Echtzeit-Farbbildverarbeitung für den Einsatz in Fahrzeugen

Gruner, Christian; Schilling, Thomas; Winter, Harald

Aglaia Gesellschaft für Bildverarbeitung und Kommunikation mbH
Tiniusstraße 12 – 15
D-13089 Berlin

E-Mail: mail@aglaia-gmbh.de

URL: www.aglaia-gmbh.de

Zusammenfassung

Vorgestellt wird das in der Aglaia GmbH entwickelte Softwaresystem „Cassandra“, das auf die Anforderungen der Echtzeit-Farbbildverarbeitung und vor allem für den Einsatz in Fahrzeugen zugeschnitten ist. Basierend auf C++ und dem Einsatz von PC-Technik ist es möglich, baugleiche Systeme für den Labor- und den Fahrzeugeinsatz zu verwenden. Damit entfallen in der Entwicklung aufwendige Portierungen und es ist ein beständiges Testen der Funktionalität im realen Fahrzeugeinsatz gewährleistet. Das streng modulare und erweiterbare System kombiniert Farb- und Grauwertverarbeitung, Entwicklungskomfort und hohe Verarbeitungsgeschwindigkeit. Eine prototypische Applikationslösung ist die Verkehrszeichenerkennung (Geschwindigkeitsbegrenzungen). Die Verarbeitungsgeschwindigkeit von 10 bis 18 Videohalbbilder/Sekunde auf einem Pentium II/400 ermöglicht Fahrzeuggeschwindigkeiten bis zu 150 km/h.

Die Problemstellung

Die Entwicklung von Visionsystemen für Fahrzeuge und mobile Roboter ist seit Jahren eine faszinierende Forschungsrichtung. Dabei ist der Bedarf an Systemen, die zu einer aktiven Erhöhung der Sicherheit im Straßenverkehr beitragen können, unübersehbar. Inzwischen wird deutlich, daß getragen von der allgemeinen Entwicklung der Rechentechnik praktisch relevante Lösungen in greifbare Nähe rücken. Entsprechende Entwicklungen sind jedoch sehr aufwendig. Insbesondere ist eine ständige Parallelität von Algorithmenentwicklung, Softwareimplementierung und praktischer Erprobung notwendig. Diese war in der Vergangenheit oft dadurch erschwert, daß die Algorithmenentwicklung in einer gewohnten und komfortablen Laborumgebung erfolgte, im Fahrzeug aus Gründen der Echtzeitforderung jedoch spezielle Systeme eingesetzt wurden, die eine Softwareportierung erforderlich machten. Der Zyklus zwischen Algorithmenentwicklung und praktischer Erprobung wurde dadurch sehr belastet.

Der Ausgangspunkt der nachfolgend dargestellten Entwicklung beruhte auf der Einschätzung, daß die Leistungsfähigkeit der PC-Technik inzwischen einen Stand erreicht hat, der Entwicklung und praktische Erprobung im Fahrzeug mit einem System erlaubt. Mit dem hier vorgestellten Softwaresystem „Cassandra“ wurde dementsprechend versucht, ein komfortables Laborentwicklungssystem für Farbbildverarbeitung zu entwickeln, das gleichzeitig für die praktische Erprobung im Fahrzeug geeignet ist. Daraus ergeben sich neben den generellen Forderungen nach Effektivität und Wiederverwertbarkeit spezifische

Anforderungen sowohl an die Entwicklungsumgebung und –methodik als auch an die Algorithmen und deren Implementierung.

Folgende Prämissen waren Grundlage der Entwicklung:

- objektorientierte Programmierung in C++ unter Windows NT
- strikte Trennung zwischen Verarbeitung und Oberfläche als Grundlage für eine spätere Portierung auf andere Betriebssysteme oder andere Hardware
- Modularität und Wiederverwendbarkeit der Komponenten
- dynamische Integration von neuen Funktionen in eine flexible Bedienoberfläche (Plugin)
- Gewährleistung einer maximalen Flexibilität zur Laufzeit
- Sicherung der Echtzeitfähigkeit von Basisalgorithmen

Das Softwaresystem „Cassandra“

Im Ergebnis dieser Überlegungen ist in der Aglaia GmbH das Softwaretool „Cassandra“ entstanden, das seine erste Bewährungsprobe in einem Projekt zur Verkehrszeichenerkennung bestanden hat. Zunächst sollen allgemeine Konzepte dargestellt werden, die die Basis der Leistungsfähigkeit bilden.

