D p o a r

a r a E , r o n n r n -

o a a r r

n

Inhaltsverzeichnis

Inhaltsverzeichnis

E

n n n n n An n n n n

n n B n p n n

n B n n n Z n n n

B n nn D n B n n n B nn n

n n B n n n B n n

Z n n n n n nn n

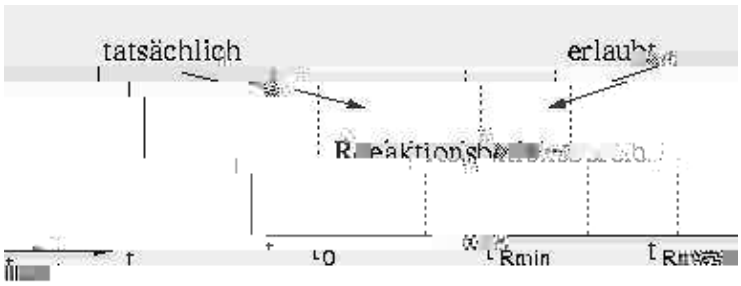
n n n n

n n n n n n nn n

- E n n n

n nn n n Z p nn

Abbildung 2-1. Reaktionsbereich



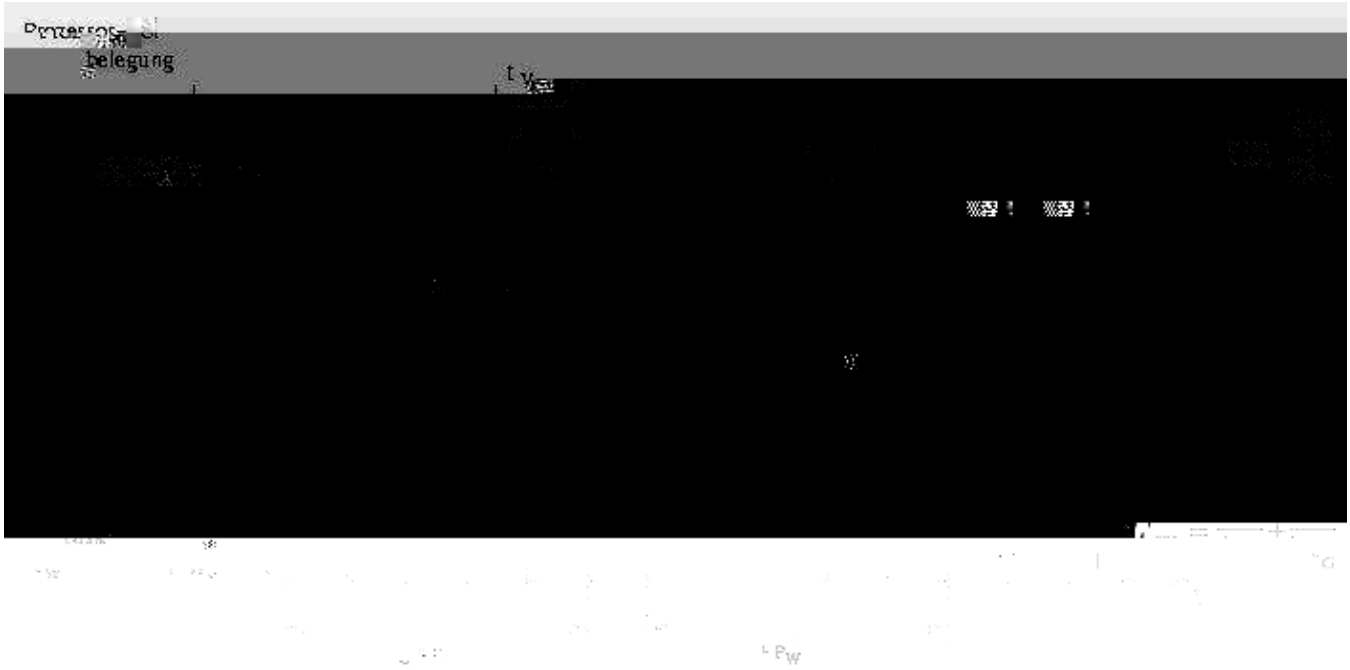
n

n n n Z p n n

n n Z p n

n $\mathbf{A_A}$ *Reaktionsbereich*

$$t_{Rmin} \leq t_R \leq t_{Rzul}$$



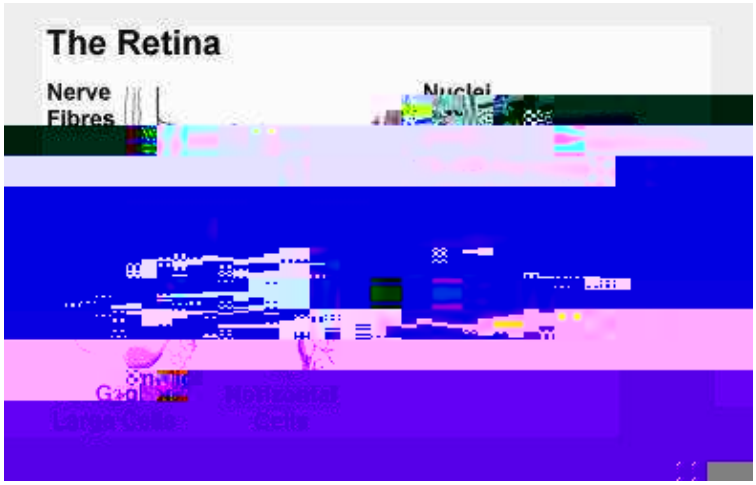
a r n E n n

An n n n n n
nn n n p nn n n nn n B n
n n n n n n n n n
n n n nn n An n n *ISS* n p p
n n D n n n p n n
n *harten Echtzeitbedingung* n n nn p n n n *weichen Echtzeitbedingung* B *Kostenfunktion einer harten und weichen Echtzeitbedingung*

Abbildung 2-5. Kostenfunktion einer harten und weichen Echtzeitbedingung

Abbildung 2-8. Die Kamera

n n A. A. A. *Das Auge [UniWup1998]* A. n A. *Die Kamera* n nn n n n n A.


The Retina
Nerve
Fibres

B n n n n n n n *Subtraktive Farbmischung* n n B n p *Das CIE-Farbsystem* n n n n n D A n n nn n n n n A n n n n B n n n n n

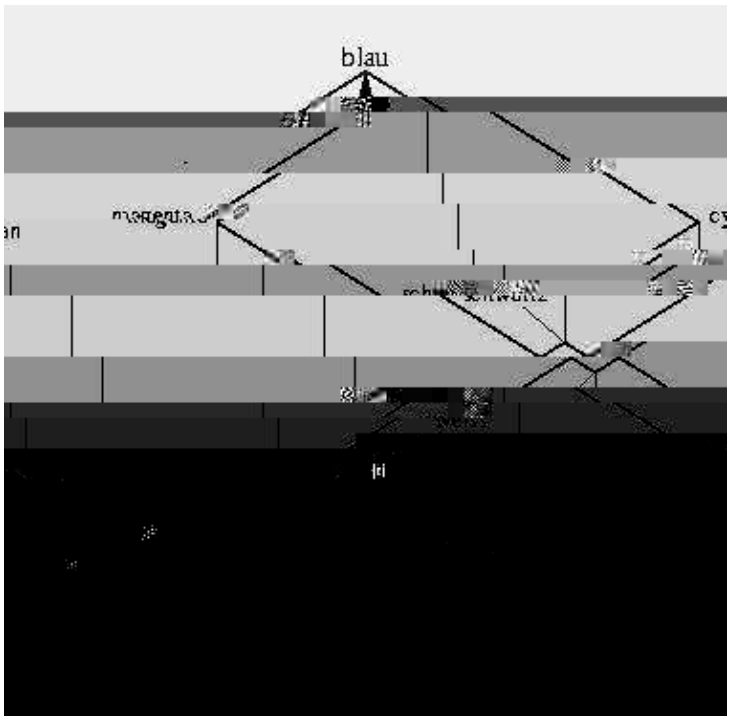
D n *Additive Farbmischung* B n q n n n n n B n n n p n D n n n p *Das CIE-Farbsystem* n n n

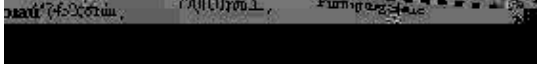
D B n A n n n n B p n n p *Additiven Farbmischung* D n n B An n n n B n n An n n

n p *Das CIE-Farbsystem* n n n n n n B A *Der RGB-Farbraum*

B nn n n n B n n n n n n D n n n n n n

n n n B





Der RGB-Farbraum und der sichtbare Farbraum [Hartl2002]

Abbildung 2-11. Der RGB-Farbraum und der sichtbare Farbraum [Hartl2002]

virtmit en Primärva enzen

virtmit e Primärva enzen

CIE-Normfarbtafel *CIE-Normfarbtafel*

[Seilnacht2002]

