslösungen zu entwickeln. Allgemein gilt, je höher die Abstraktionsebene, desto weniger Daten müssen ausgetauscht werden, woraufhin die Effizienz des Systems aus Sicht der Kommunikationslast maximiert wird. Die Reduktion der Informationsdichte kann dazu führen, dass die Präzision der verteilen Mustererkennung mittels Datenfusion verschlechtert wird. Je tiefer die Abstraktionsschicht gewählt wird, desto umfangreicher werden die Datenmengen und desto präzisere Erkennungsaussagen werden mit der Datenfusion möglich. Damit wird die Effizienz des Systems aus der Sicht der Erkennungsrate maximiert [Rus07] .

- Signalebene
  Auf der Signalebene werden direkt die von den Sensoren gemessenen Messwerte ausgetauscht und verarbeitet. Dabei ist zu beachten, dass deren Vergleichbarkeit gewährleis-

tet ist und insbesondere die Synchronisation aller beteiligten Sensoren gesichert sein muss [Rus07] .
Es kann zunächst ausgeschlossen werden, die Signalebene als verwertbare Abstraktionsebene zu verwenden, da der Kommunikationsaufwand hierfür deutlich zu hoch ist.

- Merkmalsebene
  Auf der Merkmalsebene werden Mustermerkmale ausgetauscht und fusioniert. Mustermerkmale lassen sich in Vektoren darstellen und ergeben somit die Möglichkeit, eine aus technischer Sicht skalierbare Musterbeschreibung aus dem Signal zu generieren. Die Skalierbarkeit der Vektorgröße beeinflusst die Erkennungsqualität, erlaubt aber im gleichen Zug, Mustererkennungssysteme präzise auszulasten. Die Untersuchung der Merkmalsebene sollte dann in Betracht gezogen werden, wenn zeitliche Zusammenhänge der erfassten Daten nicht präzise betrachtet werden können [Rus07] .

- Klassifikationsebene
  Die Klassifikationsebene liefert Musterklassifikationsergebnisse und berechnet mittels entsprechend zugeordneter Wahrscheinlichkeiten das Gesamtergebnis. Die Klassifikationsebene erhält für die Datenfusion die Wahrscheinlichkeiten aus Umweltinformationen, z. B. die Entfernung der Sensoren zum Ereignis oder Aprioriinformationen. Zudem können dynamische Informationen die Wahrscheinlichkeiten, mit der die Klassifikationsergebnisse einzelner Sensoren in die Fusionierung eingehen, beeinflussen [Rus07] .

3.4.2 Fusionsmodell „Omnibus“

In Abbildung 3.25 wird das Fusionsmodell „Omnibus“ vorgestellt. Das Omnibusmodell stellt den typischen Ablauf eines Mustererkennungsszenarios in Zusammenarbeit mit einer verteilten Entscheidungsfindung dar.

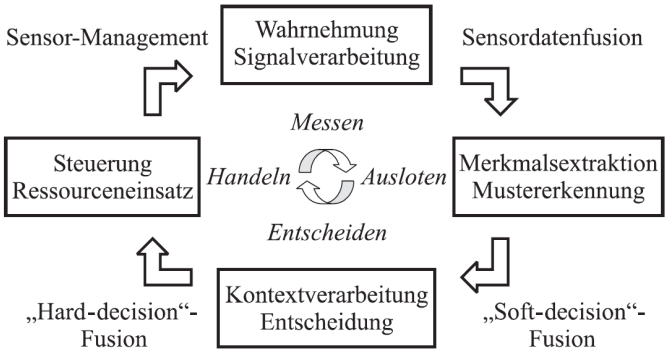

Abbildung 3.25 Fusionsmodell "Omnibus" stellt sich als Regelkreis dar [Rus07]

Das Omnibusmodell führt Transformationsschritte (Messen, Ausloten, Entscheiden, Handeln) durch, um aus gemessenen Daten, wie in der nicht verteilten Mustererkennung, eine Schlussfolgerung ziehen zu können. Zwischen den Transformationsschritten werden Schnittstellen

definiert (Sensordatenfusion, „**Soft-decision**“-Fusion, „**Hard-decision**“-Fusion, Sensormanagement), an denen Daten abgefasst und mit denen anderer Komponenten des Systems fusioniert werden können. Die Transformationsschritte korrespondieren mit den Schnittstellen und erlauben es somit, sich auf andere ähnlich modular aufgebaute Systeme übertragen zu lassen. Das Omnibusmodell verwendet im Gegensatz zu anderen Arbeiten ([Bro03] , [Kok01]) zudem die verfeinerte „Decision“-Fusion, die zwischen „Hard“- und „Soft-decision“ unterscheidet. Genau diese Verfeinerung ist der entscheidende Ansatz, der sich im Folgenden als sehr geeignet erweist.

Die entscheidenden Module der Transformationsschritte (Wahrnehmung + Signalverarbeitung, Merkmalsextraktion + Mustererkennung, Kontextverarbeitung + Entscheidung) sind identisch mit den Modulen des Standardmustererkennungsmodells aus Abbildung 3.12 und lassen sich direkt abbilden. Im Standardmodell lassen sich die Module der „Datenerhebung“ und „Vorverarbeitung“ auf das Modul „Wahrnehmung Signalverarbeitung“ des Omnibusmodells abbilden. Das Modul „Merkmale“ des Standardmodells wird auf „Merkmalsextraktion Mustererkennung“ abgebildet und die „Klassifizierung“ auf „Kontextverarbeitung Entscheidung“ des Omnibusmodells.

Das Ziel, eine erhöhte Mustererkennungsrate mittels einer verteilten Mustererkennung zu erreichen, soll mit der Fusion von Merkmalen aus der Merkmalsebene und der Fusion von Klassifikationsergebnissen aus der Klassifikationsebene erreicht werden. Die erhöhten Abstraktionsgrade reduzieren die Präzision der Erkennung, bedingt durch verlustbehaftete Datenkompressionen. Es wird dennoch insgesamt von einer Verbesserung der Klassifikationsrate im Vergleich zur lokalen Mustererkennung ausgegangen. Die qualitative Bewertung der Fusionsebenen soll in Tabelle 3.3 zusammengefasst dargestellt werden und die Entscheidung der Wahl der Merkmalsebene und der Klassifikationsebene als Untersuchungsobjekt unterstreichen. Die Merkmalsebene sowie die Klassifikationsebene bieten sich als Fusionsebenen an, da durch sie die ursprüngliche Datenmenge reduziert wird.

Tabelle 3.3 Gegenüberstellung von klassischen Fusionsebenen [Rus07]

Fusionsebene	Signalebene	Merkmalsebene	Klassifikationsebene
Art der zu fusionierenden Daten	Signale, Messdaten	Signaldeskriptoren	Symbole, Objekte, Entscheidungen, Äquivalenzklassen
Zielsetzung	Signalschätzung, Parameterschätzung	Merkmalsschätzung, Parameterschätzung	Klassifikation, Mustererkennung
Abstraktionsgrad	niedrig	mittel	hoch
Geeignete Datenmodelle	Zufallsprozesse, Zufallsvariablen	Zufallsvektoren	Wahrscheinlichkeitsverteilung
Voraussetzungen für die Fusion (räumlich/zeitlich)	Registrierung/ Synchronisation	Zuordnung der Merkmale	Zuordnung der Symbole
Typische Komplexität	hoch	mittel	gering

Auf eine Synchronisierung kann bei der Merkmals- und Klassifikationsfusion verzichtet werden. Die Komplexitäten der Merkmals- und Klassifikationsebene sind geringer als die der Signalebene. Aufgrund eines erhöhten Abstraktionsgrades ist eine Verbesserung der Erkennung und eine Senkung der Kommunikation auf der Merkmals- und Klassifikationsebene zu vermuten.

3.4.3 Sensorintegration und Datenfusion

Die Sensorintegration beschreibt den synergetischen Einsatz mehrerer Sensoren für Klassifikations-, Detektions- oder Identifikationsaufgaben. Daraus ergeben sich verschiedene Fusionskonzepte, bei denen die Daten der Sensoren miteinander kombiniert werden können. Die Fusionskonzepte sind in Abbildung 3.26 aufgezeigt und stellen sich in drei Grundtypen dar:

- Konkurrierende Integration
  Hier werden vergleichbare Sensordaten miteinander fusioniert, um deren Aussagekraft zu überprüfen und zu relativieren. Die Daten müssen unter nahezu identischen Bedingungen erfasst werden und müssen dieselben Nutzinformationen enthalten. Dieses Konzept wird u.a. zur Rauschunterdrückung verwendet. Typische Ansätze dafür sind Mittelwertbildungen. [Rus07]

- Komplementäre Integration
  Hier werden vergleichbare Sensordaten miteinander fusioniert, um deren Aussagekraft zu überprüfen und zu relativieren. Die Daten müssen nicht mehr unter 100 % identischen Bedingungen erfasst werden und sollten unterschiedliche Nutzinformationen enthalten. Dieses Konzept wird unter anderem dazu verwendet, um ergänzende Infor-

mationen zu sammeln, die im Zusammenhang eine größere Bandbreite an Informationen zur Verfügung stellt. [Rus07]

- Kooperative Integration
  Dieses Konzept erlaubt eine Aussage über die erfassten Daten mittels einer kooperativen Auswertung, bei der alle erfassten Sensordaten im Zusammenhang und gemeinsam ausgewertet werden können. Dieses Konzept sollte nur dann verwendet werden, wenn tatsächlich keine Möglichkeit zur Verwendung einer höheren Abstraktionsebene gegeben ist [Rus07] . Zudem ist es für Sensornetze aus Kommunikations- und Rechenlastgründen sinnvoll, auf die kooperative Integration zu verzichten. Die Arbeit von Brooks [Bro03] betrachtet diesen Punkt detaillierter und unterstreicht diese Aussage.

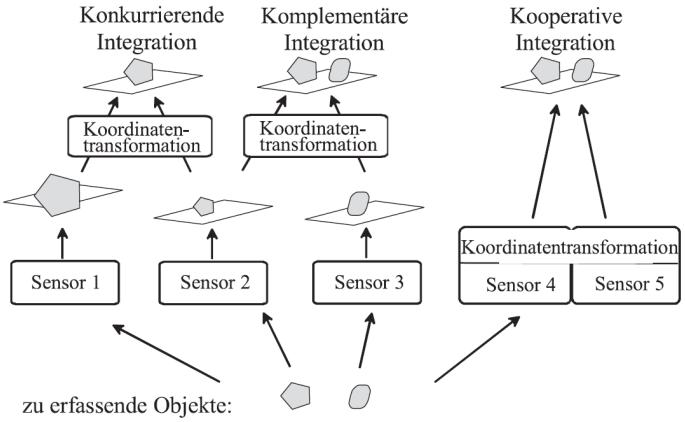

Abbildung 3.26 Grundlegende Fusionskonzepte [Rus07]

Die „Komplementäre Integration“ lässt es zu, die „Konkurrierende Integration“ darzustellen, indem gleiche Informationen so betrachtet werden, als enthielten sie unterschiedliche Nutzinformationen. Damit ist der Ansatz der „Komplementären Integration“ generell zu bevorzugen, da sich beide Konzepte mit der „Konkurrierenden Integration“ darstellen und lösen lassen. Der in dieser Arbeit verwendete Ansatz ist demnach die Implementierung der „Komplementären Integration“.

4 LOKALE MUSTERERKENNUNG

Ziel der Arbeit ist es, eine Bewegungsmustererkennung in einem verteilten Sensornetz zu realisieren. Dazu muss jeder Sensorknoten im Netzwerk in der Lage sein, Muster zu erkennen. Aus diesem Grund liegt im ersten Schritt der Schwerpunkt auf der Mustererkennung eines einzelnen Sensors. Die lokale Mustererkennung ist unabhängig vom Gesamtsystem in der Lage, Muster zu erkennen und zu klassifizieren. Dies wird als Voraussetzung für das gesamte System benötigt, da der Funkverkehr innerhalb des Sensornetzes und zur Basisstation so gering wie möglich gehalten werden soll, um die gewünschte hohe Lebensdauer des Sensornetzes beizubehalten. Um den Funkverkehr innerhalb des Sensornetzes zu reduzieren, werden nur noch die komprimierten Informationen (Merkmale oder Klassifizierung) eines Ereignisses an den nächsten Sensorknoten versendet, deren Berechnung in den folgenden Kapiteln aufgezeigt wird. Es sollen nur dann Ereignisse kommuniziert werden, wenn diese vom Sensorknoten selber als relevant angesehen werden. Die Erweiterung des Systems dahingehend, dass verteilte Erkennungsprobleme durch die Verarbeitung der so versendbaren komprimierten Informationen gelöst werden können, folgt in Kapitel 5 aufbauend auf der hier vorgestellten Lösung der lokalen Mustererkennung.

4.1 Versuchsanordnung

Es muss dem Sensorknoten nach der Kalibrierung eine zu definierende Anzahl von Mustern antrainiert werden. Jedes Muster muss mit einer wiederum vom Anwender festgelegten Anzahl von Wiederholungen in das System eingehen. Dazu wird der Sensorknoten üblicherweise für das erste Muster *n* mal auf die gleiche Weise bewegt.

Für die konkrete Versuchsanordnung werden vier geometrische Muster ausgewählt (Kreis, Quadrat, Dreieck und Bogen), die je zehnmal trainiert werden. Diese kleine Menge an Trainingsdaten soll genügen, um immer wiederkehrende Funktionsüberprüfung mit adäquatem Aufwand durchführen zu können.

Die Anzahl der Muster ist im System variabel festlegbar, wird hier jedoch auf vier beschränkt um den Testaufwand während der Erkennungstests zu begrenzen. Die Wahl der in Abbildung 4.1 aufgezeigten Muster resultiert aus ihrer Einfachheit und Nachvollziehbarkeit. Zudem soll überprüft werden, ob sich Strukturähnlichkeiten der Muster auf die Erkennungsrate auswirken. Die angesprochenen Strukturähnlichkeiten beziehen sich auf die Ecken in Dreieck und Viereck und die Rundungen in Kreis und Bogen.

Ein Muster beginnt für das System durch Bewegung und endet durch die Ruheposition. Die Ruheposition wird mittels der Kalibrierung festgelegt und wird in Kapitel 4.3.1 erläutert. Eine Ruheposition muss sich jedoch nicht mit der subjektiv empfundenen „Ruhe“ decken. Es ist jedoch nötig, dass die Ruheposition weniger Beschleunigungen verursacht als die Bewegung.

Das Erkennungssystem arbeitet mit einer Samplingfrequenz von 50 Hz. Höhere Samplingfrequenzen erreichen nicht die gewünschte Qualität der Mustererkennung und werden in Kapi-

tel 6.1 besprochen. Für die Erkennung wird ein Bewegungsmuster erzeugt, bei dem eine gewisse Grundgeschwindigkeit der Muster erreicht werden muss. Es sollen keine Pausen oder auffällige Beschleunigungswechsel während der Bewegung vollzogen werden. Eine gleichmäßige Bewegung, bei der das Muster in einem Durchgang gezeichnet wird, soll trainiert werden.

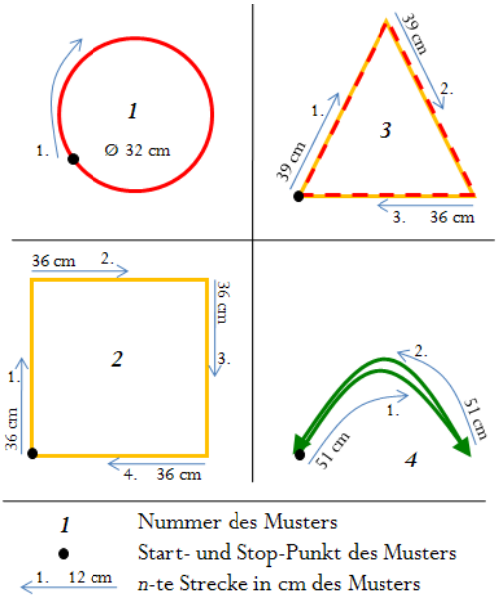

Abbildung 4.1 Musterdefinitionen, Farben entsprechen den LED's aus Tabelle 4.1

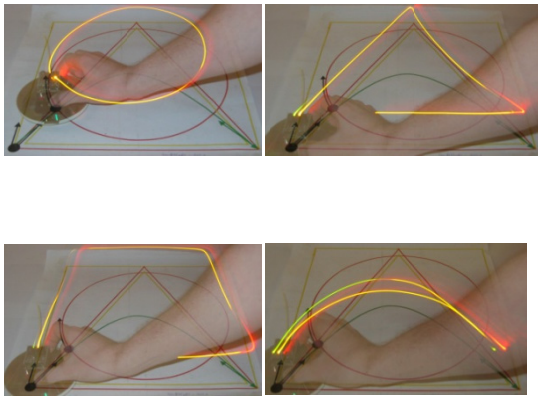

Abbildung 4.2 Ausführung der Musterbewegungen auf gedruckter Vorlage

Die erfolgreiche Aufnahme eines Musters wird durch farbliche LED's und ein akustisches Signal bestätigt (siehe Tabelle 4.1). Im Fall einer Fehlaufnahme wird kein Bestätigungssignal ausgegeben. Eine Fehlaufnahme ist ein erzeugtes Muster, das länger als die voreingestellte, maximale Musterdauer ist. Das somit nicht erfolgreich trainierte Muster muss wiederholt trainiert werden. Bei Vollendung der Trainingsphase leuchten alle LED's ohne akustisches Signal. Die geometrischen Figuren werden in Abbildung 4.1 dargestellt und die Bewegungsabläufe in Abbildung 4.2 kenntlich gemacht.

Tabelle 4.1 Sensorknotenrückmeldungen

Muster/Aktion	LED bei Erfolg	akustisches Signal
Kalibrierung	(rot) blinkt bis fertig	-
Kreis	(rot)	Piep
Viereck	(gelb)	Piep
Dreieck	(rot, gelb)	Piep
Bogen	(grün)	Piep
Fehlaufnahme	-	-
aktuelles Muster trainiert	(rot, gelb, grün)	-
Training komplett beendet	(rot, gelb, grün)	-

Die Erkennung wird nach dem Training mit denselben Mustern vorgenommen. In beliebiger vorher nicht geplanter Reihenfolge werden Muster nach den Vorgaben der Abbildung 4.1 mit dem Sensorknoten auf einer entsprechenden Druckvorlage nachgezeichnet. Der Sensor reagiert wieder wie in Tabelle 4.1 auf jedes erkannte Muster mit einem akustischen Signal in Kombination mit einer aufleuchtenden LED.

4.2 Gewichteter überwachter k-Means-Algorithmus

Ein Sensorknoten muss alle nötigen Informationen für eine Klassifizierung speichern, wenn er ohne Kommunikation mit einer Basisstation eine Klassifizierung der eintreffenden Muster durchführen soll. Für eine typische Klassifizierung werden die gespeicherten Trainingsdaten mit den neu eintreffenden Mustern verglichen. Der Speicherplatz für eine derartige Lösung ist auf den Sensorknoten jedoch nicht verfügbar. Dieses grundlegende Problem muss gelöst werden, um mit Sensorknoten eine lokale Entscheidung über ein eintreffendes Muster fällen zu können.

Der verwendete, auf einer beliebigen Metrik basierende Algorithmus „gewichteter überwachter k-Means-Algorithmus“ ist bereits in [Kal01] formal beschrieben und wird dort als „Prototype Modeler“ bezeichnet. Er stellt einen Spezialfall des k-Means-Algorithmus dar, der üblicherweise mit einem nicht überwachten Training arbeitet und wird in Kapitel 3.3.3.5 angesprochen.

Der überwachte k-Means-Algorithmus führt durch die vorherige Klassenbestimmung der Trainingsdaten (überwachtes Training) zu einer Verringerung der Trainingsdatenmenge. Es müssen nicht mehr alle Daten auf dem System gespeichert werden, sondern nur die Merkmale, die aus den jeweiligen Trainingsdaten extrahiert werden müssen. Am Ende des Trainings wird mit diesen Daten die Mittelwertbildung durchgeführt, die wiederum den Datenumfang reduziert. Aufgrund der bekannten Zugehörigkeit der Trainingsdaten zu den jeweiligen Klassen durch das überwachte Training ist es nicht mehr nötig, mehrere Iterationen durchzuführen wie bei dem k-Means, sondern es genügt exakt eine Berechnung, die für jede Klasse einen Mittelwert als Prototypen ermittelt.

Im Rahmen dieser Arbeit bedeutet dies, dass pro Merkmal und Klasse eine Referenzzahl, das sogenannte Referenzmerkmal, erzeugt wird. Alle genutzten Merkmale einer Klasse werden als Vektor definiert und ergeben dann einen Mittelpunkt (Prototypen) bzw. Referenzvektor dieser Klasse im m-dimensionalen Raum, wobei m der Anzahl der Merkmale entspricht. Dieser Ansatz widerspricht der Empfehlung für ein Mustererkennungssystem, mit möglichst umfangreichen Trainingsdaten zu arbeiten. Dennoch ist es aufgrund der technischen Bedingungen als Ansatz für eingebettete Systeme vertretbar, die Datenmenge so weit wie möglich zu reduzieren und entsprechend diesen effizienten Algorithmus zu wählen, der ein geringes Datenvolumen produziert.

Im späteren Verlauf sollen die trainierten Referenzvektoren mit neuen, unbekannten Mustern verglichen werden. Dazu werden aus den unbekannten Mustern wiederum dieselben Merkmale wie für die Trainingsdaten berechnet. Um unbekannte Muster mit den gespeicherten Trainingsvektoren vergleichen zu können, muss der Abstand des Merkmalsvektors *xv* zu dem Referenzvektor *rv* berechnet werden. Die Größe des Abstandes entscheidet über eine mögliche Zugehörigkeit des Musters zu einer Klasse. Üblicherweise werden die Abstände zu allen trainierten Klassen berechnet und der kürzeste ausgewählt, um die Zugehörigkeit damit festzulegen. Die Vektorabstände können mit verschiedenen Abstandsfunktionen $d(xv, rv)$ berechnet werden. Mittels des häufig verwendeten Euklidischen Abstandes [Dic98] werden in dieser Arbeit die Abstände von diesen m-dimensionalen Vektoren in dem von ihnen aufgespannten Vektorraum ermittelt, siehe Gleichung 8. Der Euklidische Abstand ist eine einfache gut auf den Sensorknoten umsetzbare Metrik und bietet sich daher für die nötige Abstandsberechnung an.

$$d(xv, rv) = \left||xv - rv|\right| = \sqrt{\sum_{i=1}^{m} (xv_i - rv_i)^2} \tag{8}$$

Eine Optimierung der Abstandsberechnung in Bezug auf den Rechenaufwand der Vektoren mit dem Euklidischen Abstand wird erzielt, wenn der quadratische Euklidische Abstand verwendet wird. Bei dieser Berechnung fällt das abschließende Wurzelziehen weg. Dies ist möglich, da der Euklidische Abstand eine stetig monoton steigende Funktion ist und für die Vektoren der Mustererkennung nicht der konkrete sondern nur der relative Abstand interessant ist. Die Information, welcher Vektor weiter entfernt liegt als ein anderer, bleibt somit erhalten.

Da jedes Merkmal durch den Euklidischen Abstand gleichgewichtet bewertet wird, ist korrektes Vergleichen von Vektoren nicht möglich, wenn die einzelnen Dimensionen verschiedenen Gewichtungen unterliegen. Die einzelnen Merkmale haben nahezu alle unterschiedliche Gewichtungen. Dies bedeutet, dass der Merkmalseinfluss auf den Abstand zwischen zwei Vektoren verschieden intensiv ausgeprägt ist (siehe Abbildung 4.3).

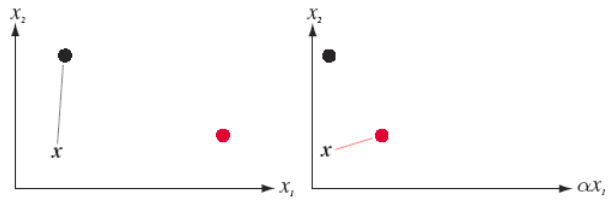

Abbildung 4.3 Skalierungsunterschiede der Achsen wirken sich dramatisch auf Euklidische Abstände aus. x_1 wird um den Faktor $\alpha = 1/3$ reskaliert und ändert damit den nächsten Nachbarn [Dud01] .

Die verschiedenen Merkmale sind demnach nicht vergleichbar. Für die Lösung dieses Problems wird in der Literatur die Merkmalsnormierung empfohlen, siehe [Kal01] , [Ver05] . Auf die so zu bezeichnende Merkmalsnormierung wird in der Komponente „Merkmalsextraktion“ in Kapitel 4.4.6 eingegangen.

4.3 Modellphasen

Die Modellphasen gliedern sich in drei getrennte Phasen auf, siehe Abbildung 4.4. In der ersten Phase wird der Sensorknoten kalibriert, wobei der Sensorknoten eine der Aufgabenstellung entsprechende Ruheposition eingenommen haben muss. Die zweite Phase trainiert den Sensorknoten auf die zu erkennenden Muster. Basierend auf die in der Trainingsphase gewonnenen Daten wird in der dritten Phase (Erkennungsphase) die eigentliche Mustererkennung durchgeführt.

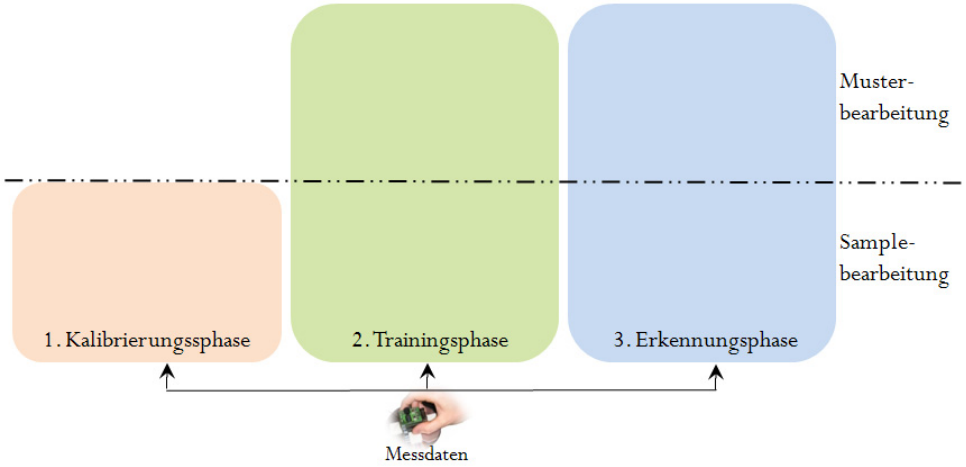

Abbildung 4.4 Phasenmodell mit den verschiedenen Bearbeitungsschichten

Das Phasenmodell wird in der Trainingsphase und der Erkennungsphase viele Softwarekomponenten gemeinsam nutzen und damit den Prozess auf die gleiche Weise durchlaufen. Die Eigenschaft des Systems, für Training und Erkennung dieselben Routinen zu verwenden, beruht auf der Entwurfsentscheidung, dass ein Distanzsystem den Vergleich der Merkmalsvektoren ermöglicht. Für die Berechnung der Distanzen sind in beiden Phasen Merkmale zu erstellen. Insofern wird ein System implementiert, das sowohl während des Trainings als auch während der Erkennung eine identische Datenfilterung, Datenkonvertierung und Merkmalsbildung durchführt, um dadurch die Berechnung der Distanzen zu ermöglichen.

4.3.1 Kalibrierungsphase

Ohne eine korrekte Kalibrierung sind die ermittelten Daten nicht verwertbar. Eine gute Kalibrierung garantiert, dass so wenig wie möglich Daten fehlerhaft sind. Die Daten sollen dem hier betrachteten Fall der Beschleunigung sowie dem tatsächlichen Verlauf einer Bewegung entsprechen, siehe Kapitel 3.2.2 [Blo06] .

Der Beschleunigungssensor der Reihe MMA73x0 hat von Werk aus für den 0 g-Zustand Spannungsschwankungen von ca. 30 bis 120 mV/g. Mögliche Einflussgrößen auf die Systemkomponenten sind das Alter, die Temperatur und Abweichungen innerhalb der Bauspezifikation bedingt durch Montage und Verpackung. All diese Größen können Schwankungen der Ausrichtung der Sensorknoten in einem Winkel von bis zu 12° ausmachen. Dieses Ausmaß an Schwan-

kung ist nicht akzeptabel und fordert somit eine Implementierung für eine automatische 0 g-Kalibrierung. Neben den genannten technisch bedingten Schwankungen wirkt die Erdanziehungskraft als Umweltfaktor auf die Kalibrierung. Die Erdanziehungskraft wirkt unmittelbar und zu jeder Zeit auf den Beschleunigungssensor. Dies bedeutet, selbst wenn der Sensorknoten flach und in Ruhe auf dem Tisch liegt, wirkt bereits 1 g auf die Z-Achse. Abhängig von der Lage des Sensors wirkt die Erdanziehungskraft auf verschiedene Achsen. Die Z-Achse sei hier exemplarisch herausgestellt. Zusätzlich unterliegen alle Achsen den oben genannten Schwankungen.

4.3.2 Trainingsphase

Der erste Schritt der Mustererkennung ist das Training. Das Ziel des Trainings ist das Lernen von Klasseninformationen, die die Mustertypen eindeutig repräsentieren. Technisch werden Referenzmerkmale für die zu erkennenden Musterklassen generiert. Während des Trainings werden Daten verarbeitet, die mittels des Beschleunigungssensors gemessen werden. Die Daten werden direkt auf dem Sensorknoten erfasst und dort in Referenzparameter umgerechnet. In diesem konkreten Fall besteht das Ziel in der Erzeugung der in Kapitel 4.4.6 beschriebenen normierten (gewichteten) Referenzvektoren. Das überwachte Training führt dazu, dass der zu verwendende Datenbestand gering bleibt und das Training lokal auf dem Sensorknoten möglich ist.

4.3.3 Erkennungsphase

Das Ziel der Mustererkennung ist die Klassifizierung eines unbekannten Musters. Die Erkennungsphase beginnt nach der Trainingsphase und verwendet bis auf die Klassengenerierung und Merkmalsauswahl dieselben Komponenten wie die Trainingsphase. Die Klassengenerierung wird in der Erkennungsphase durch die Klassifizierung ersetzt und stellt in der Erkennungsphase das erkennende Pendant zu Klassengenerierung der Trainingsphase dar. Die Merkmalsauswahl hingegen findet in der Erkennungsphase keine Anwendung, da die während der Trainingsphase selektierten Merkmale hier direkt extrahiert werden und eine weitere Auswahl unnötig ist. Zur Erkennung von Mustern werden in der Erkennungsphase auf die in der Trainingsphase gewonnen Trainingsvektoren zurückgegriffen und diese entsprechend dem im Kapitel 4.2 beschriebenem Verfahren zum Vergleich herangezogen.

4.3.4 Datenbearbeitung

Das gesamte System durchläuft während der Trainingsphase und der Erkennungsphase zwei Bearbeitungsstufen, die **Samplebearbeitung** und die **Musterbearbeitung**, siehe Abbildung 4.1. In der Samplebearbeitung wird jeder neu gemessene Datensatz, das so genannte **Sample,** einzeln in jeder Komponente bearbeitet. Ein Sample entspricht einem Tripel bestehend aus den Beschleunigungswerten für jede der drei Achsen des Sensors. Die Phase der Samplebearbeitung erfasst das Muster sampleweise und speichert es im Sensorknoten ab. Es werden Informationen

von direkt vorangegangenen Samples hinzugezogen, um zu entscheiden, ob sich das in der Erfassung befindende Muster in einem Bewegungszustand oder einem Stopzustand befindet.

In der Kalibrierungsphase muss die Stufe der Samplebearbeitung durchlaufen werden, da in der Komponente „Kalibrierung“ nur die in Kapitel 4.4.1 beschriebenen Sampleeigenschaften betrachtet werden und nicht die Mustereigenschaften.