Der `cRef`-Mechanismus

Der **`cRef`**-Mechanismus nimmt dem Programmierer eine aufwendige Arbeit ab: die Kontrolle darüber, wann ein dynamisches Objekt wieder gelöscht werden kann bzw. gelöscht werden muß. Das Löschen eines Objektes, auf das noch ein Pointer verweist, führt bei Dereferenzierung des Pointers zum Absturz. Bleiben nicht mehr benötigte Objekte erhalten, so wird die Applikation zum Speicherfresser. Für die Erreichung der gewünschten Flexibilität zur Laufzeit, insbesondere der Aufbau einer unvorhersehbaren Komplexität in der Bildanalyse und –visualisierung, ist ein solcher Mechanismus unverzichtbar.

Um diesen Mechanismus zu nutzen, werden fast alle Cassandra-Klassen von der Klasse **`cShared`** abgeleitet. **`cShared`**-Objekte enthalten einen Referenzzähler, der alle auf das Objekt vorhandenen Referenzen erfaßt. Voraussetzung ist, daß nicht normale Pointer für diese Objekte verwendet werden sondern ausschließlich die in Cassandra dafür vorgesehen Referenzobjekte vom Typ **`cRef.`** **`cRef`** ist ein Klassen-Template, in das der Typ des zu referenzierenden Objektes eingeht. Ein **`cRef`**-Objekt beinhaltet einen „echten“ Pointer auf das referenzierte Objekt und stellt eine Funktionalität zur Verfügung, die es erlaubt, daß das **`cRef`**-Objekt wie ein Pointer verwendet werden kann. Dieser Mechanismus greift ebenso, wenn die zu referenzierende Klasse nicht von **`cShared`** abgeleitet ist. In diesem Fall werden den Objekten programmintern **`cShared`**-Objekte zugeordnet, die den erforderlichen Referenzzähler bereitstellen. Damit können auch die Objekte fremder Klassen in völlig gleichartiger Weise in die automatische Verwaltung einbezogen werden

Der Signal-Slot-Mechanismus

Die Verwaltung komplexer, sich zur Laufzeit entwickelnder Abhängigkeiten zwischen Objekten, wie sie bei interaktiver Benutzeroberflächen auftreten, führen bei herkömmlicher Programmierweise schnell zu übermäßig komplizierten und fehleranfälligen Programmen. Cassandra stellt mit dem Signal-Slot-Mechanismus dem Programmierer eine leicht beherrschbare und leistungsfähige Methodik zur Verfügung, die die Programmierung von dynamischen komplexen Abhängigkeitsstrukturen extrem vereinfacht und damit schnell zu sicheren Programmen führt.

Klassen, die Signale aussenden sollen, verwenden nur spezielle Signal-Klassen und müssen nichts wissen von den Klassen, in denen die Reaktionen auf diese Signale enthalten

sind. Ebenso verwenden Klassen, die Reaktionen auf Signale enthalten nur die Slot-Klassen und müssen nichts von den Klassen wissen, die die Signale aussenden. Diese Verfahrensweise besitzt entscheidende Vorteile. Programmkomplexe können unabhängig voneinander entwickelt und angewendet werden. Der modulare Charakter der Software, die Universalität und Wiederverwendbarkeit der Module werden verstärkt. Insbesondere ist es möglich, Verarbeitungsalgorithmen und Benutzeroberfläche völlig getrennt voneinander zu entwickeln und anzuwenden. Die Verbindung zwischen Signalen und Slots, also die Festlegung von Reaktionen auf bestimmte Ereignisse, wird dynamisch zur Laufzeit hergestellt bzw. aufgehoben. Es sind Mehrfachzuordungen in beide Richtungen möglich. Ein Signal kann mehreren Slots zugeordnet werden und damit mehrere Reaktionen anstoßen. Ebenso kann ein Slot mehreren Signalen zugeordnet sein, die dann jedes für sich die dem Slot zugeordnete Reaktion auslösen.