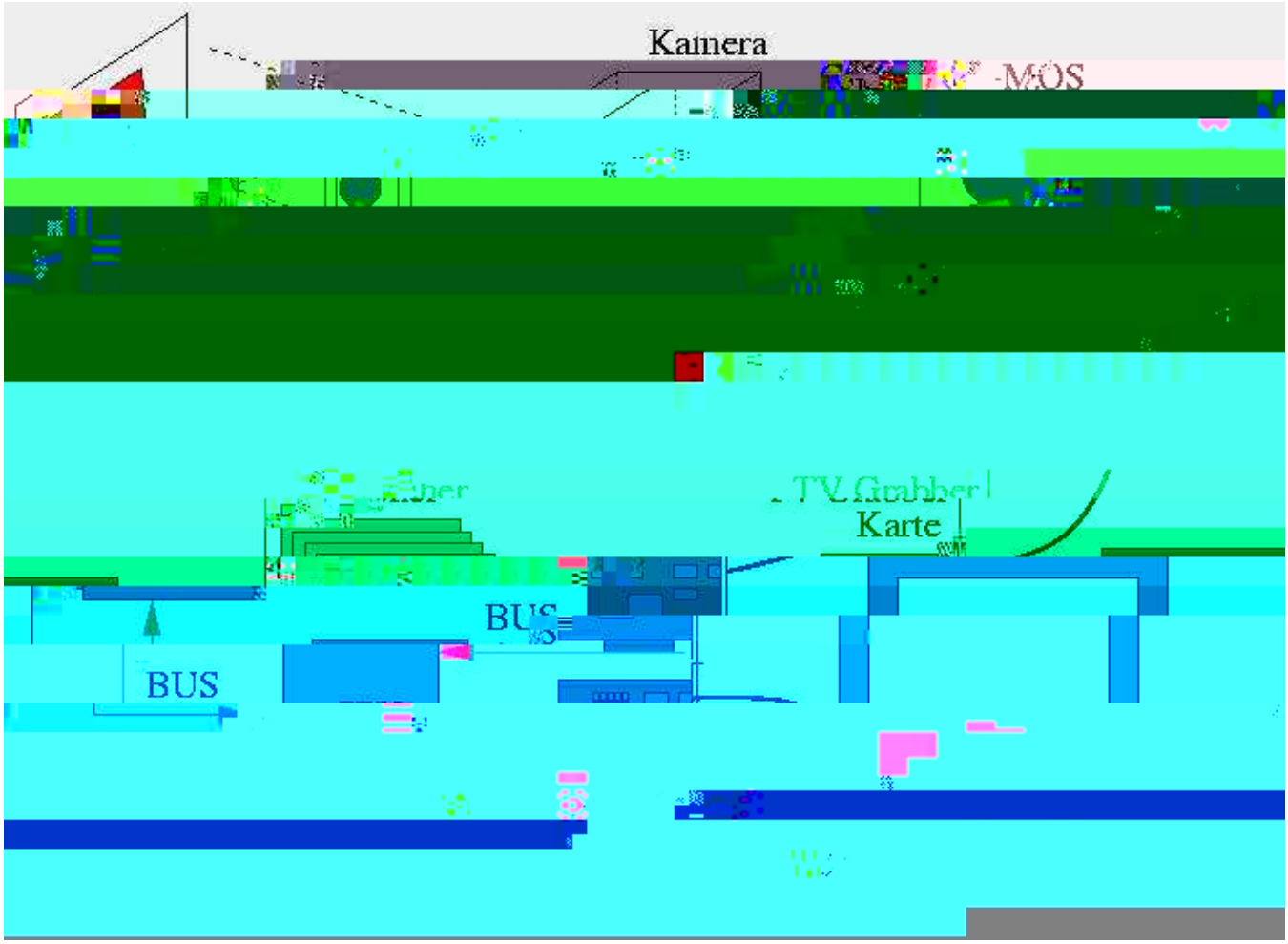
Z n

a p - r nn n , r o n

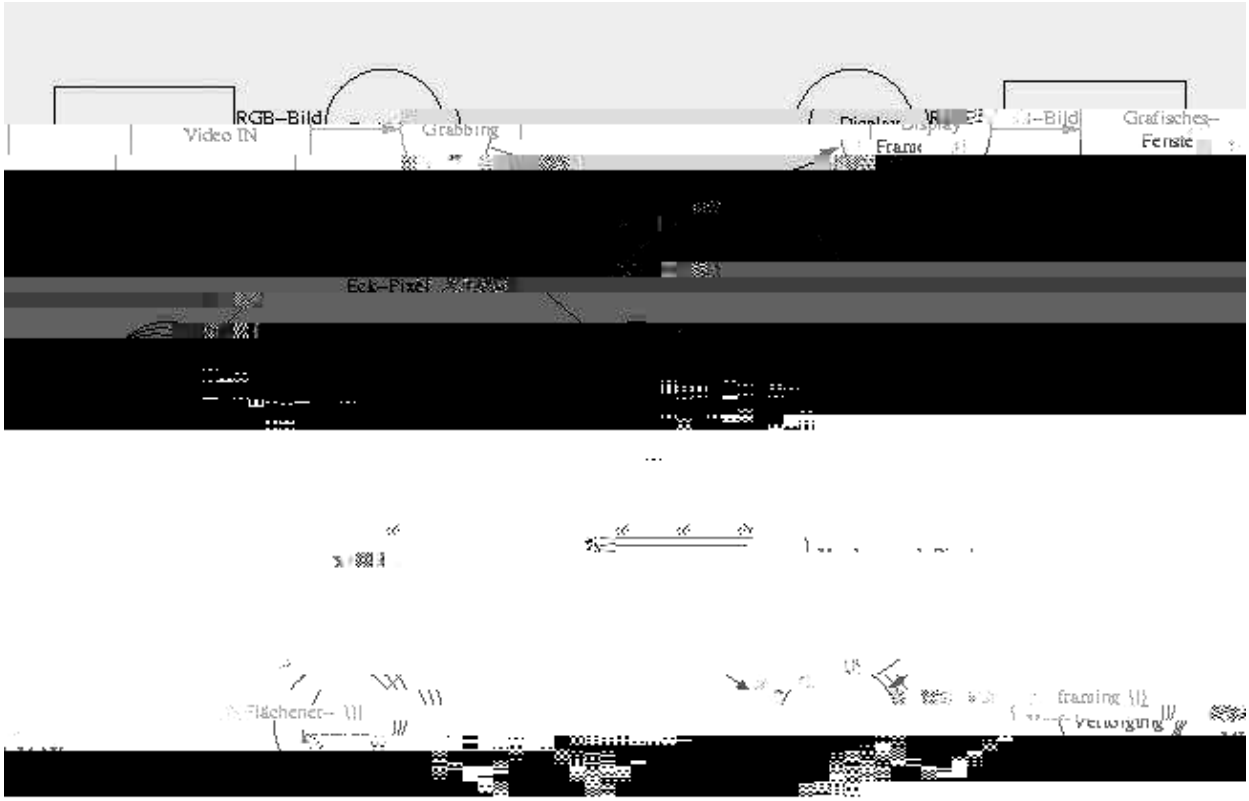
Inhaltsverzeichnis

A a n r r nn n

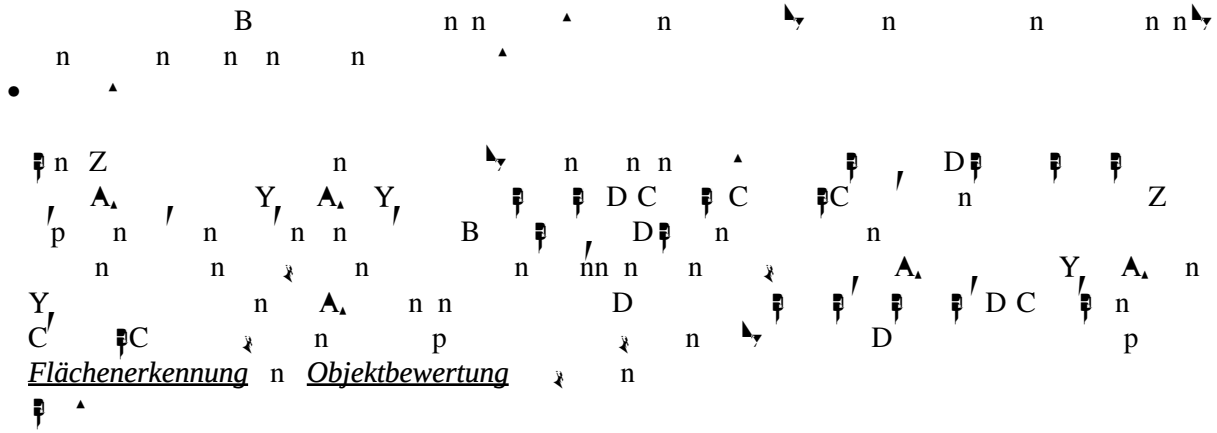
Abbildung 3-1. Aufbau einer Objekterkennung



n A n *Aufbau einer Objekterkennung* n n n n
n n n C n n n n Z B
n p n nn n n n p An n
n n n *CCD* *C-MOS*