Während der Musterbearbeitung wird in jeder Komponente das erfasste Muster und damit die Gesamtheit aller dem Muster zugehörigen Samples betrachtet und bearbeitet. Ein Muster besteht aus den gesammelten Tripel der Beschleunigungsdaten. Diese Daten werden in der Musterbearbeitung auf die Mustermerkmale reduziert und ermöglichen somit den effizienten Umgang mit Mustern in der Klassengenerierungs- und Klassifizierungskomponente.

Während der Musterbearbeitung selber können keine neuen Samples und Muster erfasst werden. Die Bearbeitung des Musters benötigt die volle Kapazität des Sensors und setzt für diese kurze Zeit die Samplebearbeitung außer Kraft.

4.4 Modellkomponenten

Für den verwendeten Ansatz muss das System laut Abbildung 3.12 leicht modifiziert werden, da die Komponenten konkret implementiert werden und es aus technischer Sicht sinnvoll ist, die Komponentendarstellung zu verfeinern. Zusätzlich muss beachtet werden, dass ein eingebettetes System mit begrenztem Speicher und Rechnerkapazität unter anderen Voraussetzungen arbeitet als Großrechnersysteme, auf denen Mustererkennungsprobleme üblicherweise gelöst werden.

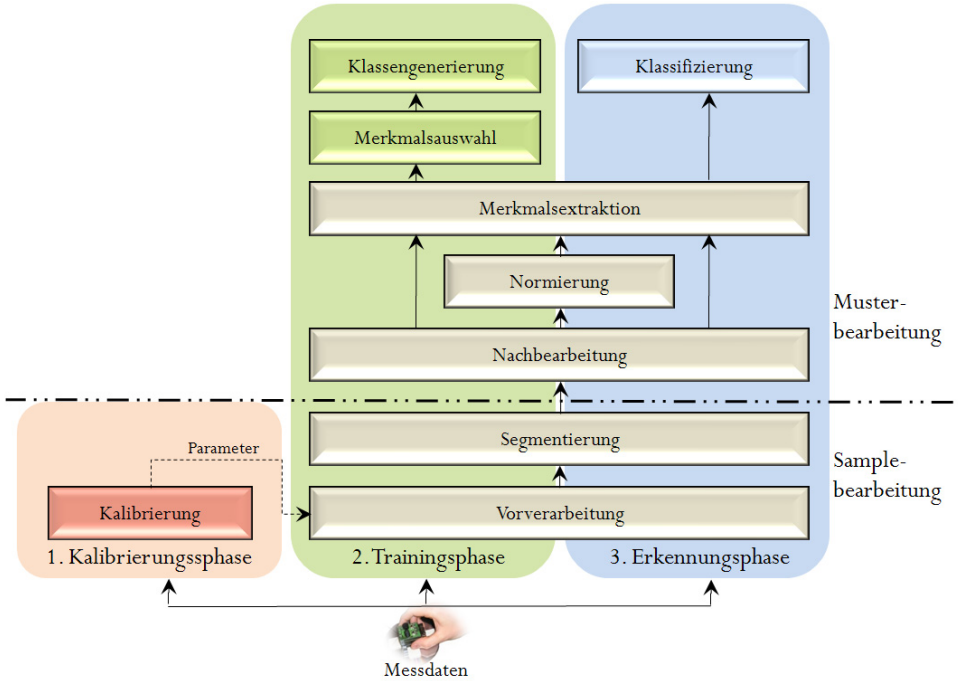

Abbildung 4.5 Phasenmodell erweitert um das konkretisierte Mustererkennungssystem

Die konkret verwendeten einzelnen Komponenten und deren Umsetzung werden im Folgenden erläutert. In dem Mustererkennungssystem in Abbildung 4.5 wird die Komponente Kalibrierung hinzugefügt. Der Sensorknoten muss, wie in Kapitel 3.2.1 angedeutet, kalibriert werden. Dies kann auch während des Systembetriebes nötig sein, insofern ist eine eigene Komponente gerechtfertigt. Die Komponente Segmentierung wird aus der Vorverarbeitung ausgelagert, da sie unabhängig von der eigentlichen Vorverarbeitung der Daten ist. Die Auslagerung der Musternormierung stellt sich insofern als praktikabel heraus, da sie nicht für jedes Feature geeignet ist. Das Modul Merkmalsauswahl wird eingefügt, da es einen entscheidenden Beitrag zur **Erkennungsrate** liefert. Die Erkennungsphase (blau) und die Trainingsphase (grün) enden beide in einer für die jeweilige Phase exklusiven Komponente. Die Klassifizierung wird nur während der Erkennung durchgeführt, ebenso wie die Klassengenerierung nur während des Trainings durchgeführt wird.

In Kapitel 4.5 wird spezifischer auf die Programmstruktur eingegangen.

4.4.1 Kalibrierung

Die exakteste Möglichkeit, den Sensorknoten auf 0 g zu kalibrieren, ist zugleich auch die aufwendigste. Wie in Kapitel 4.3.1 erwähnt, wirken konstante 1 g auf die Z-Achse, wenn diese in Richtung Erdmittelpunkt ausgerichtet ist. Der vom Sensor in dieser Ausrichtung ausgegebene Wert in Ruhelage muss gespeichert werden. Durch ein Drehen des Sensors um 180° wird der Beschleunigungssensor in die korrekte Position gebracht, um die Gegenbeschleunigung auf die Z-Achse zu erfahren. Der Gegenbeschleunigungswert in Ruhelage wird ebenfalls gespeichert. Der Mittelwert dieser beiden vom Sensorknoten ausgegebenen Werte im Ruhezustand ist der exakteste 0 g-Wert. Dieser Wert wird Offset genannt. Da die Ausgaben des Sensors zudem schwanken, sollte die gleiche Anzahl von Werten von beiden Lageseiten aufgenommen und deren Mittelwert als 0 g-Wert verwendet werden. [Sei07] verwendet 1024 Werte und weist darauf hin, dass eine höhere Wertezahl den Zielwert verbessert. Dieses Vorgehen muss für alle Achsen wiederholt werden. Da dieses Verfahren manuelles Eingreifen erfordert, ist es trotz seiner Genauigkeit nicht praktikabel. Dennoch verdeutlicht diese Technik den korrekten Umgang mit der Erdanziehungskraft eines Drei-Achsen-Beschleunigungssensors optimal.

Es wird auf das Finden eines echten Nullpunktes für die als Z-Achse fungierende Achse verzichtet, da dieses Wissen für die Mustererkennung nicht entscheidend ist. Für die Mustererkennung sind bei entsprechendem Training nur relative Bewegungen relevant. Demzufolge genügt die Annahme, dass der Sensorknoten in einer beliebigen Ausgangslage einen Ruhepunkt hat. Da trotz dieser Festlegung jede Achse, zumindest teilweise, unter den Einfluss der Erdanziehungskraft gelangt, muss einschränkend die maximal mögliche Messung jeder Achse um 1 g reduziert werden. Es wird der vom Beschleunigungssensor maximal mögliche Messbereich von ± 6 g verwendet. So reduziert sich das Messfenster auf ± 5 g. Es können folglich nur Bewegungen trainiert werden, die in alle Richtungen maximal ± 5 g verursachen, da der Sensorknoten Drehungen ausgesetzt ist.

Die maximal möglichen Beschleunigungen müssen unter dem Gesichtspunkt der korrekten Erkennung von Mustern beachtet werden. In [Bou97] und [Cha06] werden typische Bewegungen des Menschen auf ihre unterschiedlichen, erzeugten Beschleunigungen hin untersucht. Die Ergebnisse helfen einen Richtwert zu finden, um die Einstellung der Sensibilität des Sensorknotens optimal zu wählen.

Die Kalibrierung wird in der Samplebearbeitung durchgeführt. Es werden für jedes erfasste Sample die nachfolgend beschriebenen Arbeitsschritte durchgeführt, um während der Kalibrierung drei entscheidende Kalibrierungsparameter zu ermitteln:

- **Offset**
  Der Beschleunigungssensor liefert in Ruhelage keinen stabilen Nullwert, wenn er nicht optimal ausgerichtet ist. Selbst unter optimalen Umständen (Temperatur, Fertigung) schwanken die Ausgabewerte von Sensorknoten zu Sensorknoten. Daher muss ein Wert ermittelt werden, der als echter Nullpunkt fungiert. Dieser so genannte Offset wird berechnet, indem der Mittelwert der Beschleunigung ermittelt wird, die der Sensorknoten ausgibt, solange er sich in Ruhelage befindet. Dieser Wert wird später von den jeweils ausgegebenen Werten subtrahiert und ermöglicht, wie in Abbildung 4.6 dargestellt, eine Unterscheidung positiver und negativer Beschleunigung. Mit dieser Grundlage wird in der vorliegenden Arbeit festgelegt, ob ein Muster beginnt und/oder endet. Der Offset muss dynamisch bei jeder Kalibrierung neu ermittelt werden. Situationsabhängig kann sich der Offset während der Datenmessungen ändern. Der Offset stellt sich immer auf die während der Kalibrierung eingenommene Position ein. [Sei07]

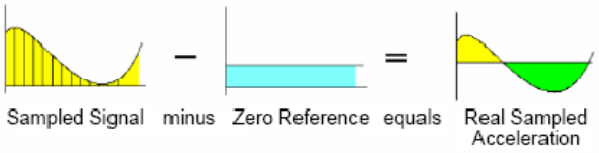

Abbildung 4.6 Vereinfachte Darstellung des Beschleunigungssignals vor und nach der Offsetbildung [Sei07]

- **Zitterbereich** (*engl. Tremble*)
  Jedes Mustererkennungsproblem kann ein verschieden ausgeprägtes Zittern beinhalten. Dieses Zittern macht es unmöglich, den eindeutigen Nullpunkt zu definieren, da ein Zittern unregelmäßige Beschleunigungsdaten produziert. Solche Daten werden im System als Bewegung interpretiert. Der Sensorknoten darf jedoch, um die Ruhelage erkennen zu können, keine Beschleunigungsdaten ausgeben. Eine Person, die beispielsweise einen Sensorknoten in der Hand hält, wird mit den Standardeinstellungen nicht in der Lage sein, den Sensorknoten in einer für den Sensorknoten ausreichenden Ruheposition zu halten, da durch das natürliche Zittern eines Menschen stetig Beschleunigungen auf den Sensorknoten wirken, die nicht seiner Nullposition entsprechen. Für einen solchen Fall gibt es die Möglichkeit, während des Trainings ein situationsabhängiges Zittern, einen so genannten Zitterbereich, zu ermitteln. So ist es, z. B. für Menschen,

möglich, den Sensorknoten nach einer Sensorbewegung innerhalb des ermittelten Zitterbereiches in eine Ruheposition zu bringen. Der Zitterbereich wird bei jeder Kalibrierung dynamisch festgelegt. Dazu orientiert sich der Algorithmus am bis dahin bereits ermittelten Offset. Der Sensorknoten misst für den Zitterbereich die Schwankungen um den durch den Offset repräsentierten Ruhepunkt und speichert die Extremwerte. [Tuc07] [Fre07]

- **Samplingintervall**
  Das System misst einen Beschleunigungswert, verarbeitet diesen und misst daraufhin wieder einen Beschleunigungswert. Zwischen den beiden Messungen muss immer die gleiche Zeit vergehen, um sicherzustellen, dass alle benachbarten Messwerte in der gleichen zeitlichen Relation zueinander stehen. Da es sich hierbei um ein eingebettetes System handelt, hat der Sensorknoten zu der Messfunktion auch die Verarbeitungsfunktion zu übernehmen. Daher folgt, dass zwischen den Messungen verschieden lang andauernde Operationen durchgeführt werden müssen. Erfolgt die Abarbeitung des Codes innerhalb des Messintervalls, muss die nächste Messung nur auf das Ende des Zeitfensters warten. Dauert die Abarbeitung hingegen länger als das vorgegebene Zeitfenster, verzögert sich die folgende Messung unbemerkt und führt zu einer ungenauen, zeitlich nicht homogenen Messreihe. Um das maximal nötige Zeitfenster zu ermitteln, muss während dieser Kalibrierungsphase der Sensorknoten bewegt werden, um so gelegentliche Mustereignisse hervorzurufen. Der Algorithmus vergrößert das Messfenster solange, bis die Mustereignisse innerhalb dieses Fensters liegen. Diese Funktion sollte zumindest nach Codeänderungen durchgeführt werden, um sicherzustellen, dass keine unnötigen Laufzeiten entstehen und um zu prüfen, ob das gewählte Samplingintervall genügend groß ist. Die theoretisch mögliche Samplingrate hängt konkret von der Geschwindigkeit ab, mit der die Vorverarbeitung und Segmentierung durchführt wird. Die Abarbeitung der Vorverarbeitung und Segmentierung muss somit schneller ausgeführt werden, als die Samples vom System gemessen werden.

Die Kalibrierung wird in der Trainingsphase und in der Erkennungsphase unter Umständen erneut nötig sein, da sich die Sensoren bedingt durch die in Kapitel 4.3.1 beschriebenen Schwankungen nach einer unbestimmten Zeit dekalibrieren können. In der erneut ausgeführten Kalibrierung wird nur der Offset und der Zitterbereich dynamisch neu ermittelt. Das ermittelte Samplingintervall bleibt erhalten, da davon ausgegangen wird, dass sich der Code und damit die maximale Laufzeit während der Datenerfassung nicht verändern. Sowohl die Mustererkennung als auch das Training kann aufgrund einer Dekalibrierung unterbrochen werden, um sich zu rekalibrieren. Nach einer Rekalibrierung wird das Training fortgesetzt und es muss nicht das gesamte Training von vorne begonnen werden. Es ist jedoch nicht zu empfehlen, während eines Trainings ein sich dekalibrierendes System als Trainingsgrundlage zu verwenden, da die erfassten Daten dann inhomogen zu einander sind. Die Dekalibrierung macht sich dann bemerkbar, wenn der Sensorknoten kontinuierlich Daten produziert, die außerhalb seiner Ruheposition liegen. Die zweite Möglichkeit der Rekalibrierung beinhaltet, dass das System nach Ablauf des

festzulegenden Kalibrierungstimers eine Kalibrierung ohne eine konkret vorgefallene Dekalibrierung startet.

4.4.2 Vorverarbeitung

Die Vorverarbeitung wird ebenso wie die Kalibrierung in der Samplebearbeitung durchgeführt, wobei ebenfalls jedes Sample direkt nach Eintreffen im Sensorknoten ausgewertet wird. Die Komponente der Datenvorverarbeitung wendet auf die eintreffenden Daten zunächst die Bitreduktion an, dann wird der zuvor in der Kalibrierung ermittelte Offset subtrahiert und anschließend das Bewegungsrauschen entfernt.

- Bitreduktion/Physikalisches Rauschen
  Selbst wenn sich der Sensorknoten in optimaler Ruhelage befindet, reichen die umliegenden Erschütterungen und die baulich bedingten Ungenauigkeiten aus, um eine ständige schwankende Datenausgabe zu erzeugen. Das Rauschen wird konkret durch eine Bitreduktion der Daten durch bitweises Verschieben der Beschleunigungswerte um 4 Bit von 12 Bit auf 8 Bit erzielt. Die auf diese Weise besser berechenbaren Nullwerte ermöglichen Musterstart- und Musterendepositionen. Der Wertebereich wird dadurch auf 0 bis 255 reduziert. Der Parameter für die Bitreduktion wird nicht dynamisch festgelegt, sondern empirisch ermittelt und ist zur Laufzeit nicht veränderbar.

- Bewegungsrauschen
  Die während einer Bewegung aufgezeichneten Daten müssen geglättet werden, um weniger Schwankungen in den Beschleunigungsdaten zu verzeichnen. Die Intensität der Glättung der Daten hängt wiederum vom jeweiligen Anwendungsfall ab und ist variabel einstellbar. Der in der Hardware integrierte Hardwaretiefpassfilter reagiert auf extreme Beschleunigungsspitzen und ermöglicht keine angepasste Tiefpassfilterung. Ein anpassbarer Tiefpassfilter wird daher implementiert. Die damit einhergehende Datenglättung führt zu deutlich weniger Störungen im Signal. Die Intensität der Filterung muss problemabhängig bestimmt werden. Ein so entwickelter Filter kann auf verschiedene Arten implementiert werden. Im vorliegenden Fall werden zwei Tiefpassfilter zur Verfügung gestellt.

  Der sogenannte **moving average** (gleitender Mittelwert) wird zur Glättung von Zeitreihen verwendet und verhält sich wie folgt: Über alle Werte der Zeitreihe werden die ersten k Elemente ab dem ersten Element gemittelt und das erste Element durch den Mittelwert ersetzt. Daraufhin werden wiederum die ersten k Elemente gemittelt jedoch beginnend ab dem zweiten Element, usw. Dies bedeutet, dass stets der Mittelwert der k rechten Nachbarelemente eines Elementes (inklusive des Elementes selber) dieses betrachtete Element ersetzt. Der **standard average** berechnet einen einfachen Mittelwert über n festzulegende Elemente und reduziert damit die Samplingfrequenz.

Die Komponente bereitet mittels der Vorverarbeitungsparameter die Rohdaten in eine auswertbare Form auf, wie in Abbildung 4.8 präsentiert. In den folgenden Darstellungen werden nur die Daten der X-Achse präsentiert, da sie zur Verdeutlichung der jeweiligen Aussage genügen und eine Darstellung aller Achsen die Übersichtlichkeit reduziert. Die konkrete Anwendung des Zitterbereiches ist in Abbildung 4.7 dargestellt.

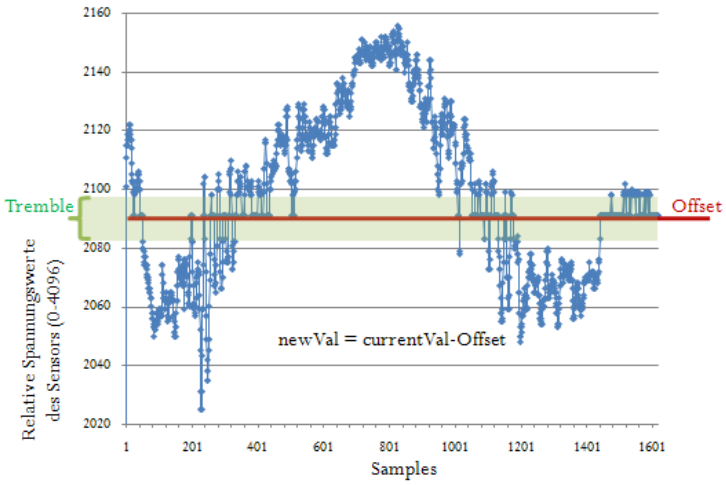

Abbildung 4.7 Reine Rohdaten des Sensors: Offset wird während einer Beispieldatenerhebung der X-Achse ermittelt. Der Zitterbereich (Tremble) erleichtert das Finden des Musterendes

Zur besseren Verdeutlichung der Funktion des Zitterbereiches werden die Beschleunigungswerte innerhalb des Zitterbereiches in Abbildung 4.7 auf den Ruhepunkt abgebildet. In der Implementierung geschieht dies jedoch nur dann, wenn das Muster tatsächlich endet.

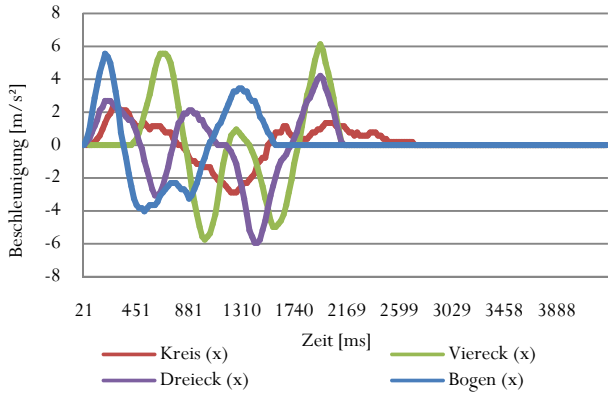

Abbildung 4.8 X-Achsen-Beschleunigungswerte (vorverarbeitet): Bewegungsrauschen mittels moving average über 8 Samples entfernt

In Kapitel 4.4.3 wird auf die Funktionsweise der Hysterese, die den Zitterbereich nutzt, eingegangen. Die Beschleunigungswerte, die zur eigentlichen Bewegung gehören, werden innerhalb des Zitterbereiches nicht verändert sondern gezählt, um nicht eine Verfälschung des Kurvenverlaufes zu bewirken. Desweiteren ist in Abbildung 4.7 der ermittelte aber noch nicht subtrahierte Offset dargestellt. In Abbildung 4.8 wird zusätzlich das Bewegungsrauschen mittels des gleitenden Mittelwertes entfernt.

4.4.3 Segmentierung

Abbildung 4.5 beschreibt die letzte Komponente der Samplebearbeitung als die Segmentierung in der Samplebearbeitung. Damit wird jedes Sample daraufhin untersucht, ob es ein Musterstart, Teil eines laufenden Musters oder ein Musterende darstellt. Die Segmentierung sorgt während der Bewegung des Sensors für die Erkennung des Musterbeginns, des Musterendes und für die Speicherung des Musterkandidaten. Ob ein Kandidat als gültiges Muster erkannt wird, entscheidet die Komponente Nachbearbeitung.

Für die Segmentierung werden zwei Zustände verwendet. Der Bewegungszustand und der Stopzustand beschreiben Zustände, die bei jeder Sampleeingabe wechseln können. Sampleeingaben entstehen mit jedem Messwert, der durch eine Bewegung erzeugt wird. Ein Sample bewirkt abhängig davon, ob es innerhalb des Zitterbereiches liegt, den Wechsel der Segmentierung in den Stopzustand oder andernfalls in den Bewegungszustand. Erst nach einer kontinuierlichen Abfolge von *N* Stopzuständen erfolgt der Wechsel in den Musterendezustand, siehe Abbildung 4.9.

Um einen Musterkandidaten zu erkennen, muss es für den Kandidaten zwei Stopzustände gebe. Zwischen diesen beiden Stopzuständen (den Zeitpunkten, in denen sich der Sensorknoten in einer Ruheposition befindet) wird die Bewegung, das eigentliche Muster, vollzogen. Ein Musterkandidat besteht demnach aus Bewegungen nach einem Stop und dem darauf folgenden Stop, der länger dauert als eine gewisse festgelegte Zeitspanne.

Es werden zunächst alle Samples des Musterkandidaten gespeichert, bis der maximale vorgesehene Speicher für Muster gefüllt ist. Der Speicher auf dem Sensorknoten agiert hier als strikt begrenzender Faktor. Es können folglich nur die Daten eines Musters erfasst und gespeichert werden, die den für das Problem vorgegebenen Speicher nicht überschreiten. Die Reduktion der Samplingrate erhöht die zulässige Musterdauer mit einer zwingend einhergehenden Reduktion der Datenauflösung. Die Verringerung der Datenauflösung erklärt sich durch die nötige Vergrößerung des Samplingintervalls und die damit einhergehende Reduktion des Messwertes pro Zeiteinheit. Dauert ein Muster länger, als der dafür vorgesehene Speicher Daten fassen kann, wird das Muster bis zum Zeitpunkt des Speicherüberlaufes gespeichert. Konsequenterweise wird das Modul der Segmentierung aber erst verlassen, wenn ein echtes Musterende gefunden wird. Eine Bearbeitung zu langer Muster wird nicht vollzogen. Diese Muster werden direkt nach dem Musterende verworfen.

In Abbildung 4.8 ist zu erkennen, dass alle Muster innerhalb der maximalen Mustergröße von ca. 4 Sekunden enden. Das Muster „Kreis“ endet bei ca. 2,6 Sekunden, „Viereck“ und „Dreieck“ bei knapp 2,2 Sekunden und der „Bogen“ bei ca. 1,6 Sekunden.

Muster mit Bewegungspausen, die innerhalb des Zitterwertes liegen, werden bei der Erfassung unterbrochen und können nicht als Ganzes erkannt werden. Der Bewegungsstop bzw. die Beschleunigungen, die in einer gewissen Zeit innerhalb des Zitterwertes liegen, signalisieren dem

Mustererkennungssystem das Ende eines Musters. Bewegungen, die den Schwellenwert des Zitterwertes überschreiten, werden als Startsignal eines Musters gewertet. Zudem ist es möglich, einen zusätzlichen Bewegungseintrittsparameter festzulegen, der bestimmt, inwieweit das Startsignal nach einem Musterende künstlich erhöht wird, um zu verhindern, dass Kleinsterschütterungen einen irrtümlichen Musterstart signalisieren. Der Bewegungseintrittsparameter wird mit dem Zitterwert akkumuliert. Fehlalarme dieser Art sind zu vermeiden und müssen heuristisch untersucht werden. Die Stop- und Startereignisse werden mittels einer Hysterese implementiert, wie sie in Abbildung 4.9 vorgestellt wird.

Die **Stop-Hysterese** (siehe Abbildung 4.9) ermöglicht das Unterscheiden einer kurzen Pause während des Musterverlaufes von einem echten länger dauernden Stop, also einem Musterende. Wenn der Beschleunigungssensor angehalten wird und die Beschleunigungswerte innerhalb des Zitterbereiches für eine gewisse einstellbare Zeit (in der Versuchsanordnung in Kapitel 4.1 beträgt diese 100 ms) um den kalibrierten Nullpunkt bleiben, wechselt die Stop-Hysterese in den eigentlichen Musterendezustand. Ab diesem Zeitpunkt wird die in Abbildung 4.5 gezeigte Musterbearbeitung gestartet. Erst wenn eine Beschleunigung einsetzt, die größer als der Zitterbereich und der Bewegungseintrittsparameter ist, verlässt das System diesen Zustand und erfasst ein neues Muster.

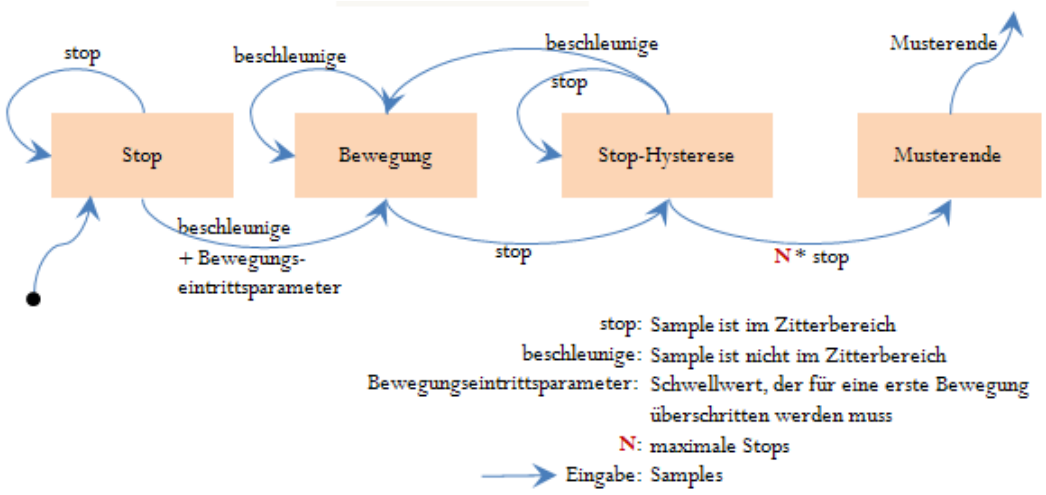

Abbildung 4.9 Stop-Hysterese

Die **Konstante Beschleunigung Hysterese** (siehe Abbildung 4.10) ermöglicht es dem Sensorknoten trotz der Einwirkung der Erdanziehungskraft, in einer Schieflage eine Stopposition bzw. ein Musterende zu finden. Ein Problem wird durch die Situation aufgeworfen, dass die Stopposition am Ende eines Musters nicht dieselbe Ruhelage wie zu Beginn des Musters haben muss. In diesem Moment verhält sich der Sensorknoten so, als ob eine konstante Beschleunigung auf ihn ausgeübt wird, insbesondere dann, wenn sich seine Achsenlage in Relation zur Erdanziehungskraft verändert hat. Dies bedeutet, dass bereits eine leichte Schieflage im Verhältnis zur Ruheposition eine konstante Beschleunigung initiiert und damit die Ruheposition nicht mehr erreicht wird. Die Konstante Beschleunigung Hysterese liefert bei diesem Problem die Lösung und wird genau dann aktiv, wenn die Beschleunigungswerte eine gewisse Zeit kons-

tant bleiben, siehe Abbildung 4.10. Die Konstante Beschleunigung Hysterese arbeitet nach dem gleichen Prinzip wie die Stop-Hysterese mit dem Unterschied, dass eine konstante Beschleunigung und nicht die Nullbeschleunigung den Stop einleitet.

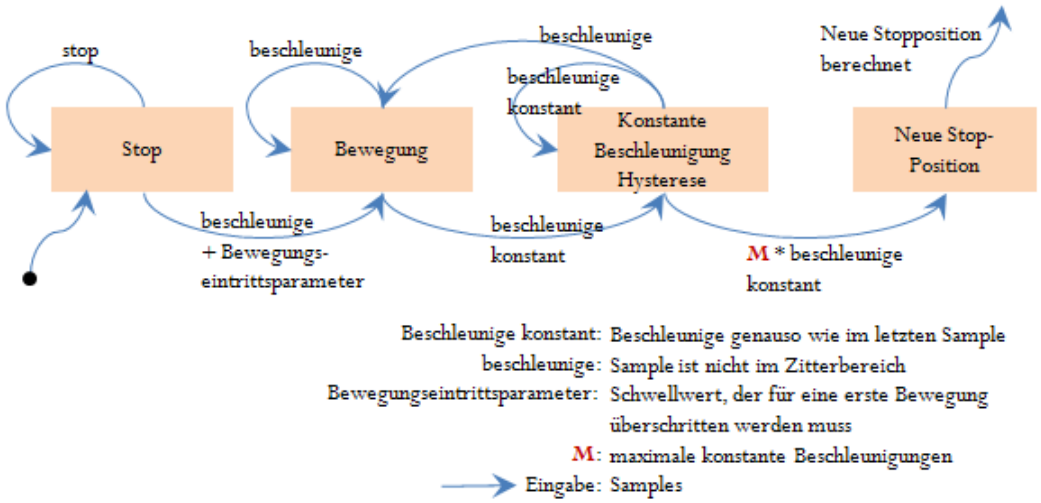

Abbildung 4.10 Konstante Beschleunigung Hysterese

Die konstante Beschleunigung darf in dem bereits angesprochenen Zitterbereich schwanken, so dass es möglich ist, innerhalb dieses Bereiches eine Ruheposition trotz Zitterns zu finden. Somit ist der Begriff der konstanten Beschleunigung so zu verstehen, dass sie in einen gewissen Spielraum konstant sein muss. Dass der Zitterbereich während einer Schieflage funktioniert, ist nicht selbstverständlich. Der Zitterbereich entsteht bisher um einen festen, bekannten Nullpunkt, der bei der Sensorkalibrierung ermittelt wird. Sobald eine feste Anzahl von Messwerten innerhalb des Zitterbereiches gefunden ist, wird die aktuelle Stopposition an dem zuvor kalibrierten Nullpunkt festgelegt, siehe Abbildung 4.11 a). Bei einer Schieflage im Verhältnis zur Ausgangskalibrierung ist dem Sensorknoten jedoch nicht bekannt, um welchen Ruhepunkt das Zittern beachtet werden muss, siehe Abbildung 4.11 b). Demzufolge muss der Sensorknoten stets eine kleine Menge von Punkten puffern, um die er einen potentiellen neuen Nullpunkt zu berechnen versucht.