Dynamischer Menüaufbau

Das Menü wird in Cassandra mittels spezieller Textdateien beschrieben und dynamisch erzeugt. In diesen Menübeschreibungsdateien werden die in der Applikation oder in den DLLs enthaltenen Menühändlerklassen (abgeleitet von **`cMenuHandler`**) in einen hierarchischen Menüaufbau mit frei wählbaren Bezeichnungen eingeordnet. Neben Menübezeichnungen können den Menühändlerklassen eine Parameterzeichenkette (zur Laufzeit änderbar), eine Statuszeilenausschrift und ggf. ein Schaltflächensymbol der Symbolleiste zugeordnet werden. Auf diese Weise ist es sehr leicht möglich, die Funktionen neuer DLLs in das Menü einzubinden. Nur der Aufbau der Symbolleiste ist den Konventionen der MFC entsprechend feststehend und muß mittels Resourceneditor bearbeitet werden.

Bildstrukturen und Zugriffsmethodik

Ein zentrales Moment der Bibliothek ist der Definitionsrahmen für Bildstrukturen und die zugehörigen Zugriffsmechanismen. Dabei werden vier hierarchisch aufeinander aufbauende Ebenen verwendet. Die unterste Ebene wird von den physischen Speicherblöcken gebildet. In der nächsten Ebene (**`cImgBuf`**) werden diese Bildpunktwerte in eine zweidimensionale Struktur aus Zeilen und Spalten gebracht, die über den Zeilen- und Spaltenindex adressiert werden. Das Koordinatensystem des Bildes, über das die Pixelwerte angesprochen werden, und der darin gültige Bildbereich werden in der Ebene darüber, dem **`cBand`**, festgelegt. Ein **`cBand`**-Objekt enthält außerdem einen Verweis auf den Bandtyp, der völlig frei definierbar ist und weiter in Subtypen, sogenannte Kanäle, strukturiert sein kann. Ein standardmäßig definierter Bandtyp ist z.B. **`cBandColor`**, der aus den Kanälen R, G und B besteht. Ein Band stellt sozusagen einen bestimmten Blick auf einen Puffer im Sinne des Koordinatensystems und des Pixeltyps dar. Es macht damit durchaus Sinn, daß auf einen Speicherbereich mehrere Bänder verweisen: das Band „**`color`**“ liefert z.B. die RGB-Werte und das Band „**`red`**“ den Rotkanal. In der obersten Schicht kann ein Bild vom Typ **`cImg`** mehrere Bänder zu einem Bild zusammenfassen. Dieses Konzept verbindet ein Höchstmaß an Flexibilität mit einem Mindestaufwand bei der Definition und Handhabung unterschiedlichster Bildstrukturen.

Die Standardbedienoberfläche

Auf Basis der genannten Konzepte ist eine Bedienoberfläche entstanden, die für viele Probleme der Farbbildverarbeitung und insbesondere für Echtzeitverarbeitung von Bildsequenzen eine gute Entwicklungs- und Testumgebung bietet. Anhand von Bild 1 soll eine Vorstellung davon vermittelt werden. Kernstück für die online-Verarbeitung der einlaufenden Kamerabilder oder gespeicherter AVI-Videodateien ist der „Videorecorder“, der über Schaltflächen der Symbolleiste bedient wird. Zur Anzeige der Videoquelle ist das Aufnahmesymbol  zu wählen. Das Anzeigefenster trägt in der Titelleiste die Bezeichnung VIDEO (Bild 1, Fenster links oben). Die einlaufenden Videobilder werden im VIDEO-

