FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Time synchronization of networked cameras with non-overlapping views using TTA

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science (BSc)

im Rahmen des Studiums

Technische Informatik

eingereicht von

Michael Boula
Matrikelnummer 0426281

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Puschner
Mitwirkung: Dipl.-Ing. Vaclav Mikolasek

Wien, Juni 12 , 2011

(Unterschrift Verfasser/in)

(Unterschrift Betreuer/in)

Inhaltsverzeichnis

Kurzfassung

Soll ein Objekt von mehreren Kameras erkannt und getracked werden können, z.B. um die Bewegung einer Person in einem Raum/Gebäude zu verfolgen, so müssen alle hierzu verwendeten Kameras synchron Bilder aufnehmen um eine konsistente Sicht auf die beobachtete Szene zu haben. Um dies zu gewährleisten, müssen die Kameras zeitlich synchronisiert werden. Die zu synchronisierenden Kameras sind Smart-Kameras die in der Lage sind Berechnungen durchzuführen und Daten über eine Ethernet-Schnittstelle auszutauschen. Diese Bachelor-Arbeit befasst sich mit der Frage, mit welcher Genauigkeit vernetzte Smart-Kameras ohne zusätzliche Hardware, die dediziert zur Synchronisierung verwendet wird, synchronisiert werden können. Da die Kameras Daten untereinander austauschen können sollen, müssen diese über Netzwerkkomponenten miteinander verbunden werden. Aus diesem Grund wird Time-Triggered Ethernet zur Synchronisierung verwendet, da hier ein spezieller Ethernet-Switch zur Synchronisierung verwendet wird, über den die verbundenen Komponenten auch in der Lage sind via Ethernet und allen darauf basierenden Protokollen zu kommunizieren. In dieser Arbeit wird gezeigt, wie TTEthernet auf Smart-Kameras etabliert wird und wie es dazu verwendet wird synchron Bilder aufzunehmen.

KAPITEL 1

Einleitung

Diese Bachelor-Arbeit ist im Zuge des CAT-Projekts[1] entstanden. Das CAT-Projekt beschäftigt sich mit der Fragestellung, ob aufgrund von Personendetektionen Rückschlüsse auf Position und Ausrichtung, von im Raum willkürlich platzierten Kameras, getroffen werden können und ob die neu gefundenen Parameter zur besseren Detektion beitragen können. Damit Rückschlüsse aufgrund von Detektionen getroffen werden können, muss feststehen, dass alle Kameras eine Szene zeit-synchron aufnehmen, also eine konsistente Sicht auf die Szene haben. Für jedes Bild, das eine Kamera aufnimmt, muss es also exakt ein korrespondierendes Bild in jeder anderen Kamera geben, d.h. die aufgenommenen Bilder müssen einander eindeutig zugeordnet werden können.

Warum die Kameras zeitlich synchronisiert sein müssen, zeigt folgendes Beispiel:

Werden zwei Kameras unsynchronisiert, mit gleicher Framerate betrieben, so wird die Zeitdifferenz zwischen den Auslösezeitpunkten der Kameras größer werden. Dies entsteht dadurch, weil die Uhren in den Kameras keine gemeinsame Referenz besitzen. Aus diesem Grund kann es sein, dass Kamera 1 exakt alle 50ms ein Bild aufnimmt (Bsp. 20fps) und Kamera 2 nur alle 55ms. (siehe Bild 1.1). Die maximale Zeitdiffenenz zwischen zwei Auslösezeitpunkten der beiden Kameras entspricht der halben Periodendauer der voreingestellten Framerate. Für das Beispiel von 20fps entspricht dies im schlechtesten Fall einer Zeitdifferenz von 25ms. Wenn beide Kameras eine horizontale Auflösung von 1280px haben, beide den selben Bildausschnitt filmen, der real einer Strecke von 10m entspricht, so entspricht 1 Pixel einem Abstand von $\frac{10m}{1028px} = 0,00781m = 7,81mm$. Geht eine Person auf geradem Wege mit ca. 5km/h von der einen Bildseite zur anderen, so legt sie $5\frac{km}{h} = \frac{5*10^3m}{3600s} = 1,39\frac{m}{s} = \frac{1,39*10^2cm}{10^3ms} = 1,39*10^-1\frac{cm}{ms}$ zurück. Im Falle der maximalen Zeitverschiebung von 25ms bedeutet das, dass die Person innerhalb dieser Zeit $1,39 * 10^-1\frac{cm}{ms} * 25ms = 3,47cm$ zurücklegt.

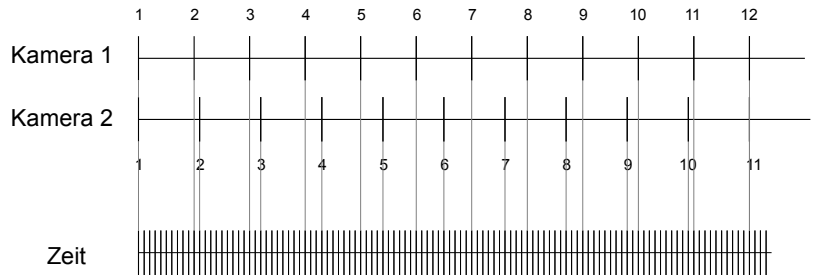

Bild 1.1: Zeitverschiebung zweier Kameras mit 10% Periodenunterschied

1) Im asynchronen Fall:

- Gehende Person: 5km/h entspricht 3,5cm oder 4px
- Laufende Person: 11km/h entspricht 7,6cm oder ca. 10px
- Radfahrer bei 30km/h entspricht 20,7cm oder ca. 27px

Alleine durch diesen Effekt kann es bereits zu Falschdetektionen kommen, wenn z.B. mehrere Personen sich zusammengedrängt bewegen (was beispielsweise auf Flughäfen oder Bahnsteigen passieren kann)

2) Im synchronen Fall (eine Zeitverschiebung von 100 μs wird toleriert):

- Gehende Person: 5km/h entspricht 0,14mm oder 0,02px
- Laufende Person: 11km/h entspricht 0,31mm oder ca. 0,04px
- Radfahrer bei 30km/h entspricht 0,83mm oder ca. 0,11px

Erst bei einem Objekt, das sich mit 280km/h durch das Bild bewegt, würden die Kameras eine Differenz von 1 Pixel erkennen. Außerdem wird durch Bild 1.1 deutlich, dass nach längerer Zeit nicht mehr eindeutig feststeht, welches Bilderpaar der Kameras denselben Zeitpunkt zeigt. In Bild 1.1 ist Aufnahme 6 von Kamera 2 nicht mehr eindeutig Aufnahme 6 oder 7 von Kamera 1 zuordenbar, d.h. bereits nach fünf Bildern (in unserem Fall 250ms nach Start der Aufnahme) kann bereits keine eindeutige Zuordnung mehr getroffen werden. Demnach können Bilderpaare nur mit einer gewissen Wahrscheinlichkeit gefunden werden.