Data Dictionary:



D n A *Verschachteltes Grabben Version 2* n n C n A

Verschachteltes Grabben

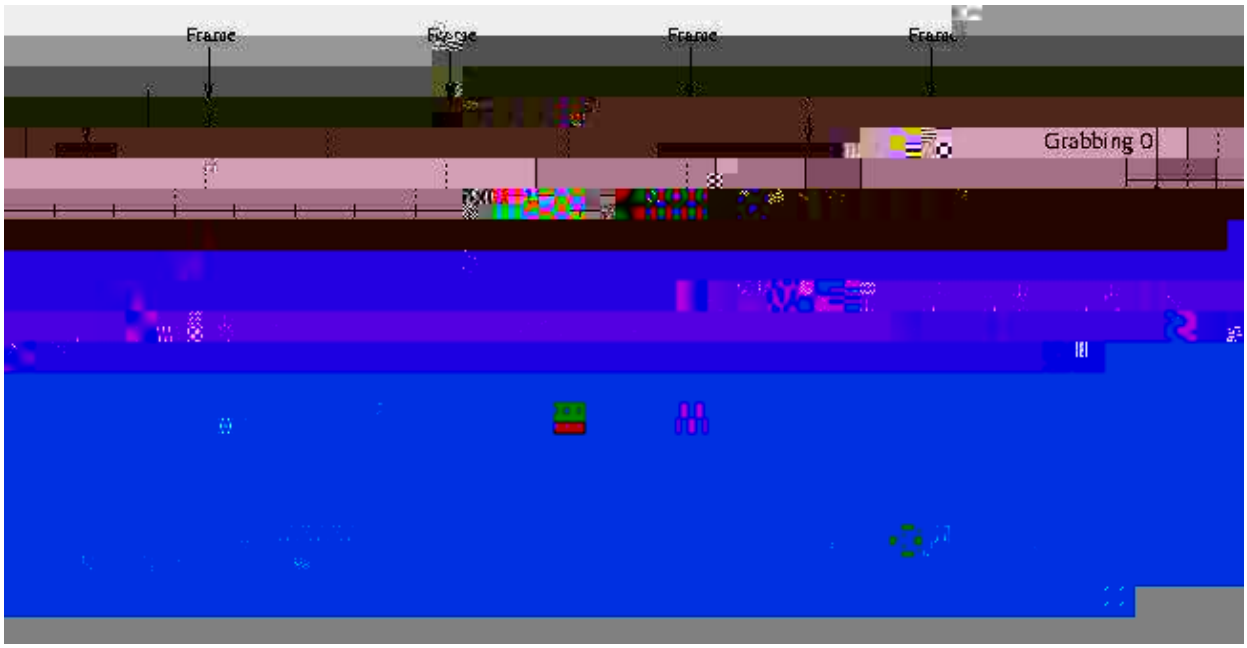
Abbildung 3-4. Verschachteltes Grabben Version 2

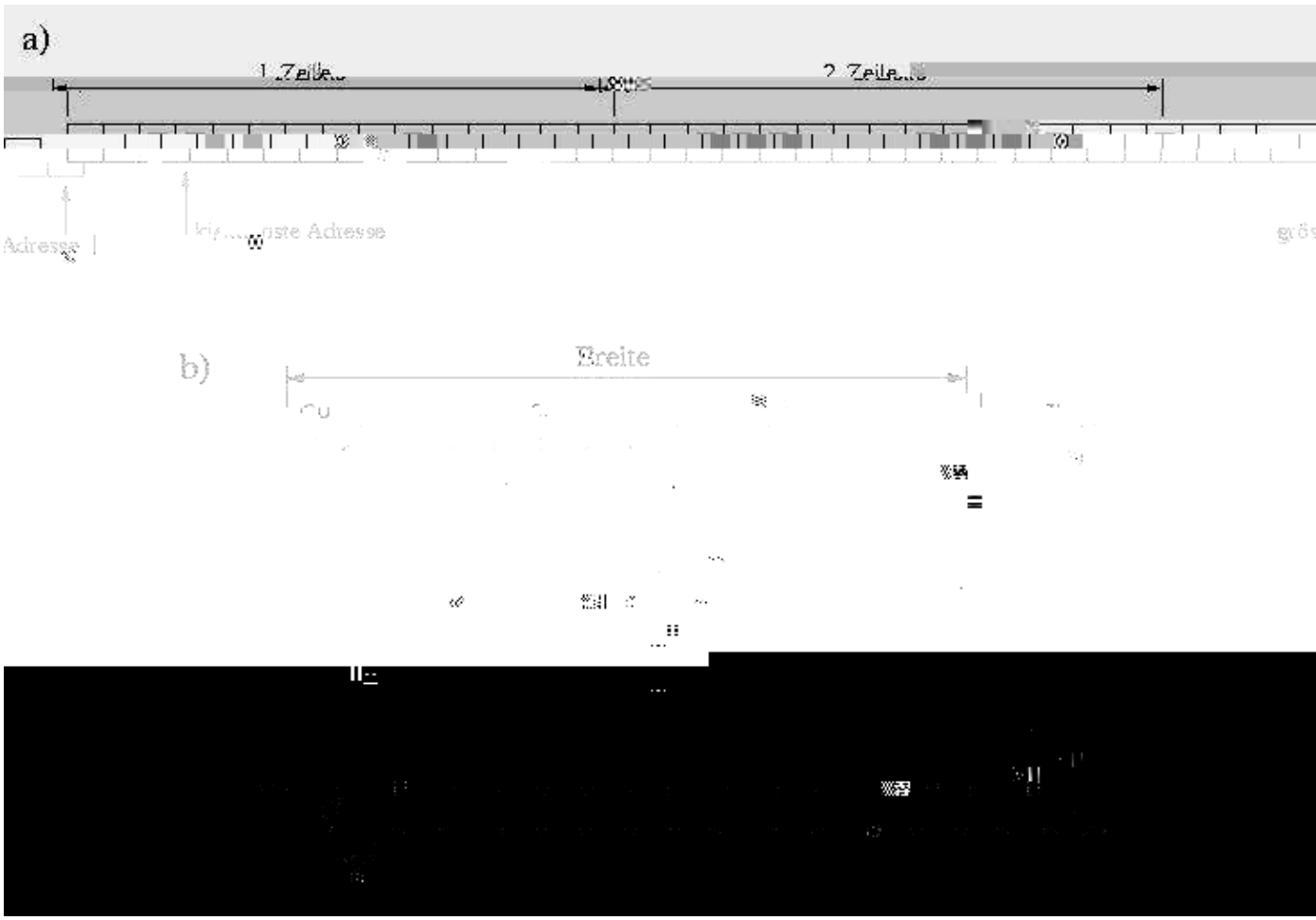
```
FRAME=0;

OPEN();
INIT();
GRAB(FRAME);

while (!Abbruch){
    GRAB(!FRAME);
    SYNC(FRAME);
    WORK(FRAME);
    FRAME = !FRAME;
}
CLOSE();
```

n n n A *Verschachteltes G*

n n Z n n n

n D A n n An


Frame
Frame
Frame
Frame
Grabbing 0

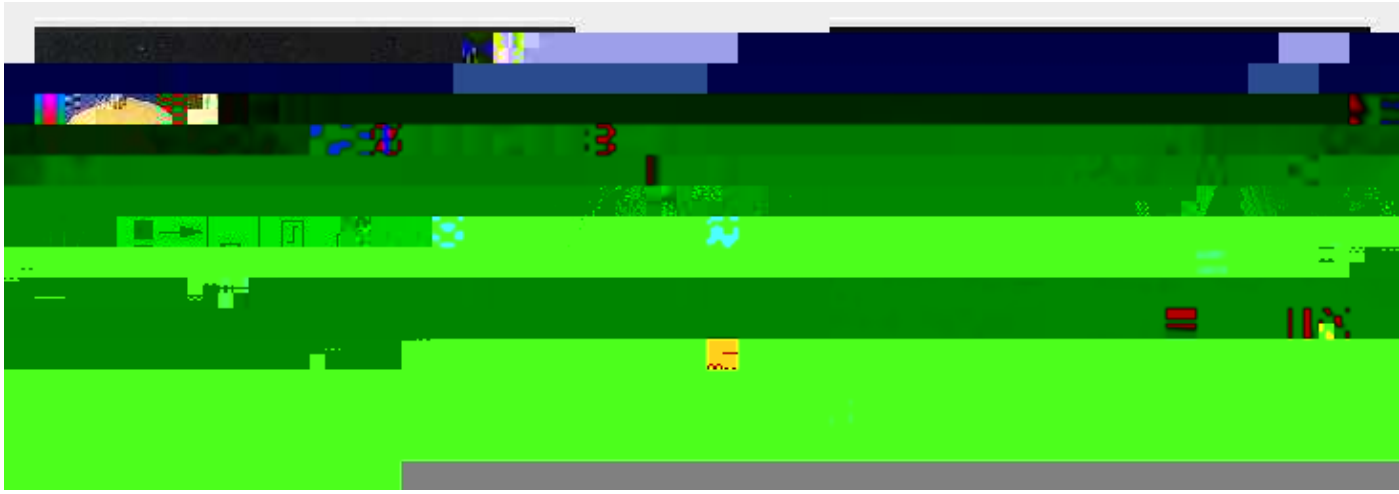


B B n n n n B n n n n A *Die Verarbeitungsrichtung der Segmentierung und der Flächenerkennung - Teilbild b)* n Z n n n D n n p n B B n An n n n A n n Y A n n p n n D n n p n p n n

n p n n n n n n n n Y A n n n n n n p n nn n n D p n n n n Y

a r o n

n n n p n n n n n n n B n n n n n n n



n n n n n n n n nn n
n n n D n nn *logisch UND-Verknüpft*
n nn nn n n *Zweistufiges Threshold für jede der drei Farben* n n B n *Beispielcode: Zweistufiges Threshold für jede der drei Farben*