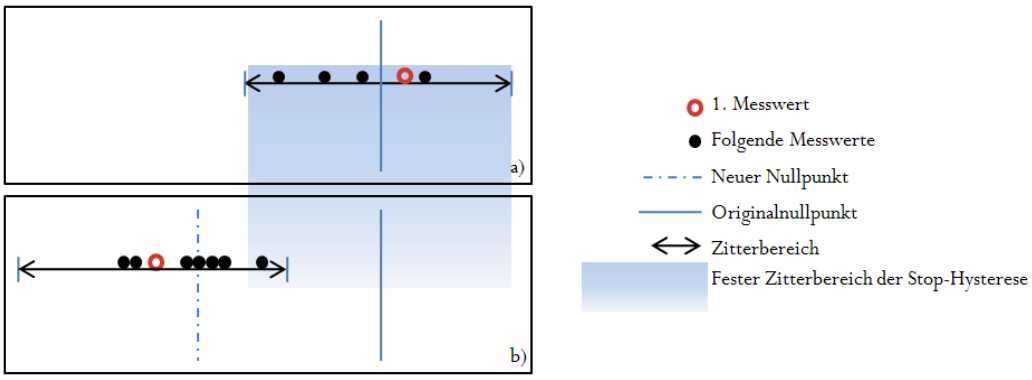

Abbildung 4.11 a) beschreibt schematisch die Berechnung des Nullpunktes mit der Stop-Hysterese b) beschreibt schematisch die Berechnung des Nullpunktes mit der Konstante Beschleunigung-Hysterese

In Abbildung 4.11 b) wird mit der Stop-Hysterese kein neuer Nullpunkt gefunden, da alle Messwerte, bis auf einen, außerhalb des Zitterbereiches vom Originalnullpunkt liegen. Der Unterschied der neuen Nullpunktberechnung befindet sich darin, dass um einen anderen ersten Messwert die folgenden Messwerte betrachtet werden. Liegen diese m folgenden Messwerte innerhalb des festgelegten Zitterbereiches um den ersten Messwert, wird für diese der Mittelwert berechnet. Der so berechnete Mittelwert stellt den neuen Nullpunkt dar. Der Mittelwert ist deshalb zu verwenden, da der erste Wert (roter Kreis in Abbildung 4.11) nicht die tatsächliche Mittelposition wiedergibt und so in Kürze eine weitere Neuberechnung des Nullpunktes angestoßen würde. Sobald einer der folgenden Messwerte außerhalb des Zitterbereiches liegt, wird dieser als neuer erster Messwert verwendet. Die Anzahl der folgenden Messwerte in Abbildung 4.11 b) sollte höher sein als die Anzahl der Messwerte für den Originalnullpunkt aus Abbildung 4.11 a), da der Originalnullpunkt Vorrang haben sollte vor einer Neuberechnung eines neuen Nullpunktes. Damit muss gelten: $M > N$ für den Automaten der Abbildung 4.9 und der Abbildung 4.10. Ist die neue Ruheposition gefunden, wird der Nullpunkt (Offset) korrigiert und gilt ab diesem Zeitpunkt für alle kommenden Messungen als „Originalnullpunkt“. Demzufolge findet ein Muster in einer im Verhältnis zur Ausgangsposition ermittelten Schieflage innerhalb des Zitterbereiches ein Stopereignis. Die vorgestellten Hysteresen werden in Abbildung 4.12 in Kombination dargestellt.

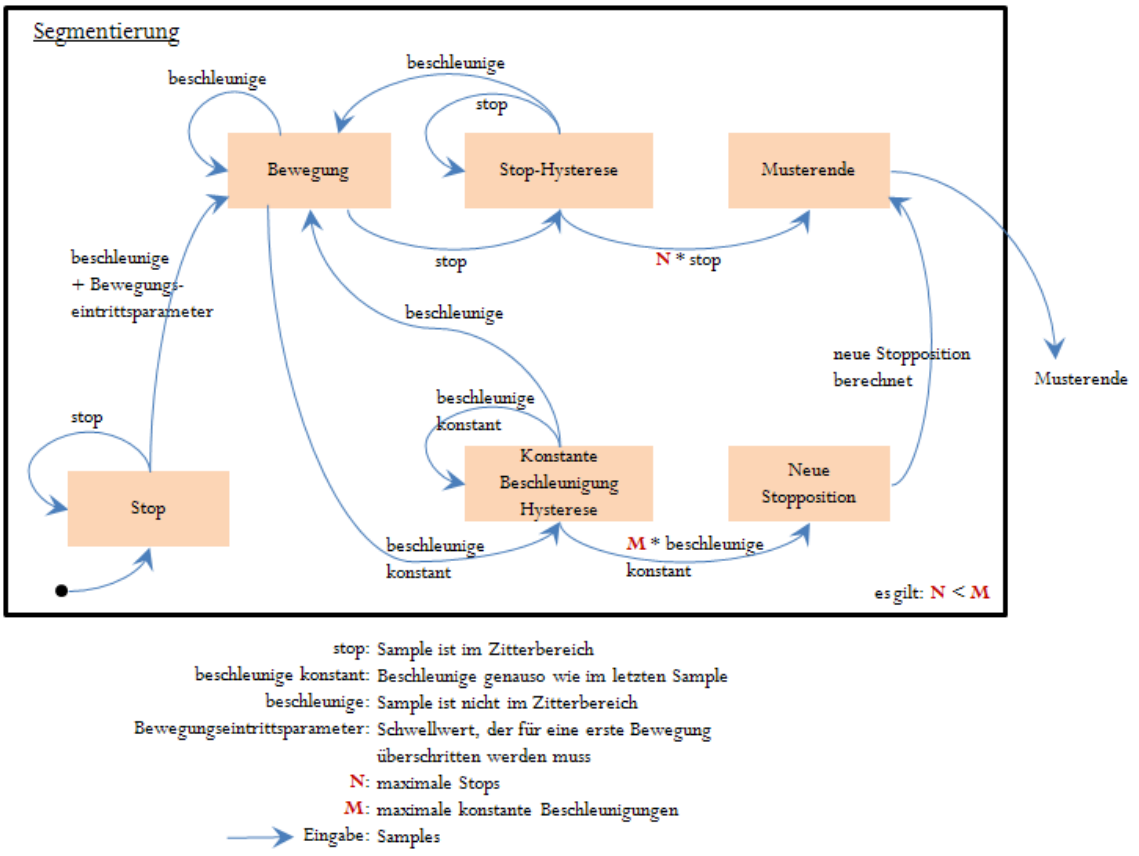

Abbildung 4.12 Kombination der Stop-Hysterese und der Konstante Beschleunigung Hysterese im Modul Segmentierung

Befinden sich die Hysteresen im Bewegungszustand und ein Stopereignis trifft ein, greift sowohl die Konstante Beschleunigung Hysterese als auch die Stop-Hysterese. Dies liegt darin begründet, dass ein Stopereignis ebenfalls einer konstanten Beschleunigung entspricht. Da ein Stopereignis jedoch höher zu priorisieren ist als eine neu berechnete Stopposition, muss die Anzahl der nötigen Stopereignisse N für die Stop-Hysterese kleiner sein als die Anzahl der konstanten Beschleunigungen M für die Konstante Beschleunigung Hysterese. Die höhere Priorität der Stop-Hysterese erklärt sich dadurch, dass weniger Berechnungsaufwand in der Verwendung der vorab berechneten Stopposition liegt. Zudem wird die Exaktheit der Mustererkennung erhöht, wenn dieselben Ausgangpositionen wiedergefunden werden. Desweiteren wird in dieser Arbeit davon ausgegangen, dass die Wahrscheinlichkeit für den Stop des Musters an der letzten gelernten Stopposition größer ist als an einer neuen Position. Am Ende beider Hysteresen gilt, dass ein Musterende eingetroffen ist und die Segmentierung verlassen wird.

Auf der einen Seite ist die Suche nach konstanten Beschleunigungsdaten ein geeigneter Weg, um eine neue Ruheposition zu ermitteln. Auf der anderen Seite fallen dadurch Muster aus dem Erkennungsrepertoire heraus, die eine gewisse Zeit konstante Beschleunigungen innehaben und danach weitere Bewegungen vollziehen. Solche Muster werden frühzeitig aufgeteilt. Nur eine Vergrößerung des Zeitrahmens, in dem das Muster durch eine konstante Beschleunigung auf allen Achsen geprägt sein muss, um die besprochene neue Ruhelage zu finden, umgeht dieses Problem. Zusammenfassend kann gesagt werden, dass Muster mit konstanter Beschleunigung auf allen Achsen nur beschränkt erfasst werden können.

4.4.4 Nachbearbeitung

In der ersten Komponente der Musterbearbeitung (siehe Abbildung 4.5) wird das Muster nachbearbeitet und entschieden, ob das Muster überhaupt ausgewertet wird oder nicht. Die Nachbearbeitung beschränkt sich darauf, dass das Muster exakt zugeschnitten wird und entsprechende Informationen über das Musterende gespeichert werden. Für den Fall, dass das Muster die Speichergrenze in der Segmentierung überschritten hat, wird in der Nachbearbeitung das aktuelle Muster verworfen und keine weiterreichende Bearbeitung durchgeführt. Wird ein Muster als gültig erkannt, wird es für die weitere Bearbeitung freigegeben und verbleibt bis zur Klassifizierung im Speicher des Sensors enthalten. Ein Muster gilt als gültig, wenn es vollständig auf dem Sensorknoten gespeichert werden kann. Eine Komprimierung der Daten durch die Merkmalsextraktion in Kapitel 4.4.6 führt zudem dazu, dass nicht die gesamten Musterinformationen erhalten bleiben müssen.

Während der Trainingsphase wird ein Muster, das verworfen werden muss, nicht gespeichert. Daraus ergibt sich, dass es nochmals trainiert werden muss, um das Training zu vollenden.

4.4.5 Normierung

Um in der Lage zu sein, Muster während der Musterbearbeitung vergleichen zu können, werden diese normiert. Die Normierung auf eine gemeinsame Zeit wird **Zeitnormierung** genannt und bildet alle Werte linear auf eine gegebene maximale Musterdauer ab. Bei der Zeitnormierung geht aufgrund der Angleichung der Musterdauer die Information über die Dauer verloren. Daher ist es unter Umständen notwendig, die Informationen über die Musterdauer vor der Zeitnormierung des Musters zu sichern. Es wird Anwendungsfälle geben, in denen das Merkmal „Musterdauer" uninteressant ist, dann kann dieses Merkmal ignoriert werden. Es gibt jedoch eine Reihe von Merkmalen, für die das normierte Muster nicht verwendet werden sollte. Dazu zählen Merkmale, die den konkreten Amplitudenverlauf beschreiben. Maximal- und Minimalmerkmale zählen dazu. Der Informationsverlust ist besonders bei Mustertypen zu beachten, deren Eigenschaften auf Intensität, Zeit und Exaktheit der Bewegungsabläufe abzielt. Andere Merkmale, wie die implementierten Histogramme, müssen auf einer normierten Mustergrundlage erstellt werden. Die Musternormierung ist in dieser Implementierung als Option zu sehen und zu verstehen.

Im zweiten Schritt der Normierung müssen die Amplituden normiert werden, um so den maximalen Wertebereich aller Merkmale in ein vergleichbares Maß zu transformieren. Die Normierung der Amplituden wird **Wertnormierung** genannt. Es existieren zwei verschiedene Wertnormierungen. Die **deformierende Wertnormierung** deformiert die Verhältnisse der Intensitäten der Amplitudenmaxima. Dies hat den Vorteil, dass ähnliche Muster nahezu gleich werden und dabei eine gute Wertauflösung erreicht wird. Die Werteauflösung verteilt sich optimal über den Normierungsbereich von 0 bis 255. Als Nachteil stellt sich ein, dass die Intensitäten der Amplitudenmaxima nicht mehr voneinander abgesetzt sind. Eine deformierende Wertnormierung mit Zeitnormierung ist in Abbildung 4.13 dargestellt.

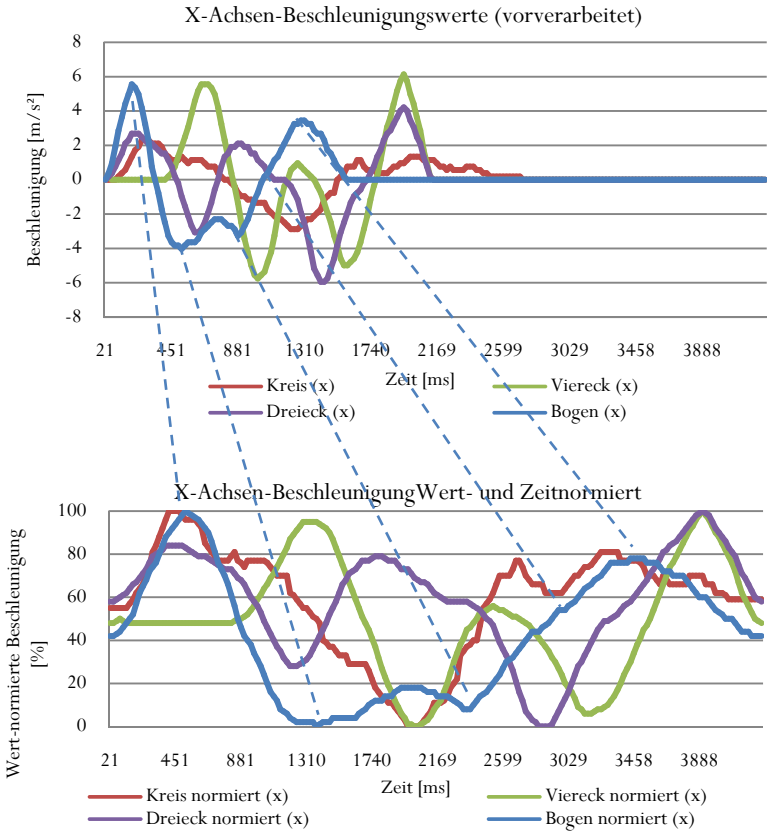

Abbildung 4.13– Prozess der Zeit- und deformierenden Wertnormierung

Wenn die Intensitätsunterschiede der Muster wichtiger sind als die gute Auflösung des Kurvenverlaufs selber und als die somit erreichte gute Ähnlichkeit der Muster innerhalb einer Klasse, muss die **deformierungsfreie Wertnormierung** gewählt werden. Dazu muss aber bekannt sein, in welchem Bereich der maximale Ausschlag der Klassen liegt. Da dies nicht mit nur einem Trainingslauf ermittelt werden kann, muss dieser maximale Amplitudenausschlagswert geschätzt werden. Je besser er geschätzt wird, umso besser wird der gesamte Normierungsbereich ausgenutzt. Das optimale Ausnutzen des Normierungsbereiches ist deswegen wichtig, weil davon die Unterscheidbarkeit der Werte stark abhängt. Es stehen 256 verschiedene Werte für die Unterscheidung zur Verfügung. Der Normierungsbereich kann zwar erhöht werden, dies ist aber nur begrenzt möglich, da, je mehr Merkmale verwendet werden, durch die Folgeberechnung des Euklidischen Abstandes Wertüberläufe entstehen können. Eine deformierungsfreie Wertnormierung mit Zeitnormierung ist in Abbildung 4.14 zu sehen.

Die blauen, gestrichelten Markierungen heben die Normierung des Musters „Bogen“ in Abbildung 4.13 und Abbildung 4.14 hervor. Die Zeitnormierung wird mit Hilfe der linearen Interpolation vorgenommen. Alle Muster werden in dieser Implementierung in den Wertebereich 0 bis 255 normiert. Üblicherweise werden Normierungen im Bereich von 0 bis 1 normiert. Da der Sensorknoten keine FPU (Floating Point Unit) besitzt, wird, soweit dies möglich ist, auf die Berechnung von Fließkommazahlen verzichtet. Eine äquivalente Prozentdarstellung ist daher die in Abbildung 4.13 gewählte Einheit.

Im konkreten Fall der zu erkennenden, geometrischen Figuren (Kreis, Viereck, Dreieck, Bogen) wird die deformierende Wertnormierung gewählt, da speziell die Form erkannt werden soll und nicht in welchem Zeitraum die Form gezeichnet wird. Die Mustererkennung ist damit teilweise invariant gegen die Art und Weise, wie die Form gezeichnet wurde.In der Bildverarbeitung ist es beispielsweise üblich, Bilder auf die gleiche Größe zu bringen. Dabei kommt es möglicherweise zu Deformationen. Im Fall dieser Bewegungsanalyse werden alle Muster auf die gleiche Dauer hochskaliert, um zu vermeiden, dass ähnliche Muster nicht vergleichbar sind, weil sich Ihre Kurvenverläufe aufgrund verschiedener Musterlängen zu sehr unterscheiden.

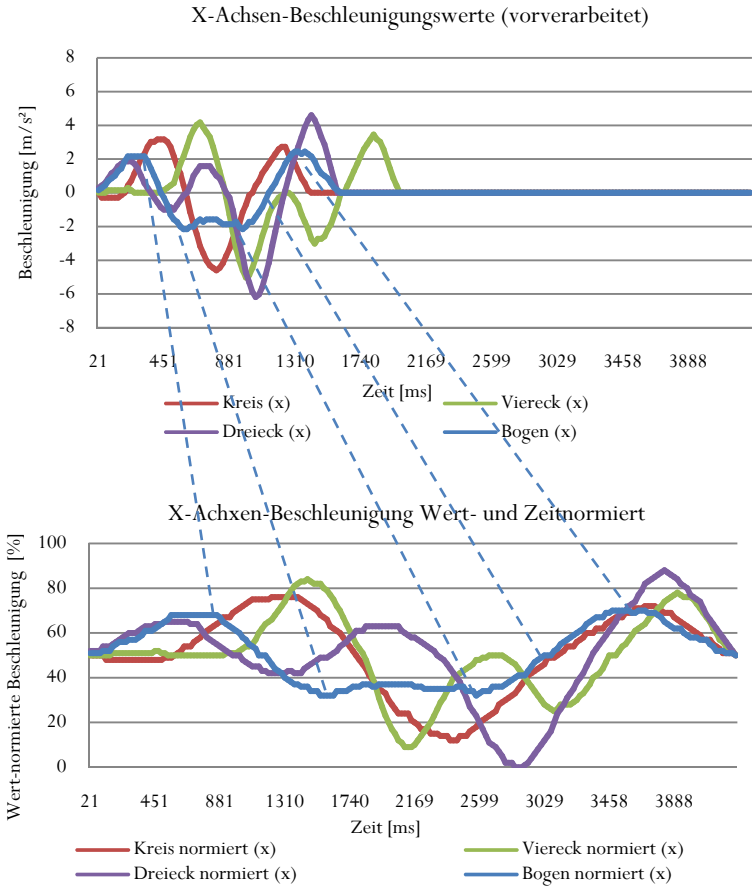

Abbildung 4.14 Prozess der Zeit- und deformierungsfreien Wertnormierung

Die deformierungsfreie Wertnormierung wirkt zunächst intuitiver und sinnvoller, hat aber den entscheidenden Nachteil, dass sie nur gut funktioniert, wenn ein nahezu optimaler, maximaler Amplitudenausschlagswert geschätzt wird. Ist dieser Wert ungünstig gewählt (siehe Abbildung 4.15), ist ein deutlicher Genauigkeitsverlust der Messkurven festzustellen. Der Grund für den Genauigkeitsverlust liegt im reduzierten Wertebereich, auf den die Kurven abgebildet werden. Je kleiner dieser Bereich ist, umso ungenauer werden die Daten, da insbesondere keine Fließkommazahlen auf dem Sensorknoten hardwareunterstützt verwendet werden können und jeder Wert auf eine ganze Zahl abgebildet werden muss. Eine geeignete Möglichkeit, um einen guten Schätzwert zu erhalten, ist eine heuristische Herangehensweise, bei der zunächst die maximalen Amplitudenwerte beobachtet und daraufhin als Schätzwert für die nächste Erhebung verwendet werden.

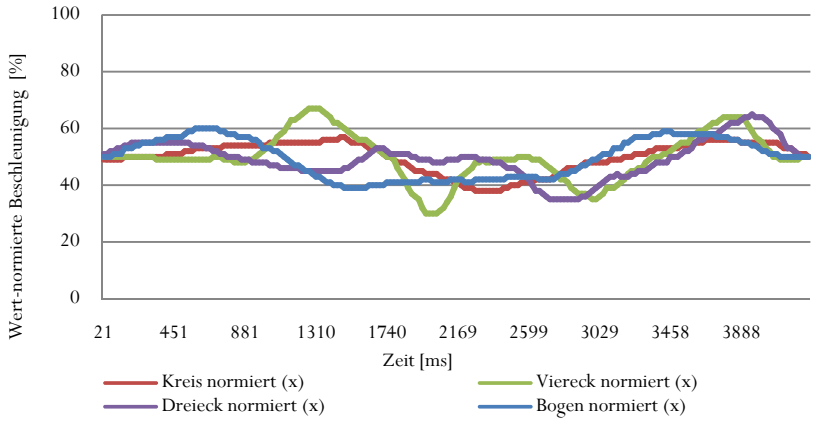

Abbildung 4.15 Prozess der Zeit- und deformierungsfreien Wertnormierung (maximaler Amplitudenausschlagswert ungünstig geschätzt)

Für eine Automatisierung dieser Technik müsste in einer weiteren (zu implementierenden) Trainingsphase, die der ersten vorangestellt ist, der maximale in den Trainingsdaten vorkommende Amplitudenwert ermittelt werden.

4.4.6 Merkmalsextraktion

Die Merkmalsextraktion in der Musterbearbeitung greift hier auf das gesamte Muster zurück. In der Merkmalsextraktion werden verschiedene Mustercharakteristika basierend auf normierten Daten (siehe Kapitel 4.4.5) berechnet. Nachdem alle Merkmale erzeugt sind, müssen die Merkmale aufgrund ihrer unterschiedlichen Skalierung (siehe Abbildung 4.3) normiert werden. Die folgenden Merkmale basieren auf dem zunächst nicht normierten, vorverarbeiteten Muster. Diese Merkmale eignen sich insbesondere für Mustervergleiche, bei denen die Muster in ihren Intensitäten unterschieden werden sollen:

- maximale Amplitude des Musters
- minimale Amplitude des Musters
- Musterdauer (entspricht der Anzahl der Samples)
- Mittelwert der Amplitude des Musters
- Varianz über das gesamte Muster

Es ist möglich, das System um andere Merkmale zu erweitern. Einige verschiedene Möglichkeiten sind in Kapitel 3.3.3.4 vorgestellt. Die Möglichkeiten diesbezüglich sind derart vielseitig, dass eine vollständige Behandlung in dieser Arbeit nicht möglich ist. Statistische Merkmale basierend auf vorverarbeiteten Daten werden zunächst im System verwendet, um mit minimalem computertechnischem Aufwand eine Mustererkennung zu erzielen. Die folgenden Merkmale basieren auf den normierten Werten des Musters und eignen sich daher insbesondere zur Unterscheidung von Mustern, deren Form verglichen werden sollen.

Histogramm

Ein weiteres verwendetes, etwas aufwändigeres Verfahren ist die Ermittlung von Histogrammwerten. Hier interessiert insbesondere die Information über die Amplitudenänderung innerhalb einer Histogrammklasse. Diese Information beinhaltet damit eine statistische Aussage, die stets das gesamte Muster betrifft, da Amplituden des gesamten Musters in Histogrammklassen einsortiert werden.

Um diese Information bzw. das Merkmal zu erhalten, werden die Musterwerte nach dem Amplitudenausschlag sortiert (siehe Abbildung 4.16) und die Samples in gleichgroße Klassen aufgeteilt. Dieses Merkmal ergibt sich aus der Differenz der maximalen und minimalen Amplitude innerhalb einer Histogrammklasse. Die so erzeugten Differenzen werden als Merkmale spezifiziert und lassen sich mit den Differenzen anderer Muster vergleichen, wenn sie aus der gleichen Histogrammklasse stammen. Die Klassengröße ist zudem variierbar und Histogrammklassen enthalten immer die gleiche Anzahl von Samples. An dem Beispiel des Kreises wird in Abbildung 4.16 für die ersten beiden Histogrammwerte aufgezeigt, wie diese Werte extrahiert werden. Dazu werden die Amplitudenweiten der je 25 Samples umfassenden Histogrammklassen ermittelt.

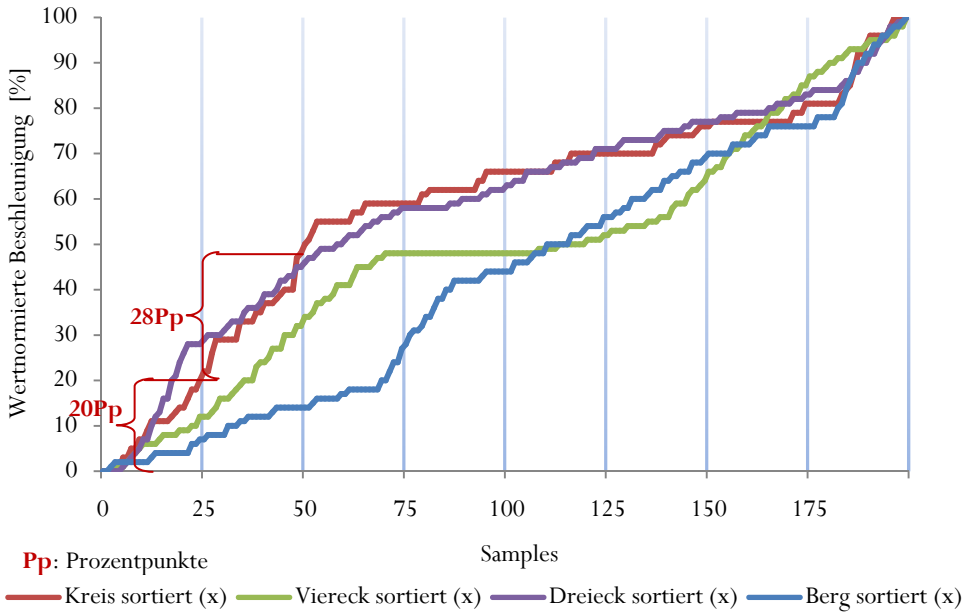

Abbildung 4.16 Sortierte X-Amplitudenwerte für die Histogrammbildung mit angedeuteter Umsetzung der Merkmalsbildung für die Figur Kreis

Die Amplitudenweitendarstellung ist für alle geometrischen Figuren aus der Versuchsanordnung in Kapitel 4.1 in Abbildung 4.17 dargestellt.

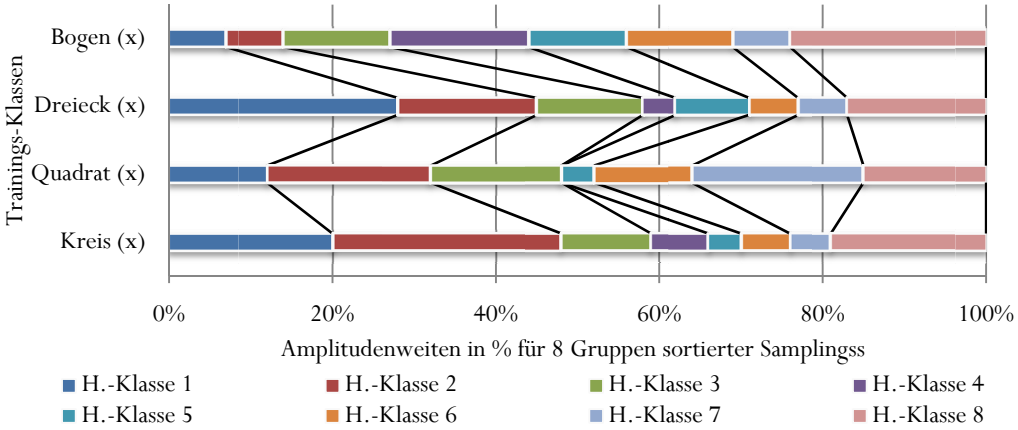

Abbildung 4.17 Vergleich von X-Achsen Histogrammwerten für die jeweiligen Histogrammklassen (Merkmale) verschiedener Musterklassen

Zu beachten ist die Menge der Merkmale, die auf diese Weise entstehen. Der Merkmalsvektor wächst mit jedem weiteren Merkmal an und erhöht die Dimensionalität. Da für jede Achse die gleichen Merkmale erstellt werden, entstehen für die Histogrammmerkmale bereits 24 Merkmale pro Klasse, da jede Klassen acht Merkmale pro Achse erzeugt. In Abbildung 4.17 werden die Merkmale des Histogramms für die X-Achse erzeugt. Dies entspricht einem 24-dimensionalen Vektorraum. Die Merkmale M_i jeder verwendeten Achse werden für jeden Merkmalstyp hintereinander für jede der drei Achsen *x, y* und *z* in den Merkmalsvektor $\underline{x}$ geschrieben (siehe Gleichung 9). Die Merkmale werden demnach auf lokaler Ebene fusioniert. Damit ist M_{ia} das Merkmal der i-ten Gruppe und der zweite Index *a* entspricht der Achse, für die das Merkmal erhoben wird.

$$\underline{x} = (M_{1x}, M_{1y}, M_{i}z, M_{2x}, M_{2y}, M_{2z}, \ldots, M_{nz}) \tag{9}$$

Die Frage nach der Anzahl der Histogrammklassen wird in der Literatur kontrovers diskutiert (siehe [Hyn95] , [Legg07]). Es existieren verschiedene Regeln, mit denen approximativ eine optimale Histogrammklassenzahl ermittelt wird, um möglichst aussagekräftige Darstellungen der Daten zu erhalten. Da die Datenkurven jedoch variieren, ist es kaum möglich, eine einheitliche Aussage nach den üblichen Regeln (Sturges - 1926, Scott - 1979, Freedman und Diaconis - 1981) zu gewinnen. Nach der Sturges-Regel liegt eine optimale Histogrammklassenzahl für die Versuchsanordnung aus Kapitel 4.1 zwischen sieben und neun Klassen *k* für 200 Samples *s*, siehe Gleichung 10.

$$k = 1 + \log_2 s = 1 + 3{,}3 * \log_{10} s \tag{10}$$

Andere Regeln liefern Klassenzahlen von bis zu 80. Die Histogrammklassenzahlen dieser Regeln sind stark variabel und auf dieses Problem nicht direkt anwendbar, da in der vorliegenden Arbeit unterschiedliche Datenreihen verwenden werden und, wie oben angedeutet, auf eine möglichst geringe Histogrammklassenzahl hoher Wert gelegt wird. Da bei der Sturges-Regel nur die Menge der Daten einfließt, wird diese als Orientierung für die Versuchsanordnung genutzt. Daraus ergeben sich konkret acht Histogrammklassen pro Achse, da sich insbesondere die Anzahl von 200 Samples gut in acht Teile zerlegen lässt und die Anzahl der Merkmale gering gehal-

ten werden soll. Eine Veränderung der Histogrammklassenzahl für andere Probleme ist folglich nicht auszuschließen und muss heuristisch ermittelt werden.

Merkmalsnormierung

Nachdem der Merkmalsvektor erstellt ist, müssen alle Merkmale, um im Vektorraum vergleichbar zu sein, auf einen gemeinsamen vergleichbaren Vektorraum, abgebildet werden. Um diese Normierung der Merkmale sowohl für die Trainingsdaten als auch für die zu klassifizierenden Daten durchführen zu können, werden die maximalen und minimalen Werte des jeweiligen Merkmales klassenübergreifend während des Trainings erlernt. Diese maximalen und minimalen Werte der jeweiligen Merkmale der Trainingsdaten werden für jedes zu extrahierende Merkmal gespeichert und bilden somit einen Erwartungsraum für das jeweilige Merkmal über alle Klassen. Alle Merkmale werden, um vergleichbar zu sein, auf diesen Erwartungsraum linear abgebildet.

Um sicher zu gehen, dass dieser Erwartungsraum auch bei geringeren Trainingsdatenmengen bereits ausreichend beschrieben ist, wird dieser um eine Pufferzone von 10 % nach oben und unten ausgeweitet. Somit werden also auch Muster, die geringfügig außerhalb des trainierten Erwartungsraumes liegen, plausibel bearbeitet. Für die Behandlung von Merkmalen, die dennoch außerhalb des Erwartungsraumes liegen, bieten sich zwei mögliche Behandlungsroutinen an. Die verwendete Routine bildet den Wert auf den nächsten Wert, also das Maximum oder Minimum, ab. Für eine musterabweisende Routine bietet es sich an, festzulegen, dass ab einer gewissen Anzahl von Merkmalen, die außerhalb des Merkmalraumes liegen, das Muster zurückgewiesen und als ungültig deklariert wird. Abhängig von dem Mustertyp ist jeweils eine Alternative zu wählen oder anzupassen.