Gegenstand dieser Bachelor-Arbeit ist die Zeit-Synchronisierung der verwendeten Kameras (siehe Kapitel 2) ohne zusätzlichen Hardware-Aufwand und dem Gewährleisten eines Mindestmaßes an Ausfallsicherheit.

Wir haben die Smart-Kameras nach Linux portiert, damit TTEthernet verwendet werden kann. Um die Genauigkeit der Synchronisierung zu zeigen, haben wir ein Experiment mit zwei Kameras durchgeführt. Jede Kamera wurde von uns so konfiguriert, dass sie ein Rechtecksignal erzeugt. Mit Hilfe eines Oszilloskops haben wir die Zeitdifferenz zwischen den Signalen der beiden Kameras und somit die Genauigkeit der Synchronität gemessen (wären die Kameras ideal synchron, wäre die Differenz gleich 0s). Es konnte gezeigt werden, dass die Kameras mit Hilfe von TTEthernet synchronisiert werden können und dass die Genauigkeit der Synchronisierung für unseren Anwendungsfall ausreichend ist.

Die Bachelor-Arbeit ist wie folgt strukturiert:
Kapitel 2 beschreibt den ursprünglichen Systemaufbau und dessen Probleme. Kapitel 3 beschreibt die von uns gewählte Methode und deren Verwendung zum Lösen des Synchronisierungsproblems. Kapitel 4 beschreibt, wie wir Genauigkeit der Synchronisierung getestet haben und befasst sich mit der Interpretation der Messergebnisse.

KAPITEL 2

Systembeschreibung

Im Zuge des CAT-Projekts sollen die Kameras nicht nur synchron Bilder aufnehmen, sondern auch als verteiltes System in der Lage sein Detektionen und Bildverarbeitung selbst zu berechnen. Aus diesem Grund wurden Smart-Kameras, also Kameras, die eine CPU und gängige PC-Interfaces besitzen, verwendet. Die verwendeten Kameras (Sony XCI-V100C[4]) sind mit dem x86 kompatiblen VIA Eden 1GHz, 512 MB SD-Ram, einem 1000BaseT Netzwerkinterface und einem FPGA der Xylinx Virtex5 Familie ausgestattet.

Da die Architektur x86 kompatibel ist, können die Kameras mit WinXPe (Embedded Version von Windows XP) bzw. theoretisch mit jedem x86 kompatiblen Betriebssystem betrieben werden. das aufgenommene Bild kann auf einem Bildschirm ausgegeben werden, der direkt an die Kamera angeschlossen wird. Über das Netzwerkinterface können die Kameras mit einem Ethernet-Switch verbunden werden und so Daten untereinander austauschen.

Der bisherige Systemaufbau ist in Bild 2.1 zu sehen. Alle Kameras sind zusätzlich zum Netzwerk an einen Pulsgenerator oder auch Hardware-Trigger angeschlossen. Dieser erzeugt ein Rechtecksignal, dessen negative Flanke allen Kameras zeitgleich das Signal zur Aufnahme eines Bildes gibt.

Problematisch an dieser Methode sind folgende Punkte:

- Single Point of Failure:
  Alle Kameras, die synchronisiert werden sollen, müssen an denselben Pulsgenerator angeschlossen werden. Hat dieser eine Fehlfunktion oder wird zerstört, so stoppt die Aufnahme auf allen Kameras.

- höhere Kosten:
  Energiekosten für den Betrieb des Pulsgenerators;
  Anschaffungskosten für den Pulsgenerator und die Verkabelung der Kameras

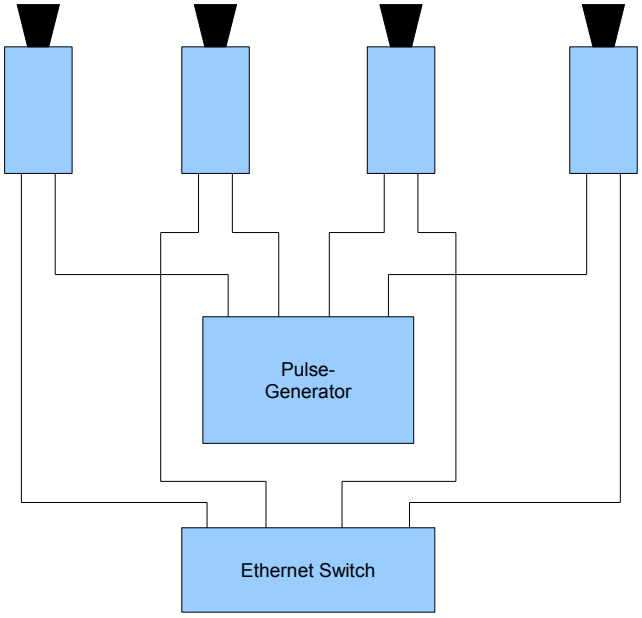

Bild 2.1: Systemaufbau mit Pulsgenerator zur synchronen Bildaufnahme

KAPITEL 3

Realisierung

Bei der Auswahl der Synchronisierungsmethode wurde das Hauptaugenmerk auf die Vermeidung zusätzlicher Verkabelung und Hardware, die rein zur Synchronisierung verwendet wird, gelegt um Zusatzkosten zu minimieren. Damit alle Kameras untereinander Daten austauschen können, müssen diese wie oben erwähnt an einen Ethernet-Switch angeschlossen werden. Die einfachste und naheliegendste Synchronisationsmethode ist also eine, die lediglich mit Hilfe des Ethernet-Protokolls funktioniert, da bei dieser Methode die bereits bestehende Netzwerkinfrastruktur mitverwendet werden kann.