D n p n p n n n n o_tracing::threshold
B p pp n D n o_tracing::set_rgb_threshold nn n
n n n

Abbildung 3-11. Zweistufiges Threshold für jede der drei Farben

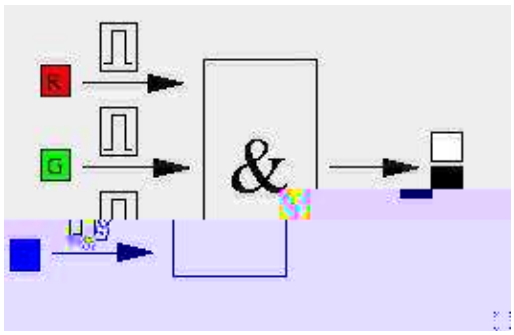


Abbildung 3-12. Beispielcode: Zweistufiges Threshold für jede der drei Farben

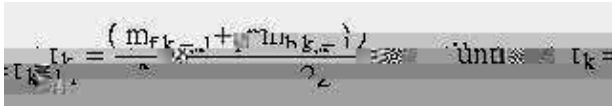
```
if (     (ROT_MAX   >= Q_Punkt[rot]   > ROT_MIN  )
      && (GRUEN_MAX >= Q_Punkt[gruen] > GRUEN_MIN)
      && (BLAU_MAX  >= Q_Punkt[blau]  > BLAU_MIN ) ){
      Z_Punkt=0;
}
```

n l n

n n p *Isodata Algorithmus* n n C

Z B nn n n B n n n n n n D n n B n n n

$$\text{Mittelwert} = \frac{\sum_{i=0}^{n} (\text{Helligkeit}_i \cdot \text{Menge}(\text{Helligkeit}_i))}{\sum_{i=0}^{n} \text{Menge}(\text{Helligkeit}_i)}$$



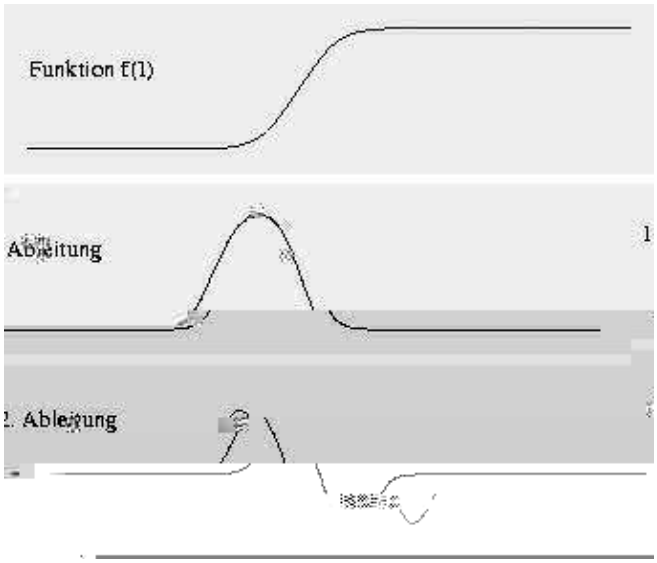
n nB n n n n n
A n n n An n A
C l n

A ap r o n

B n n n n n n

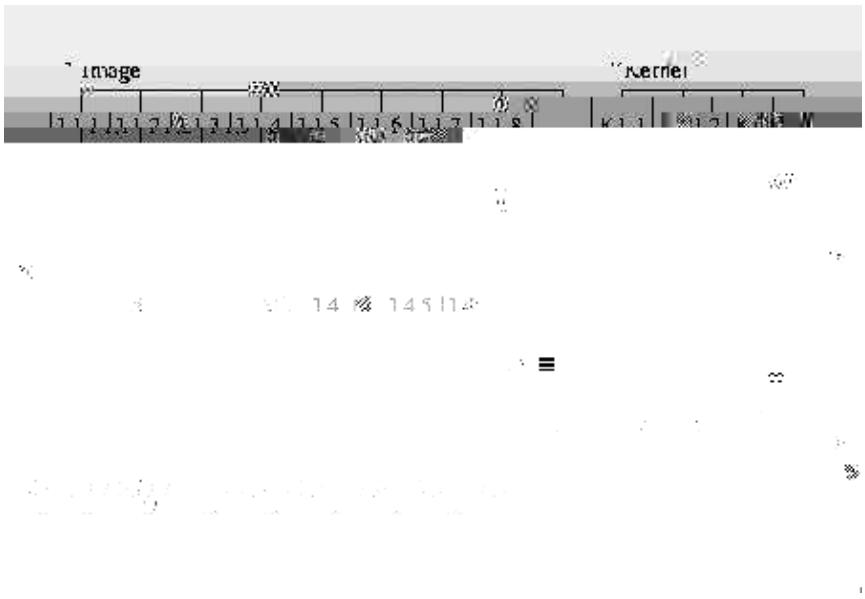
A *Eine Kante mit ihrer ersten und zweiten Ableitung* Z nn n n n n
A n n n n n nn A n n n
n n n A n n n n

Abbildung 3-19. Eine Kante mit ihrer ersten und zweiten Ableitung



An n A n n B n n n *Convolution*
n n n nn n A n n n l *Kernel* n
n n n A n n n l *Kernel*

Convolution n n D n n n
n n n n n B n



C n n n *Kernel* n n n n n B n n *Kernel* p n n n A n n n n n n p D A n nn p n n n p A *Veran2phauung eines Image und eines Kernel* B n n n n A *Berechnung einer Convolution*

Abbildung 3-21. Berechnung einer Convolution

A n B n n p *Formel zur Berechnung einer Convolution* D Z n n p n n Z n n p n n *Kernel*

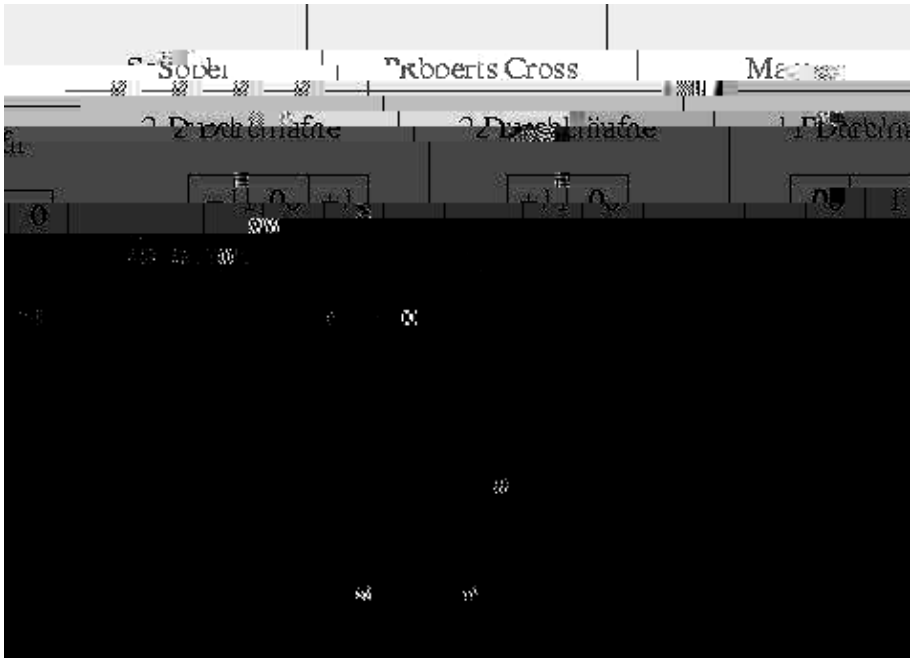
Abbildung 3-22. Formel zur Berechnung einer Convolution

n n n p B n n n n n A n

D n n *Kernel* n n nn n n p n A n n n *Sobel* n *Roberts Cross*

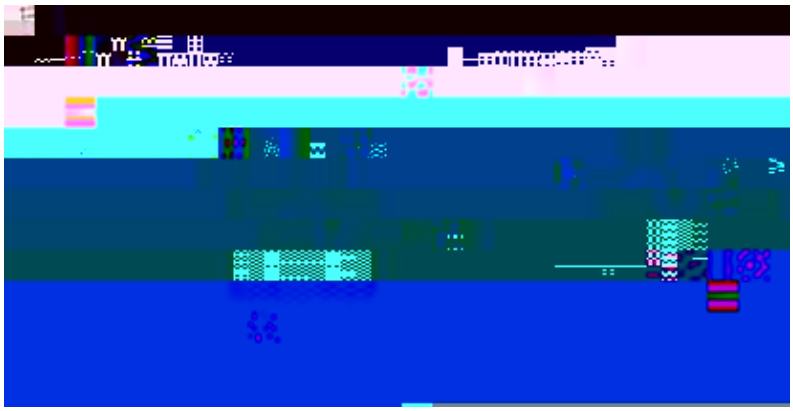
n *Kernel* n A n n *Marr* B A n n n n n *Laplace Tran2formation* D n n n