Ein erster Ansatz zur gleichverteilten Normierung der Merkmalscluster ist es, während der Berechnung des Euklidischen Abstandes die quadrierte Differenz der Merkmale der zu vergleichenden Vektoren durch die jeweilige Merkmalsvarianz zu dividieren. Der Ansatz scheitert zunächst an der Tatsache, dass für die Berechnung der Varianz alle Trainingsdaten gespeichert vorliegen müssen. Innerhalb des Sensorknotens ist dies aus speichertechnischen Gründen nicht möglich. Der Verschiebungssatz der Varianz erlaubt es dennoch, die Varianz während des Trainings zu berechnen, ohne alle Daten puffern zu müssen. Aufgrund der fehlenden FPU und der damit verbundenen zusätzlichen Berechnungsschritte führt der Versuch innerhalb der zeitlichen Möglichkeiten dieser Arbeit zu keinem zufriedenstellenden Ergebnis. Als Ansatz sei dieser Gedanke jedoch noch nicht verworfen.

Aus den genannten Gründen wird auf eine Normierung der Merkmale, die zu einer Gleichverteilung der Merkmale führt, verzichtet. Die Verteilung wird mittels einer linearen Normierung approximiert.

4.4.7 Merkmalsauswahl - Training

Die geeignete Merkmalsauswahl ist ein komplexes Problem und erfordert ein erweitertes Trainingssystem, welches verschiedene Untermengen von Merkmalen gegen eine komplett gespeicherte Trainingsmenge testet [RPo06] . Für die meisten Mustererkennungsprobleme werden zumeist festgelegte Merkmalsvektoren für genau ein Problem definiert und verwendet. Der hier vorgestellte Algorithmus bietet die Freiheit, das System auf verschiedene Musterprobleme anzupassen. So muss das Auswählen von geeigneten Merkmalen für jedes Problem neu durchgeführt werden. Automatisierte Lösungen zur Merkmalsauswahl sind äußerst komplex und deren Umsetzungen bedürfen eines deutlich größeren Zeitrahmens als den dieser Diplomarbeit. Aus diesem Grund kann für die Komponente der Merkmalauswahl auf Kapitel 7.1 verwiesen werden. Nachfolgend wird auf die manuelle Merkmalsauswahl der Versuchsanordnung aus Kapitel 4.1 eingegangen.

Es wird zunächst der Abstand der Merkmale zu anderen Klassen betrachtet, um zu entscheiden, ob die Merkmale die Klassen gut charakterisieren. Dazu werden einige Merkmale einer Achse in Abbildung 4.18 aufgetragen. Jene Merkmale, welche die Klassen signifikant differenzieren, werden markiert und in der Mustererkennung aktiviert. Da es bei den geometrischen Figuren nicht darum geht, festzustellen, ob ein Muster mit einer bestimmten Bewegungsintensität erzeugt wurde, werden derartige Merkmale an dieser Stelle nicht betrachtet.

Bei Betrachtung von Merkmal 6 in Abbildung 4.18 ist festzustellen, dass das Muster „Kreis“ und „Quadrat“ nicht differenziert wird.

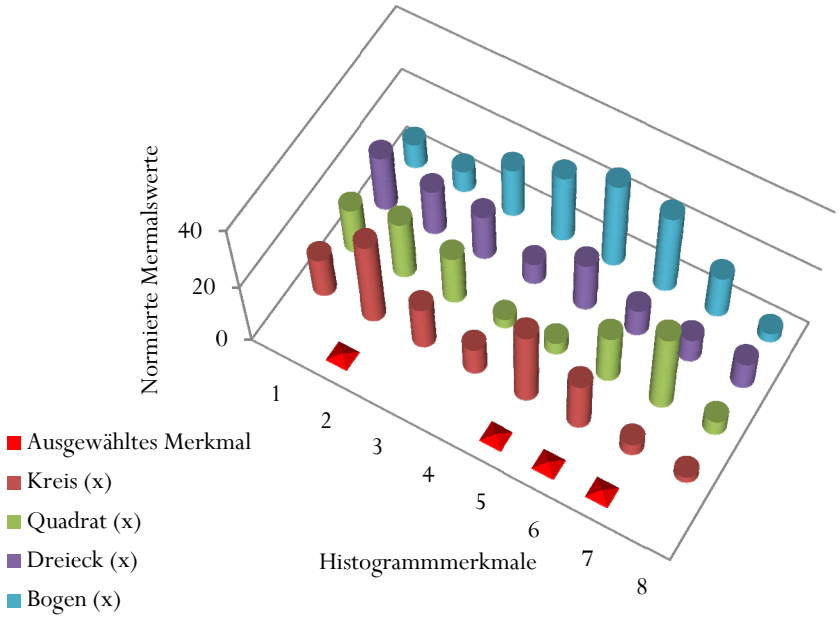

Abbildung 4.18 Merkmalsauswahl aus den Histogrammmerkmalen der X-Achse

Der Euklidische Abstand zwischen diesen Mustern wird gegen Null gehen. Merkmal 6 hat den Vorteil, dass es wie kein anderes Merkmal das Dreieck vom Bogen gut unterscheidet. Um diese

positive Eigenschaft zu nutzen, muss ein Kompromiss eingegangen werden. Merkmal 7 und Merkmal 5 differenzieren die beiden Muster „Kreis“ und „Viereck“ optimal und gleichen somit das für diese beiden Muster nachteilige Merkmal 6 aus. Unter der Annahme, dass die Merkmalsnähe innerhalb der Klassen gegeben ist, werden hier die Merkmale so ausgewählt, dass sie die Muster im Zusammenspiel möglichst eindeutig unterscheiden können. Merkmale können sich gegenseitig in ihrer Aussagekraft neutralisieren und sollten daher nur ausgewählt werden, wenn sie dazu beitragen, Klassen von einander abzugrenzen.

4.4.8 Klassengenerierung - Training

Die Trainingsphase endet mit der Klassengenerierung. Es wird für alle in der Trainingsphase kumulierten Merkmalswerte der Mittelwert der einzelnen Klassen gebildet und somit das Klassenzentrum definiert, siehe Kapitel 4.3.2. Zusätzlich werden die Minima und Maxima des jeweiligen Merkmals zur Verfügung gestellt, um mit deren Hilfe in der Erkennungsphase die Merkmalsnormierung vorzunehmen. Die generierten Merkmale, welche nun als Referenzklassenvektoren fungieren, werden ebenfalls linear auf die bekannten Minima und Maxima normiert und stellen somit in der Metrik vergleichbare Vektoren dar. Die Minima und Maxima beschreiben ein Intervall, in welchem Merkmalswerte erwartet werden. Dieses Intervall wird, wie in der Merkmalsnormierung beschrieben, auf einen festen Wertebereich linear abgebildet. Es wird im System folglich für jede Klasse ein Merkmalsvektor gespeichert. Dieser Referenzklassenvektor stellt den Mittelpunkt aller Trainingsdaten des jeweiligen Musters dar. Der Euklidische Abstand zu genau diesen mehrdimensionalen Mittelpunkten entscheidet über die Zugehörigkeit eines Musters zu einer Klasse.

4.4.9 Klassifizierung - Erkennung

Die Klassifizierung ist erreicht, wenn ein Muster evaluiert werden kann und ein vorangegangenes Training erfolgreich abgeschlossen ist. Für den Merkmalsvektor des unbekannten Musters werden im Folgenden die Euklidischen Abstände zu allen Klassenzentren berechnet und die dabei kleinste Entfernung identifiziert. Der minimale Euklidische Abstand ist der zu betrachtende Abstand. Die Musterklasse, zu der der Abstand des unbekannten Musters am kürzesten ist, wird nach der Beschreibung des k-Means-Algorithmus aus Kapitel 4.2 als die am ehesten zuzuordnende Klasse gewertet.

Ist der Abstand größer als ein vorher heuristisch ermitteltes Maß, so muss das Muster zurückgewiesen werden und kann keiner Klasse zugeordnet werden. Diese Form der Zurückweisung, auch Distanzzurückweisung genannt, wird hier jedoch nicht umgesetzt.

4.5 Implementierung des Automaten

In Abhängigkeit von dem jeweiligen Arbeitsablauf werden Trainingsroutinen oder Mustererkennungsroutinen aktiv. In Abbildung 4.19 wird der konkrete Automat dargestellt.

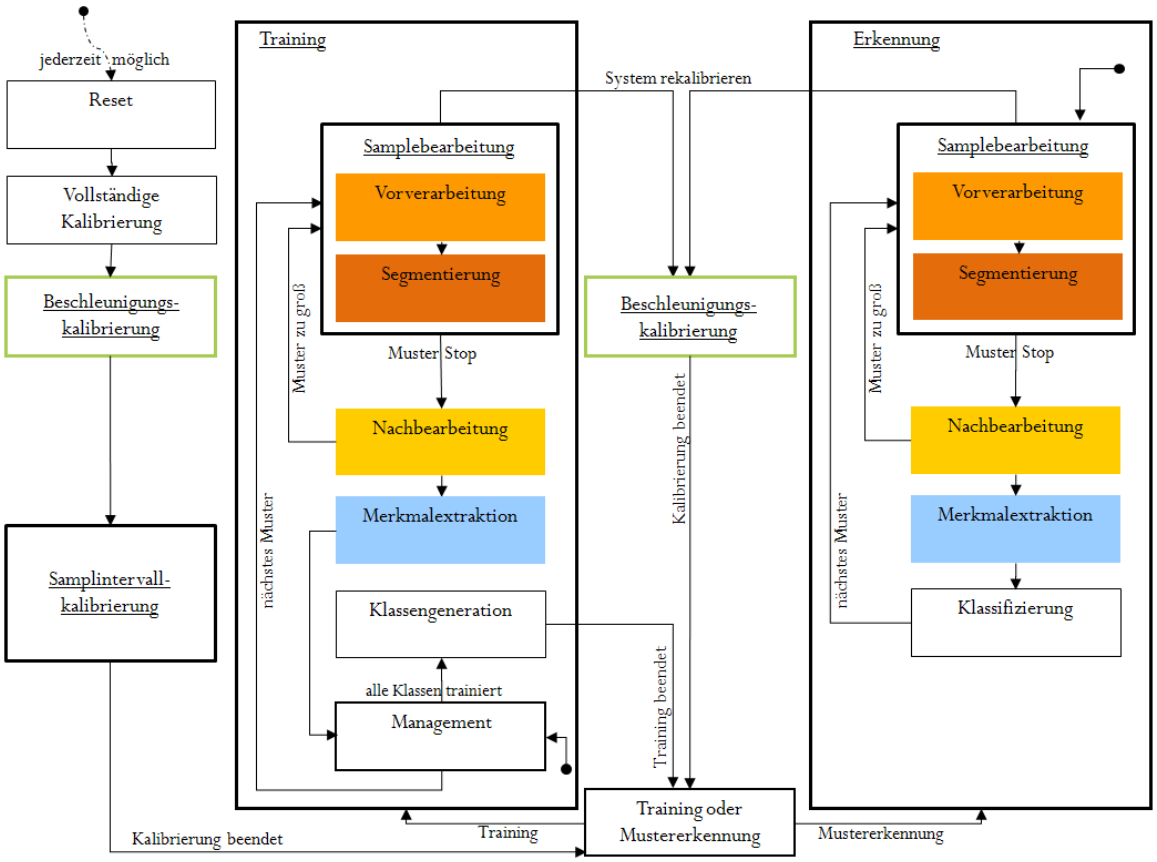

Abbildung 4.19 Automatendarstellung Mustererkennung für lokale Erkennung

Um die Komplexität des Automaten zu reduzieren, wird er nach Harel, wie in [Har86] beschrieben, in mehrere Detailansichten unterteilt. Alle Automatenteile, deren Beschriftungen unterstrichen dargestellt sind, finden sich als Detailansichten in weiteren Automaten wieder.
Die in Kapitel 4.3.2 besprochene Trainingsphase findet sich ebenso im Zustand „Training“ wieder, wie die in Kapitel 4.3.3 besprochene Erkennungsphase in dem Zustand „Erkennung“ und die in den Kapiteln 4.3.1 und 4.4.1 besprochene Kalibrierung in den so bezeichneten Kalibrierungszuständen. Die jeweiligen Phasen werden in Abbildung 4.5 vorgestellt.

Das System startet mit dem Zustand „Reset“, welcher jederzeit aufgerufen werden kann und das gesamte System inklusive des Trainings neu startet und initialisiert. Alle Daten, die während eines potentiell vorrangegangenen Trainings aufgenommen wurden, werden in diesem Zustand gelöscht. Der nachfolgende Zustand „Vollständige Kalibrierung“ startet die Beschleunigungskalibrierung und die Samplingintervallkalibrierung. Auf den Detailautomat der Beschleunigungskalibrierung wird in Abbildung 4.20 vertiefend eingegangen. Der Automat der Samplingintervallkalibrierung wird in Abbildung 4.21 vertieft dargestellt. Inhaltlich wird auf die Kalibrierung in Kapitel 4.3.1 und Kapitel 4.4.1 eingegangen. Der Zustand der Beschleunigungskalibrierung

kann sowohl während des Trainings als auch während der Erkennung eingenommen werden. Die Beschleunigungskalibrierung wird insbesondere durch eine Dekalibrierung des Systems gestartet oder auch durch die periodische Rekalibrierung die im System voreingestellt ist.

Der Zustand „Training oder Mustererkennung“ leistet die klare Trennung zwischen Training und der Erkennung. Eine mögliche Rekalibrierung während des Trainings erfordert die Möglichkeit, über genau diese Schnittstelle die Fortsetzung des Trainings zu realisieren. Das Management innerhalb des Trainings beobachtet die Anzahl der trainierten Muster der jeweiligen Mustertypen und hält somit die Prozesskontrolle innerhalb des Trainings vor. Nach Abschluss des Trainings beginnt die Erkennung.

Die Farbigkeit der Zustände innerhalb des Trainings und der Erkennung symbolisiert die Codegleichheit der Zustände beider Hauptzustände „Training“ und „Erkennung“. Die Unterschiede beschränken sich auf die bereits in den jeweiligen Modellkomponenten (in Kapitel 4.4) genannten Unterschiede und bedürfen hier keiner weiteren Erläuterung. Für die Beschleunigungskalibrierung gilt dies verstärkt, da deren Code vollständig identisch ist und nur der Übersichtlichkeit halber mehrfach dargestellt wird. Eine Detaildarstellung der Samplebearbeitung befindet sich in Abbildung 4.22**,** deren Interaktion insbesondere mit den in Kapitel 4.4.3 vorgestellten Hysteresen der Segmentierung erläutert wird.

Im Folgenden wird auf die Detaildarstellung der Zustandsdiagramme eingegangen. Die Beschleunigungskalibrierung zerlegt sich in die Ermittlung des Nullpunktes und des Zitterwertes. Beide Kalibrierungsparameter werden in Kapitel 4.4.1 besprochen. Der Automat aus Abbildung 4.20 sammelt für jeweils beide Parameter während der Kalibrierung Sensordaten, bis die vorgegebene Datenmenge ermittelt ist und berechnet die für jede Achse nötigen Parameter. Der Zustand „Zittern“ verwendet für seine Berechnung bereits den im Zustand „Nullpunkt“ ermittelten Offset, um den Streubereich um den Offset zu ermitteln. Der Streubereich könnte theoretisch in der Praxis falsch berechnet werden, zumindest dann, wenn der Sensorknoten nach der Offsetberechnung anderen Beschleunigungen ausgesetzt ist als während der Offsetberechnung. Daher ist in der Praxis darauf zu achten, dass keine unbeabsichtigten Beschleunigungen während der Kalibrierung auf das System wirken.

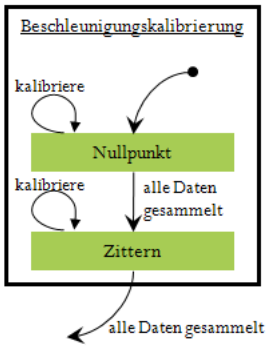

Abbildung 4.20 Automatendarstellung Beschleunigungskalibrierung

Im Idealfall sollten die Zustände „Nullpunkt“ und „Zittern“ den identischen Beschleunigungen bzw. Störungen ausgesetzt sein, um einer guten Kalibrierung zu genügen. Eine optimale Kalibrierung ist aufgrund der nicht zu verhindernden, zeitlich versetzten Berechnung der Parameter nicht möglich. Die zeitliche getrennte Berechnung erklärt sich aufgrund der aufeinander basierenden Teilberechnungen. Es ist daher dafür zu sorgen, dass keine unterschiedlichen Störungen während dieser beiden Zustände auf das System einwirken. Es wird also davon ausgegangen, dass während der Kalibrierungsschritte vergleichbare Störungen auf das System einwirken.

Die Samplingintervallkalibrierung verwendet eine leicht abgewandelte Form der Samplebearbeitung aus Abbildung 4.22, um das korrekte Samplingintervall zu ermitteln (siehe Abbildung 4.21). Der Unterschied beschränkt sich darauf, dass ein zeitmessender Zustand eingefügt wird, der in der Lage ist, festzustellen, wie lange die Bearbeitung eines Samples gedauert hat. Die Dauer der Bearbeitung hängt davon ab, welche Codeteile durchlaufen werden, wie in Kapitel 4.4.1 erläutert ist. Der zeitmessende Zustand misst folglich die Zeit vor dem Eintreffen eines Samples und nach seiner erfolgreichen Bearbeitung. Die Sampleintervallkalibrierung misst demnach für jedes Ereignis, Stopereignis oder Bewegungsereignis die Zeit und speichert die längste Dauer als Referenzwert. Es sollen sowohl Bewegungs- als auch Stopereignisse geprüft werden. Das bedeutet, dass der Sensorknoten bewegt werden muss, um die Kalibrierung an dieser Stelle erfolgreich durchführen zu können. Würde der Sensorknoten wie während der Beschleunigungskalibrierung ruhig gehalten werden, bekäme die Samplingkalibrierung keine Bewegungsereignisse zur Verfügung gestellt und das Ergebnis würde verfälscht. Stellt dass System fest, dass das Zeitfenster zu klein ist und für die Samples demnach mehr Zeit zur Verarbeitung benötigt wird als zunächst im Code angenommen, so wird diese automatisch angepasst und eine entsprechende Textausgabe an die Konsole gesendet. Diese Prozedur wiederholt sich in einer vorher festgelegten Anzahl von Samples.

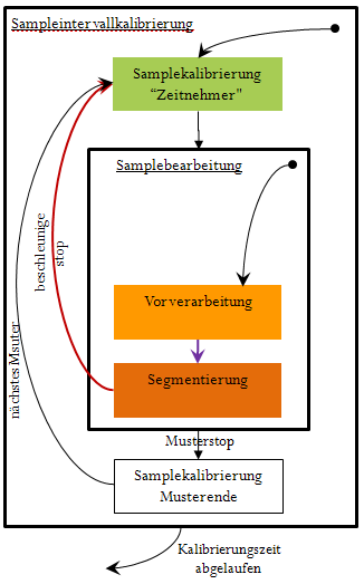

Abbildung 4.21 Automatendarstellung Sampleintervallkalibrierung

Endet eine Mustereingabe, wird die Musterbearbeitung während der Samplingkalibrierung nicht durchgeführt, da diese keinen Einfluss mehr auf die verbrauchten Rechenzeiten während des Samplings hat.

Wird vom System die Samplingfrequenz während der Sampleintervallkalibrierung verringert, sollte diese in der Implementierung angepasst werden. Das System sollte daraufhin erneut gestartet werden. Die Beschleunigungskalibrierung basiert letztendlich auf der voreingestellten Samplingfrequenz und sollte nicht während des Programmlaufes geändert werden. Nach der Sampleintervallkalibrierung kann dies jedoch nötig sein, zumindest dann, wenn sich der Quellcode derart geändert hat, dass die Abarbeitung der Berechnungen länger dauert als bei der letzten Sampleintervallkalibrierung.

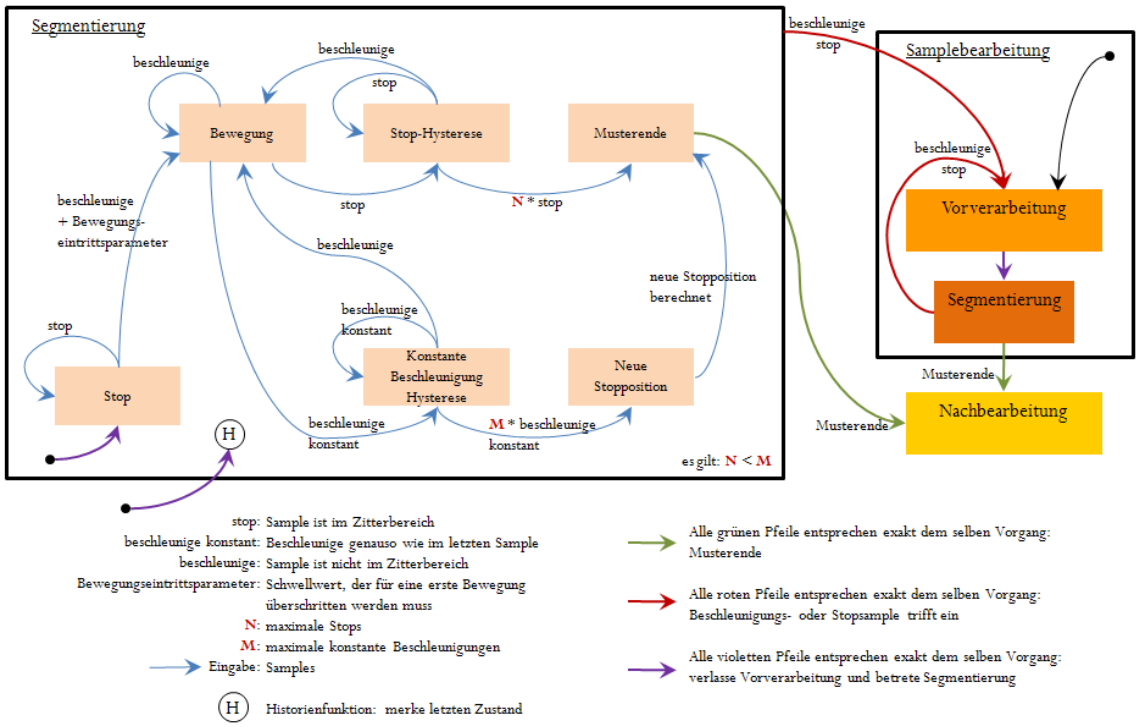

Abbildung 4.22 Automatendarstellung der Samplebearbeitung und Segmentierung

Die Samplebearbeitung besteht aus der Vorverarbeitung (siehe Kapitel 4.4.2) und der Segmentierung (siehe Kapitel 4.4.3). Jedes Sample wird in der Vorverarbeitung aufbereitet und dann an die Segmentierung weitergereicht. Der Segmentierungsautomat wird nach jedem Zustandswechsel, der durch eine Beschleunigung oder ein Stop initiiert wird, verlassen, wobei mittels der Historienfunktion, wie in [Har86] definiert, der letzte Zustand erhalten bleibt. Die Historienfunktion wird durch das große „H“ innerhalb der Segmentierung dargestellt. Nach dem Verlassen aufgrund eines solchen Ereignisses wird das nächste Sample vorverarbeitet und bearbeitet. Die Segmentierung wird an der Stelle fortgesetzt, wo sie eingangs verlassen wurde. Wird die Segmentierung für ein Muster das erste Mal betreten, beginnt sie wie alle Automaten im Startelement (schwarzer Punkt). Die roten Pfeile beschreiben in dieser Darstellung dieselben Ereignisse („Stop“, „Beschleunigung“) und sind damit identisch. Gleiches gilt für die grünen Pfeile, die das Ende eines Musters andeuten. In beiden Fällen wird die Segmentierung und da-

mit sogar die Samplebearbeitung verlassen, es wird hier ebenfalls das Musterende dargestellt. Detailbeschreibungen der Hysteresen werden in Kapitel 4.4.3 umfangreich dargelegt.

Die Nachbearbeitung entspricht exakt den Beschreibungen des Kapitels 4.4.4. Während der Merkmalsextraktion wird direkt überprüft, ob die Merkmale auf einer normierten Datengrundlage zu berechnen sind oder nicht. Daher wird die Musternormierung direkt in den Zustand der Merkmalsextraktion eingearbeitet. Eine Merkmalsauswahl wird, wie in Kapitel 4.4.6 beschrieben, manuell vorgenommen und entfällt daher als Zustand. Die Klassengeneration während der Trainingsphase erfolgt konsequent nach den Beschreibungen des Kapitel 4.4.8, ebenso wie die Klassifizierung der Erkennungsphase, welche nach den Beschreibungen des Kapitel 4.4.9 implementiert ist.

5 VERTEILTE MUSTERERKENNUNG

In diesem Kapitel werden geeignete Methoden zur verteilten Entscheidungsfindung für das Einzelsensorssystem untersucht. Für die verteilte Mustererkennung wird das in Kapitel 3.4 erläuterte Omnibusmodell verwendet. Das Omnibusmodell aus Abbildung 3.25 lässt sich leicht auf das in dieser Arbeit entwickelte Mustererkennungssystem der Abbildung 4.5 abbilden, da beide Systeme die typischen Elemente eines Mustererkennungssystems enthalten und modular aufgebaut sind. Es sind die Abbildungen aus Tabelle 5.1 vorzunehmen, um das Omnibusmodel hier einsetzen zu können.

Tabelle 5.1 Abbildung Omnibus-Modell auf das Mustererkennunssystem dieser Arbeit

Omnibusmodell	**Mustererkennungssystem**
Elemente Wahrnehmung Signalverarbeitung	Vorverarbeitung Segmentierung
Merkmalsextraktion Mustererkennung	Nachbearbeitung Normierung Merkmalsextraktion
Kontextverarbeitung Entscheidung	Klassifizierung
Steuerung Ressourceneinsatz	beliebiges übergeordnetes System zur Verwaltung von Ressourcen sowie die Alarm- und Handlungsinstanz

Alle entscheidenden und zu verwendenden Schnittstellen (siehe Tabelle 5.2) des Omnibusmodells stehen in dem hier genutzten Modell zur Verfügung, da die modulare Trennung der Mustererkennungskomponenten dies ohne weiteres zulässt. An diesen Schnittstellen ist es möglich, Daten im System abzugreifen und innerhalb der entsprechenden Fusionsebene zu fusionieren. Von nicht allen Schnittstellen wird Gebrauch gemacht.

Tabelle 5.2 Omnibus-Schnittstellen mit ihren Fusionsebenen und der in dieser Arbeit eingesetzten Fusionsebenen

Omnibusmodell - Schnittstellen	Fusionsebene	Verwendet
„Data"-Fusion (Sensordatenfusion)	Signalebene	Nein
Merkmalsfusion („Soft-decision"-Fusion)	Merkmalsebene	Ja
Klassifikationsfusion („Hard-decision"-Fusion)	Klassifizierungsebene oder Symbolebene	Ja
Sensor-Management	Infrastruktur außerhalb der Erkennungsebenen	Nein

Die Signalebene soll aufgrund der hohen Komplexität ihrer nötigen, zeitlichen Synchronisation und des damit verbundenen hohen Datenaufwandes nicht betrachtet werden. Die zentrale Zielsetzung dieser Arbeit, ist Fusionen auf der Signalebene zu vermeiden und unterschiedliche Wege der Datenfusionen zu untersuchen.

Der Funkverkehr, der nach der verteilten Erkennung einen gewünschten Alarm des Systems an der Basisstation auslöst, wird hier nicht betrachtet, da eine Basisstation in dieser Arbeit nicht angesteuert wird. Es ist jedoch davon auszugehen, dass der für die Alarmgebung auf wenige Bit reduzierte Datenverkehr problemlos mit üblichen Routingprotokollen zu lösen sein wird. Ressourcenverteilende Lösungsansätze sind ein weiterer zu untersuchender Bereich des verteilten Mustererkennungssystems, bei dem die Sensorknoten je nach Bedarf bestimmten Beobachtungsregionen zugeordnet werden können, dessen Betrachtung hier nicht weiter verfolgt wird.

5.1 Methoden

Für die Implementierung wird die komplementäre Integration gewählt, da sie zusätzlich die konkurrierende Integration abdeckt. Die komplementäre Integration wird verwendet, um Musterteile mit verschiedenen oder gleichen Nutzinformationen zu einem Muster zusammenzufügen. Muster mit gleichen Nutzinformationen werden genauso behandelt wie verschiedenartige Musterteile. Im Zusammenhang betrachtet ist ein gültiges Muster, das aus n identischen Teilen besteht, äquivalent zu einem gültigen Muster, das aus n unterschiedlichen Teilen besteht. Zum Beispiel besteht in einer Methode A ein Muster aus drei Teilen. Jeder der drei beteiligten Sensoren liefert als Klassifikationsergebnis eine Zahl. Ein Muster M besteht aus drei Einsen und wird mit der komplementären Integration zu (1, 1, 1) fusioniert und als Muster M erkannt. Eine andere Methode B erkennt das dreiteilige Muster N (1, 2, 3) und kann ebenfalls mit der komplementären Integration fusioniert werden. Mit der konkurrierenden Integration wäre eine Fusion der Daten ausschließlich für Methode A möglich, da deren Mittelwert alle Teilinformationen des Musters beschreibt. Die komplementäre Integration ermöglicht es hingegen, beide Mustertypen mit einer Integrationsform zu fusionieren.

In beiden Methoden wird das Muster im obigen Beispiel auf der Klassifikationsebene fusioniert. Ein erster Untersuchungsansatz ist genau dieser.

Nachfolgend wird die Grundmethodik der Datenfusion erläutert. Die konkrete Umsetzung für diese Arbeit und Wahl der Methoden wird in Kapitel 5.2.6 vorgenommen.

- Klassifikationsfusion
  Die Methodik der Klassifikationsfusion fusioniert Klassifikationsergebnisse verschiedener Sensorknoten durch Mittellwertbildung oder verschiedene Formen der gewichteten Analysen. Üblicherweise werden den Klassifikationsergebnissen Gewichte zugeordnet, die den entsprechenden Einfluss der Ergebnisse regulieren. Gewichte können z. B. auf Erfahrungswerten beruhen, die die Qualität und Zuverlässigkeit der Sensorknoten beschreiben. Ein weiteres Kriterium kann z. B. die Entfernung der Sensorknoten zum vermuteten Ereignis sein. Eine erhöhte Entfernung zum Ereignis bedeutet folglich eine entsprechend verringerte Einflussnahme des Ergebnisses des Sensorknotens auf die Evaluation. Eine detaillierte Beschreibung eines gewichteten Evaluationsansatzes ist in der Ausarbeitung nach Liu Chun-Ting [Liu06] zu finden. Aufgrund der hohen Komprimierung der Ereignisdaten wird das Datenaufkommen dieser Methodik eher gering eingeschätzt und die Ergebnisqualität unter Umständen auf diese Weise negativ beeinflusst.

- Merkmalsfusion
  Die Merkmalsfusion verwendet die von den an der Auswertung beteiligten Sensorknoten ermittelten Merkmale und bestimmt aus diesen Daten im Zusammenhang betrachtet ein Klassifikationsergebnis. Für die Merkmalsauswertung wird dann ein Klassifikator eingesetzt, der die Merkmale gewichtet oder ungewichtet betrachtet und klassifiziert. Bei der Merkmalsfusion werden mehr Daten versendet als bei der Klassifikationsfusion, wodurch ein erheblich größerer Kommunikationsaufwand erwartet wird. Die Merkmale bieten aufgrund ihrer Vielzahl die Möglichkeit, präzisere Informationen und feinere Abstufungen der Erkenntnisse über ein Ereignis zu vermitteln als die Klassifikationsergebnisse. Daher wird erwartet, dass die Merkmalsfusion unter Annahme der Verwendung geeigneter Merkmale bessere Erkennungswerte liefert als die Klassifikationsfusion.