Fenster angezeigt (ggf. wie in Bild 1 mit Vergrößerung oder anderen Anzeigeoptionen) und in einem Ringpuffer mit vorgebbarer Länge abgespeichert. Dabei überschreibt das aktuelle Bild immer das älteste. Der Bildeinzug kann über das Aufnahmesymbol wieder unterbrochen werden. Danach kann man sich die im Ringpuffer gespeicherte Bildsequenz anschauen. Zur Navigation stehen folgende Funktionen zur Verfügung:  schneller Bildrücklauf,  vorheriges Bild,  Abspielen der Sequenz im Ringpuffer,  nächstes Bild,  schneller Bildvorlauf,  endloser, zyklischer Bildlauf (wenn aktiviert). Die im Ringpuffer enthaltene Bildsequenz kann als AVI-Datei auf die Festplatte geschrieben und wieder in den Ringpuffer eingelesen werden. Die entsprechenden Schaltflächen ( Lesen,  Schreiben) können auch bei laufender Aufnahme und Verarbeitung benutzt werden. Verzeichnisse und Dateinamen werden als Parameter der Menüpunkte festgelegt. Beim Schreiben wird an den als Parameter festgelegten Dateinamen die nächste freie vierstellige Nummer angehangen. Beim Lesen wird die Nummer der ersten Datei intern weitergezählt.

Im Menü “Videoauswertung“ sind Verarbeitungsalgorithmen enthalten, die automatisch mit allen Aufnahme- und Wiedergabefunktionen des Videorecorders verbunden werden können. In Bild 1 sind “Verkehrszeichenerkennung“ und „Gradientenbild“ (Fenster links unten) aktiviert. Für die Verkehrszeichenerkennung sind die Ausgabeoptionen „Labelbild“ (Fenster links Mitte) und „Anzeige erkannter Verkehrszeichen“ (mittleres Fenster) aktiviert. Weitere in Bild 1 enthaltene Fenster sind das Infofenster (Mitte oben: Darstellung der RGB-Werte unterlegt mit den entsprechenden Farben), das Navigationsfenster (rechts oben: Darstellung des aktuellen Fensters in voller Größe mit Anzeige des aktuellen Ausschnittes), der Linienschnitt für die im Gradientenbild dargestellte Linie (rechts unten) und ein Vorschaufenster für die im aktuellen Verzeichnis enthaltenen AVI-Dateien. Bis auf letzteres werden alle Fenster bei laufender Aufnahme oder sonstiger Wiedergabefunktionen des Videorecorders automatisch aktualisiert.

Applikationsbeispiel Verkehrszeichenerkennung

Als erste Applikation, bei der sich auch beweisen mußte, ob eine ausreichende Echtzeitfähigkeit auf normaler PC-Technik erreicht werden kann, wurde ein System zur Erkennung von Verkehrszeichen, eingeschränkt auf Geschwindigkeitsbegrenzungen, entwickelt. Die erreichte Verarbeitungsgeschwindigkeit von 10 bis 18 Videohalbbilder/Sekunde auf einem Pentium II/400 ermöglicht eine Erkennung bei Fahrzeuggeschwindigkeiten bis zu 150 km/h und bestätigt damit die gewählte Vorgehensweise. Die wichtigsten Verfahrensschritte der Verkehrszeichenerkennung sind Farbbildsegmentierung, Single-Pass-Regionenanalyse, Kandidatenfindung, hierarchisches Fuzzy-Matching und Ergebnisfolgenanalyse. Diese Schritte werden immer über dem gesamten Bild ausgeführt.

Farbbildsegmentierung

Aus Performance-Gründen erfolgt die Farbbildsegmentierung nur durch pixelweise Farbklassifikation über zwei Look-up-Tabellen. Die erste transformiert die RGB-Werte mit 3*6 Bit in einen 8-Bit-Wert der sich zusammensetzt aus Buntartklasse (2 Bit für Rot und Unbunt, später kommen Blau und Gelb hinzu) und Grauwert (6 Bit). Diese große Tabelle wird nur einmal bei Programmstart und bei Änderung der Parameter über Menü der Bedienoberfläche berechnet. Die zweite Tabelle trennt die Grauwerte in Weiß und Schwarz. Die dabei verwendete Schwelle ist dynamisch und wird aus der Vorgeschichte abgeleitet, wobei vor allem auf bereits erkannte Verkehrszeichen Bezug genommen wird. Im Ergebnis liegt ein sogenanntes Labelbild vor, das nur noch die Farbklassen Rot, Schwarz und Weiß

enthält.