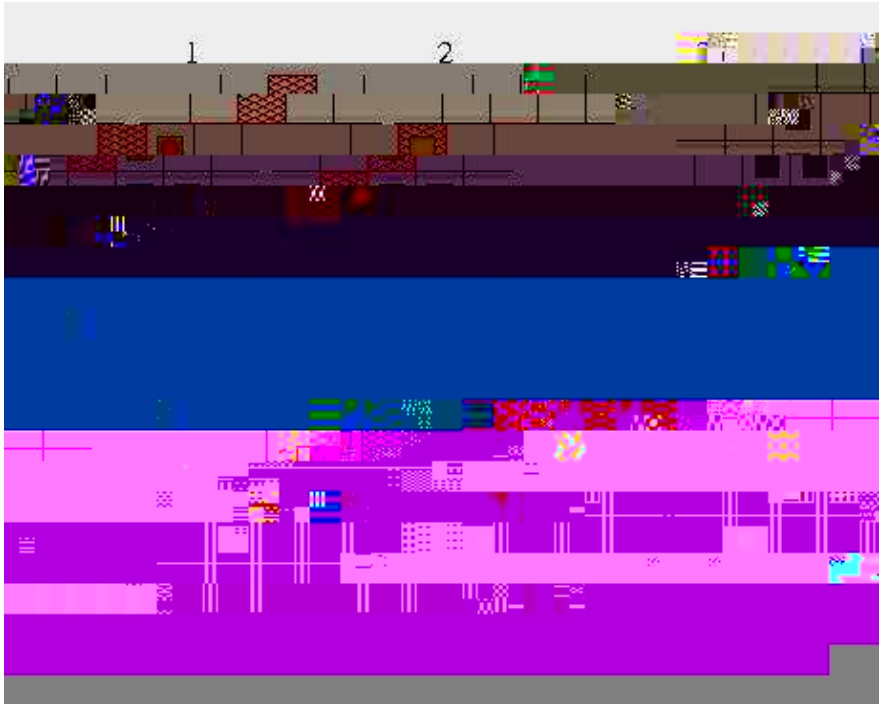
D n n n n n B n D *Sobel* n
n n n B n n n n
n n D nn D



Z p nn n n n

n r nn n

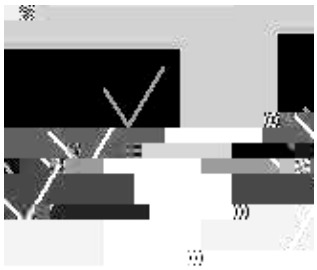


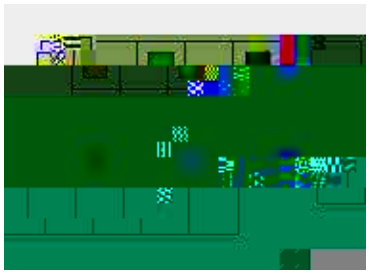


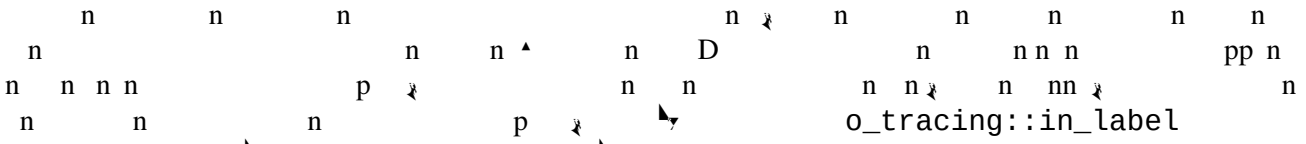
n A *8-connectivity Nachbarerkennung* n An n n n n n
A n nn n
n n n n n n p n D n B n n
A n n An n n p n
nn A n nn n n n
n n n n nn n n n
nn n n

Abbildung 3-27. 8-connectivity Nachbarerkennung

o_tracing::labeling







D o_tracing::in_label

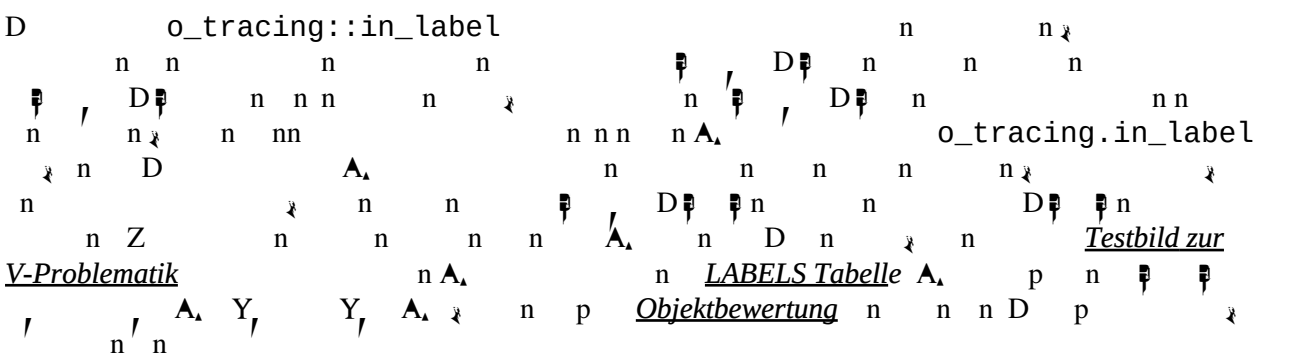
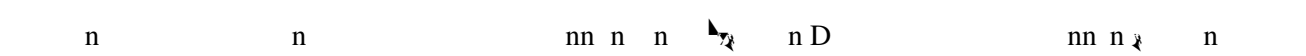


Abbildung 3-30. LABELS Tabelle

INDEX	NEXT_I.	MENGE	X_MAY	X_MIN	Y_MAX	Y_MIN	ROOT
0	0	0	0	0	0	0	0
1	2	8	71	62	28	13	1
2	4	9	67	55	39	19	1
3	4	6	61	28	52	28	1
4	0	21	60	38	70	30	1



r n

Um die MENGE stärker in die Gewichtung einfliesen zu lassen, wird die Dichte mit der MENGE

r a n r o n

Unter Subframing versteht man, dass nicht das gesa9te Bild nach dem Objekt durchsucht wird, sondern nur der Teilausschnitt, in welchem das Objekt erwartet wird.

Diese Technik hat den VorteiiaeVorteiür die Vorgänge des Thresholding und Labeling nur ein kleiner Bereich bearbeitet werden muss. Dies wirkt sich auch auf die Anzahl der von der Selektion zu bearbeitendernung/Obj aus. So ist es möglich, in jedem dieser Bereiche Rechenleistung einzusparen und dem Wunsch nach Einhaltung dern1. und 2. Echtzeitbedingung entgegenzukommen.

Je nach ***Frame-Grabber-Karte*** wird sogar das Grabben eines Subframes aus dem Bildstrom unterstützt. Diese Technik hätte zwar den VorteiiaeVortesich das Datenaufkommen schon beim Grabvorgang reduzieren liesse. Allerdings würden bei dernVerfolgung mehrerernung/Obj mehrere solcher Subgrabvorgänge anfallen, wodurch wiedernmehr Zeit in Anspruch genommen werden müsste. Zum anderen hätte die Applikation mit dynamischen Speicherbereichen zu arbeiten, watesich ebenfalls negativ auf die Bearbeitungszeit des Prozesses auswirken würde.

Ein anderer Vorteii des Subframing ist, dass nur dernBereich um das Objekt durchsucht wird, so dass Störungen außerhalb des Bereiches nicht wahrgenommen werden.

Nachteiihaft bei dieser Technik ist, dass ein Objekt, das nicht an der erwarteten Position, sondern außerhalb

Nr, daenlebewegersM ni Objekungsut es mök istaieliesse.