- Kooperative Fusion
  Die kooperative Fusion verknüpft die Klassifikationsfusion und die Merkmalsfusion auf verschiedenste Weise. Es sind diverse Kombinationen denkbar, zwei von ihnen werden in Kapitel 5.2.6 konkretisiert. Der allgemeine Ansatz dieser Methode beschreibt, dass bei nicht ausreichender Qualität der Daten für die Klassifikationsfusion diese Daten um die der Merkmalsfusion auf unterschiedliche Weise ergänzt werden. Die Erwartung an die Ereigniserkennungsqualität dieser Methode ist hoch, da die Vermutung besteht, dass Schwächen der Klassifikationsfusion durch die Merkmalsfusion ausgeglichen werden kann und dabei der Kommunikationsaufwand im Verhältnis zur Erkennungsleistung ge-

ringer ist als bei der Merkmalsfusion. Die zu bestätigende Annahme wäre dabei, dass die Merkmalsklassifikation eine höhere Erkennungsqualität hat als die Klassifikationsfusion. Aufgrund der bei steigender Sensorknotenzahl wachsenden Dimension des Merkmalsvektors in der Merkmalsfusion ist die hier prognostizierte Verbesserung nicht gesichert. Mit praktischen Versuchen der Klassifikationsfusion und der Merkmalsfusion muss die erwartete Verbesserung der Klassifizierungsaussagen bewiesen werden.

5.2 Verteilte Mustererkennung – Architektur

Das Omnibusmodell erlaubt es, in der Architektur des hier vorgestellten Mustererkennungssystems an geeigneten Stellen ein Modul anzusetzen, welches in der Lage ist, Klassifikationsergebnisse und/oder Merkmale der betrachteten Muster zu propagieren. Das für den verteilten Einsatz zu verwendende Modul erhält den Namen „Radio“. Zunächst muss die Kommunikation zwischen den Sensorknoten aufgebaut werden. Es wird sich hier auf eine einfache Lösung im Bereich der Funkübertragung beschränkt, da der Fokus auf der verteilten Mustererkennung und nicht auf der Optimierung des Kommunikationsprotokolls liegt.

Da alle nötigen Daten für die Verteilung in der lokalen Erkennung erzeugt werden, müssen sie folglich weiterhin auf die gleiche Weise erzeugt werden und gewähren die Wiederverwendung des lokalen Erkennungssystems. Für das verteilte Erkennungssystem müssen die Datenstrukturen erweitert werden, um die Trainings- und Erkennungsdaten anderer Sensorknoten speichern zu können. Das verteilte Erkennungssystem wird auf das vorhandene lokale Erkennungssystem aufgesetzt. Abbildung 5.1 zeigt die integrierte Sendetechnik, sowie eine weitere Phase, die „Hello“-Phase. Weiterhin werden auf das lokale Erkennungssystem neue Zustände aufgesetzt, die die Verteilung der Daten ermöglichen. Die Zustände und deren Zusammenhang sind nachfolgend beschrieben.

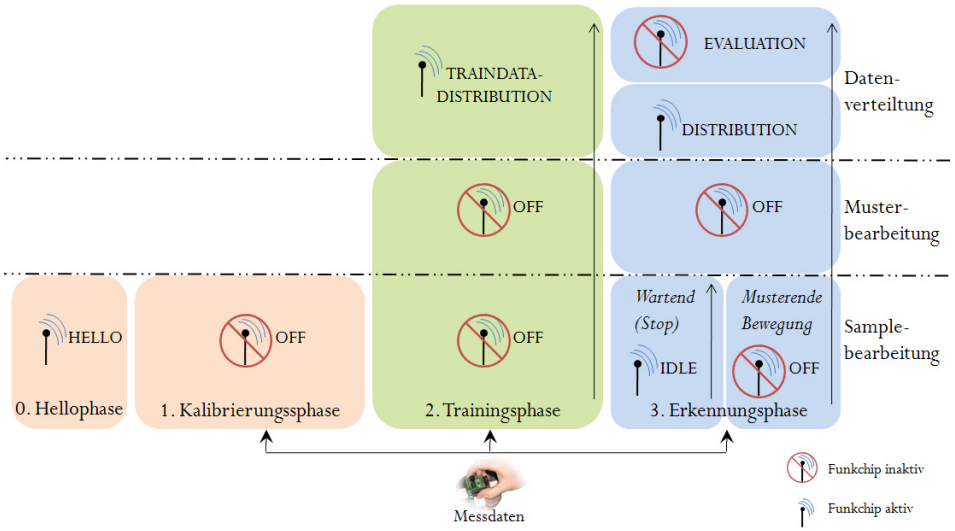

Abbildung 5.1 Lokales Mustererkennungsmodell um die verteilte Erkennung ergänzt, basierend auf Abbildung 4.5

Im Folgenden werden bisher aus der lokalen Mustererkennung unbekannte Elemente besprochen.

5.2.1 „Hello"-Zustand

Die Sensorknoten befinden sich nach dem Einschalten direkt im „Hello"-Zustand bzw. in der „Hello"-Phase und haben somit noch vor der Kalibrierung ihre Nachbarn registriert. Im „Hello"-Zustand tauschen die Sensorknoten ihre Identität aus. Alle beteiligten Sensorknoten müssen zur gleichen Zeit eingeschaltet werden, da der „Hello"-Zustand nach einer fest vorgegebenen Zeit verlassen wird und nur innerhalb dieser Zeitspanne „Hello"-Nachrichten akzeptiert werden. Die Initialisierung der Sensorknoten untereinander beschränkt sich demnach auf die „Hello"-Phase. Jeder Sensorknoten propagiert seine Identität einmal und speichert die empfangenen Identitäten inklusive seiner eigenen Identität ab.

In den hier betrachteten verteilten Mustererkennungsmethoden sind stets alle registrierten Sensorknoten an der Erfassung beteiligt. Eine vorgegebene Anzahl von Sensorknoten darf innerhalb eines Sensorknotens registriert werden, weitere Sensorknoten werden nicht akzeptiert. Die verteilte Mustererkennung kann nur erfolgreich arbeiten, wenn die vorgegebene Anzahl der zu registrierenden Sensorknoten bei wenigstens einem Sensorknoten erfolgreich verzeichnet werden kann. Die Sensorknoten, die ihre Nachbarknoten registriert haben, können im späteren Verlauf deren Daten in die Klassifizierung mit einfließen lassen. In der „Hello"-Phase wird, nachdem alle Sensorknoten registriert sind, eine feste Reihenfolge der Sensorknoten festgelegt (einfache Sortierung nach Größe der ID). Nach Abschluss dieser Phase arbeitet das System mit den bekannten Modellkomponenten, die im Einzelnen in Kapitel 4.4 erläutert werden.

5.2.2 „Off"-Zustand

In dem sogenannten „Off"-Zustand können keine Nachrichten empfangen und gesendet werden. Dieser Zustand wird vor allem dann eingenommen, wenn das System nicht durch Interrupts gestört werden darf. Gründe für die Notwendigkeit eines unterbrechungsfreien Arbeitens sind in erster Linie die Zeitpunkte der Datenerfassung, also des aktiven Samplings von Bewegung. Das System sollte zudem nicht unterbrochen werden, wenn es sich in der Kalibrierungsphase befindet, da es eben gerade dort zu Erfassungsverzögerungen kommt, wenn Nachrichten behandelt werden müssen. Diese Verzögerung kann bei einmaligem Auftreten toleriert werden, ein solches Auftreten von Interrupts kann jedoch nicht als einmalig garantiert werden. Das Senden und Empfangen von Daten wird folglich nicht während der gesamten Samplebearbeitung gestattet, da durch einen Interrupt Verzögerungen während der Musteraufzeichnung entstehen können, die das Muster beeinflussen und nachhaltig verändern. Weiterhin wird festgelegt, dass während der Berechnung der Klassifizierung ebenfalls keine Nachrichten empfangen werden dürfen, da dies die Fertigstellung unnötig verzögert und zudem zu diesem Zeitpunkt kein Mehrwert von einer Kommunikation ausgeht.

Es kann vorkommen, dass ein Sensorknoten kurzfristig nicht antwortet. In diesem Fall befindet sich der Sensorknoten im „Off“-Zustand und erkennt, bearbeitet ein Muster oder befindet sich in der Kalibrierungsphase. Antwortet ein Sensorknoten längerfristig nicht, ist er ausgefallen. Gründe für einen Sensorknotenausfall kann ein physikalischer Defekt oder ein Energieversorgungsproblem sein.

5.2.3 „Distribution“-Zustand

In dem sogenannten „Distribution“-Zustand befindet sich der Sensorknoten, nachdem er ein Muster klassifiziert hat. Ist ein Muster klassifiziert, wird davon ausgegangen, dass die registrierten Nachbarknoten ebenfalls ein Muster erkannt haben oder dies zumindest in Kürze passiert sein wird. Die Mustererkennung auf verschiedenen Sensorknoten dauern unterschiedlich lange, da die eintreffenden Muster aufgrund der Beschaffenheit einer Bewegung nie exakt gleich sein können, selbst wenn die Sensorknoten örtlich dicht beieinander liegen. Die Wartezeiten der Firmware auf Antworten anderer Sensorknoten sind zu kurz, um zu garantieren, dass Nachrichten wirklich bei dem Zielsensor ankommen. Um sicherzustellen, dass die Erkennung anderer Sensorknoten bereits abgeschlossen ist, versendet der Sensorknoten erst nach einer gewissen Wartezeit seine Daten. Während des gesamten „Distribution“-Zustandes kann der Sensorknoten bereits Daten empfangen. Die Sendeabfolge der Nachrichten ist in Abbildung 5.2 illustriert.

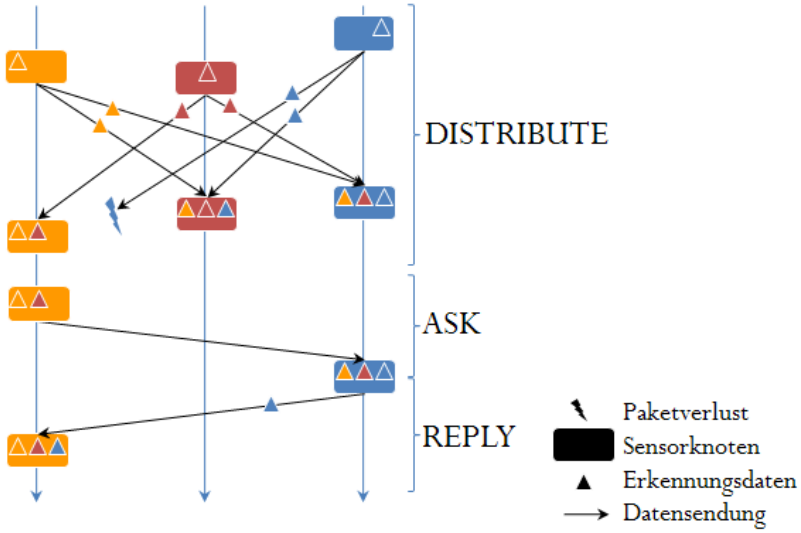

Abbildung 5.2 Sendefolge der Pakete mit DISTRIBUTE-, ASK-, REPLY- Nachrichten für ein verloren gegangenes Paket

In einem ersten Schritt werden die Daten in einer sogenannten DISTRIBUTE-Nachricht zusammengestellt. Der Dateninhalt der Nachricht besteht aus den Erkennungsdaten der erkannten Klasse und der erkannten Merkmale. Die DISTRIBUTE-Nachricht wird per Broadcast verteilt, zudem wird nicht auf DISTRIBUTE-Nachrichten geantwortet. Alle Sensorknoten speichern die so erhaltenen Erkennungsdaten der Nachbarknoten lokal ab.

Im zweiten Schritt prüfen die Sensorknoten, ob sie von jedem registrierten Nachbarknoten die Erkennungsdaten erhalten haben. Falls Daten eines Sensorknotens fehlen, sendet der entsprechende Sensorknoten eine ASK-Nachricht aus, auf die dann der angefragte Sensorknoten mit einer REPLY-Nachricht antwortet. Die Versendung einer ASK-Nachricht tritt nur dann ein, wenn der Sensorknoten nicht innerhalb einer gewissen Zeit die gewünschten DISTRIBUTE-Nachrichten der registrierten Sensorknoten erhalten hat.

Die REPLY-Nachricht enthält die Erkennungsdaten und bleibt immer unbeantwortet. Da die Sensorknoten bei Erhalt einer ASK-Nachricht immer mit einer REPLY-Nachricht antworten, ist während des „Distribution"-Zustands gesichert, dass die Merkmale und Klassifizierungsdaten ausgetauscht werden können. Selbst wenn sich die Onlinezeiten nur knapp überschneiden und möglicherweise die DISTRIBUTE-Nachricht einen Sensorknoten nicht erreicht hat, weil er zu spät mit der Mustererkennung abschließen konnte, ist der Datenaustausch mit hoher Wahrscheinlichkeit gewährleistet.

Wenn keine Pakete verloren gehen, können die Sensorknoten im Idealfall bereits nach n DISTRIBUTE Paketen (n = Anzahl Sensorknoten) ihre Daten ausgetauscht haben. Für jedes nicht - übertragene DISTRIBUTE-Paket oder nicht beantwortete ASK-Paket werden durch den ASK-REPLY-Mechanismus noch einmal maximal zwei Pakete versendet. Um die maximale Wiederholung von Nachrichtensendungen zu begrenzen, darf jeder Sensorknoten inklusive der DISTRIBUTE-Nachricht zusammen nur $n-1$ ASK-Nachrichten verschicken.

Aufgrund dessen, dass Stabilitätsprobleme mit dem in der Firmware implementierten Unicast-Protokoll vorliegen, muss jede Nachricht mittels Broadcast übermittelt werden. Der so entstehende Sende-Overhead ließe sich mit entsprechenden Korrekturen in zukünftigen Versionen der Firmware verbessern.

5.2.4 „Idle"-Zustand

In dem sogenannten „Idle"-Zustand ist der Sensorknoten in der Lage, auf Nachrichtenanfragen (DISTRIBUTE- oder ASK-Nachrichten) zu antworten, kann aber selber keine Datenanfrage stellen. Dies ist auch nicht nötig, da sich der Sensorknoten in diesem Zustand in der Ruheposition befindet und keine Beschleunigung erfahren hat, die ihm einen Musteranfang signalisieren würde. Diese Ruheposition entspricht dem Stop-Zustand der Stop-Hysterese aus Abbildung 4.9. Der Sensorknoten wird in diesem Zustand immer antworten, dass er keine Muster empfangen hat. Diese Antwort wird mittels einer REPLY-Nachricht verschickt. REPLY-Nachrichten enthalten normalerweise das Klassifizierungsergebnis und/oder Merkmale des erkannten Musters. Während des „Idle"-Zustandes enthalten die REPLY-Nachrichten immer das für alle Sensorknoten gleichermaßen bekannte Flag, dass sich der angefragte Sensorknoten im „Idle"-Zustand befindet. Der „Idle"-Zustand hilft in zukünftigen Arbeiten zu unterscheiden, ob ein Sensorknoten ausgefallen ist oder ob sich bei einem Sensorknoten keine Muster ereignet haben.

5.2.5 „Train-Distribution“-Zustand

Dieser Zustand wird für die Methode 2 – Merkmalsfusion, wie in Kapitel 5.2.6 beschrieben, genutzt, welche die Merkmalsverteilung nach einem Training ermöglicht. Nach Abschluss des gesamten Trainings und der damit erfolgreichen Erstellung der Referenzklassenvektoren beginnt die Verteilung der Referenzklassenvektoren per Broadcast an die registrierten Nachbarknoten. Jeder trainierte Referenzklassenvektor wird dabei in einem eigenen Paket versendet. Für jede trainierte Klasse versendet jeder Sensorknoten folglich eine Nachricht.

Die Sensorknoten speichern die erhaltenen Merkmale in einer festgelegten Reihenfolge ab. Die so gespeicherten Merkmale fungieren als Referenzvektor für das verteilte Muster. Sollen nicht die Merkmale verteilt werden sondern die Klassen, muss eine andere Vorgehensweise verwendet werden. Die „Train-Distribution“ wird in dem Fall der Methode 1 – Klassifikationsfusion, wie in Kapitel 5.2.6 beschrieben, nicht verwendet. Beispiele zur Auswertung von verteilten Klassifizierungsergebnissen werden im „Evaluation“-Zustand behandelt (siehe Kapitel 5.2.6).

Genau wie die Sensorknoten in der „Hello“-Phase, wie in Kapitel 5.2.1 beschrieben, einer festen Reihenfolge zugeordnet werden, werden die Merkmale anderer Sensorknoten in der gleichen Reihenfolge gesichert. Der Merkmalsvektor der einzelnen Sensorknoten wird somit um die Merkmalsvektoren der registrierten Nachbarknoten ergänzt und vergrößert sich um den Faktor der Anzahl der an der Registrierung beteiligten Sensorknoten. Werden beispielsweise drei Sensorknoten mit je sechs Merkmalen für die Erkennung eines Ereignisses verwendet, vergrößert sich der Referenzklassenvektor jeder Kasse für die verteilte Erkennung auf 18 Merkmale bzw. Dimensionen.

5.2.6 „Evaluation“-Zustand

Es gibt nach dem Austausch der Daten verschiedene Möglichkeiten, diese auszuwerten. In Abhängigkeit von den Daten, die ausgetauscht werden, können die jeweiligen Evaluationsalgorithmen verwendet werden. Für die Auswertung dieser Daten dürfen jedoch keine Datenlücken vorkommen, da eine Datenlücke oder ein fehlendes Teilmuster so gewertet wird, als ob einer der Sensorknoten keine Daten gesendet hat und damit auch kein Muster erkannt hat. Es kann zwar zwischen einem aktiven Sensorknoten, der kein Muster erkannt hat, und einem ausgefallenem Sensorknoten, der nicht antwortet (siehe „Idle“-Zustand), unterschieden werden, jedoch ist die Konsequenz in dieser Arbeit dieselbe: Es kann keine erfolgreiche Auswertung stattfinden. In einem fortgeschrittenen System könnte die verteilte Mustererkennung soweit angepasst werden, dass sie auf eine veränderte Sensorknotenzahl entsprechend dynamisch reagiert. Dazu zählt das Auswählen eines Auswertungsknotens, der sich aus der Nachbarschaft die nötigen Daten zusammensucht und diese entsprechend dem Vorkommen bewertet.

Im Folgenden werden vier in dieser Arbeit verwendeten Evaluationsmethoden beschrieben, die sich auf die in Abbildung 4.1 vorgestellten geometrischen Figuren stützen. Deren Konkretisierung für den „Evaluation“-Zustand wird aufgezeigt. Die Konzepte, auf denen die untersuchten

Methoden basieren, werden in Kapitel 5.1 erläutert. In allen Methoden gilt der gleiche Versuchsablauf:

Drei Sensorknoten sollen dieselbe geometrische Figur nachzeichnen. Erkannt wird nach der Fusionierung, welches Muster die drei Sensorknoten zeichnen sollten. Ein dedizierter Auswertungsknoten wird nicht gewählt, daher werden die Daten auf allen Sensorknoten fusioniert. Es genügt jedoch zunächst zu zeigen, dass eine verteilte Erkennung überhaupt erfolgreich und mit Erkennungsratenverbesserungen durchzuführen ist. Für die Optimierung des Systems durch die Wahl eines Auswertungsknoten wird auf Kapitel 7.1verwiesen.

- Methode 1 – Klassifikationsfusion
  Die drei geometrischen Figuren werden auf der Klassifikationsebene fusioniert. Werden mehr als 50 % (es sind auch andere Stimmverhältnisse denkbar) der Figuren als „gleich" erkannt, wird zugunsten der Mehrheit entschieden und als Ausgabe die Mehrheitsentscheidung gewählt, ansonsten kann keine Entscheidung getroffen werden. Erkennen beispielsweise zwei von drei Sensorknoten einen Kreis, wird das Ergebnis als Kreis festgelegt. Das Nicht-Treffen einer Entscheidung wird als falsche Entscheidung (falsch positiv) gewertet.

  Die Klassifikationsdaten werden in dem „Distribution"-Zustand verteilt. Alle gesammelten Klassifikationsergebnisse können daraufhin nach einem Demokratie-Entscheid bewertet werden.

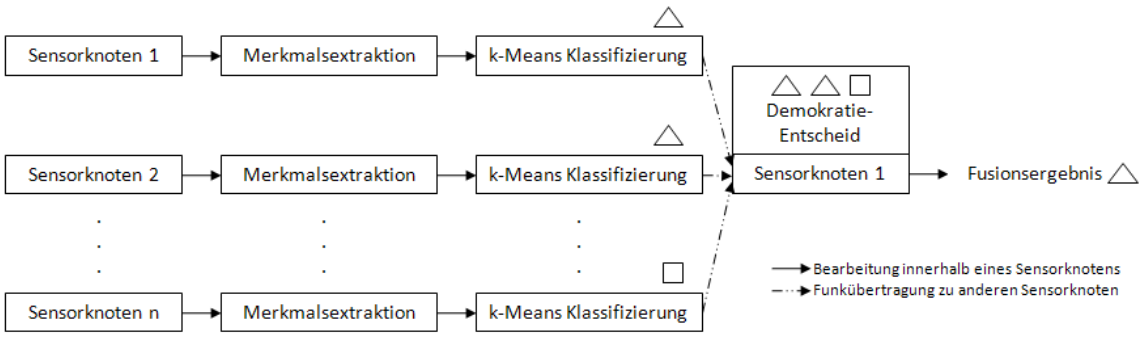

Abbildung 5.3 abstrakte Darstellung der Klassifizierungsfusion

Im hier vorgestellten Demokratieverfahren werden die Ergebnisse der Sensorknoten alle gleichgewichtet bewertet. Wie in der abstrakten Darstellung in Abbildung 5.3 gezeigt, findet damit eine Fusionierung auf der Klassifizierungsebene statt.

- Methode 2 – Merkmalsfusion
  Die drei geometrischen Figuren werden auf der Merkmalsebene fusioniert. Die extrahierten Merkmale der Sensorknoten werden in vorgegebener Reihenfolge (kleinste Sensorknoten-ID zuerst) in einem gemeinsamen Merkmalsvektor abgelegt. Nach demselben Prinzip, wie die Mustererkennung dieser Arbeit auf dem Einzelsensoren vollzogen

wird, wird die Mustererkennung mit den Merkmalen aller beteiligten Sensorknoten durchgeführt. Dabei vergrößert sich die Dimension der Merkmalsvektoren um den Faktor der Anzahl der Sensorknoten. Jeder der n Sensorknoten liefert aus seiner lokalen Mustererkennung zunächst m Merkmale, die in der verteilten Lösung zu einem Merkmalsvektor der Dimension $m \cdot n$ zusammengesetzt werden.

Die extrahierten Referenzklassenvektoren des jeweiligen Sensorknotens müssen bereits im „Train-Distribution"-Zustand an alle Nachbarknoten verteilt werden.

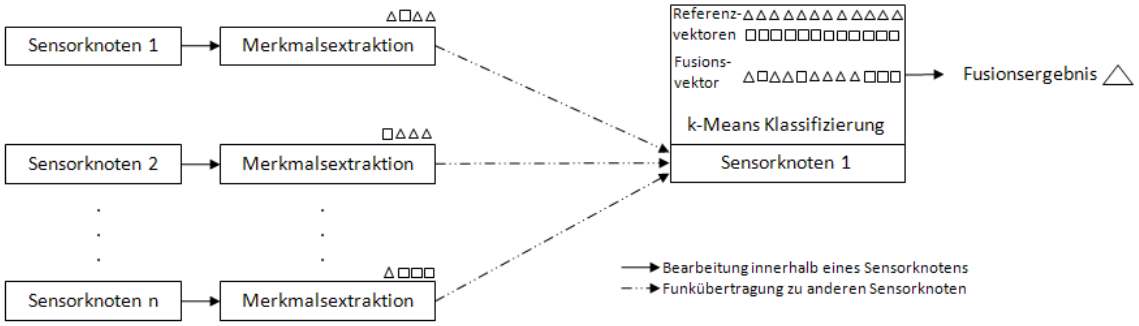

Abbildung 5.4 vereinfachte Darstellung der Merkmalsfusion

Berechnet werden wieder die Euklidischen Abstände aus den in der Trainingsphase gebildeten fusionierten Referenzklassenvektoren zu dem unbekannten Muster, das aus mehreren Sensorknoten einen fusionierten Merkmalsvektor liefert. Es gilt hier ebenfalls, dass der kürzeste Euklidische Abstand die Entscheidung über das Klassifikationsergebnis liefert. Wie aus der abstrakten Darstellung in Abbildung 5.4 hervorgeht, findet damit eine Fusionierung auf der Merkmalsebene statt. Die Merkmalsfusion hat damit den enormen Vorteil, dass der Klassifizierungsalgorithmus aus der lokalen Klassifizierung unverändert übernommen werden kann.

- Methode 3 – Kooperative Fusion
  In einem kritischen Entscheidungsfall können mit dieser Methode zusätzlich die Merkmalsdaten angefordert werden. In dieser Arbeit soll die kooperative Fusion nach dem Prinzip des Mehrheitsentscheides betrachtet werden. Sobald mehr als 50 % der Sensorknoten eine gemeinsame Entscheidung fällen können, wird diese verwendet. Andernfalls werden die Merkmalsdaten angefordert und eine Merkmalsfusion durchgeführt. In der Implementierung ist eine separate Betrachtung der Methode 3 nicht nötig, da sich die nötigen Messwerte aus den Analysen der Methode 1 sowie der lokalen Erkennung direkt ableiten lassen. Die Merkmalsfusionsergebnisse der Methode 2 werden für die Methode 3 dann herangezogen, wenn in der lokalen Erkennung keine 50%-Mehrheit ein und dasselbe Klassifizierungsergebnis feststellt.

- Methode 4 - Kooperative Fusion mit Vetorecht
  Diese Methode basiert auf Methode 3. Es wird im Unterschied zu Methode 3 das Demokratierecht durch ein für alle Sensorknoten gültiges Vetorecht ersetzt. Diese kooperative Fusion soll bereits dann Merkmalsdaten anfordern, wenn auch nur ein Sensorknoten ein Klassifikationsergebnis liefert, welches sich von denen der anderen unterscheidet. In der Implementierung ist eine separate Betrachtung der Methode 4 nicht nötig, da sich die nötigen Messwerte aus den Analysen der Methode 1 sowie der lokalen Erkennung direkt ableiten lassen. Die Merkmalsfusionsergebnisse der Methode 2 werden für die Methode 4 dann herangezogen, wenn während der lokalen Erkennung bereits ein einzelner Sensorknoten eine andere Klassifizierung als die übrigen Sensorknoten feststellt.

Nach Abschluss der Evaluation muss als Folge dieses Ergebnis an eine hier nicht weiter betrachtete Basisstation gesendet werden, die über vorhandene Multihop-Netztopologien erreichbar sein muss.

5.2.7 Erweiterung des Automaten

Die Erweiterung des Zustandsautomaten um die verteilte Mustererkennung ist in Abbildung 5.5 dargestellt. Um die Komplexität des Automaten zu reduzieren, wird der Automat nach [Har86] in mehrere Detailansichten unterteilt. Die Abbildung 5.5 basiert auf den Beschreibungen der Abbildung 4.19. Die rote Umrandung hebt die für verteilte Erkennung eingeführten Zustände, wie in Kapitel 5.2 beschrieben, hervor. Die Aktivierung und Deaktivierung des Funkchips sind durch entsprechende Symbolik gekennzeichnet. Im „Hello“-Zustand werden die Identitäten der Sensorknoten ausgetauscht. Im „Train-Distribution“-Zustand werden in Methoden 2, 3 und 4 alle trainierten Merkmale ausgetauscht. Im Falle der Methode 1 werden an dieser Stelle keine Daten ausgetauscht.

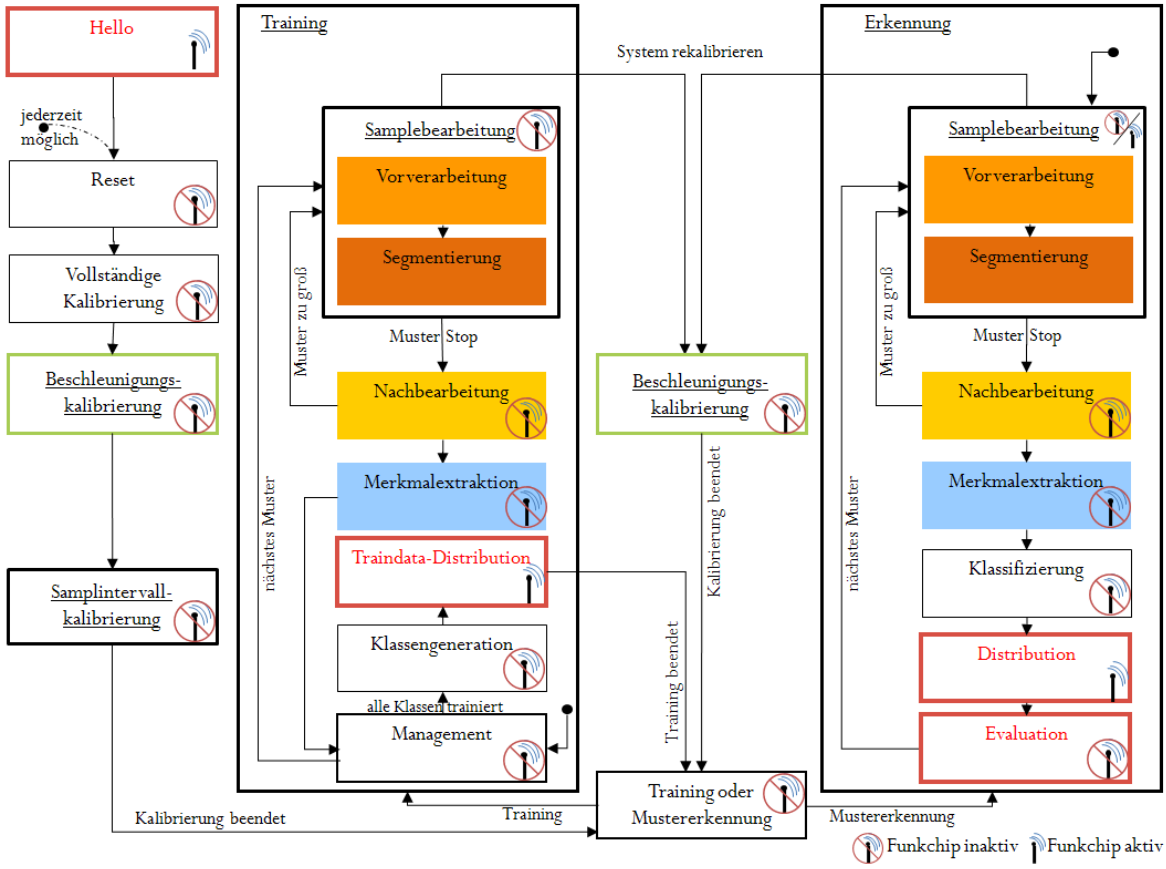

Abbildung 5.5 Um die Zustände der verteilten Mustererkennung (rote Elemente) erweiterter Automat

Der „Distribution“ -Zustand verteilt für das aktuell erkannte Muster methodenabhängig jeweils das erkannte Muster und die Merkmale. Für die Untersuchungen in dieser Arbeit werden beide Daten übertragen, die hier weiterhin als Erkennungsdaten betitelt werden. In dem „Evaluation“-Zustand werden die eigenen Daten mit denen der Nachbarsensorknoten kombiniert und ausgewertet. Die Ausgabe des Evaluationsergebnisses erfolgt anhand der LED’s des MSB-S. Für den Fall eines Alarms muss eine entsprechende Implementierung für die an die Evaluation anzuschließende Alarminstanz erfolgen, siehe Kapitel 7.1. Der Funkchip wird immer dann deaktiviert, wenn das System nicht gestört werden sollte. Dies gilt insbesondere während der Kalibrierung, des Trainings der verschiedenen Klassen sowie der konkreten Mustererfassung. Es ist möglich, sogar während der Musterbearbeitung in der Erkennungsphase den Funkchip zu aktivieren. Da die Zeit, in der der Chip aktiv ist, jedoch nur sehr kurz ist und zudem die Ergebnisberechnungen verzögert werden können, wird der Funkchip in der aktuellen Betrachtung ebenfalls inaktiv geschaltet. Die Berechnung der Klassifizierung ist zeitkritisch, da ein Sensorknoten die Ergebnisse in der verteilten Erkennung versendet und dafür nur das gemeinsame Sendezeitfenster zur Verfügung hat. Eine Aktivierung des Funkchips könnte folglich während der Musterbearbeitung dazu führen, dass sich die Sendefenster nicht mehr überschneiden.