Regionenanalyse

Im nächsten Schritt wird mittels eines äußerst effektiven Single-Pass-Verfahren das Labelbild in verkettete Listenstrukturen („Regionenliste“) umgewandelt. Diese Strukturen beschreiben die Form aller im Labelbild enthaltenen zusammenhängenden Regionen mittels Konturlisten (jede Region hat genau eine Außenkontur und keine oder beliebig viele Innenkonturen). Außerdem werden einfache Merkmale und die topologische Beziehungsstruktur zwischen den Regionen ermittelt.

Kandidatenbildung

Durch Auswertung der Regionenliste werden Hypothesen über Positionen und Größen möglicher Verkehrszeichen ausgedrückt in umschreibenden Rechtecken abgeleitet. Im einfachsten Fall kann ein Verkehrszeichen vermutet werden, wenn ein roter Ring vorliegt in dessen Inneren genügend weiße Regionen vorhanden sind. Daneben werden aber auch andere Hypothesen zugelassen, die möglichst allen denkbaren Störungen entsprechen. So werden auch Fragmente roter Ringe oder weiße kreisförmige Objekte zum Ausgangspunkt von Hypothesen genommen. Weiterhin werden, da sowohl Außen- als auch Innenkonturen von Regionen gestört sein können, verschiedene Größenannahmen zugelassen. Letztendlich entstehen Kandidatengruppen, d.h. Gruppen von Rechtecken, wobei eine Gruppe ein mögliches Verkehrszeichen repräsentiert.

Hierarchisches Fuzzy-Matching

Für die Erkennung wird ein Klassifikator eingesetzt, der auf der Grundlage eines Fuzzy-Matchings der Kandidaten-Teilbilder mit Fuzzy-Masken und fuzzy-logischen Ausdrücken Zugehörigkeiten zu Hypothesen (durch die fuzzy-logischen Ausdrücke gegeben) bestimmt. Die besten Hypothesen für jede Kandidatengruppe bilden dann die Ergebnismenge. In dieser sind dann zu jeder Hypothese ein Gütemaß angegeben.

Das Matching erfolgt in drei Stufen. Für jeden der rechteckigen Bildausschnitte aller Kandidatenlisten wird zunächst geprüft, ob eine der gegebenen äußeren Formen (hier nur roter Kreisring) eingenommen wird. Das beste der Rechtecke pro Kandidatengruppe wird ausgewählt und weiterverwendet, wenn das Ergebnis eine bestimmte Güte überschreitet. Wird die vorgegebene Güte nicht erreicht, so wird die Kandidatengruppe gestrichen. Es handelt sich dann um kein relevantes Verkehrszeichen. Das lokale Wissen über das Vorhandensein eines Verkehrszeichens führt dann zu einer besseren Farbsegmentierung, die sowohl das weitere Matching, wie auch die globale Segmentierung der weiteren Bilder positiv beeinflußt. Für jeden Kandidaten werden dann Hypothesen über die möglichen enthaltenen Piktogramme berechnet. Da diese Auswahl über alle möglichen Piktogramme erfolgt, werden nur grobe Ausdrücke berechnet (ohne Korrelation), die aber die Lösungsmenge schon erheblich einschränken. Auch hier kann es zum Verwerfen des Kandidaten kommen, wenn kein Piktogramm mit einer ausreichenden Güte erkannt wird. In einer verfeinerte Klassifikation der Piktogramme werden die Güten der Hypothesen neu berechnet. Zur Anwendung kommt dabei auch eine Korrelation, um kleine Verschiebungen der Bilder gegenüber dem Verkehrszeichenmodell auszugleichen. Außerdem kann man sich auf signifikante Untergebiete beschränken, in denen sich die groben Hypothesen wesentlich unterscheiden.

Ergebnisfolgenanalyse

Aufgabe der Folgenanalyse ist es, aus den unsicheren und unscharfen Aussagen der Matchingstufe in der zeitlichen Abfolge eindeutige Entscheidungen über das Vorhandensein von Verkehrszeichen zu treffen. Dabei besteht natürlich das Dilemma, daß frühestmögliche Erkennung und sichere, eindeutige Erkennung im Widerspruch stehen.