3.1 TTEthernet

Time-Triggered-Ethernet (kurz TTEthernet oder TTE) erweitert das Ethernet-Protokoll um Mechanismen und Nachrichten, die zeitlich gesteuerte Kommunikation ermöglichen. Es wird eine gemeinsame Zeitbasis erzeugt (global sparse time), die für alle teilnehmenden Knoten gilt. Außerdem wird Ethernet um Time-Triggered-Nachrichten kurz TT-Nachrichten erweitert. TT-Nachrichten werden zyklisch versandt, d.h. dass eine Nachricht z.B. immer 1ms nach Beginn einer neuen Runde versandt wird und dass eine Runde 10ms dauert. Diese Einteilung der Sende- und Empfangszeitpunkte der Nachrichten muss a priori, also vor Inbetriebnahme festgelegt werden. Durch diese vorab Definition aller Nachrichten kann der TT Scheduler, der den Ablauf der Kommunikation einteilt garantieren, dass alle TT-Nachrichten ankommen ohne mit anderen Nachrichten zu kollidieren. [5]

Um eine globale Zeit einzuführen werden die Kameras nach folgendem Schema synchronisiert: *(speziell im von TTTech weiterentwickelten TTE [2])*
Die Teilnehmer werden in Synchronisation-Master und Synchronisation-Clients unterteilt. Alle Synchronisation-Master senden zum gleichen Zeitpunkt der globalen Zeit einen Protocol Control Frame (PCF) an den Compression-Master (meist der TTEthernet-Switch). Der Compression-Master berechnet dann aus den Empfangszeitpunkten eine korrigierte Zeit und sendet diese an

alle Teilnehmer zurück, also die Synchronisation-Masters und die Synchronisation-Clients.

TTEthernet generiert also eine gemeinsame Zeitbasis und die Teilnehmer können via Ethernet-Nachrichten (bzw. darauf aufbauenden Protokollen) aber auch mit Hilfe von TT-Nachrichten kommunizieren. Die gemeinsame Zeitbasis kann aber auch genutzt werden um Ereignisse zu vordefinierten Zeitpunkten auszulösen. Diese Time-Triggered-Tasks werden, wie auch das Senden von Time-Triggered-Messages, mit einer gewissen Genauigkeit, zum selben Zeitpunkt auf allen synchronisierten Kameras ausgelöst.

3.2 Verwendung der TTE-Mechanismen

Der oben beschriebene Mechanismus des zeitgleichen Auslösens eines Ereignisses kann für unsere Anwendung verwendet werden, um in vordefinierten Abständen gleichzeitig auf allen Kameras einen Task auszulösen.

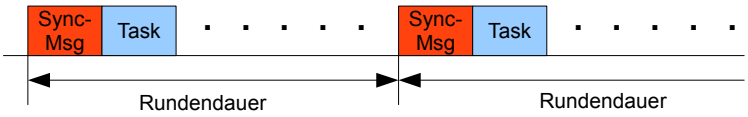

Bild 3.1: TTE rundenbasierender Schedule

Wie schon im Unterkapitel 3.1 erwähnt und wie im obigen Bild dargestellt, ist die Kommunikation rundenbasierend, d.h. es wird definiert zu welchem Zeitpunkt relativ zum Rundenbeginn Nachrichten gesendet bzw. Tasks ausgelöst werden. Dieser erzeugte Schedule wird dann jede Runde ausgeführt. Erzeugt man einen Schedule wie im obigen Bild, so kann die Framerate alleine durch die Rundendauer definiert werden. Wählt man eine Rundendauer von 50ms, so beträgt auch die Zeit zwischen zwei aufeinander folgenden Tasks 50ms und demnach entsteht eine Framerate von 20fps. Möchte man eine höhere Framerate erzeugen, verkürzt man die Rundenzeit (bzw. verlängert um die Framerate zu senken).

Nachteil dieser Methode ist, dass bei hohen Frameraten die verbleibende Restzeit innerhalb einer Runde minimiert wird und dadurch nur wenige zusätzliche TT-Nachrichten gesendet werden können.

Eine weitere Möglichkeit bestünde darin den Task öfter pro Runde auszuführen und damit die Rundendauer zu verlängern, sodass alle geforderten TT-Nachrichten übertragen werden können. Das Zeitintervall zwischen den Tasks muss immer gleich groß gewählt werden, da so die Framerate bestimmt wird. Außerdem muss darauf geachtet werden, dass die Zeit innerhalb einer Runde genügend oft resynchronisiert wird, da die Uhren in den Kameras auseinander bzw. zueinander driften und diese damit Zuordnung zur globalen Zeit verlieren. Schedules für diese Methode sind also etwas komplizierter zu erzeugen als für die erste Methode.

Da für unsere Anwendung in erster Linie kein Übertragen von Daten gefordert ist und uns mehr interessiert, wie sich die Synchronisierungsqualität durch verschiedene Frameraten beeinflussen lässt, beschränken wir uns auf die erste Methode und variieren die Rundenzeit um so die Framerate festzulegen.

3.3 Integration des TTE-Treibers

Die Firma TTTech, die TTEthernet in Zusammenarbeit mit der TU Wien entwickelt hat, bietet unter anderem eine Software Implementierung der oben erwähnten Mechanismen in Form eines Treibers für das Betriebssystem Linux an. Der Treiber enthält OpenSource-Schnittstellen, damit er an die verwendete Hardware angepasst werden kann. Die eigentlichen Mechanismen zur Synchronisation und zum Nachrichtenversand sind im proprietären Teil des Treibers implementiert. Wie schon im Kapitel 2 beschrieben, sind die Kameras x86-kompatibel und werden mit WinXPe ausgeliefert. Damit der angebotene Treiber verwendet werden kann, müssen die Kameras nach Linux portiert und der Treiber modifiziert werden. Als Betriebssystem wird die von TTTech empfohlene Distribution, Ubuntu 8.04, welche auf einem Linux 2.6er Kernel basiert, verwendet.