Die Samplingbearbeitung der Erkennungsphase ist um die Zustände „Idle“ und „Off“ erweitert, siehe dazu die Detaildarstellung in Abbildung 5.6, deren Grundlagen mit Abbildung 4.22 eingeführt werden.

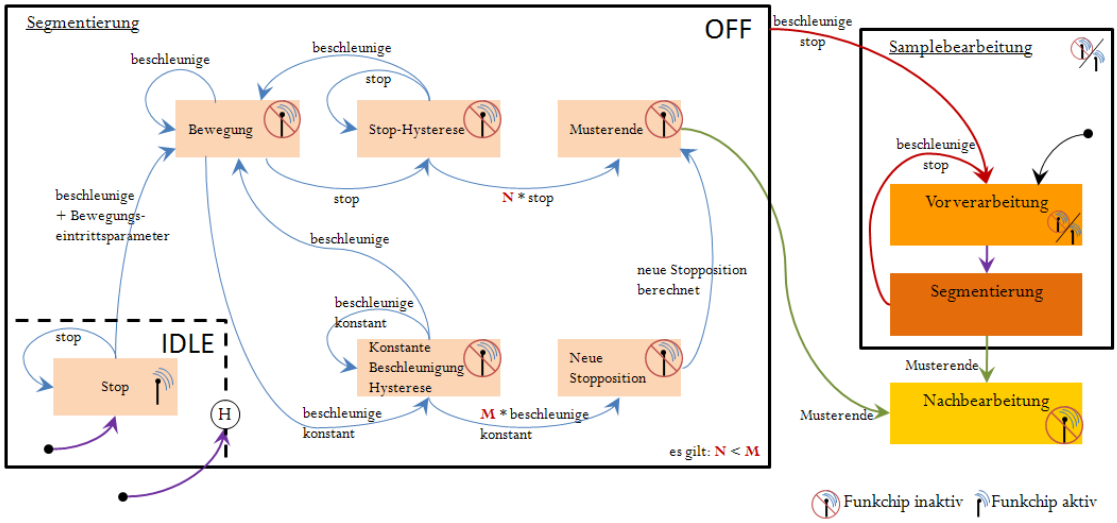

Abbildung 5.6 Die Samplebearbeitung der Erkennungsphase ist um die verteilten Erkennungszustände „Idle“ und „Off“ erweitert. Zusätzlich werden die Funkchipaktivitäten aufgezeigt

In Abbildung 5.6 werden sowohl der „Idle“-Zustand als auch der „Off“-Zustand in der Zustandsimplementierung visualisiert. Eine detaillierte Beschreibung der Grundlagen, auf denen Abbildung 5.6 basiert, wird in Kapitel 4.5 mittels Abbildung 4.22 vorgenommen. Die Samplebearbeitung während der Erkennungsphase befindet sich im „Idle“-Zustand (siehe Kapitel 5.2.3), solange sich der Sensorknoten in der Ruhephase befindet. Erst bei Erzeugung von Beschleunigungen und dem damit verbundenen Wechsel in die Beschleunigungshysterese wird das Funkmodul ausgeschaltet, um eine Unterbrechung der Mustererfassung zu verhindern. Bei jedem erfassten Beschleunigungswert wechselt der Zustand von der Vorverarbeitung in die Segmentierung. Der „Idle“-Zustand bleibt solange erhalten bis ein Muster beginnt. Es wird das Funkmodul deaktiviert und es bleibt für jedes folgende Sample (Beschleunigungswert) deaktiviert. Nach erneutem Einfinden des Sensorknotens in der Ruhephase endet das Muster und es folgt die Musterbearbeitung. Erst wenn die Musterbearbeitung (beginnend bei der Nachbearbeitung) abgeschlossen ist und sich der Sensorknoten wieder im „Idle“-Zustand befindet, kann das Funkmodul in der Samplebearbeitung der Erfassungsphase aktiviert werden. Dies geschieht, um möglichen Anfragen nach Musterergebnissen mitteilen zu können, dass kein Muster erfasst wurde. Damit ist es möglich, im Falle eines ausgefallen Sensorknotens, dies anhand einer solchen Nachricht zu unterscheiden.

Während der Vorverarbeitung ist der Funkchip, abhängig von dem letzten durchlaufenen Zustand, aktiv oder inaktiv. Beim ersten Eintritt in die Samplebearbeitung ist der Funkchip deaktiviert, da das System mit diesem Zustand initialisiert wird. Ändert sich während der Segmentierung der Zustand von „Idle“ zu „Off“, bleibt dieser Zustand in der Vorverarbeitung des nächsten Samples erhalten. Die Vorverarbeitung wechselt folglich nie aktiv den Radiozustand. Nur in der Segmentierung wird der Radiozustand in Abhängigkeit von der Beschleunigung aktiv verändert. Die meiste Zeit sollte sich das System in dem „Idle“-Zustand befinden, wenn davon ausgegangen wird, dass Muster in der Realität selten vorkommen. Eine Zaunüberwachung sollte beispielsweise so funktionieren, dass der „Idle“-Zustand nur dann verlassen wird, wenn wirklich eine

Beschleunigung der Stärke stattfindet, die einem Überwindungsversuch oder anderen Ereignissen gleichkommen könnte. Abgesehen von natürlichen Umwelteinflüssen wie Wind und Tieren, die eine ständige Zaunbewegung verursachen können, ist zu prüfen, inwieweit sich die hier vorgestellten Techniken in der Realität als einsetzbar erweisen.

Es ist zu analysieren, inwieweit der Energieverbrauch durch die lange Funkbereitschaft im „Idle"-Zustand die Lebensdauer des Systems verringert und gegebenenfalls an dieser Stelle den Funk zu deaktivieren.

6 AUSWERTUNG

Es werden in den folgenden Kapiteln Analysen der Systemparameter der lokalen und verteilten Erkennung durchgeführt. Es werden bewertende Analysen der Erkennungssysteme untereinander und zu einem verwandten Erkennungssystem vorgenommen. Die Bewertung des Kommunikationsaufwandes im Bezug zur erreichten Erkennungsrate wird ebenso erstellt wie eine Betrachtung der Ergebnisse unter Berücksichtigung probandenspezifischer Einflüsse. Für die Beurteilung der Erkennungssysteme werden typische Kennwerte vorgestellt und eingesetzt.

6.1 Systemparameter

Das System ist durch verschiedene Parameter veränderbar. Insbesondere die Samplingfrequenz könnte unter Umständen die Qualität des Systems beeinflussen. Eine Erhöhung der verwendeten Frequenz von 50 Hz auf 100 Hz lässt eine verbesserte Erkennung erwarten. Es werden drei verschiedene Frequenzen experimentell untersucht, 50Hz, 100 Hz und 130 Hz. Das Ziel der Untersuchung ist dabei, die Erkennungsqualität bisheriger Untersuchungen mit 50 Hz unter Auswahl geeigneter Merkmalen und Parameteränderungen zu verbessern oder wenigstens beizubehalten. Für die 50 Hz-Einstellung werden sechs Histogrammmerkmale ausgewählt, die als Vorgabe für die erhöhten Samplingfrequenzeinstellungen gelten.

Die Erkennungsqualität kann durch die Erhöhung der Samplingfrequenz bei gleicher Merkmalszahl jedoch nicht erreicht werden. Dies wird darauf zurückgeführt, dass bei erhöhter Samplingfrequenz die Datendetails mehr Informationen in das Muster einbringen, als sechs Merkmale ausreichend charakterisieren können. Um die bisherige Erkennungsqualität zu erreichen, wird für die erhöhte Samplingfrequenz eine größere Merkmalszahl notwendig. Die Erkennungsqualität des 50 Hz-Systems wird bei einer Samplingfrequenz von 100 Hz mit neun Merkmalen und bei 130 Hz erst bei 14 Merkmalen erreicht.

Schwerpunktmäßig werden Histogrammmerkmale ausgewählt, da diese eine höhere Invarianz gegenüber dem konkreten Bewegungsablauf der Geometrie der Muster aufweisen können. Da zunächst die Verwendung von je acht Histogrammmerkmalen pro Beschleunigungsachse implementiert wird (siehe Kapitel 4.4.6), sind bei 14 verwendeten Merkmalen nahezu alle Merkmale in der Verwendung. Eine differenzierte Auswahl zwischen geeigneten und ungeeigneten Merkmalen ist damit nicht mehr möglich.

Aufgrund der erhöhten Samplingfrequenz muss zusätzlich der verfügbare Speicherplatz auf 250 bis 300 Samples erweitert werden. Die Merkmalszahl der Achsen wird auf je 16 Klassen pro Achse erhöht, um das Problem der zu geringen Zahl von Merkmalen entgegenzuwirken. Trotz dieser Optimierung der Menge der auswählbaren Histogrammparameter erzielt das System keine Verbesserung. Damit ist für die Histogrammmerkmale anzunehmen, dass keine weiteren Erkennungsoptimierungen mit Histogrammmerkmalen zu erreichen ist. Alternative Merkmale, die in Zukunft zu implementieren sind, können an diesem Punkt eingreifen.

Eine erhöhte Samplingfrequenz hat auf den Sensorknoten negative Folgen. Zum einen ist zu beachten, dass bei steigender Frequenz der Speicherverbrauch auf den Sensorknoten entsprechend zunimmt, da in konstanter Zeit mehr Daten erhoben werden. Neben dem proportional zur Samplingfrequenz steigenden Platzbedarf auf den Sensorknoten ist der steigende Merkmalsaufwand zu beachten, der im Folgenden einen erhöhten Rechenaufwand bei der Berechnung des Euklidischen Abstandes mit sich bringt. Eine Rechtfertigung für eine erhöhte Samplingfrequenz kann demnach nur eine deutliche Verbesserung der Erkennungsqualität sein. Ist diese Verbesserung nicht zu erzielen, wie in dem hier vorliegenden Experiment, sollte darauf verzichtet werden.

6.2 Kennwerte

Für die Betrachtung der Ergebnisse werden verschiedene statistische Kennwerte betrachtet, um eine Klassifikationsbewertung vornehmen zu können. Die verwendeten Kennwerte basieren auf dem Prinzip der bedingten Wahrscheinlichkeit und verwenden die nachfolgenden Begriffe:

- richtig positiv: wird verwendet, wenn ein gesuchtes Muster korrekt klassifiziert wird
- richtig negativ: wird verwendet, wenn ein gesuchtes Muster nicht vorliegt und dies korrekt festgestellt wird
- falsch positiv: wird verwendet, wenn ein gesuchtes Muster gefunden wird, obwohl es nicht vorliegt
- falsch negativ: wird verwendet, wenn das gesuchte Muster nicht klassifiziert wird, obwohl es vorliegt

Die **Sensitivität** (*engl.: sensitivity*) wird als das Verhältnis von richtig erkannten Mustern eines Typs zu allen bisher erfolgten Mustern dieses Typs definiert. Dies entspricht der Wahrscheinlichkeit, dass ein gesuchtes Muster richtig erkannt wird.

$$Sensitivität = \frac{\#richtig\ positiv}{\#richtig\ positiv + \#falsch\ negativ}$$

Die **Spezifität** (*engl.: specificity*) wird als das Verhältnis von richtig erkannten anderen Mustern zu allen anderen erzeugten Mustern definiert. Dies entspricht der Wahrscheinlichkeit, dass es keinen Fehlalarm gibt.

$$Spezifität = \frac{\#richtig\ negativ}{\#richtig\ negativ + \#falsch\ positiv}$$

Relevanz und Segreganz sagen aus, mit welcher Wahrscheinlichkeit eine positive Aussage wirklich positiv ist und eine negative Aussage wirklich negativ ist.

Die **Relevanz** (*engl.: positive predictive value*) beschreibt das Verhältnis von richtig erkannten Mustern zu richtig erkannten Mustern und den fälschlich als richtig erkannten Mustern. Dies entspricht der Wahrscheinlichkeit, dass ein erkanntes Muster aus vielen verschiedenen Mustern wirklich das erkannte Muster ist.

$$Relevanz = \frac{\#richtig\ positiv}{\#richtig\ positiv + \#falsch\ positiv}$$

Die **Segreganz** (*engl.: negative predictive value*) ist das Verhältnis von korrekt nicht erkannten anderen Mustern zu allen nicht erkannten Mustern. Dies entspricht der Wahrscheinlichkeit, dass ein Muster, wenn es nicht als solches erkannt wurde, tatsächlich nicht das gesuchtes Muster war.

$$Segreganz = \frac{\#richtig\ negativ}{\#richtig\ negativ + \#falsch\ negativ}$$

Die **Korrektklassifikationsrate** (*engl.: accuracy*) ist der Anteil aller richtig klassifizierten Objekte. Mit der Korrektklassifikationsrate werden die Aussagen der Kennwerte Sensitivität, Spezifität, Relevanz und Segreganz zusammengefasst.

$$Korrektkl. - Rate = \frac{(\#richtig\ positiv + \#richtig\ negativ)}{\#richtig\ positiv + \#richtig\ negativ + \#falsch\ positiv + \#falsch\ negativ}$$

6.3 Lokale Erkennung

Zur Langen Nacht der Wissenschaft 2007 konnte bereits mit einem ersten lokalen Erkennungssystem eine Erhebung durchgeführt werden. Besucher wurden gebeten, die geometrischen Figuren aus Abbildung 4.1 nachzuzeichnen. Die geometrische Figur wurde den Versuchspersonen als eine gleichmäßige Bewegung vorgeführt, um sicher zu stellen, dass keine großen Pausen und Beschleunigungswechsel in die Bewegung einfließen. Die Probanden sollten daraufhin die geometrischen Figuren mindestens je einmal wiederholen, siehe Abbildung 6.1. Die Ergebnisdaten werden mit den in Kapitel 6.2 vorgestellten Kennwerten bewertet.

Abbildung 6.1 Besucher der Langen Nacht der Wissenschaft 2007 erzeugen Evaluationsergebnisse für das zu diesem Zeitpunkt bestehende, lokale Erkennungssystem

6.3.1 Bewertung – Kennwerte

Die Ergebnisse basieren auf 70 Versuchen pro Muster bzw. Klasse. Es konnten folglich während der langen Nacht der Wissenschaft 280 Muster erfasst werden. Die Auswertungsergebnisse der Abbildung 6.2 zeigen, dass die Sensitivität im Mittel bei 65 % liegt. Die Spezifität erreicht im Mittel einen Wert von 88,4 %.

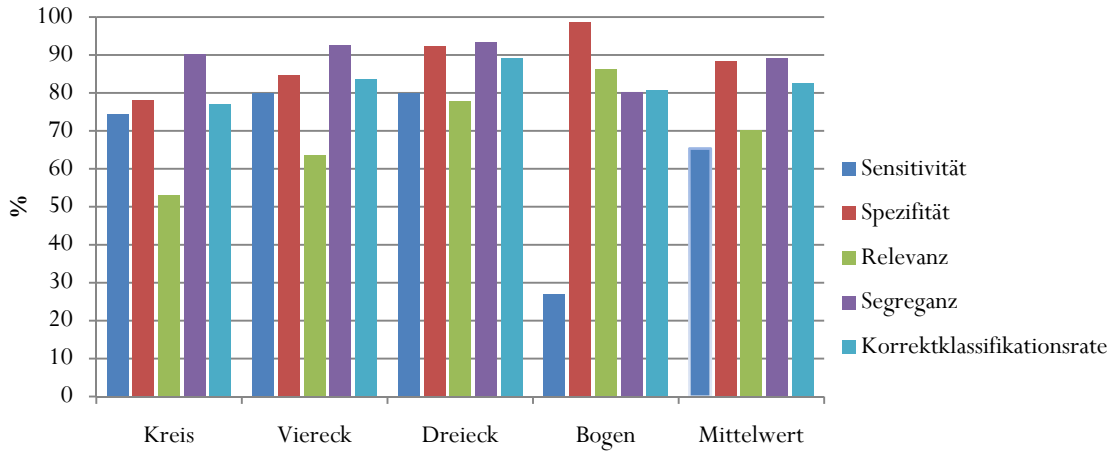

Abbildung 6.2 Ergebnisse der lokalen Erkennung während der Langen Nacht der Wissenschaft 2007

Dieser erste Versuch wurde zum damaligen Zeitpunkt bereits als Erfolg gewertet, da insbesondere nicht der Trainierende, sondern projektfremde Personen die Muster nachzeichneten. Deutliche Probleme sind in der Sensitivität des Bogens (27 %) zu verzeichnen. Das bedeutet, dass bei einer Fehlerkennung eines Musters gehäuft das Ergebnis „Bogen“ ausgegeben wurde. Der so einhergehende hohe und vermeintlich gute Wert der Spezifität des Bogens von 98,5 % relativiert sich damit. Zudem zeigen die Korrektklassifikationsrate und die niedrigen Relevanzen, dass das System zu diesem Zeitpunkt mit einer guten Musterdifferenzierung noch erhebliche Schwierigkeiten hatte.

Im weiteren Verlauf der Arbeit werden Programmfehler im System behoben sowie die in Kapitel 4.4.6 eingeführte Pufferzone implementiert. Außerdem wird die Normierung der Daten in dem Wertebereich 0 bis 100 auf 0 bis 255 erhöht. Der vergrößerte Wertebereich lässt eine höhere Genauigkeit bei der Daten- und Merkmalsnormierung zu.

Ein weiterer Versuchslauf mit dem lokalen Erkennungssystem wird von drei Probanden vollzogen (siehe Abbildung 6.3), die während des Projektverlaufes und des Trainings der Sensorknoten keinen Kontakt zum Projekt und keine Kenntnis von dem Projekt hatten. Insgesamt wurde jedes Muster 480-mal durch alle Probanden erzeugt. Die so entstehenden Ergebnisse sind daher mit denen der Langen Nacht der Wissenschaft 2007 gut vergleichbar.

Abbildung 6.3 Versuchsaufbau mit projektfremden Probanden zur Erfassung von lokalen und verteilten Ereignissen

Die Ergebnisse sind in Abbildung 6.4 dargestellt. Im Mittel ist eine Verbesserung der Spezifität auf 92,9 %, der Sensitivität auf 78,8 % und der Korrektklassifikation auf 89,3 % zu beobachten. Die so ermittelten Daten unterstreichen die gute Funktionalität der Erkennung. Anhand der verhältnismäßig niedrigen Werte der Sensitivität des Kreises und der Relevanz des Bogens sowie der Spezifität und der Relevanz des Dreiecks ist zu vermuten, dass hier ein Zusammenhang besteht. Durch die Versuchsreihen wird diese Vermutung bestätigt. Konkret wird tatsächlich bei der Durchführung eines Kreismusters in gehäufter Form das Muster des Bogens erkannt und bei einer Person insbesondere bei der Durchführung des Dreiecks gehäuft ein Viereck erkannt. Aufgrund der strukturellen Ähnlichkeiten des Bogens und des Kreises liegt eine besondere Schwierigkeit darin, diese beiden Muster gut zu unterscheiden. Eine ähnliche Problematik liegt für das Dreieck und das Viereck vor. Die Strukturen der runden Figuren untereinander sowie der eckigen Figuren untereinander sind ähnlicher zueinander als die der eckigen zu den runden Figuren.

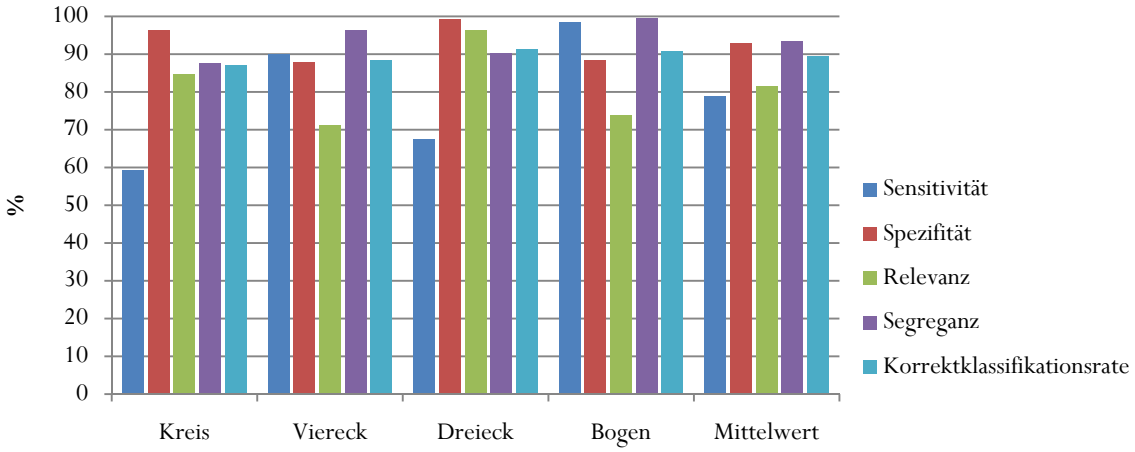

Abbildung 6.4 Ergebnisse der lokalen Erkennung nach Verbesserungen im Erkennungssystem

Im Vergleich mit den Ergebnissen der Arbeit von Wittenburg et *al.* [Wit07] in Abbildung 6.5 geht deutlich hervor, dass das hier vorgestellte System eine deutlich höhere Genauigkeit erzielt. Insbesondere werden hier exemplarisch vier verschiedene Muster für die Erkennung innerhalb eines Systems differenziert. In der Arbeit von Wittenburg et *al.* hingegen ist ein festgelegtes Ereignis aus sechs Ereignissen zu erkennen. Es musste folglich bei Nichtauftreten des festgelegten Ereignisses kein Unterschied gemacht werden, welches Ereignis stattgefunden hat. Die sehr hohe Sensitivität bei einer verhältnismäßig geringen Spezifität lässt vermuten, dass das Hauptereignis „über den Zaun klettern“ zu oft als „falsch positiv“ erkannt wurde und damit zu der hohen Erkennungsrate geführt hat. Die im Vergleich dazu relativ guten Werte des optimierten Erkennungssystems dieser Arbeit unterstreichen die Genauigkeit der Erkennung. Auf der Basis dieser Erkennungsrate wird in Kapitel 6.4 die verteilte Erkennung bewertet.

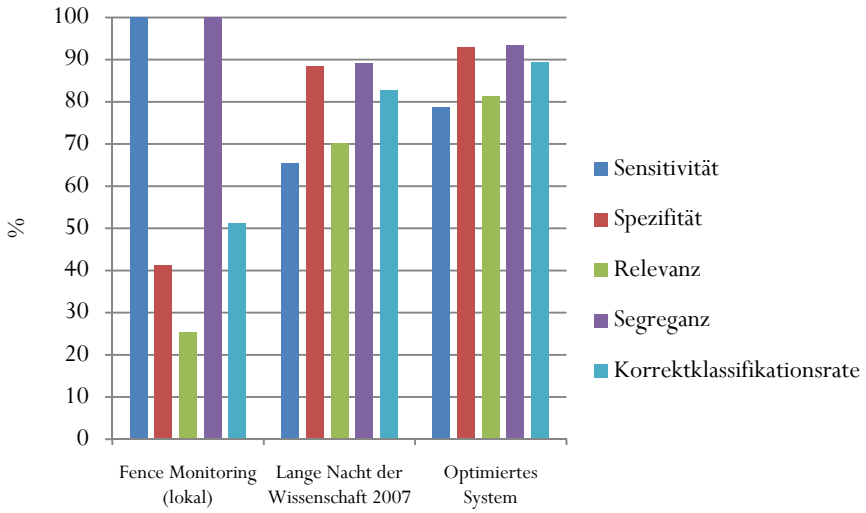

Abbildung 6.5 Systemvergleich lokaler Erkennungssysteme. Es werden die Mittelwerte des hier vorgestellten System und deren Optimierung mit der Arbeit von Wittenburg [Wit07] verglichen

Verbesserungen des hier vorgestellten Systems in der lokalen Erkennung im Vergleich zur Arbeit von Wittenburg et *al.* sind in der Sensitivität nicht zu verzeichnen, was jedoch an der hohen „falsch positiv"-Rate des Fence Monitoring-Systems liegt. In einer direkten Gegenüberstellung der Kennwerte wird bei der optimierten lokalen Erkennung eine Verringerung der Sensitivität um 21,3 Prozentpunkte und eine Verbesserung der Spezifität um 51,6 Prozentpunkte zum Fence Monitoring festgestellt.

Die Verbesserungen des Systems der Langen Nacht der Wissenschaft 2007 zum optimierten lokalen System schwanken zwischen 4,3 Prozentpunkten in der Segreganz bis hin zu 13,4 Prozentpunkten in der Sensitivität. In der Steigerung der Korrektklassifikationsrate um 6,7 Prozentpunkte werden zusammenfassend die Auswirkungen der in diesem Kapitel angesprochenen Systemkorrekturen aufgezeigt.

Im Zusammenhang betrachtet, stellt die 100 %ige Sensitivität bei einer geringen Spezifität das System von Wittenburg et *al.* als überempfindlich dar, da offensichtlich viele Ereignisse falsch als Warnung interpretiert werden. Die einseitige Betrachtung der Sensitivität ist somit ungeeignet. Dies führt zum Schluss, dass selbst eine Verringerung der Sensitivität bei hoher Spezifität ein verlässliches Erkennungssystem beschreibt. Ein weiterer guter Vergleichsfaktor ist die Korrektklassifikationsrate, welche im Vergleich zur Arbeit von Wittenburg et *al.* im optimierten System um 38,3 Prozentpunkte verbessert wird.

6.4 Verteilte Erkennung

Die Ergebnisse der verteilten Erkennung teilen sich in vier untersuchte Methoden auf: Die in Kapitel 5.2.6 vorgestellte Methode der Klassifikationsfusion, die Methode der Merkmalsfusion, die Methode der kooperativen Fusion und die Methode der kooperativen Fusion mit Vetorecht. Es soll untersucht werden, inwieweit sich die Erkennungsgenauigkeit (Sensitivität, Spezifität, Relevanz, Segreganz und Korrektklassifikationsrate) durch den verteilten Erkennungsansatz verändert. Es soll im Weiteren überprüft werden, ob der Einsatz der kooperativen Fusion unter Beachtung des aufkommenden Datenvolumens rentabel ist. Darüberhinaus soll überprüft werden, in welchen Schwankungsgraden die Erkennungsqualität von den jeweiligen Probanden abhängt.

6.4.1 Methodenbewertung - Kennwerte

Für die verteilte Erkennung wurden drei projektfremde Probanden ausgewählt, die während des Projektverlaufes und des Trainings der Sensorknoten keinen Kontakt zum Projekt und keine Kenntnis von dem Projekt hatten. Die Probanden sollten jeweils dasselbe Muster mit einem Sensorknoten auf der Vorlage aus Abbildung 4.1 erzeugen. Die Bewegung muss nahezu zeitgleich von allen Probanden vollzogen werden, damit die Daten im Zeitfenster der Übertragung ausgetauscht werden können. Es wurden von jedem Probanden insgesamt 160 Ereignisse pro Muster gezeichnet. Nach Abschluss einer Bewegung tauschen die Sensorknoten ihre Daten aus.

Trotz der Einfachheit des Übertragungsprotokolls und der Koordinierung dreier Menschen, die 160 Versuche wiederholt zeitgleich durchführen sollten, sind 97 % der Datenübertragungen in allen Versuchen erfolgreich, siehe Abbildung 6.6.

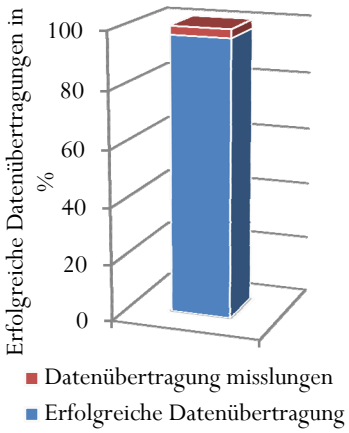

Abbildung 6.6 Ereignisauswertungen nach Kommunikation: 97 % aller Versuche der verteilten Erkennung konnten nach erfolgreicher Datenübertragung ausgewertet werden

Die Betrachtung der verteilten Erkennung ist auf der Basis einer 97 %igen Wahrscheinlichkeit, dass die Übertragung der Daten funktionieren wird, als sinnvoll zu erachten. In der Methode 1 der Klassifikationsfusion (siehe Abbildung 5.3) bewerten die Sensorknoten die Ergebnisse nach der Mehrheitsentscheidung. Ist bei den mehr als 50 % der Sensorknoten dasselbe Ergebnis verzeichnet worden, wird die Klassifikationsfusion aller Klassendaten die Mehrheit als endgültige Entscheidung festlegen. Die Ergebnisse werden per LED angezeigt. Das System hat weiterhin Schwierigkeiten mit der Kreiserkennung, die sich basierend auf den Problemen der lokalen Erkennung hier fortführen und in der Sensitivität mit 63% und einer Segreganz von 88,8% ausdrücken, siehe Abbildung 6.7. Der Kreis wurde demnach, wenn er erzeugt wurde, nicht so oft erkannt wie andere Muster. Zudem zeigt die hohe Spezifität auf, dass das Muster „Kreis“ nicht durch Fehlalarme belastet ist und die ebenfalls hohe Relevanz zeigt auf, dass, wenn ein Kreis gefunden wurde, er es wirklich war. Folglich hat der Kreis Probleme überhaupt erkannt zu werden.

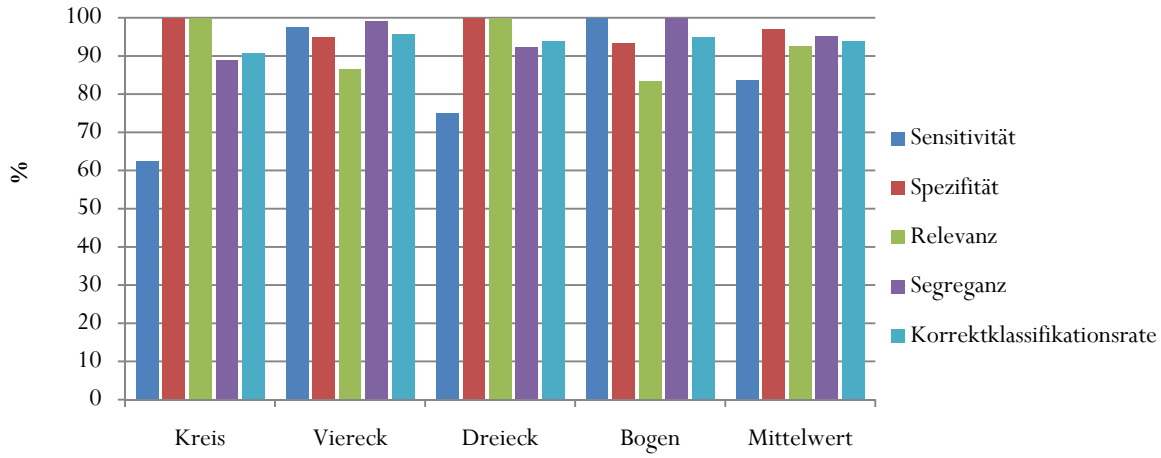

Abbildung 6.7 Ergebnisse der verteilten Erkennung mittels Klassifikationsfusion

Die verteilte Erkennung soll im Folgenden mittels der Merkmalsfusion (Methode 2) ausgewertet und mit der Klassifikationsfusion verglichen werden. Der besondere Unterschied zur Klassifikationsfusion ist die Verwendung der verteilt erfassten Merkmale und deren Fusion in einem vergrößerten Merkmalsvektor. Die Auswertung ergibt eine deutliche Verbesserung der Sensitivität auf 91,9 % und eine geringe Verbesserung der Spezifität auf 97,3 %. Zudem ist die Korrektklassifikationsrate auf 95,9 % angestiegen. Gegenüber der Klassifikationsfusion stellt sich damit die Merkmalsfusion deutlich besser dar. Das bei der Merkmalsfusion entstehende erhöhte Datenvolumen soll an dieser Stelle ignoriert werden. Die grafische Darstellung der Ergebnisse für die Merkmalsfusion ist in Abbildung 6.8 aufgezeigt. Insbesondere fällt auf, dass die Kreiserkennung deutlich bessere Ergebnisse erzielt, wird sie verteilt mit der Merkmalsfusion ausgeführt (vergleiche dazu Abbildung 6.4 und Abbildung 6.7 mit Abbildung 6.8).