Die Hypothesen der Matchingstufen werden in eine Liste einsortiert, die für jedes Verkehrszeichen in zeitlicher Folge die Hypothesen aller vergangenen Analyseschritte enthält. Wenn Mehrfachhypothesen entstanden sind (z.B. 80 oder 30 bei einem Kandidaten), so wird jede Hypothese einzeln mit ihren Alternativen eingeordnet. Für jede Verkehrszeichenhypothese wird eine Folgengüte berechnet, die die Sicherheit des Erkennens dieses Zeichens angibt. Übersteigt die Güte eines oder mehrerer Zeichen ein Mindestmaß, so werden diese Zeichen als erkannte Verkehrszeichen betrachtet. Alternativen zu erkannten Zeichen werden aus der Liste gestrichen. Ebenso wird die Liste wird von veralteten Einträgen befreit, d.h. Hypothesen, die sich nicht durchsetzen konnten, werden wieder vergessen.

In den Bildern 2 bis 5 ist dieser Prozeß nachvollziehbar. In Bild 2 wird zwar ein Kandidat gefunden, aber nicht als Verkehrszeichen erkannt. In Bild 3 wird eine 80 mit der Güte 0.66 gefunden. Diese Güte steigt in dem nächsten Bild auf 1.51 und übersteigt schließlich in Bild 5 das Mindestmaß (Ausrufezeichen in der Titelzeile).