Das Problem einer Softwarerealisierung eines echtzeitfähigen Systems ist jedoch, dass Betriebssystemfunktionen die Vorhersagbarkeit einer Echtzeitanwendung verschlechtern. Im schlimmsten Fall kann es dazu kommen, dass ein Echtzeittask nicht vor Ablauf seiner Deadline ausgeführt werden kann, weil das Betriebssystem höhere Priorität hat. Um diesen Umstand zu umgehen müssen Funktionen des Betriebssystems unterbrochen werden können und den Echtzeittasks muss Vorrang gegeben werden. Dies kann mit Hilfe eines von kernel.org zur Verfügung gestellten Patches gewährleistet werden.[3] Da für die von TTTech empfohlene Linux-Distribution ein angepasster Treiber zur Verfügung gestellt wird, müssen die OpenSource-Schnittstellen nicht weiter angepasst werden.

Jede Kamera muss separat konfiguriert werden, d.h. der Treiber jeder Kamera muss auf den im Unterkapitel 3.2 erwähnten Schedule angepasst werden. Sobald der Kamera ihre Rolle im Kommunikationssystem zugewiesen worden ist, kann der Treiber kompiliert und geladen werden. Werden die Kameras nun an den TTEthernet-Switch angeschlossen, so werden sie zeitlich synchronisiert und beginnen zu den a priori festgelegten Zeitpunkten Nachrichten zu senden bzw. Tasks zu exekutieren.

KAPITEL 4

Machbarkeitsnachweis

Um zeigen zu können, dass Smartkameras mit Hilfe von TTEthernet zeitlich synchronisiert werden können und dadurch in der Lage sind ohne zusätzliche Hardware zeitsynchron Bilder aufzunehmen, muss eine systemunabhängige Messung durchgeführt werden. Es besteht die Möglichkeit die einzelnen Knoten mit Hilfe von Software, die von der Firma TTTech zur Verfügung gestellt wird, zu überprüfen, jedoch wollten wir eine vom Protokoll unabhängige Methode zum Test der Synchronität entwickeln.

Die Kameras verfügen über eine I/O-Schnittstelle, die direkt mit der General Purpose I/O (GPIO) der CPU verbunden ist. Der in Unterkapitel 3.2 beschriebene Task wird für unseren Versuchsaufbau zum Erzeugen eines Rechtecksignals verwendet. Hierzu wird bei jedem Aufruf des Tasks der aktuelle Zustand am Ausgang der Kamera (GPIO-Ausgang) getoggelt. Das erzeugte Rechtecksignal wird von einem Oszilloskop gemessen und jede dargestellte Flanke des Signals entspricht einem Aufruf des Tasks. Schließt man mehrere zeitlich synchronisierte Kameras die denselben Task ausführen an dasselbe Oszilloskop an, so kann man eine Zeitdifferenz zwischen den Flanken der gemessenen Rechtecksignale messen. Diese Zeitdifferenz ist ein Maß für die Genauigkeit der Synchronisation. Um die Synchronität der beiden generierten Signale zu zeigen, beschränkten wir uns in unseren Versuchen auf die Verwendung von zwei Kameras. Der Prinzipaufbau der für den Test verwendet wurde, ist in Bild 4.1 zu sehen.

Ziel des CAT-Projekts ist, dass die Kameras Detektionen berechnen und untereinander austauschen. Dabei werden CPU und Speicher der Kameras stärker belastet als beim schlichten Aufnehmen von Bildern. Aus diesem Grund wird überprüft, ob die Auslastung der Kameras einen Einfluss auf die Synchronität der Bilder hat. Die Auslastung der Kameras wird durch Tasks wie Kopieren, Suchen und Ausgaben am Terminal erzeugt. Außerdem wird in weiterer Folge überprüft, ob die Framerate, also die Dauer der Rundenzeit und somit auch die Auslastung des Netzwerks und des Switches einen Einfluss haben (je mehr Runden pro Sekunde gestartet werden, umso mehr Synchronisationsnachrichten müssen verschickt werden und umso höher ist der Time-Triggered-Traffic). Die Kameras sollen bei 20fps betrieben werden, wobei die theoretisch maximale Framerate laut Produktinformationen 90fps[4] beträgt.

Die Zeitdifferenz der Flanken, die von zwei synchronisierten Kameras erzeugt werden, wird also bei

- 20 fps
  - volle Auslastung der Kameras (Load-Betrieb)
  - Idle-Zustand der Kameras (Idle-Betrieb)
- 90 fps
  - volle Auslastung der Kameras (Load-Betrieb)
  - Idle-Zustand der Kameras (Idle-Betrieb)

gemessen.

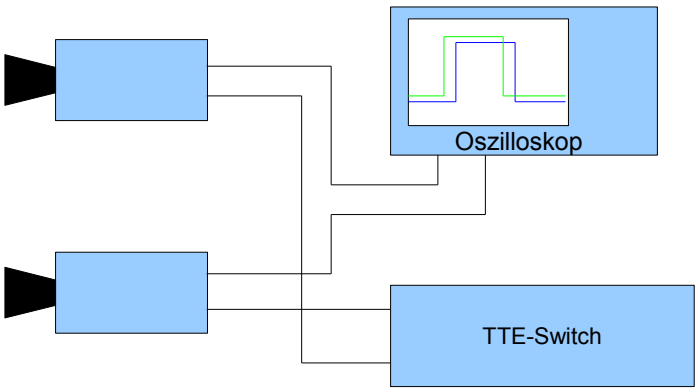

Bild 4.1: Versuchsaufbau

4.1 Ergebnisse und deren Interpretation

Wie schon in Unterkapitel 3.2 festgelegt erzeugen zwei Kameras zum selben Zeitpunkt der globalen Zeit ein Ereignis (Pegeländerung am Ausgang der Kamera). Gemessen wurde die Zeitdifferenz zwischen den erzeugten Ereignissen. Für jede der oben erwähnten Konfigurationen wurden 10.000 Werte gemessen und die Verteilungsparameter aus diesen Stichprobe ermittelt. Eine Übersicht bzw. einen Vergleich dieser Verteilungsparameter findet man in Tabelle 4.1 und einen grafischen Vergleich aller Verteilungen zeigt Bild 4.8.