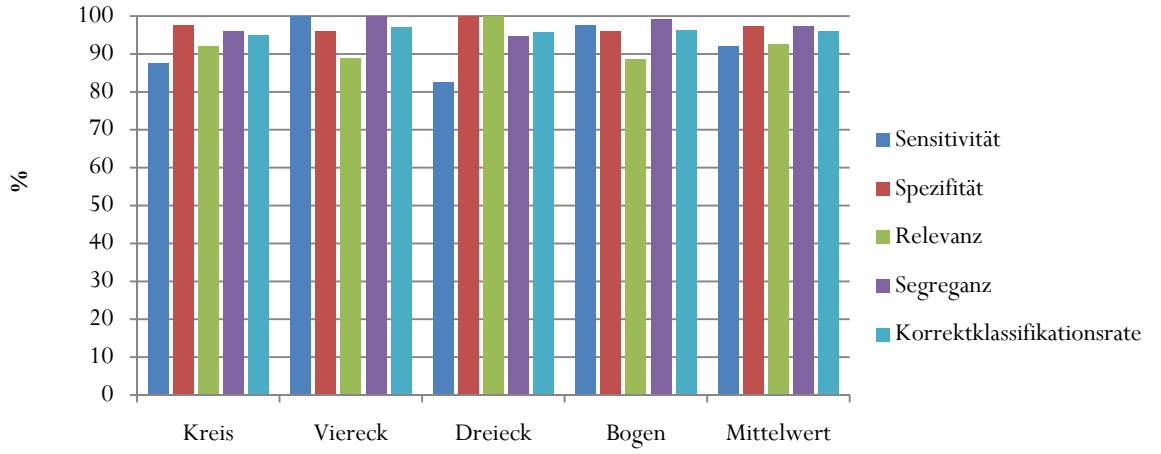

Abbildung 6.8 Ergebnisse der verteilten Erkennung mittels Merkmalsfusion

In Abbildung 6.9 werden die Auswirkungen verteilter Erkennungsmethoden gemittelt und im Vergleich mit dem verteilten Erkennungssystem Fence Monitoring betrachtet. Vergleicht man die Fusionsmethoden untereinander, so bleibt die Spezifität mit Schwankungen um ± 1 % relativ konstant. Die Verbesserung der Sensitivitäten von der Klassifikationsfusion zur kooperativen

Fusion beträgt in der Sensitivität 6,9 Prozentpunkte und von der kooperativen Fusion zur Merkmalsfusion noch 1,3 Prozentpunkte. Die Verbesserungen der Korrektklassifikationsrate zeigen leichte Steigerungen von der Klassifikationsfusion zur kooperativen Fusion um 1,6 Prozentpunkte und von der kooperativen Fusion zur Merkmalsfusion noch 0,6 Prozentpunkte. Die Unterschiede der Erkennungsqualitäten entsprechen den Erwartungen aus Kapitel 5.1. Der Zuwachs an Erkennungsqualität durch die Merkmalsfusion ist so deutlich zu erkennen. Insbesondere der Qualitätszuwachs in der Merkmalsfusion wird durch einen erhöhten Kommunikationsaufwand erreicht. Es ist daher abzuwägen, welche Methode einzusetzen ist. Die kooperative Fusion mit Vetorecht (Methode 4) setzt sich von allen Methoden knapp ab. Der Unterschied der Erkennungsqualität liegt zur Merkmalsfusion jedoch stets unter einem Prozentpunkt, womit eine Gleichstellung der beiden Methoden in Betracht zu ziehen ist. Aufgrund des zusätzlich anfallenden Datenvolumens wird die Methode 4 gegenüber der Merkmalsfusion (Methode 2) unter dem Gesichtspunkt der Kommunikation neu bewertet. Dazu wird in Kapitel 6.5 der Kommunikationsaufwand der verschiedenen Methoden miteinander verglichen.

Im Vergleich zur Arbeit von Wittenburg et *al.* in Abbildung 6.9 zeigt sich, dass gegenüber dem Fence Monitoring System eine Weiterentwicklung erzielt werden konnte, die das hier vorgestellte Erkennungssystem in jeder Methode als verlässlich beschreiben lässt. Die verteilte Erkennung wird durch die Merkmalsfusion in der Sensitivität um 5,2 Prozentpunkte, in der Spezifität um 44,0 Prozentpunkte, in der Relevanz um 65,3 Prozentpunkte, in der Segreganz um 2,1 Prozentpunkte und in der Korrektklassifikationsrate um 37,0 Prozentpunkte verbessert. Die Methode 4 mit Vetorecht erhöht die Korrektklassifikationsrate gegenüber dem Fence Monitoring sogar um 37,4 Prozentpunkte.

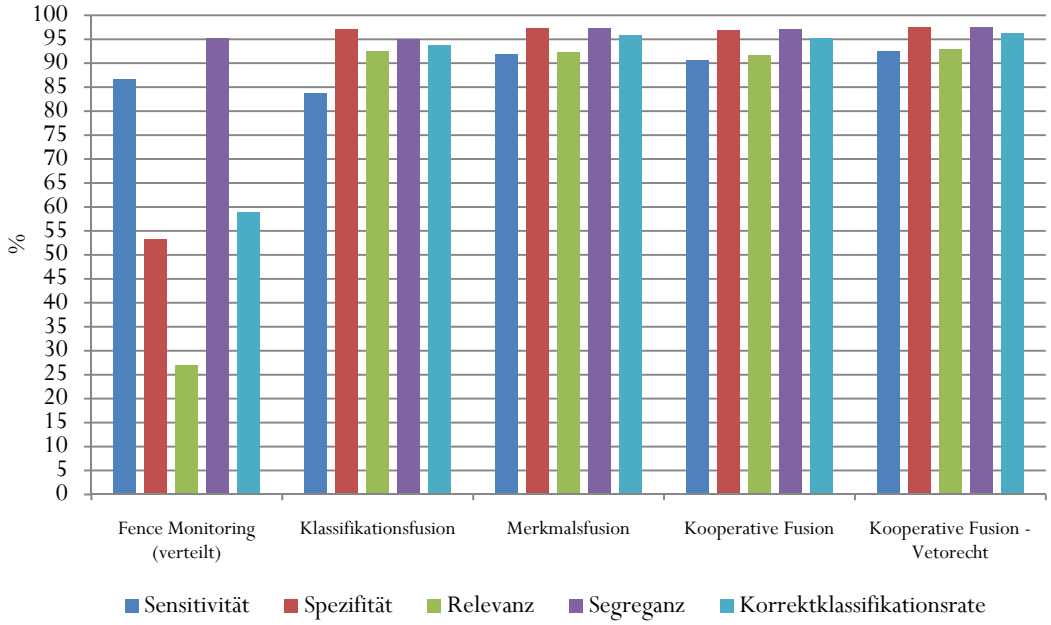

Abbildung 6.9 Statistische gemittelter Wertevergleich zur Bewertung der vorgestellten verteilten Methoden dieser Arbeit mit Fence Monitoring

Nahezu alle jeweils gemittelten Parameter erreichen in der verteilten Erkennung des hier entwickelten Systems Kennwerte über 90 %. Dies zeigt, dass die Gesamtqualität der verteilten Erkennung des hier vorgestellten Systems als sehr gut zu bezeichnen ist. Die höchste Erkennungsqualität erreicht die kooperative Fusion mit Vetorecht für die Kennwerte der Sensitivität mit 93 %, der Spezifität mit 98 % und der Korrektklassifikationsrate mit 96 %. Die zuverlässigste und genaueste Erkennung leistet damit die kooperative Fusion mit Vetorecht.

6.4.2 Lokale vs. verteilte Erkennung

Im Vergleich zur lokalen, optimierten Lösung wird die Spezifität für die Klassifikationsfusion im Mittel auf 97,1 %, die Sensitivität auf 83,8 % erhöht. Die genannten, deutlichen Verbesserungen sowie die Verbesserungen der gemittelten Korrektklassifikationsrate auf 93,8 % zeigen, dass sich bereits ein einfacher Mehrheitsentscheid in einem verteilten Erkennungssystem positiv auswirkt. Damit ist erwiesen, dass eine verteilte Erkennung bereits auf der Ebene der Klassifizierung eine Verbesserung der Gesamterkennung mit sich bringen kann.

Die Vermutung, dass sich Muster in einer verteilten Mustererkennung besser beobachten und analysieren lassen [Rus07] , wird hier bestätigt. Die verteilte Erkennung führt in dem hier vorgestellten System mit jeder verteilten Methode zu einer verbesserten Erkennung, wird sie mit dem lokalen Erkennungssystem verglichen.

In Abbildung 6.10 wird das Fence Monitoring und das hier entwickelte System mit ihren optimalen Erkennungsqualitäten im Vergleich dargestellt. Die Verbesserung der Sensitivität von dem optimierten lokalen Erkennungssystem zur Merkmalsfusion betragen 13,1 Prozentpunkte, die der Spezifität 4,4 Prozentpunkte, die der Relevanz 11,0 Prozentpunkte, die der Segreganz 4,0 Prozentpunkte und die der Korrektklassifikationsrate 6,6 Prozentpunkte. Diese Betrachtung ist insbesondere deswegen interessant, da sie keine Veränderungen der Erkennungstechnik oder deren Parameter im lokalen Erkennungssystem erfahren hat. Die Verbesserung lässt sich ausschließlich auf die Anwendung der verteilten Fusion zurückführen. Damit ist gezeigt, dass die verteilte Erkennung eine deutliche Steigerung der Erkennungsqualität mit sich bringt.

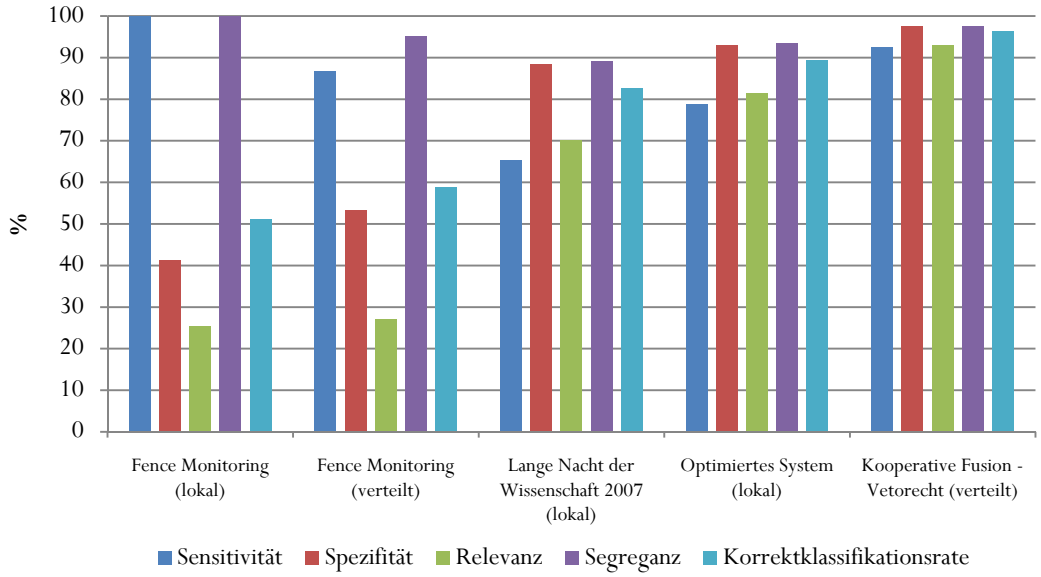

Abbildung 6.10 Vergleich des gesamten Fence Monitoring Systems mit den in dieser Arbeit vorgestellten lokalen und erkennungstechnisch besten verteilten System

Der Fortschritt der Systemqualitäten ist deutlich an den Kennwerten ablesbar und stellt sich besonders prägnant in der Korrektklassifikationsrate dar. Die Kennwerte sind zusammen betrachtet mit dem hier eingeführten System und weiter verfeinerten verteilten Erkennung besser geworden. Die Aussagekraft der Kennwerte ist im Zusammenhang betrachtet besonders mächtig. Wie anhand der Abbildung 6.10 gut zu erkennen ist, müssen sinkende Kennwerte nicht zwingend einer Verschlechterung der Gesamterkennung aussagen. Als Beispiel wird die Sensitivität aus Abbildung 6.10 hervorgehoben welche vom lokalen Fence Monitoring zum verteilten Fence Monitoring um 13 Prozentpunkte fällt und dann um weitere 22 Prozentpunkte zum lokalen System der Langen Nacht der Wissenschaft 2007 absinkt. Die Aussage des sinkenden Kennwertes der Sensitivität, steht aber im Zusammenhang mit den steigenden Kennwerten, die eine deutlich bessere Gesamterkennung beschreiben, bei der die Systeme in der genannten Reihenfolge weniger Fehlalarme produzieren (Aussage der Spezifität) und gleichzeitig steigt die Wahrscheinlichkeit, dass ein erkanntes Ereignis wirklich das erkannte Ereignis ist. Die positiven Effekte übersteigen insgesamt in der Erkennung die negativen Effekte und führen zur kontinuierlichen Steigerung der Korrektklassifikationsrate.

6.5 Kommunikation

Im folgenden Abschnitt wird die anfallende Kommunikation analytisch, die bei den jeweiligen verschiedenen Methoden erwartet wird, untersucht. Interessant ist insbesondere für ein skalierbares System, wie ein Sensornetz, das Kommunikationsverhalten bei unterschiedlicher Anzahl von Sensorknoten. Alle hier im Funkverkehr betrachteten Datenvolumen basieren auf der Annahme, dass ein verlustfreier Datenverkehr hergestellt wird. Damit wird allein die anfallende Datenmenge betrachtet. Es werden Datenvolumen untersucht, die für die jeweiligen Methoden anfallen. Für die Kooperationsmethoden ergibt sich Gleichung 11, in der die anfallende Datenmenge B_k für n Sensorknoten ermittelt wird. K entspricht der Bytegröße eines Paketes mit Klas-

sifikationsdaten. Die Variable p ist die Wahrscheinlichkeit, dass der Sensorknoten mit den Klassifikationsdaten bereits eine erfolgreiche Klassifikationsfusion durchführen kann. p wird hier primäre Auswertungsquote genannt. R drückt in Bytes aus, wie groß ein Broadcast-Anfragepaket ist, das eine Merkmalsfusion zusätzlich initiieren soll. M entspricht der konstanten Paketgröße eines Antwortpaketes ohne Merkmale. Für jedes mit diesem Paket versendete Merkmal wird ein Byte zusätzlich benötigt. Damit entspricht die Anzahl m der Merkmale der Anzahl der zusätzlich benötigten Bytes. Die Antwortpakete M sind von jedem beteiligten Sensorknoten zu versenden, außer von dem Sensorknoten, der die Anfrage mit dem Paket R stellt.

$$B_k = K * n + (1 - p) * (R + (M + m) * (n - 1)) \quad (11)$$

Das aufkommende Datenvolumen B_{Kf} der Klassifikationsmethoden ergibt sich aus Gleichung 12 und das Datenvolumen B_{Mf} für die Merkmalsfusion aus Gleichung 13.

$$B_{Kf} = K * n \quad (12)$$

$$B_{Mf} = M * n \quad (13)$$

Die für die Analyse veranschlagten Bytegrößen sind für die in den Gleichungen 11-13 verwendeten Konstanten in Tabelle 6.1 eingetragen und lassen sich aus den Datenstrukturen der Implementierung direkt ableiten.

Tabelle 6.1 Paketgrundgrößen gemäß Implementierung in Bytes

Konstante	K	R	M
Bytes	21	14	20

K ist um ein Byte größer als M, da K immer einen Klassifikationswert übermittelt. R als Anfragepaket benötigt nur eine minimale Paketgröße, während K und M zusätzlich einen Zeitstempel beinhalten, um sicher stellen zu können, dass veraltete Ereignisse nicht betrachtet werden.

6.5.1 Theoretische Analyse

In diesem Kapitel wird das Datenaufkommen aufgrund theoretischer Annahmen bezüglich p analysiert. Die Methoden 3 und 4, werden unter verschiedenen Qualitäten der Erkennungsrate betrachtet. Kann in der Kooperationsmethode die Klassifikationsfusion nicht durchgeführt werden, müssen Daten für die Merkmalsfusion angefordert werden. Das Anfordern weiterer Datenpakete bedeutet konkret, dass ein Anfragepaket ausgesendet wird und die Merkmale von allen anderen Sensorknoten zusätzlich zu den bereits ausgetauschten Klassifikationsmerkmalen zurückgesendet werden müssen. In der vorliegenden Betrachtung in Abbildung 6.11 ergibt sich, dass ab einer primären Auswertungsquote von ca. 79 % die Kooperationsmethoden unrentabel werden, da für dieses p mehr Daten versendet werden müssen, als bei der Merkmalsfusion.

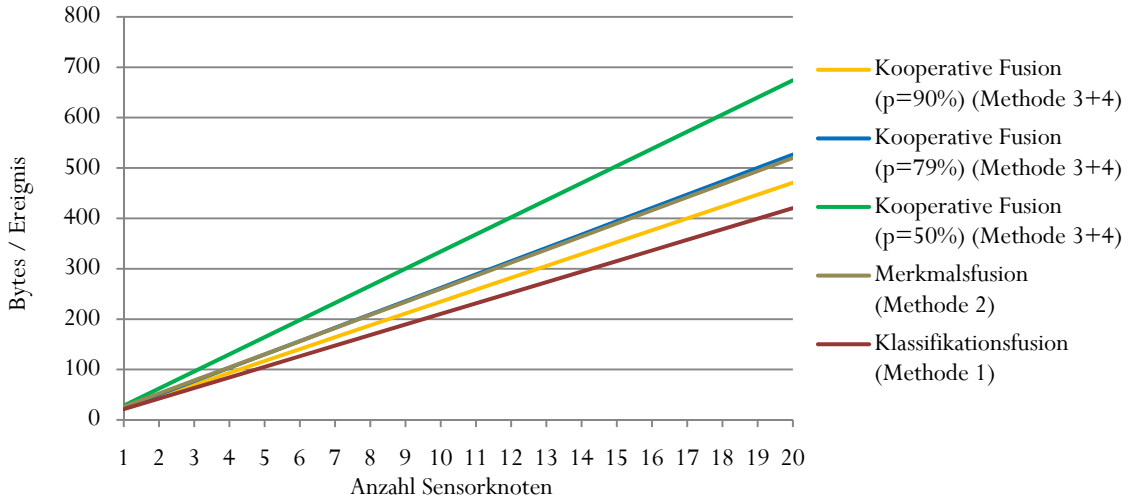

Abbildung 6.11 Spezifischer Methodenvergleich des Datenvolumen bei sechs Merkmalen mit theoretischer primärer Auswertungsquote der Methoden 3 bzw. 4

Die Klassifikationsfusion erzeugt nach Gleichung 11 ein geringeres Datenaufkommen, was darauf zurückzuführen ist, dass nur die stark komprimierten Klassifikationsdaten der höchsten Abstraktionsebene zusätzlich versendet werden müssen. Bei der Merkmalsfusion hingegen steigt der Aufwand aufgrund des konstanten Merkmalsaustausches bei jeder Klassifizierung.

Eine konkrete Empfehlung zur Verwendung der Klassifikationsfusion bedingt durch den geringen Datenverkehr widerspricht der Empfehlung für die Merkmalsfusion mit Vetorecht (Methode 4) aufgrund einer verbesserten Korrektklassifikationsrate. Eine Entscheidung für eine Methode kann aus der Betrachtung der Abbildung 6.11 nur unter dem Gesichtspunkt des geringsten zu erwartenden Datenaufkommen getroffen werden. Als Kompromiss bietet die kooperative Fusion (Methode 3) eine Möglichkeit, ein optimales Kosten-Nutzen-Verhältnis wählen zu können. In Abhängigkeit von veränderlichen Kennwerten und veränderlicher Zahl der Merkmale wird diese Aussage jedoch neu zu bewerten sein.

Die Datenvolumen ändern sich mit steigender Merkmalsmenge. Eine Veränderung der Datenaufkommen ist somit zu erwarten und führt zu einer nötigen Neubewertung der Empfehlung für eine Methode. In Abbildung 6.12 wird nach Gleichung 11 ermittelt, ab welcher Merkmalsmenge sich der Datenaufwand der Merkmalsfusion soweit erhöht, dass er für alle weiteren Merkmalsmengenerhöhungen über dem der kooperativen Fusion (für p= 90%) liegt

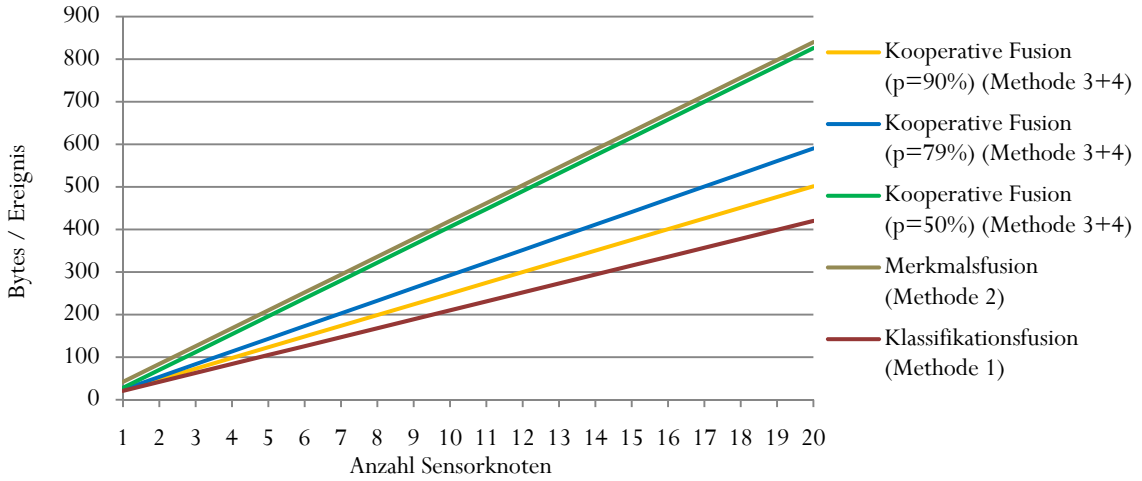

Abbildung 6.12 Spezifischer Methodenvergleich des Datenvolumens bei 22 Merkmalen mit theoretischer primärer Auswertungsquote der Methode 3 bzw. 4

Bei einer Erhöhung der Merkmalszahl wird ab 22 Merkmalen das Datenvolumen der Merkmalsfusion bereits bei einer Sensitivität von 90 % umfangreicher als das einer kooperativen Fusion mit einer Sensitivität von 50 % (siehe Abbildung 6.12). Damit ist eine weitere Neueinschätzung der optimal einzusetzenden Methode für Veränderung der Merkmalsmenge nötig.

6.5.2 Spezifische Analyse

Im Gegensatz zu Kapitel 6.5.1, in welchem das Datenaufkommen aufgrund theoretischer Annahmen bezüglich *p* analysiert wird, soll hier das spezifische *p* betrachtet werden. Aus den Auswertungen der vollzogenen Experimente kann das spezifische *p* für die hier vorgestellte Versuchsanordnung konkret ermittelt und in Abbildung 6.13 mit den Kooperationsmethoden betrachtet werden.

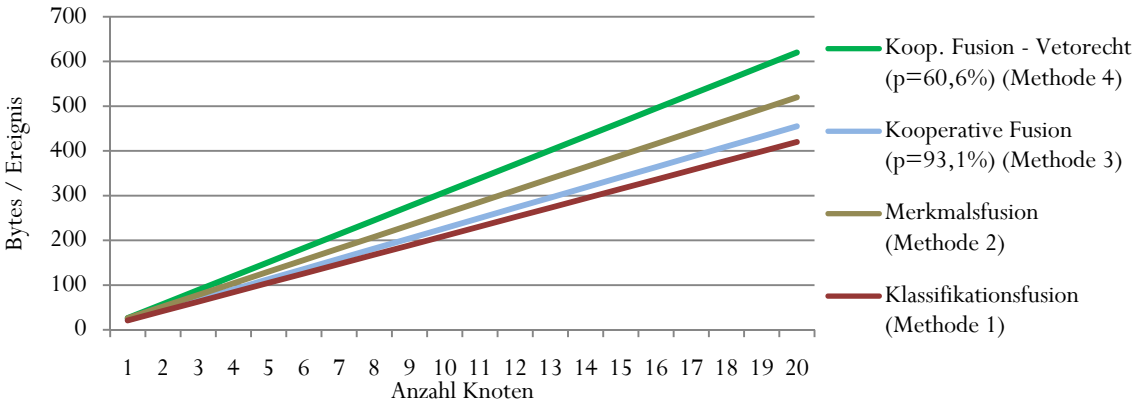

Abbildung 6.13 Spezifischer Methodenvergleich des Datenvolumens bei sechs Merkmalen. *p* wird mittels Versuchen konkret ermittelt

In der kooperativen Fusion (Methode 3) werden, wie in Kapitel 5.2.6 beschrieben, die Merkmale anderer Sensorknoten dann angefordert, wenn in dem hier vorgestellten Anwendungsfall

alle drei Sensorknoten ein unterschiedliches Klassifikationsergebnis vorlegen. Mit einer Wahrscheinlichkeit von p=93,1 % tritt dieser Fall in durchgeführten Experimenten nicht ein und ein weiteres Datenanfordern ist nicht nötig.

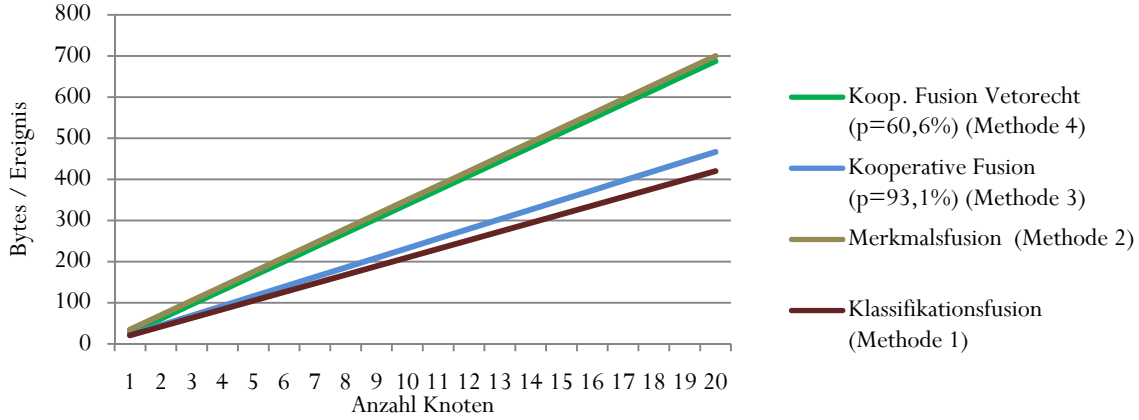

Abbildung 6.14 Spezifischer Methodenvergleich des Datenvolumens bei 15 Merkmalen mit spezifischem p der Methode 3 und 4

Zusätzlich wird die kooperative Fusion mit Vetorecht (Methode 4) betrachtet. Die kooperative Fusion mit Vetorecht soll bereits dann Daten anfordern, wenn auch nur ein Sensorknoten ein Klassifikationsergebnis liefert, welches sich von denen der anderen unterscheidet. Mit einer Wahrscheinlichkeit von p=60,6 % tritt dieser Fall in durchgeführten Experimenten nicht ein. Die kooperative Fusion erreicht damit eine der höheren Datenvolumen der hier betrachteten Methoden und wird von der Merkmalserkennung erst bei der Verwendung von 15 Merkmalen im Datenvolumen übertroffen. Damit ist die Merkmalsfusion bei Betrachtung des Datenaufwandes der Methode 4 vorzuziehen, wenn weniger als 15 Merkmale verwendet werden, siehe Abbildung 6.14. Eine Neubewertung dieser Aussage ist dennoch zu vollziehen, da jede Merkmalsmengenänderung auch eine Änderung der Erkennungsqualität verursachen kann und sich damit das spezifische p ändert. Die hier gegebenen Analysen geben demnach Richtungen vor, auf denen in Zukunft aufgebaut werden kann und an denen zukünftige verteilte Erkennungssysteme eine Orientierung finden können.

6.5.3 Analyse der Rentabilität

Unter dem Gesichtspunkt einer Kosten-Nutzen-Analyse werden in Abbildung 6.15 die Korrektklassifikationsrate der jeweiligen Methoden und der entsprechenden Datenvolumen im Verhältnis zueinander betrachtet. Damit ist eine theoretische Aussage möglich, wie sehr ein versendetes Byte statistisch die Qualität der Erkennung beeinflusst. Aufgrund der guten Aussagekraft der Korrektklassifikationsrate über die Gesamtqualität der Mustererkennung wird diese im Folgenden für eine theoretische Kosten-Nutzen-Analyse als Nutzfaktor eingesetzt. Das Aufkommen der zu versendenden Bytes dagegen ist als Kostenfaktor zu interpretieren. Das theoretische Kosten-Nutzen-Verhältnis „Korrektklassifikationsrate pro versendetes Byte“ wird für das hier eingesetzte Sensornetz betrachtet, in welchem drei Sensorknoten miteinander kommunizieren müssen, um ein Muster zu erkennen.

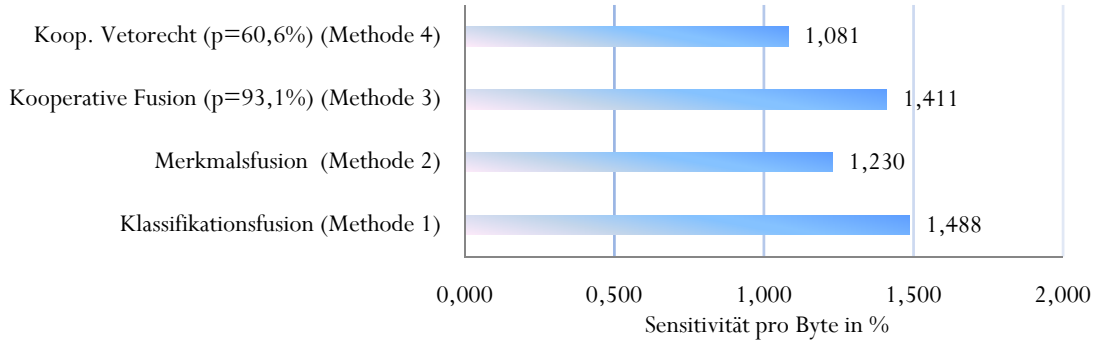

Abbildung 6.15 Korrektklassifikationsrate pro versendetes Byte in % bei drei Sensorknoten

Abbildung 6.15 bewertet die Klassifikationsfusion (Methode 1) am rentabelsten mit 1,488 % erreichter Korrektklassifikationsrate pro versendetes Byte. Die kooperative Fusion (Methode 3) mit einer Rentabilität von 1,411 % erreichter Korrektklassifikationsrate pro versendetes Byte schneidet bereits ungünstiger ab. Am unrentabelsten wirken sich die bei leichten Unsicherheiten angeforderten Merkmalsdaten der kooperativen Fusion mit Vetorecht aus. Die Merkmalsfusion ist eine präzise Methode, deren Qualität mit einem höheren Datenaufkommen erreicht wird, das hier konkret 1,23 % pro Byte beträgt. Nicht zu empfehlen ist damit die kooperative Fusion mit Vetorecht (Methode 4), da sie mit 1,081 % Korrektklassifikationsrate pro Byte als am unrentabelsten einzustufen ist.

Die Empfehlungen beziehen sich in dieser Betrachtung auf die Rentabilität. Eine gute Rentabilität muss nicht die besten Erkennungsergebnisse liefern, wie die Auswertungen dieser Betrachtung im Vergleich zu den Betrachtungen des Kapitels 6.4.2 zeigen. Demnach ist im Einzelfall abzuwägen, welche Kriterien (die Erkennungsqualität, das Datenaufkommen der Kommunikation, die Rentabilität der Erkennungsqualität im Verhältnis zur Kommunikation) als Entscheidungsfaktor für eine der Methoden im Einsatz vorrangig wichtig sind.