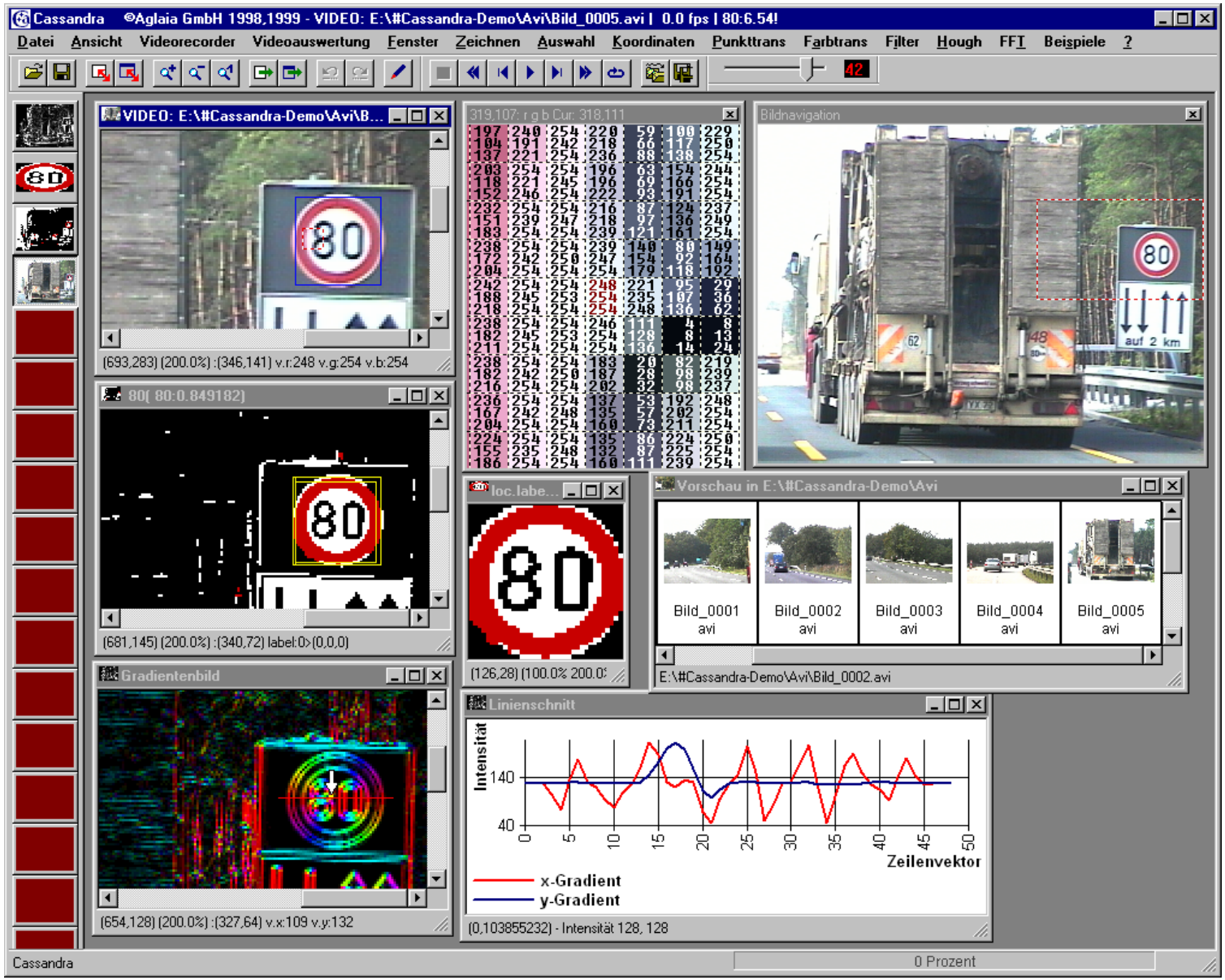
Cassandra ©Aglaia GmbH 1998,1999 - VIDEO: E:\#Cassandra-Demo\Avi\Bild_0005.avi | 0.0 fps | 80:6.54!
Datei Ansicht Videorecorder Videoauswertung Fenster Zeichnen Auswahl Koordinaten Punkttrans Farbtrans Filter Hough FFT Beispiele ?
VIDEO: E:\#Cassandra-Demo\Avi\B...
(693,283) (200.0%) :(346,141) v.r:248 v.g:254 v.b:254
80(80:0.849182)
(681,145) (200.0%) :(340,72) label:0>(0,0,0)
Gradientenbild
(654,128) (200.0%) :(327,64) v.x:109 v.y:132
319,107: r g b Cur: 318,111
loc.labe...
(126,28) (100.0% 200.0
Bildnavigation
auf 2 km
Vorschau in E:\#Cassandra-Demo\Avi
Bild_0001 avi
Bild_0002 avi
Bild_0003 avi
Bild_0004 avi
Bild_0005 avi
E:\#Cassandra-Demo\Avi\Bild_0002.avi
Linienschnitt
Intensität
140
40
0 5 10 15 20 25 30 35 40 45 50
Zeilenvektor
x-Gradient
y-Gradient
(0,103855232) - Intensität 128, 128
Cassandra
0 Prozent