	idle		load	
	20fps	90fps	20fps	90fps
Mittelwert (μs)	21,08	21,33	35,23	34,06
Standardabweichung (μs)	0,98	2,55	3,41	4,24
Schiefe	3,94	4,42	3,09	1,94
Exzess	39,45	22,19	25,86	16,21
Kleinster Wert (μs)	11,49	8,67	15,23	10,10
Median (μs)	20,90	20,80	34,76	33,75
99% Perzentil	25,04	33,74	45,85	46,63
Größter Wert (μs)	37,33	46,31	76,96	76,94

Tabelle 4.1: Vergleich der Verteilungsparameter

Ohne weitere Kenntnis über das verwendete System, könnte man vermuten, dass die gemessenen Zeitdifferenzen normalverteilt sind. Wie man in Bild 4.2 erkennen kann, ist die den Stichproben zugrundeliegende Verteilung jedoch viel steiler als eine Normalverteilung, was man auch am Exzess der einzelnen Verteilungen erkennen kann (für eine Normalverteilung wäre dieser Wert 0). Eine interessante Beobachtung ist, dass alle Verteilungen rechtsschief sind, d.h. es werden häufiger Werte beobachtet die kleiner sind als der Mittelwert. Die Schiefe wird jedoch durch einige Ausreißer beeinflusst, die wesentlich größer sind als der Mittelwert. Diese Beobachtung ist ebenfalls in dem in Bild 4.3 dargestellten Boxplot erkennbar. Der Boxplot stellt die in Tabelle 4.1 stehenden Verteilungsparameter grafisch dar. Der Strich in der Box entspricht dem Median, die Länge der Box dem Interquartilsabstand. Die Grenzen der Box entsprechen also dem oberen bzw. dem unteren Quartil. Die Verlängerung der Box nennt man Whiskers, welche in diesem Boxplot beim dreifachen Interquartilsabstand liegen. Bei den durch Ringe dargestellten Ausreißern handelt es sich um extreme Ausreißer die eher rechts vom Mittelwert liegen, was zu den Schleifen führt. Hierdurch wird verdeutlicht, dass aufgrund von Mittelwert und Median keine Zusicherungen bzgl. Synchronität getroffen werden dürfen, sondern einzig der Worst-Case (maximaler Wert) ausschlaggebend ist.

Betrachtet man die kleinsten gemessenen Werte, so erkennt man, dass der kleinste Wert sowohl im Idle-Betrieb als auch im Load-Betrieb für 90fps geringer ist als für 20fps. Dies ist höchstwahrscheinlich auf das bei 90fps geringere Synchronisierungsintervall zurückzuführen (siehe Unterkapitel 3.2).

Eindeutig zu erkennen ist (siehe Bild 4.4 und Bild 4.5), dass die Auslastung der Kameras starken Einfluss auf die Synchronität hat. Im Mittel ist die Zeitdifferenz der beiden Ereignisse zwischen 59-67% im Load-Betrieb höher als im Vergleich zum Leerlauf (idle). Interessant ist jedoch, dass die Framerate offenbar wenig Einfluss auf die Synchronität hat (siehe Bild 4.6 und Bild 4.7), da sowohl Mittelwert als auch Median in beiden Lastszenarien (idle und load) von der Framerate fast nicht beeinflusst werden. Betrachtet man Bild 4.5, erkennt man für den Idle-Betrieb eine Anhäufung von Messwerten, die um den Mittelwert der Verteilung für den Load-Betrieb liegen. Interessant ist, dass es, wie in Bild 4.4 zu sehen, für die Verteilungen im Idle-Betrieb bei 20fps kein solches Phänomen gibt. Anders ausgedrückt bedeutet dies,

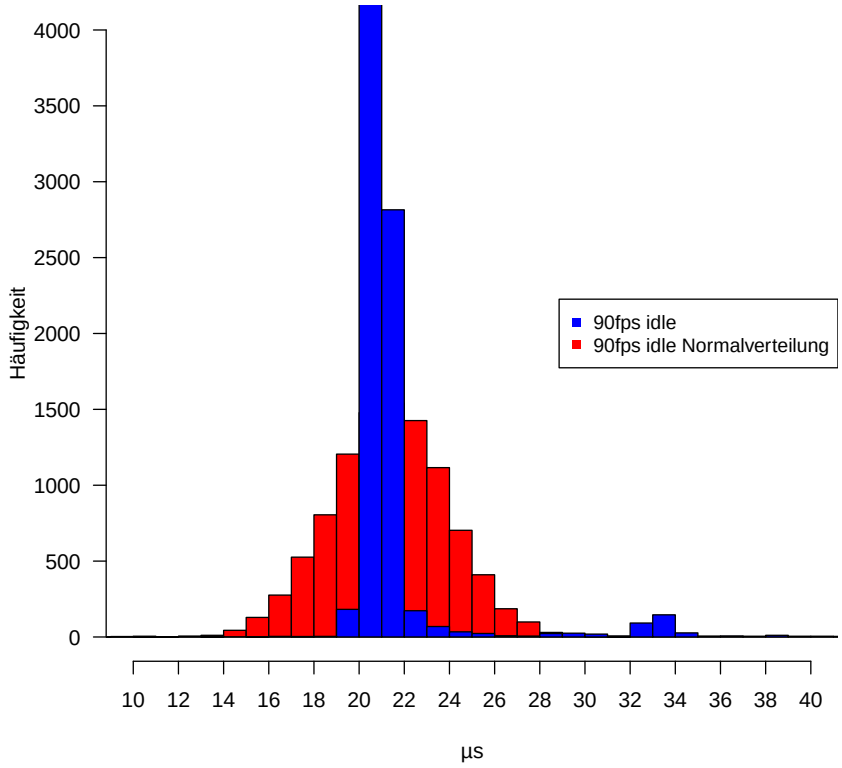

Bild 4.2: Vergleich der Verteilung von 90fps Idle-Betrieb mit einer Normalverteilung mit selbem Mittelwert und Standardabweichung