6.6 Probandenspezifische Analyse

Es werden Ergebnisse von drei projektfremden und einer projektvertrauten Person erfasst und die Streuung der Ergebnisse verglichen. Die projektvertraute Person nahm das Training für die Sensorknoten vor, während die projektfremden Personen keinerlei Kontakt während des Projektverlaufes und des Trainings zum Projekt hatten. Es ergeben sich Resultate, die von der projektvertrauten Person besser ausfallen als die der projektfremden. Die Unterschiede in der Erkennungsqualität lassen sich auf die regelmäße Auseinandersetzung der projektvertrauten Personen mit dem Bewegungsablauf der geometrischen Figuren zurückführen. Aus diesem Grund basieren alle bisherigen Auswertungen der Ergebnisse auf Versuchen mit projektfremden Probanden. Verbesserungen der Kennwerte lassen sich durch projektvertraute Personen in einem relativ geringen Rahmen, wie in Abbildung 6.16 zu sehen, erzielen.

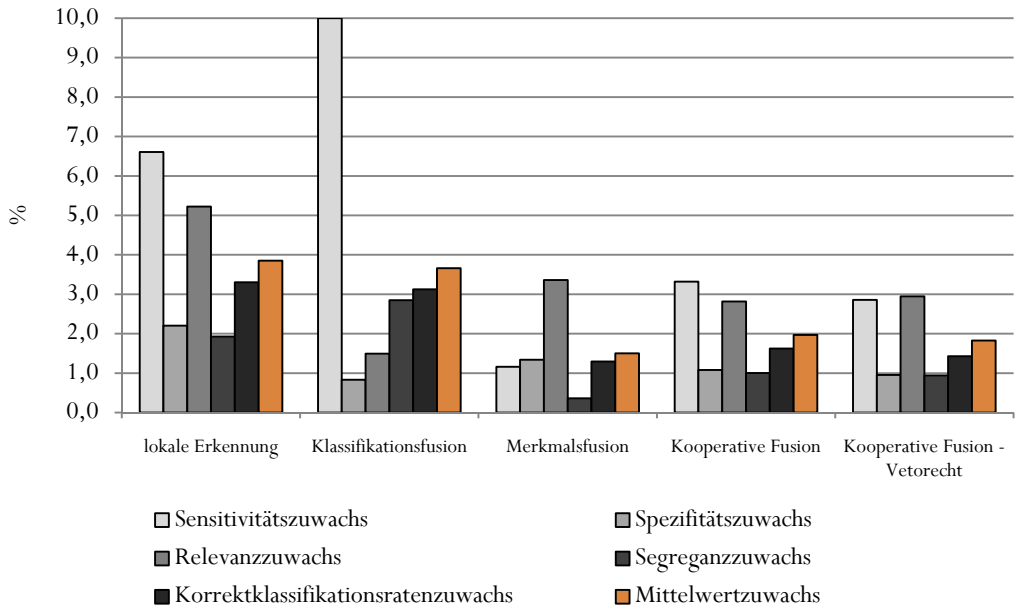

Abbildung 6.16 Vergleich von Kennwertverbesserungen (in Prozentpunkten) durch projektvertraute Personen

Die Ergebnisse der projektvertrauten Personen liegen im Mittel um 1,5 Prozentpunkte bis 3,9 Prozentpunkte über denen der projektfremden Personen. Aufgrund der Vertrautheit mit dem System ist dieser Unterschied zu erwarten. Diese Ergebnisse zeigen zudem auf, in welchem Maß das System durch projektvertraute Personen positiv beeinflusst bzw. verfälscht werden kann. Eine zukünftige Untersuchung des Mustererkennungssystems sollte unter Berücksichtigung dieser Ergebnisse vollzogen werden, um den Rahmen des Einflusses auf Erhebungsergebnisse durch projektnahe Personen möglichst kontrollierbar und gering zu halten.

Die projektfremden Probanden sowie die projektvertrauten Probanden erzielen unterschiedliche Qualitäten der Ergebnisse aus der lokalen Erkennung. Die Streuung der so erzeugten für jeden Probanden gemittelten Kennwerte zeigt auf, wie das System auf unterschiedliche Bedienung reagiert. Die erzielten Kennwerte streuen, wie in Abbildung 6.17 gezeigt, zwischen 2,7 und 9,7 Prozentpunkten. Das System wurde von der projektinternen Person trainiert und erreicht durch diese Person die besten Kennwerte.

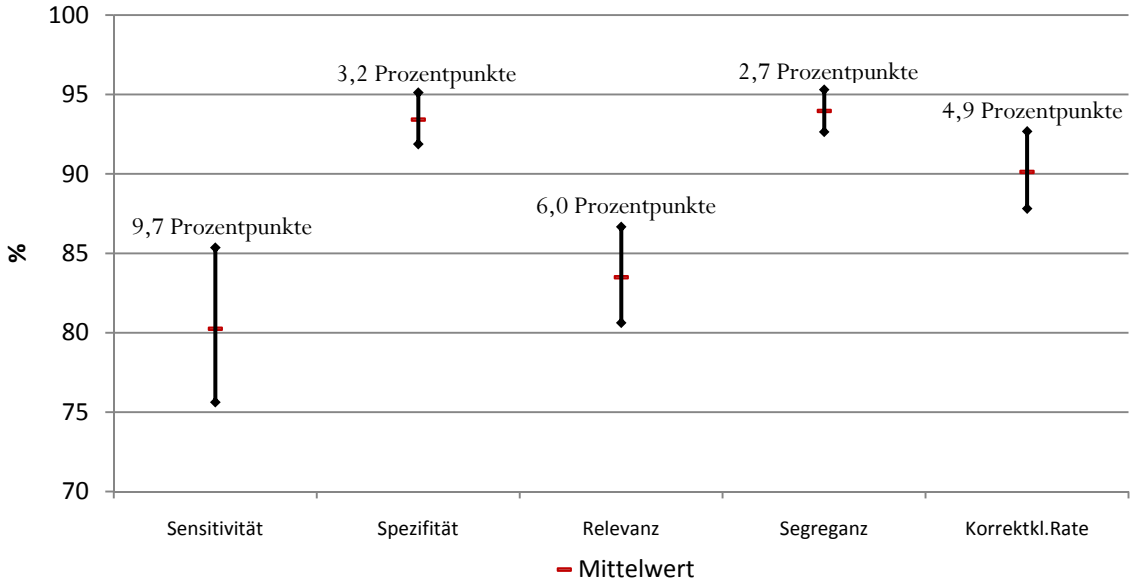

Abbildung 6.17 Streuweite erzielter Kennwerte der lokalen Erkennung der jeweiligen Probanden (ein projektinterner Proband, drei projektfremde Probanden)

Das Training ist somit als personenspezifisch zu bezeichnen, da die besten Ergebnisse mit der trainierenden Person erreicht werden. Wohingegen es trotzdem in der Lage ist, in einem sehr guten Rahmen die Bewegung projektfremder Personen, die das System nicht trainiert haben, zu erfassen. In der verteilten Erkennung beläuft sich der Streuunterschied in einem Bereich von 1,6 bis 10,6 Prozentpunkten, siehe Abbildung 6.18. Anhand der Darstellungen ist zu erkennen, dass aufgrund der medianen Lage der Mittelwerte zu den erhobenen Kennwerten, die Erhebungen nahezu gleichverteilt sind.

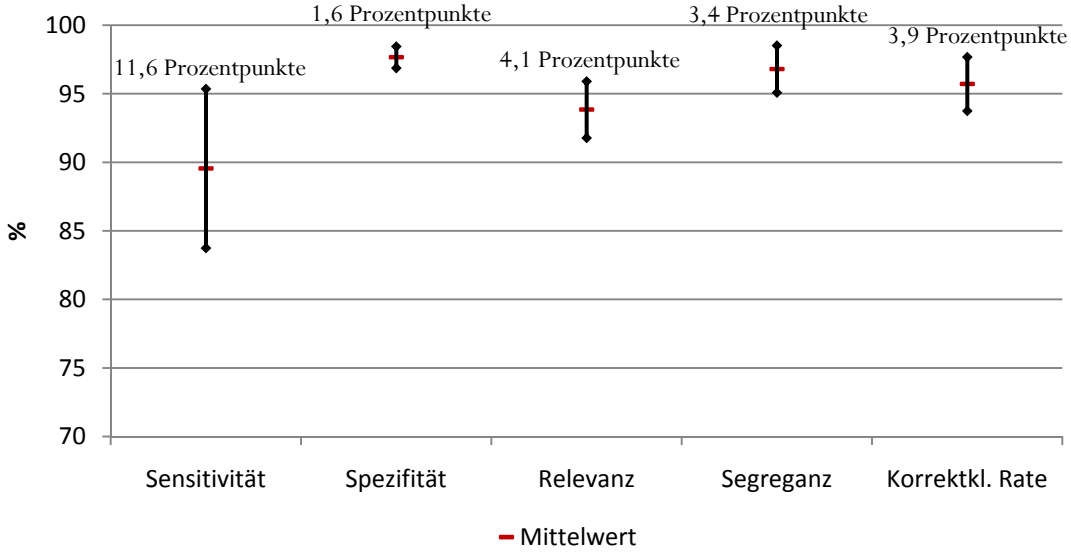

Abbildung 6.18 Streuweite erzielter gemittelter Kennwerte der verteilten Erkennung der jeweiligen Probanden (ein projektinterner Proband, drei projektfremde Probanden)

7 ZUSAMMENFASSUNG UND AUSBLICK

In dieser Arbeit wird das Erkennen von Ereignissen in drahtlosen Sensornetzen mittels einer lokalen und einer verteilten Datenauswertung und Interpretation realisiert. In der lokalen Mustererkennung wird mit einem einzelnen Sensorknoten gearbeitet. Die verteilte Mustererkennung setzt die Erfassungen von Ereignissen mit mehreren Sensorknoten um. Die beobachtenden Sensorknoten tauschen auf der Klassifikations- und Merkmalsebene Daten aus, um die verteilte Erkennung durchzuführen.

Für den Sensorknoten MSB 430 der ScatterWeb Plattform wird ein Erkennungssystem entwickelt, für das bekannte Mustererkennungstechniken verwendet und angepasst werden, um Restriktionen drahtloser Sensornetze berücksichtigen zu können. Für die Erkennung von Mustern wird der dem MSB 430 zugehörige Dreiachsen-Beschleunigungssensor MMA7260 eingesetzt. Basierend auf einer automatischen Sensorkalibrierung ermöglicht das Erkennungssystem das Training unterschiedlicher lokaler und verteilter Muster, deren nachfolgende lokale sowie verteilte Erfassung und deren Klassifizierung. Kein System, konnte bisher durch die Fusion komprimierter Daten innerhalb eines Sensornetzes eine vergleichbare Erkennungsrate erzielen.

Es wird in verschiedenen untersuchten Methoden gezeigt, dass die verteilte Erkennung gegenüber der lokalen Erkennung eine Weiterentwicklung darstellt, die eine deutliche Ereigniserkennungsverbesserung bewirkt. Die Erkennungsqualität der verteilten Erkennung wird eingehend untersucht und mit einer Sensitivität von 92,5 %, einer Spezifität von 97,5 % und einer Korrektklassifikationsrate mit 96,3 % als hoch eingestuft und kann im Vergleich zu lokalen Erkennung die Korrektklassifikationsrate um 6,9 Prozentpunkte steigern. Im Vergleich mit der lokalen Erkennung des Fence Monitoring Projektes kann eine Steigerung um 45,1 Prozentpunkte zur verteilten Erkennung des hier vorgestellten Systems erreicht werden.

Das verteilte Erkennungssystem basierend auf dem Omnibusmodell wird mit einer komplementären Sensorintegration implementiert, die es ermöglicht, eine konkurrierende Integration durchzuführen. Damit ist das System für Erkennungsaufgaben einsetzbar, bei denen gleiche oder verschiedene Nutzinformationen betrachtet werden sollen.

Die Rentabilität des Erkennungssystems in Bezug auf die versendeten Daten werden untersucht. Schlussfolgernd wird die höchste Rentabilität der Klassifikationsfusion zugesprochen. Es lassen sich anhand der durchgeführten Analysen konkrete Empfehlungen für die Verwendung einer Fusionsmethode in unterschiedlichen Anwendungsfällen geben. Unter Beachtung der verschiedenen Einflussstärken von Probanden und Systemparametern werden Analysen bezüglich der lokalen und verteilten Architektur vollzogen. Trotz inhärenter Restriktionen in drahtlosen Sensornetzen ist es möglich, eine verteilte Ereigniserkennung zu erstellen, die von den verteilten Fähigkeiten der Sensornetze profitiert.

Das Gesamtsystem leistet sehr gute Erkennungswerte von verteilten Ereignissen in Sensornetzen durch komprimierte Beschleunigungsdaten und deren Fusion auf verschiedenen Abstraktionsebenen innerhalb des Sensornetzes.

7.1 Weiterführende Arbeiten

In Zukunft sollte eine dynamische Sensorknotenverwaltung zum Einsatz kommen, um Sensorausfälle und Anmeldungen zu koordinieren. Zur weiteren Optimierung des Funk-Datenaufkommens ist ein Wahlsystem für die automatische Festlegung eines Auswertungsknoten zu erstellen.

Aufgrund einer fehlenden FPU in der Hardware ist eine Untersuchung nötig, inwieweit der Einsatz einer FPU-Emulation auf dem Sensorknoten Verbesserungen in der Erkennung verwirklichen kann.

Es ist zu untersuchen, ob sich Wavelettransformationen, mittlere Steigung, das Parzen-Fenster oder Regressionsparameter als Merkmale in der Ereigniserkennung auf dem Sensorknoten umsetzen und einsetzen lassen.

Um die Merkmalsauswahl zu optimieren und zu automatisieren, bietet sich eine Lösung durch ergänzende Hardware an. Der externe einmalige Einsatz dieses Trainingssystems legitimiert sich insofern, da es nach dem Training während der späteren Ereigniserkennung im Sensornetz nicht mehr verwendet werden muss.

Ein externes System, das während des Trainings in der Lage ist, alle Trainingsdaten zu speichern, kann bei der Merkmalsauswahl präzisere Entscheidungen treffen. Mittels einer Kovarianzmatrix ließen sich dort z. B. Abhängigkeiten in den Merkmale erkennen und damit von der Verwendung ausschließen. Durch eine Kreuzvalidierung der Daten könnten signifikante Merkmale bestimmt werden. Der Einsatz der Kreuzvalidierung kann auf dem externen System oder unter Umständen auf einem mit mehr Speicher ausgestattetem Sensorknoten durchgeführt werden.

Es ist zu untersuchen, in welchem Maß eine Erfassung von parallel auftretenden Ereignissen nötig und gegebenenfalls realisierbar ist. Es wird zu prüfen sein, in welchem Umfang eine Erkennung basierend auf reinen Beschleunigungsdaten tatsächlich nutzbar sein wird. In Kombination mit weiteren Sensoren wie Mikrofon, IR-Sensor und Kamera könnte eine in der Realität eingesetzte Mustererkennung wie der Zaunüberwachung zum Einsatz kommen.

7.2 Abschließende Bewertung

Die in wissenschaftlichen Arbeiten bisher nicht vollzogene Konsequenz aus dem Einsatz eines drahtlosen Sensornetzes ist die Ausnutzung redundanter Komponenten als Vorteil für eine fusionierende Ereigniserkennung. In dieser Arbeit kann belegt werden, dass in der Ereigniserkennung die Erkennungsqualität von der durch mehrere Sensorknoten gemeinschaftlich durchgeführten Analyse eines Ereignisses profitiert. Die hier vorliegende Arbeit kann durch die konkrete Implementierung und analytische Untersuchung eines verteilten Erkennungssystems überzeugend darlegen, dass unterschiedliche Fusionseffekte auf verschiedenen Datenebenen existieren und diese einen deutlichen Qualitätssprung für das verteilte System gegenüber der lokalen Erkennung bewirken. Aufgrund der erfolgreichen Klassifizierung innerhalb des Sensornetzes ohne zusätzliche externe Hardware oder Server-Sensorknoten hebt sich dieser Ansatz von vergleichbaren Arbeiten ab und stellt einen Fortschritt in der verteilten Ereigniserkennung dar.

8 BIBLIOGRAPHIE

[Bao04] Bao, L., Intille., S. *Activity Recognition from User-Annotated Acceleration Data.* s.l. : In: Lecture Notes In Computer Science, ISSU 3001, pages 1-17, 2004.

[Blo06] Blotny, A. *Deployment, Calibration and Data Validation Techniques for Wireless Sensor Networks. Seminararbeit.* [PDF] Berlin : Freie Universität Berlin, 2006.

[Bok06] Bokareva, T., Hu, W., Kanhere,S., Ristic, B., Gordon, N., Bessel, T., Rutten, M., Jha, S. *Wireless Sensor Networks for Battlefield Surveillance.* [PDF] Brisbane, Queensland, Australia : In: Proceedings of The Land Warfare Conference, LWC, 2006.

[Bou97] Bouten, C.V.C. Koekkoek, K.T.M. Verduin, M. Kodde, R. Janssen, J.D. *A triaxial accelerometer and portable data processing unit for theassessment of daily physical activity.* s.l. : In: Biomedical Engineering, IEEE Transactions on, Volume: 44, Issue: 3 On page(s): 136-147, 1997.

[Bro03] Brooks, R. R., Ramanathan, P., Sayeed, A. M. *Distributed Target Classification and Tracking in Sensor Networks.* [PDF] Pennsylvania State University, University of Wisconsin : In: Proceedings IEEE 91(8) 11163-1171, 2003.

[Cha06] Chang, M. *Review of Clinical Applications with Human Accelerometry.* [PDF] DK-2100 Copenhagen Denmark : In: Technical Report no. 06/12 ISSN: 0107-8283, 2006.

[Col51] Collatz, L. *Differentialgleichungen.* s.l. : In: Teubner Verlag, 1990.

[Dic98] Dick de Sousa Guimarães, G. *Eine Methode zur Entdeckung von komplexen Mustern in Zeitreihen mit Neuronalen Netzen und deren Überführung in eine symbolische Wissensrepräsentation.* Marburg/Lahn : Fachbereich Mathematik der Philipps-Universität Marburg, 1998.

[Dua07] Duarte, M. Sensor Networks Research Group, 2007. [Online] Electrical and Computer Engineering Department University of Wisconsin-Madison, 2007. http://www.ece.wisc.edu/~sensit/.

[Dud01] Duda, R. O., Hart, P. E., Stork, D. G. *Pattern Classification.* New York, Chichester, Weinheim, Brisbane, Singapore, Toronto : In: A Wiley-Interscience, 2001.

[Endl05] Endler, M. *Large scale body sensing for Infectious Disease Control.* Rio de Janeiro, Brazil : In: Position paper submitted to the Sentient Future Competition, 2005.

[Fer99] Ferber, J. *Multi-Agent systems: An introduction to Distributed Artificial Intelligence.* Essex, England : In: Addison-Wesley, 1999.

[Fre07] Freescale. 2007. http://www.freescale.com/. *±1.5g - 6g Three Axis Low-g Micromachined Accelerometer. Technische Daten.* [Online] 2, 2007.

[Har86] Harel, D. *SATECHARTS: A VISUAL FORMALISM FOR COMPLEX SYSTEMS.* [PDF] North-Holland : Sience of computer programming, 8 (3), 231 – 274, 1987.

[Hea01] Headon R., Curwen, R. *Recognizing Movements from the ground Reaction Force.* [PDF] Orlando, Florida : In: ACM International Conference Proceeding Series; Vol. 15, Proceedings of the 2001 workshop on Perceptive user interfaces, 2001.

[Hyn95] Hyndman, R. *The problem with Sturges' rule for constructing histograms. A short note, 1995.*

[JXi06] J., Xiaoyi. Mustererkennung-Vorlesung WS06. *http://cvpr.uni-muenster.de/teaching/ws06/mustererkennungWS06/ , 2006.* [Online]

[Jaf061] Jafari, R., Noshadi, H., Ghiasi, S., Sarrafzadeh,M. *Adaptive electrocardiogram feature extraction on distributed embedded systems.* [PDF] University of California : In: IEEE Transactions on Parallel and Distributed Systems. 17 (8), pp. 797-807, 2006.

[Jaf06] Jafari, R., Noshadi, H., Ghiasi,S., Sarrafzadeh,M. *Adaptive Medical Feature Extraction for Resource Constrained Distributed Embedded Systems.* [PDF] Los Angeles, Davis, University of California : In: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops, Page: 506 , 2006.

[Jua02] Juang, P.,Oki, H.,Wang, Y., Martonosi, M., Peh, L., Rubenstein,D. *EnergyEfficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet.* San Jose : In: ASPLOS-X conference, 2002.

[Kal01] Kalton, A., Langley, P., Wagstaff, K., Yoo, J. *Generalized Clustering, Supervised Learning, and Data Assignment.* New York, NY, USA : In: ACM Press, 2001.

[Kok01] Kokar, M. M., Tomasik, J. A., Weyman, J. *Data vs. Decision Fusion in the Category Theory Framework.* Northeastern University Boston, Universite d'Auvergne France, Department of Mathematics Northeastern University Boston, 2001. : In: Proceedings of FUSION 2001 - 4th International Conference on Information Fusion, Vol. 1, pages , 2001.

[Legg07] Legg, P., Rosin, P., Marshall, D., Morgan, J. *Improving Accuracy and Efficiency of Registration by Mutual Information using Sturges Histogram Rule.* [PDF] s.l. : In: Proc. Medical Image Understanding and Analysis, pp. 26-30, 2007.

[LiD02] Li, D., Wong,K.D., Hu, H.Y., Sayeed,A.M. *Detection, Classification and Tracking of Targets in Distributed Sensor Networks.* [PDF] USA : In: IEEE SignalProcessing Magazine 19(2) 17-29, 2002.

[Liu06] Liu Chung-Ting, Huo Hong, Fang Tao, Shen Xiao. *Classification Fusion in Wireless Sensor Networks.* Shanghai Jiaotong University 200240, Wuhan University 430079, Shanghai, Wuhan : In: Acta Automatica Sinica, 2006.

[Liup06] Liu, J., Sun,J. ,Wang, S. *Pattern Recognition: An overview.* [Pdf] Jilin University, 130012 Changchun, China : In: IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.6, 2006.

[Lli97] Llinas, J., Hall, D. L. *An introduction to multi-sensor data fusion.* State University of New York at Buffalo, The Pensylvania State University : In: Proceedings of the IEEE Volume 85, Issue 1, Jan 1997 Page(s):6 - 23, 1997.

[Mat06] Maaten, L., Boon, Boon, P. 2006. *COIN-O-MATIC: A fast system for reliable coin classification.* Berlin : In: MUSCLE CIS Coin Recognition Competition Workshop (2006), 2006.

[Mat05] Matrinez, K., Padhy, P., Riddoch, A., Ong, H.L.R., Hart, J.K. *Glacial Environment Monitoring using Sensor Networks.* [PDF] United Kingdom : In: Proceedings of Real-World Wireless Sensor Networks (in press), Stockholm, Sweden, 2005.

[Mat07] Mattern, F. *Allgegenwärtige Informationsverarbeitung – Technologietrends und Auswirkungen des Ubiquitous Computing.* [Pdf] s.l. : In: Allgegenwärtige Datenverarbeitung – Wie möchten wir in Zukunft leben? pp. 11-38, 2007.

[Mat03] Matttern, F., Römer, K. *Drahtlose Sensornetze.* [Pdf] Zürich : Springer, In: Informatik-Spektrum, Vol. 26, No. 3, pp. 191-194, 2003.

[Nie03] Niemann, H. *Klassifikation von Mustern.* s.l. : Springer-Verlag, Berlin 1983.

[RPo06] Polikar, R. *Pattern Recognition.* Rowan University. Glassboro, New Jersey : In: Wiley Encyclopedia of Biomedical Engineering, 2006.

[Rip96] Ripley, B. *Pattern Recognition and Neural Networks.* s.l. : In: Cambridge University Press, 1996.

[Rud06] Rudloff, A., Lauterjung, J., Zschau, J. *Der Deutsche Beitrag zur Einrichtung eines Tsunami-Frühwarnsystems.* s.l. : In: Notfallvorsorge (Walhalla-Verlag), Heft 1/2006, S. 10-12, 2006.

[Rus07] Ruser, H., Puente León, F. *Informationsfusion - Eine Übersicht.* [Hrsg.] Oldenburg Verlag. Universität der Bundeswehr München, Technische Universität München : In: Technisches Messen 74, Oldenburg Verlag, 2007.

[Sca071] ScatterWeb Homepage. *Computer Systems &TelematicsWorking Group, Freie Universität Berlin, 2007.* [Online] http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/.

[Sch92] Schalkoff, R.J. *Pattern Recognition: Statistical, Structural and Neural Approaches.* s.l. : In: Wiley, 1991.

[Schi05] Schiller, J., Liers, A., Ritter H. *ScatterWeb: A wireless sensornet platform for research and teaching.* Institute of Computer Science, Freie Universität Berlin : In: Computer Communications Volume 28, Issue 13, 2 August 2005, Pages 1545-1551, 2005.

[Sch07] Schmidt, M. SLEWS - A Sensorbased Landslide Early Warning System. [Online] Department of Engineering Geology and Hydrogeology / Lehrstuhl für Ingenieurgeologie und Hydrogeologie, RWTH Aachen University, 2007. http://www.slews.de/.

[Sch051] Schoch, D., Sala, M. *BIN IT! The Intelligent Waste Management System.* Zurich : In: The Sentient Future Competition, 2005.

[Sch05] Schütze, S. *Merkmalauswahlverfahren zur Lokalisierung der Bindungsstellen von Transkriptionsfaktoren. Diplomarbeit.* [PDF] Jena : Friedrich-Schiller-Universität Jena, 2005.

[Sei07] Seifert, K., Camacho, O. *Implementing Positioning Algorithms Using Accelerometers. Application Note.* s.l. : In: Freescale Semiconductor, Inc., 2007.

[Til02] Tilak, S.,Abu-Ghazaleh,N.,Heinzelmann,W. *A Taxonomy of Wireless Micro-Sensor Network Models.* [PDF] Binghamton University, Binghamton, University of Rochester, Rochester : In: ACM SIGMOBILE Mobile Computing and Communications Review Volume 6 , Issue 2, 2002.

[Tuc07] Tuck, K. 2007. freescale. *Implementing Auto-Zero Calibration Technique for Accelerometers. Application Note.* [Online] 2007. http://www.freescale.com/files/sensors/doc/app_note/AN3447.pdf.

[Ver05] Ververidis, D., Kotropoulos, C. *Sequential forward feature selection with low computational cost.* Department of Informatics, Aristotle University of Thessaloniki, Greece : In: Proceedings of European Signal Processing Conference, EUSIPCO, 2005.

[WälS07] Wälchli, M., Skoczylas, P., Meer, M., Braun, T. *Distributed event localization and tracking with wireless sensors.* [PDF] Bern, EPFL, Switzerland : In: 5th International Conference on Wired/Wireless Internet Communications (WWIC 2007), 2007.

[Wei91] Weiser, M. The Computer for the 21st Century. [Online] Scientific American, 265, 1991. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.

[Werm06] WernerAllen, G. Monitoring Volcanic Eruptions with a Wireless Sensor Network. 2006. [Online] http://www.eecs.harvard.edu/~mdw/proj/volcano/.

[Wer06] Werner-Allen, G., et al. *Fidelity and Yield in a Volcano Monitoring Sensor Network.* [PDF] s.l. : In: Procedings of the Seventh USENIX Symposium on Operating System Design and Implementaition, Seatle USA, 2006.

[Wer05] Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J., Welsh, M. *Monitoring Volcanic Eruptions with a Wireless Sensor Network.* Harvard University, University of New Hampshire, University of North Carolina : In: Proc. European Workshop on Sensor Networks (EWSN'05), 2005.

[Wit07] Wittenburg G., Terfloth K., Villafuerte, F. L., Naumowicz, T., Ritter, H., Schiller, J. *Fence Monitoring - Experimental Evaluation of a Use Case for Wireless Sensor Networks.* Freie Universität Berlin, Germany : In: European Workshop on Wireless Sensor Networks, Delft, Niederlande, 2007.

[Witf07] Wittenburg, G. *Fence Monitoring - A Use Case for Wireless Sensor Networks - AG Computer Systems & Telematics - Freie Universität Berlin.* . [Online] Freie Universität Berlin, 2007. http://cst.mi.fu-berlin.de/projects/FenceMonitoring/.

9 ABBILDUNGSVERZEICHNIS

10 TABELLENVERZEICHNIS

ERKLÄRUNG

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig angefertigt und ohne fremde Hilfe verfasst habe, keine außer den von mir angegebenen Hilfsmitteln und Quellen dazu verwendet habe und die den benutzten Werken inhaltlich oder wörtlich entnommen Stellen als solche kenntlich gemacht habe.

Berlin, den 24. Oktober 2007

Norman Dziengel

AUFGABENSTELLUNG

Verteilte Ereigniserkennung in Sensornetzen

Überblick

Die Verarbeitung und Aggregation von Daten ist ein integraler Bestandteil der Vision von drahtlosen Sensornetzen, da man sich von der Reduktion des Datenvolumens eine längere Lebensdauer des Netzes verspricht. In den vergangenen Jahren wurden unterschiedlichste Ansätze zur Datenvorverarbeitung in Sensornetzen vorgestellt, wobei der Fokus allerdings stark auf der Aggregation von Rohdaten oder Anfragen nach diesen lag. Erst seit Kurzem wird auch die Erkennung von Ereignissen betrachtet.

Verteilte Ereigniserkennung in Sensornetzen differenziert sich durch zwei Merkmale von bisherigen Forschungsansätzen: Einerseits wurde im Bereich der Sensornetze bisher darauf verzichtet, auf den Sensorknoten selbst die Rohdaten aus den Sensoren in einer komplexen Art und Weise zu interpretieren. Andererseits betrachten gängige Ereigniserkennungsansätze nicht den hier vorliegenden Spezialfall, dass dasselbe Ereignis parallel von mehreren Beobachtern aufgezeichnet und zuverlässig erkannt werden soll, wobei die Beobachter allerdings in der Lage sind, zu diesem Zweck Information auszutauschen. Hinzu kommt die allgemeine Ressourcenknappheit auf den Sensorknoten, die Anpassungen von bekannten Algorithmen erforderlich machen.

Ziel dieser Arbeit ist es, in einem bestehenden Rahmenwerk zur verteilten Ereigniserkennung die einzelnen Komponenten bezüglich der Genauigkeit des Gesamtsystems zu optimieren. Dies schließt die Aufzeichnung von Ereignismustern auf den Sensorknoten ein. Hierzu sollen unterschiedliche algorithmischen Herangehensmöglichkeiten untersucht und mit einander in Testläufen verglichen werden.
Aufgaben

Eine Diplom- oder Masterarbeit in diesem Bereich umfasst folgende Punkte:

- Einarbeitung in die ScatterWeb Plattform und den bestehenden Ansatz zur verteilten Ereigniserkennung
- Implementierung unterschiedlicher Strategien zur Ereigniserkennung auf den ScatterWeb Sensorknoten
- Evaluation der verteilten Ereigniserkennung anhand von geeigneten Testläufen und Optimierung des Gesamtsystems

Umfeld

Verteilte Ereigniserkennung in Sensornetzen ist derzeit ein aktuelles Thema in der Forschung. Die grundsätzliche Möglichkeit der Ereigniserkennung mit Hilfe der ScatterWeb Plattform wurde am Fachbereich bereits gezeigt, die Vorgehensweise ist gut dokumentiert und auf der Fence Monitoring Homepage beschrieben. Für den praktischen Teil der Arbeit stehen ScatterWeb Sensorknoten zur Verfügung, die am Fachbereich entwickelt wurden und ebenfalls auf der ScatterWeb Homepage voll dokumentiert sind.

Zur Vernetzung der Aktivitäten der Arbeitsgruppe gibt es regelmäßige Treffen aller Studien- und Diplomarbeiter sowie der wissenschaftlichen Mitarbeiter der Arbeitsgruppe montags von 10:00 bis 12:00 Uhr.