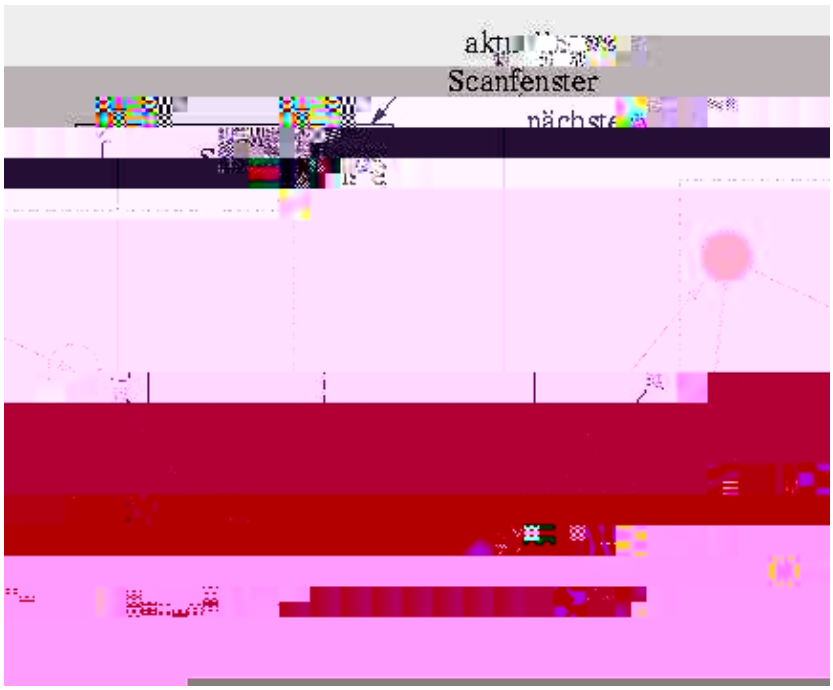
min. und max. Koordinaten, die Mitte eines um den Körper ausgespannten Viereckes errechnet. Addiert man diese zu den min. Koordinaten des Objektes, erhält man die Lage des Objektes im Frame. Zieht man von diesem Punkt die Koordinaten des im vorhergehenden Frame genauso errechneten Punktes ab, so ergibt sich die Richtung, in die sich das Objekt seit dem letzten Frame bewegt hat. Addiert man diese Richtung zur aktuellen Position des Objektes, ergibt sich der Punkt, an dem das Objekt erscheinen wird, wenn es seine Richtung und Geschwindigkeit beibehält.

Der Subframe sollte ein Quadrat um diesen Punkt darstellen, das mindestens um ein Pix h Punknes duiochsen größer ist als die viereckige Fläche, in der sich das Objekt befindet. Würde die Subframegröße genauso groß oder kleiner als die Objektgröße gewählt o b en, hätte das Objekt keine Möglichkeit größere Ausmaße anzunehmen, wie es nesspielswesse durch eine Drehung eines nicht kressrunden Körpers geschieht. Da das Fenster jedoch nes jeder Möglichkeit kleiner o b en würde, ergäbe sich nach kurzer Zeit eine Subframegröße von null. Um dieses zu vermes duiund um sicherzustellen, dass das Objekt auch noch erkannt wird, wenn es nicht genau Punkder errechneten Position erscheint, wird z.B. in der Besspielapplikation die Fenstergröße doppelt so groß wie die Objektbreste gewählt. Darüber hinaus wird das Subfenster noch durch einen dynamischen Faktor in die Bewegungsrichtung vergrößert, der von der Geschwindigkeit des Objektes abhängt.

Sobald sich ein Objekt dem Rand des Frames (nicht Subframe) nähert, besteht die Gefahr, dass der Subframe die Grenzen des Frame Array oder die logische Höhen- Bresteneinteilung verletzt. Deshalb müssen diese Grenzen überwacht und gegebenenfalls der Subframe eingeschränkt o b en.

Bewegt sich das Objekt zu unregelmäßig oder wurde der Subframe zu klein gewählt, kann es geschehen, dass das Objekt verloren geht (siehe Abb. *Verlust des Objektes*). In einem solchen Fall muss der Subframe wieder vergrößert o b en, um das Objekt wieder zu finden.

Abbildung 3-38. Verlust des Objektes



Wurde das Objekt erfolgresch erkannt, kann begonnen w b en, diese Informationen w sterzugeben. In der

Beispielapplikation geschieht dies durch die Wiedergabe des durch die Kamera aufgenommenen Bildes in einem Fenster. Ein verfolgtes Objekt wird hierbei durch kleine Ecken an den Kanten des Subframefensters hervorgehoben. Die entsprechende Imlementierung befindet sich in der Methode `o_tracing::draw_tracing_frame`.

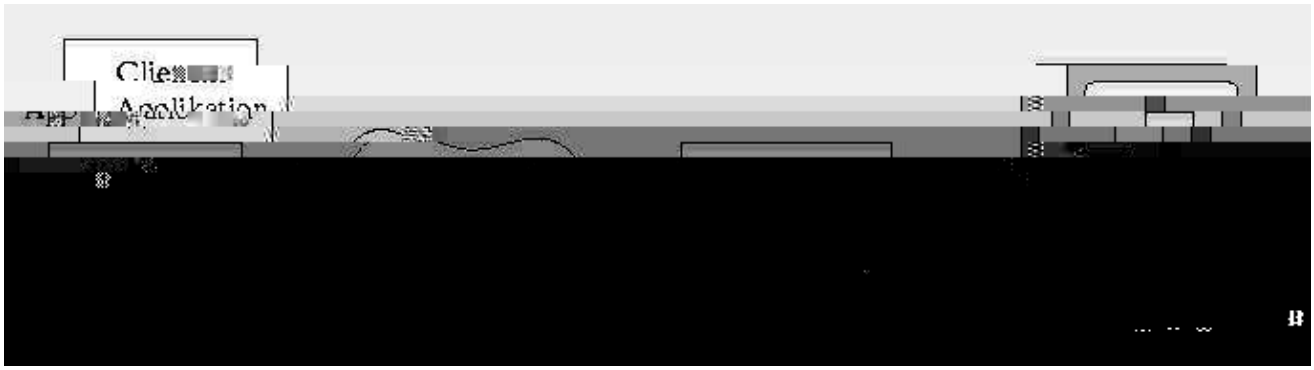
Zurück
Objektbewertung

Zum Anfang
Nach oben

Nach vorne
Grafisches-Fenster

X n o C n r r n p

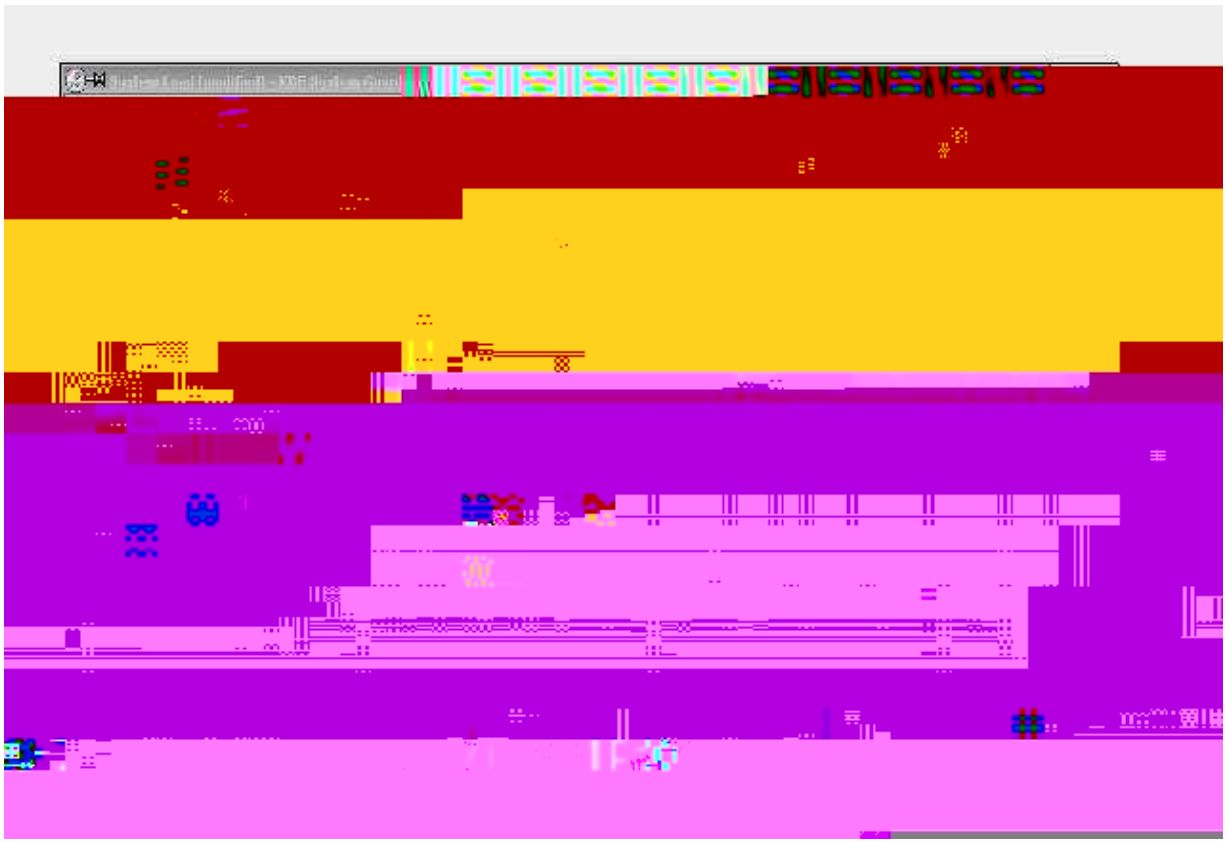
Wie in Abb. *Das X-Window-Client-Server-System* dargestellt, wird das X-Window System in eine Client-Seite und eine Server-Seite unterteilt. Auf der Server-Seite läuft der X-Server, eine Applikation, die den Zugriff auf die Eingabe und Ausgabe Schnittstellen wie Tastatur, Maus, Monitor oder ähnlic7em steuert. Auf der Client-Seite werden eine oder mehrere Applikation betrieben, welc7e über Nachric7ten mit dem Server kommunizieren können. Ein Beispiel für eine solc7e Applikation ist das Beispielprogramm dieser Arbeit. Diese Unterteilung in Client und Server hat die Vorteile, dass eine Applikation auf einem Rechner ablaufen kann, während sie durch einen anderen Rechner über ein Netzwerk gesteuert wird. Natürlic7 können sic7 Client und Server auc7 auf dem gleic7en Rechner befinden. Ein weiterer Vorteil dieser Trennung ist, dass beide Programmteile nic7t in der gleic7en Programmiersprac7e geschrieben worden e m oTnsden. s0 rnic7r aus,wrend sie über dase mhweinlic7r X Prtokollr kommuniziere.s Ben derZuwornnung derBeugrife7 Client unm Serverfäell7 aue, dass dies. Eirteilung nic7t der in der NetzwetRechike üleic7enDbefiiationienrspnic7. Ien Servere m Rechner bzrnicnest, nerInformkatiolen oder Recenglestlung für Clienss
Nac vorine]TXr-Skm'jmrX9Subfic7 üb / VDevolgTL atio380.09m'j215.72 714 Tm lc7e über Nver-Sy3b.