dass sich bei 90fps Idle-Betrieb Werte ergeben, die dem Load-Betrieb entsprechen, aber die bei 20fps Idle-Betrieb jedoch nicht auftreten. Vermutlich wird die Ausführung des Tasks, der die Events am Ausgang der Kamera erzeugt, dadurch verzögert, weil zum Ausführzeitpunkt Funktionen vom Betriebssystem ausgeführt werden, die nicht unterbrochen werden können. Um die höhere Framerate zu erreichen, muss der eventerzeugende Task 4,5x öfter ausgeführt werden, wodurch die Wahrscheinlichkeit einer Taskkollision steigt. Würde man die Framerate noch weiter steigern, würden mehr Tasks kollidieren und sich dadurch die Verteilung des Idle-Betriebs näher zur Load-Betrieb Verteilung hin verschieben (bzw. mit dieser übereinstimmen). Diese Verzögerung des eventerzeugenden Task ist verantwortlich für die schlechtere Synchronität der gemessenen Ereignisse im Load-Betrieb der Kameras. Um die Auslastung der Kameras zu erreichen werden diese mit diversen Funktionen kontinuierlich belastet, was jedoch dazu führt, dass die Wahrscheinlichkeit einer Kollision stark steigt.

Der gemessene Worst-Case ist 76,96μs Zeitdifferenz. Betrachtet man das 99% Perzentil,

so erkennt man, dass bei 90fps und Load-Betrieb 99% aller Werte unter 46,63μs liegen. Das bedeutet, dass die Auftrittswahrscheinlichkeit eines Wertes, der größer als 46,63μs ist, gleich 1% ist. Die Auftrittswahrscheinlichkeit eines Wertes, der größer als 70,79μs ist, beträgt 0,1%. Da es jedoch nur um einen gemessenen Worst-Case handelt, wird dazu geraten von einem realen Worst-Case auszugehen, der um 100μs Zeitdifferenz liegt. Wie in Kapitel 1 gezeigt wurde, ist diese Synchronisierungsqualität jedoch für die meisten Anwendungen ausreichend.