Bild 1

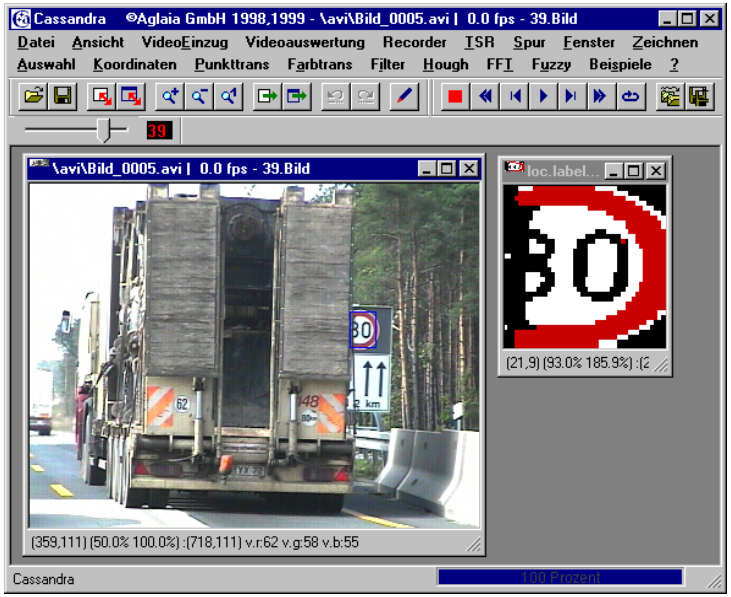

Bild 2

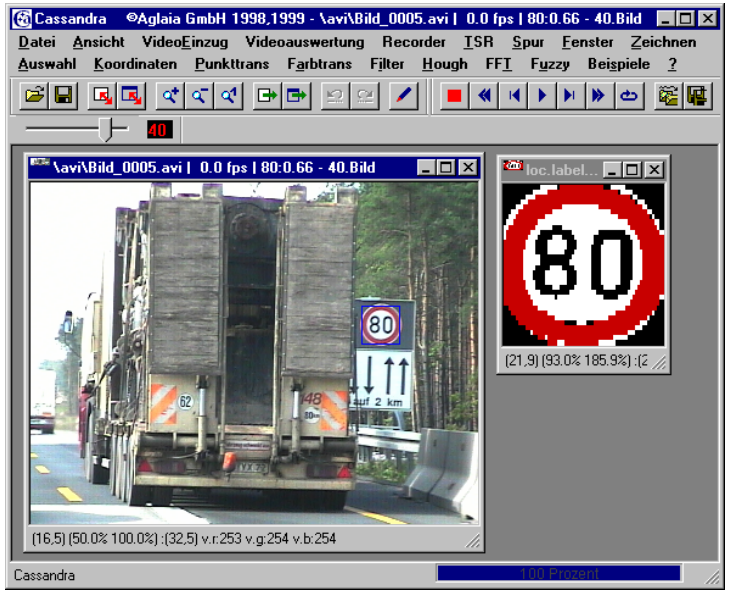

Bild 3

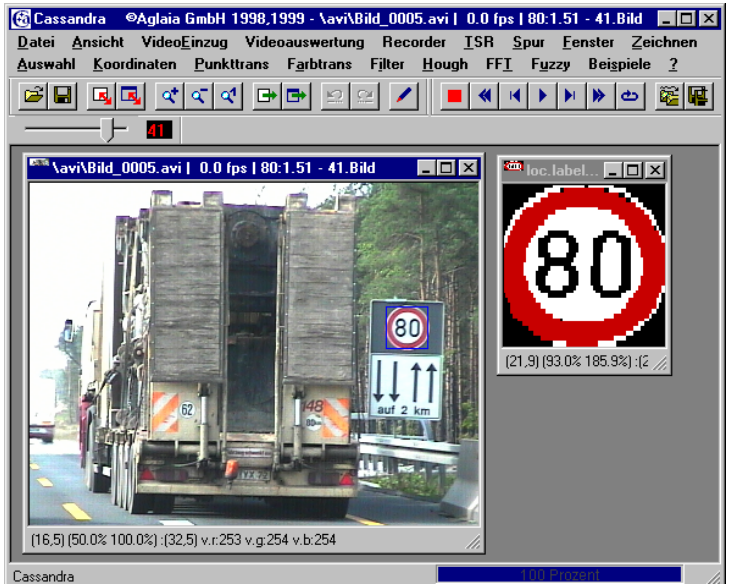

Bild 4

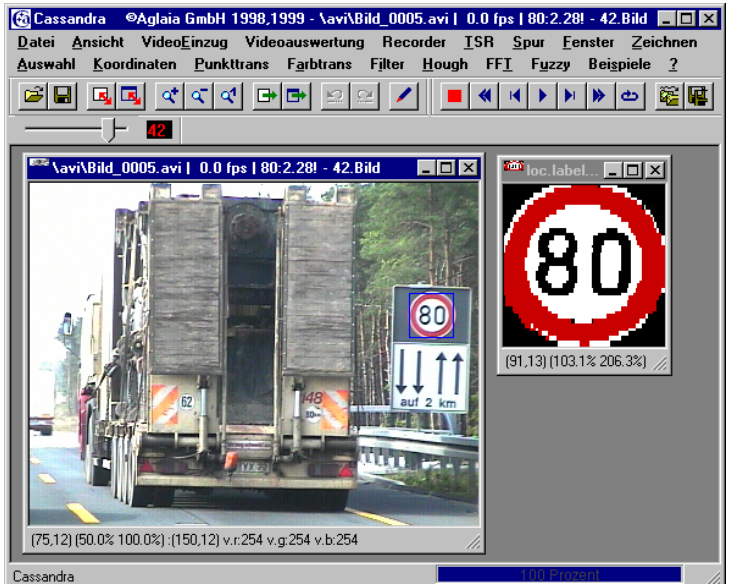

Bild 5