der Nachrichten verwendet.

- Das Versenden wird gezielt ausgelöst. Gerade bei der Beispielapplikation, wo sich das Fenster ohne zutun des Benutzers ständig verändert, würde das Warten auf die erste Bedingung zu lange dauern.

Auf der Client-Seite können die eingehenden Nachrichten durch Methoden der Xlib in ihrer



Alle Tests wurde mit einer Auflösung von 320 x 240 Bildpunkten durchgeführt.

Bei dieser Abbildung (Abb. *Systen und Benutzer Last*) wird die Ausl
"User-Load" unter Verwendung einer an die *Frame-Grabber-Karte* angeschlossenen Kamera angegeben. Load ist hierbei ein Maß dafür, wie viele Prozesse bereit sind die CPUs zu nutzen, geteilt durch die Anzahl der im System zur Verfügung stehenden CPUs. Bei einem Load unter 100 Prozent sind die CPUs in der Lage innerhalb eines Zeitabschnittes alle anstehenden Prozesse abzuarbeiten. Je weiter sich dieser Wert der 100 Prozent Grenze nähert, desto stärker ist die CPU ausgel

und "User-Load" beruht darauf von wem die CPU beansprucht wird. Die Beispielapplikation beispielsweise erzeugt hauptsächlich "User-Load" und nur sehr wenig "System-Load". "System-Load" wird von der Beispielapplikation z.B. durch das Aufrufen von *System-Calls* erzeugt. Natürlich erzeugt das Betriebssystem selbstständig eine oft ungleichmässige Menge an "System Load".

Bei der Abbildung der Lasten mit KDE System Guard 1.1.0 fällt auf, dass teilweise User-Load zu System-Load wird. Dieses Verhalten ist wahrscheinlich auf eine Fehlzuordnung zurückzuführen, deshalb ist es sinnvoll in diesen Fällen das "System-Load" dem "User-Load" anzurechnen.

In der Abbildung ist zu erkennen, dass, wenn die *drei Papierkugeln in Subframes* verfolgt werden, die CPU zu

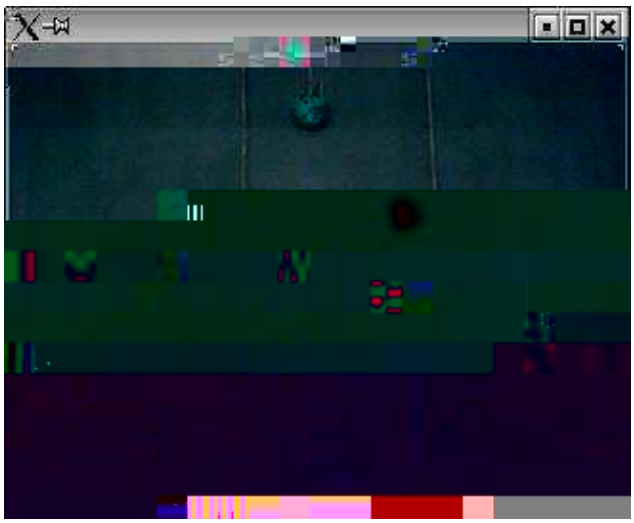
ca. 20-50% ausgel

erreichen. Wird hingegen der Thresholdwert so ungeschickt gewählt, dass *drei mal das gesamte Bild*

bearbeitet werden muss, ist die CPU maximal ausgel

auf 5-6 Bilder pro Sekunde.

Abbildung 5-6. drei Mal das gesamte Bild mit einer USB-Kamera aufgenommen



Dadurch, dass die Load-Darstellung nicht nur von der Beispielapplikation abhängt, sondern auch durch, mit dieser Applikation konkurierende Prozesse, beeinflusst werden, können die dort dargestellten Werte nicht die Beispielapplikation hundertprozentig repräsentieren. Für eine grobe Betrachtung der RechenkernauslasispiRechenkerna

o nA a n

Möchte man genauer untersuchen wie sich Rechenleistung innerhalb einer Applikation verteilt, oder wie oft eine Methode aufgerufen wird, so ist es möglich die Applikation mit der Compiler-Option "-gp" und je nach gcc Version dem Compilerparameter "-fprofile-arcs" zu compilieren (siehe Manpage zu gprof). Wird die Applikation nun erfolgreich ausgeführt und beendet, kann mit Hilfe der Programme gprof oder kprof eine von der Applikation erzeugte Profile-Datei ausgewertet werden.

Im Folgenden sind Auszüge von Profilen dargestellt. Bei den Auszügen wurden die Parameter der Methoden entfernt, um eine höhere Übersichtlichkeit zu erhalten. Alle Profile wurden während einer 120 sekündigen Laufzeit der Beispielapplikation erzeugt.

Die Überschriften der einzelnen Profil-Spalten bedeuten:

- % time

  Der prozentuale Anteil der Programm-Laufzeit, welche diese Methode verbraucht hat.
- cumulative seconds

  Die fortlaufende Anzahl der Sekunden dieser Methode und der ihr übergeordneten Methoden.
- self seconds
-

  Die Anzahl der Sekunden, die für diese Methode als Laufzeit erfasst wurden. Die Tabelle ist nach diesem Wert sortiert.

  DieAnzahl derAufgerden diesFunkikatden.
- sems/aallime
-

  DMigtessgewahlderSekunden, dinfür diese Methode uAufgerdebergeordnetenMethooer dAuffgeach
-

ihr keine andere Applikation die Rechenzeit streitig macht.

```
0.00     30.53     0.00        3     0.00     0.00  o_tracing::set_rgb_threshold()
0.00     30.53     0.00        3     0.00     0.00  o_tracing::set_total_threshold()
0.00     30.53     0.00        3     0.00     0.00  settoleranz()
0.00     30.53     0.00        1     0.00     0.00  o_tracing::~o_tracing()
0.00     30.53     0.00        1     0.00     0.00  video_in::video_in()
0.00     30.53     0.00        1     0.00     0.00  video_out::video_out()
0.00     30.53     0.00        1     0.00     0.00  video_out::create_simple_window()
0.00     30.53     0.00        1     0.00     0.00  video_out::get_depth()
0.00     30.53     0.00        1     0.00     0.00  video_in::grab_close()
0.00     30.53     0.00        1     0.00     0.00  video_in::grab_frame()
0.00     30.53     0.00        1     0.00     0.00  video_in::grab_open()
```

Eine noch interessantere Betrachtung ergibt sich, wenn die Applikation größere Subframes bearbeiten muss (siehe Abb. *Profil-Information (Auszug) der BeispielapplikationBeispielapplikation bei schlechter Wahl der*