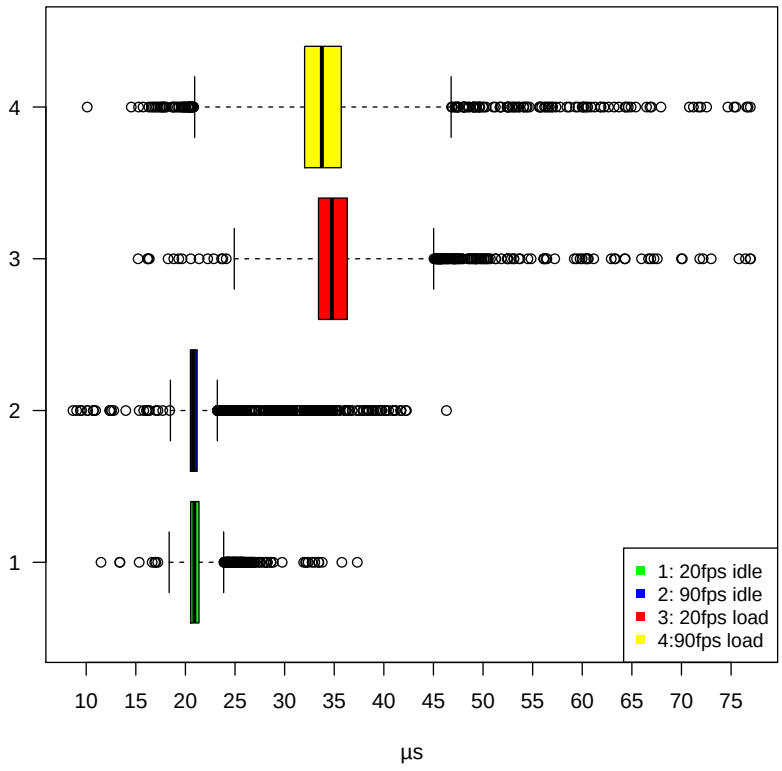

Bild 4.3: Boxplot aller Messungen im Vergleich

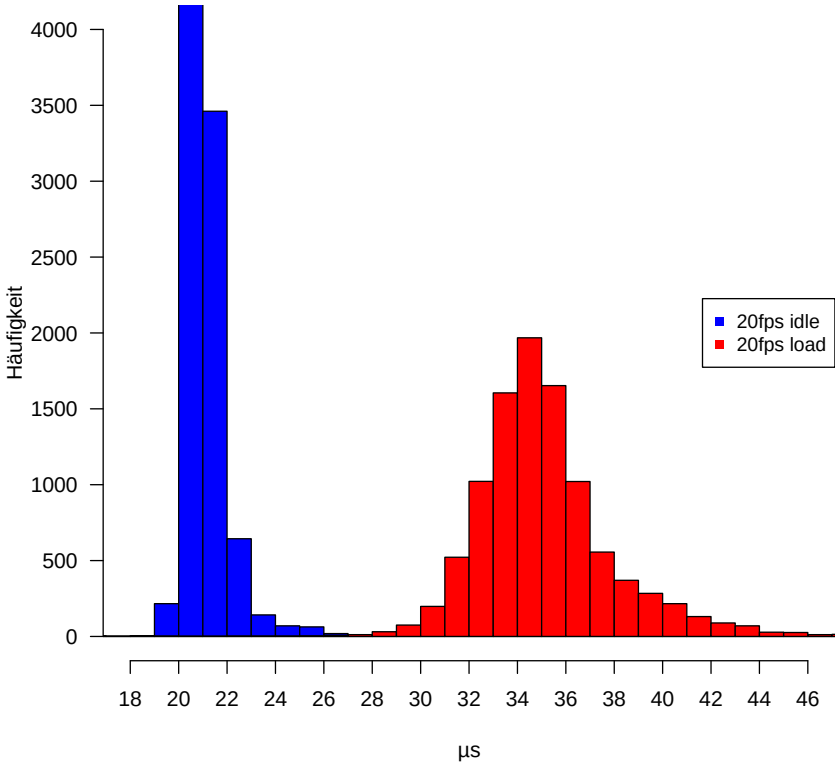

Bild 4.4: Vergleich der Messergebnisse bei 20fps

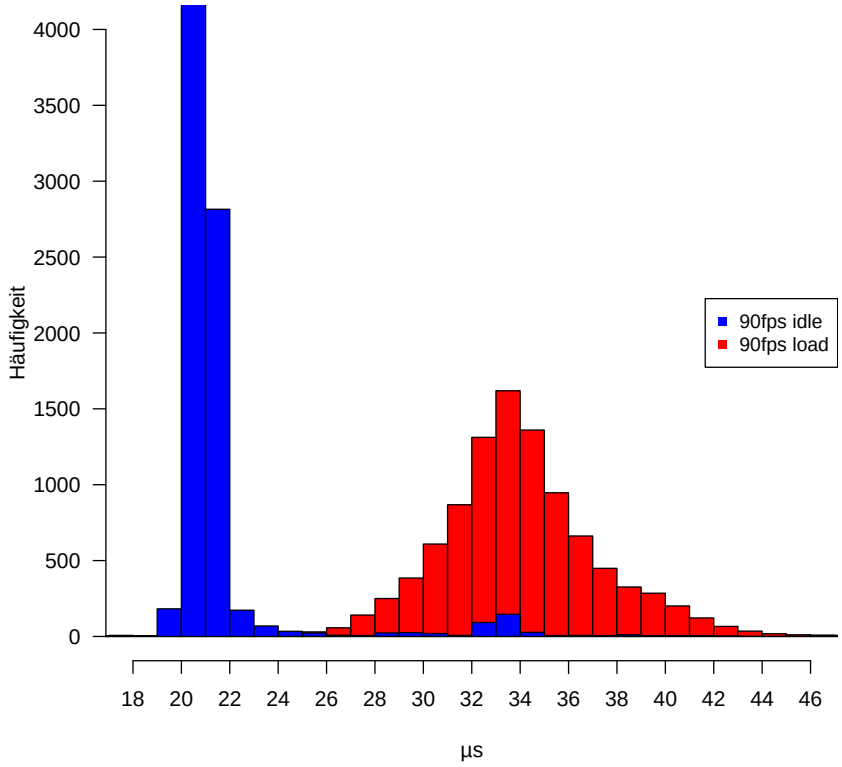

Bild 4.5: Vergleich der Messergebnisse bei 90fps

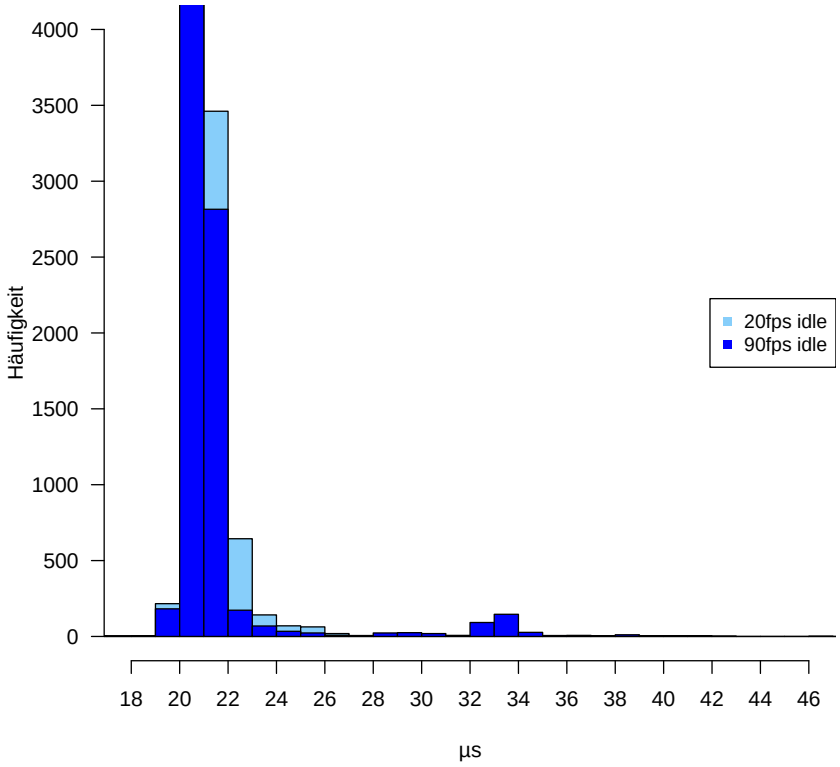

Bild 4.6: Vergleich der Messergebnisse im Idle-Betrieb der Kameras

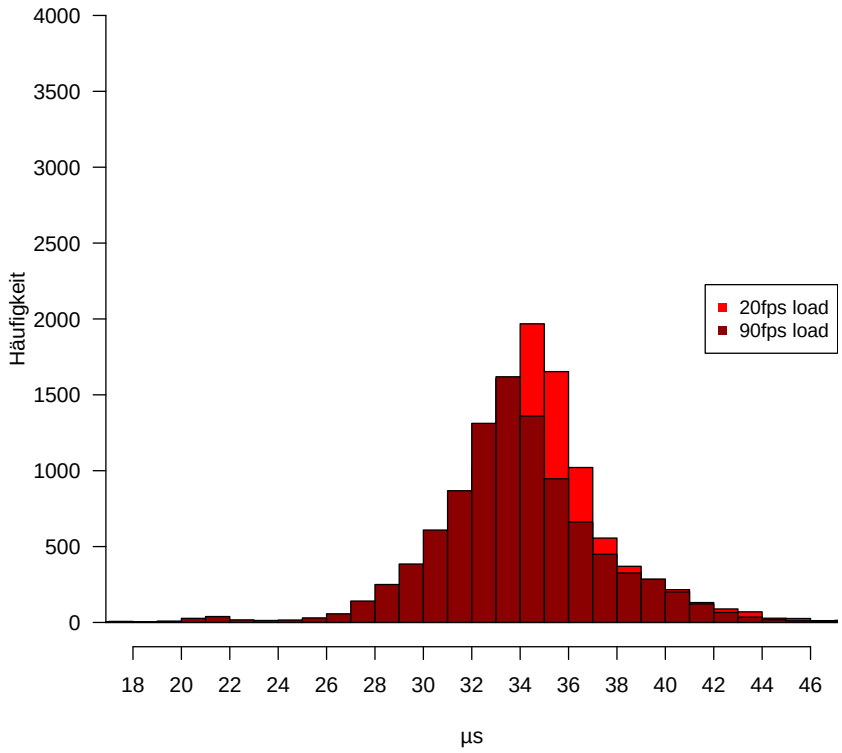

Bild 4.7: Vergleich der Messergebnisse im Load-Betrieb der Kameras

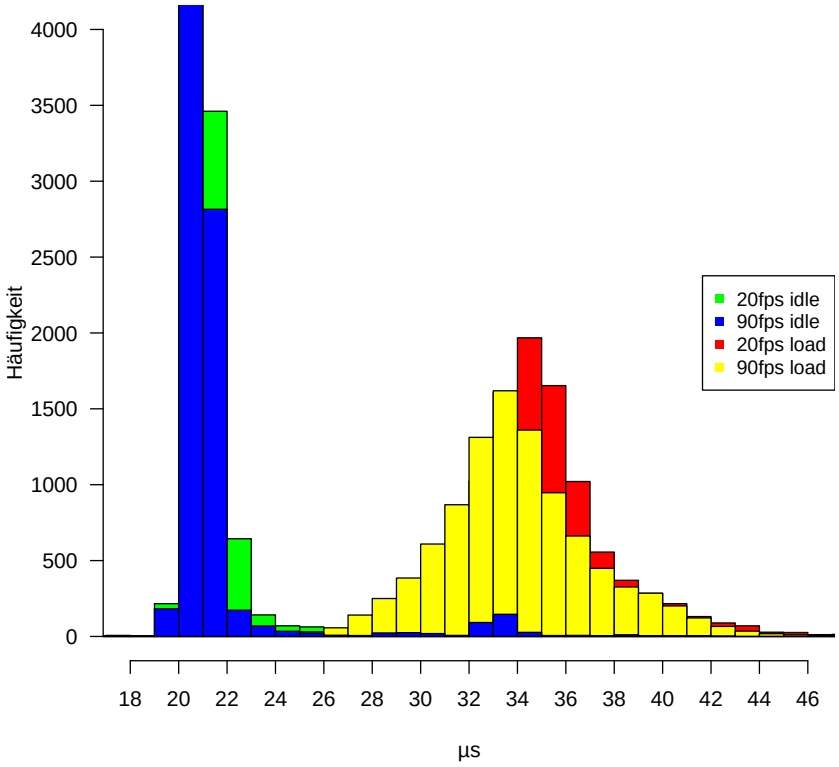

Bild 4.8: Vergleich aller Messergebnisse

KAPITEL 5

Zusammenfassung und Ausblick

Die Fragestellung, mit deren Lösung sich diese Arbeit befasst hat, lautet, ob auf Smart-Kameras TTEthernet etabliert werden kann und ob die Kameras damit synchronisiert werden können. Es wurde gezeigt, dass die Kameras dank x86-Kompatibilität unter Linux betrieben werden können und dass sie durch geringe Modifikationen echtzeitfähig gemacht werden können. Aufgrund dieser Vorarbeit konnte der von TTTech zur Verfügung gestellte TTEthernet-Treiber an die Kameras angepasst und in Betrieb genommen werden. Ein Schedule zum zeitsynchronen Auslösen von Bildern wurde sowohl für den Betrieb von 20fps als auch 90fps erstellt und es konnte gezeigt werden, dass die Kameras zeitsynchron innerhalb einer mittleren Zeitdifferenz von ca. 35μs bzw. im Worst-Case ca. 77μs Bilder aufnehmen können.

Offene Aufgaben, die in dieser Arbeit nicht ausgeführt werden konnten, sind folgende:

- Verwendung des Tasks um die Aufnahme eines Bildes auszulösen und somit einen zeitlich synchronisierten Stream auf allen verwendeten Kameras zu erzeugen.
- Messen der Synchronisierungsqualität, wenn zum Einstellen der Framerate die zweite in Unterkapitel 3.2 beschriebene Methode verwendet wird. Hier können größere Rundenzeiten verwendet werden, da der Task zum Auslösen von Bildern öfter pro Runde aufgerufen wird. Außerdem kann die Qualität der Kamerauhren überprüft werden, da seltener synchronisiert wird und somit der Drift der Uhren wesentlichen Einfluss zeigen könnte.
- Messen des größten Resynchronisierungsintervalls unter der Vorraussetzung, dass die Synchronisierungsqualität nicht darunter leidet.
- Verwendung von TTEthernet zum Austausch von Detektionsdaten.

Danksagung

Besonderer Dank gilt Roman Pflugfelder vom Austrian Institute of Technology, Vaclav Mikolasek von der Technischen Universität Wien und Michael Walloch für deren Unterstützung. Diese Arbeit wurde durch den Wiener Wissenschafts-, Forschungs-, und Technologiefonds im Rahmen des Projekts CAT ICT08-030 gefördert.

Bildverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] CAT - calibration and tracking,
Link: http://www.cat-project.at/

[2] TTTech Computertechnik AG
TTEthernet_Article.pdf, 2009
Link: http://www.tttech.com/fileadmin/content/white/TTEthernet/TTEthernet_Article.pdf

[3] RTwiki,
Link: https://rt.wiki.kernel.org/index.php/Main_Page

[4] Sony Corporation,
Broschüre: XCISX100C_Final pdf.pdf, 2008
Link: http://ws.sel.sony.com/PIPWebServices/RetrievePublicAsset/StepID/SEL-asset-142946/original/XCISX100C_Final%20pdf.pdf

[5] Hermann Kopetz, Astrit Ademaj, Petr Grillinger und Klaus Steinhammer
The Time-Triggered Ethernet (TTE) Design
Vienna University of Technology Real-Time Systems Group 2005