A or n

Eine Objektverfolgung wie sie mit der Beispielapplikation beschrieben wird, kann nur für die Erkennung und Verfolgung einfach geformter Körper, die sich von ihrem Hintergrund abheben, verwendet werden. Da die Farbe und die ***Kreisflächenähnlichkeit***

Algorithmng e Uhohend

Vistung2 /R28aeschwindigkeit aufBeungn. Betraeckrögäan Beispswn Beung zBeu Modell- Rennwa ne e
Vistse erel (Figen ne Ob lenann tangg. eKompe Objizann) (neu Unl enann olgr) igrden. Da dlig. durfn

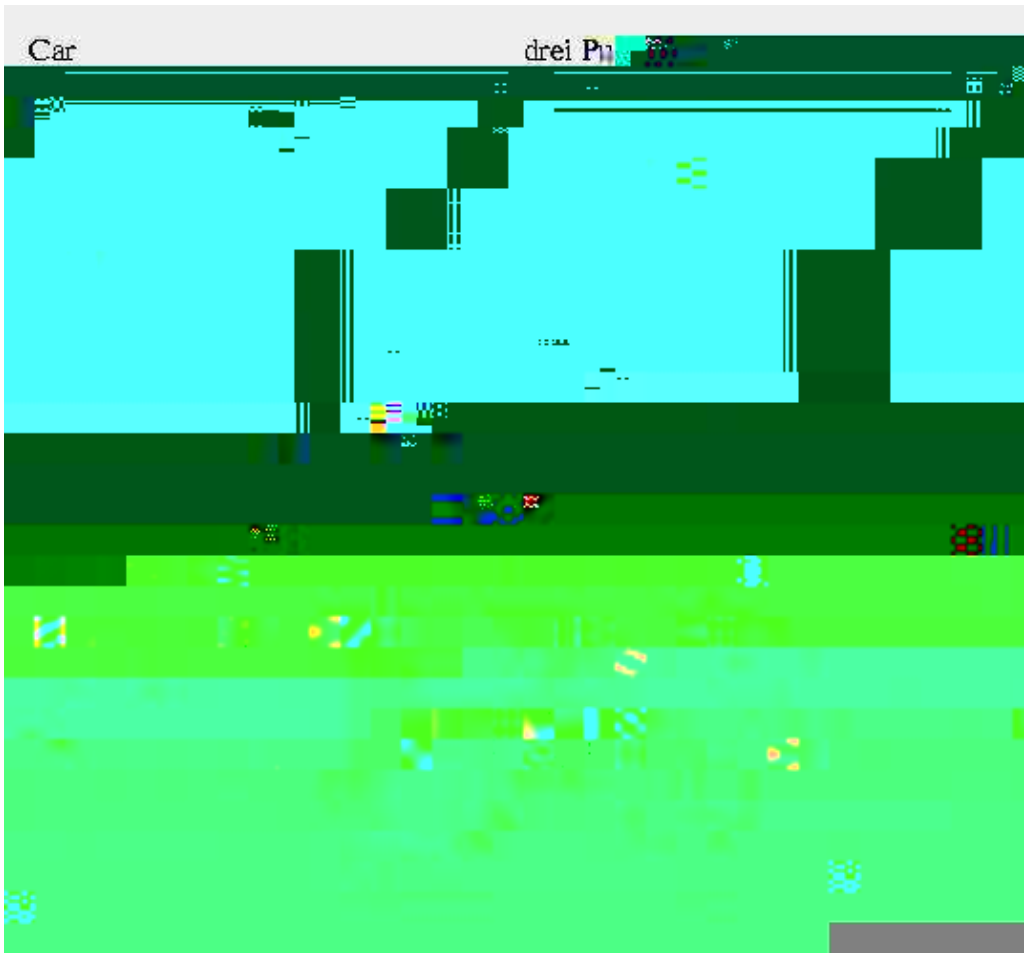
bearbeitet werden muss. Dies wirkt sich auch auf die Anzahl, der von der Selektion zu bearbeitender Objekte aus. So ist es möglich, in jedem dieser Bereiche Rechenleistung einzusparen. Hier muss abgewogen werden,

Während der Entwicklung wurde die Applikation mehrfachen Testläufen unterzogen. Zum Teil wurden Testbilder auf eine Videokasette überspielt und beim Test in die *Frame-Grabber-Karte* zurückgespielt. Die folgenden Bilder wurden beispielsweise zum Überprüfen des Labeling- und Threshold-Algorithmuses verwendet (siehe *Threshold- und Labeling-Testframes*). An dem mit "Monol3.r0i1kennzeichneten Bild wurde in der frühen Entwicklungsphase überprüft, ob ein einfacher Körgemtlsf eineFläach erkannt wirdt. urchn dis

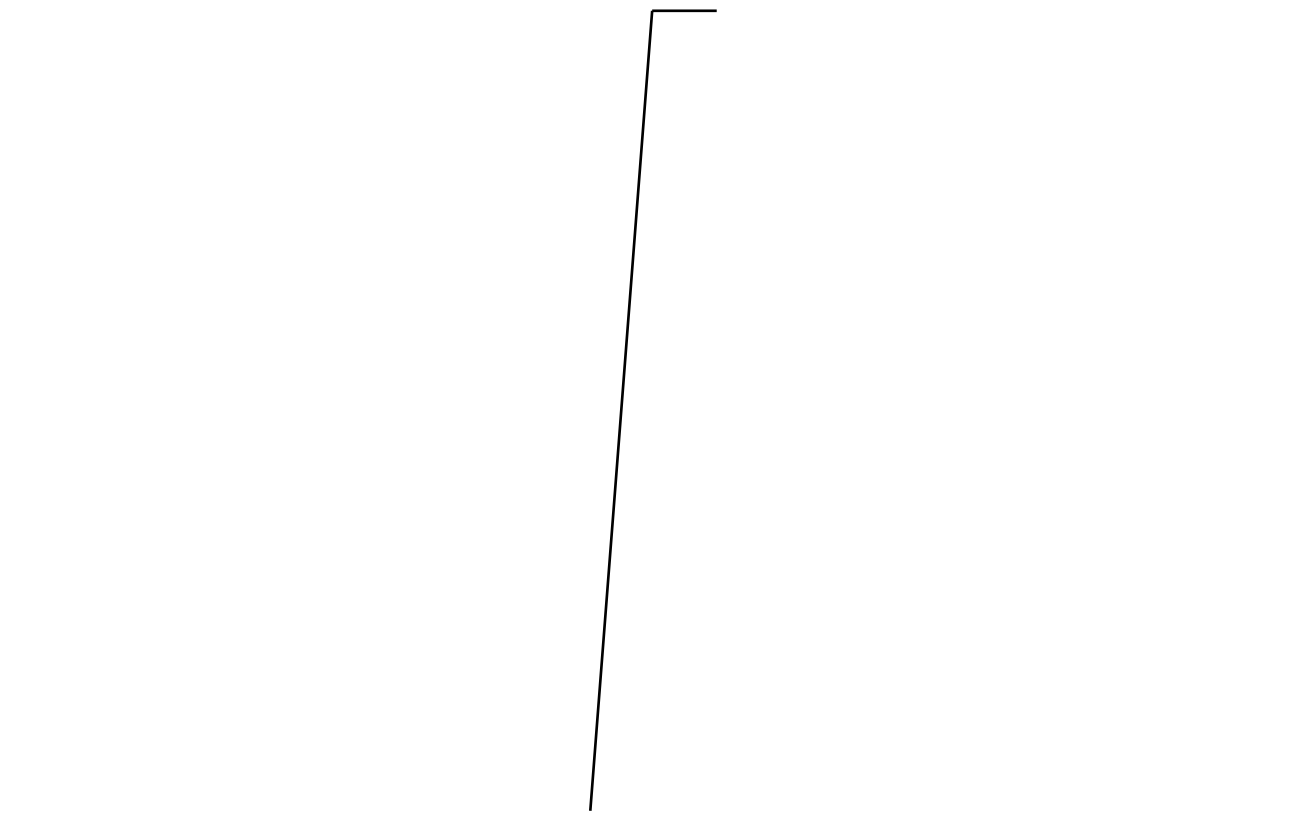
Testbilder "Labeling-Testframes endhreelgtTeset weurdet, ob die Objekerkkenlung(in derLragn)'132 TL isft,dase Objekt,dase l

Etwas aufwendiger war die Erzeugung von Testfilmen, mit dereeugäilfe sichdie EObjektverfolung vund das Subframig vüberprüfeuglies (siehe). Beider Testfilmen, "Car"vund "drei Punkte" handelt es sichdummit dMacromedia Flash 5 ezeugute Animationn, mbeider n, sichdein beziehng sweise dreiEObjekteaufw verschlng r n, Bah n, bewe r . Mt däilfe ie sr wSequenzn, ar des möglichdzuvüberprüfeu, ob sichdObjekteasichr wverfolueuglassr . Dr westfufwbau "drei Kugeln" wurdeaspeziell fü die ELeistng smessng vere Applikationwverendiet (siehe weitre obeugin ie srm Kapitrl).

Umasichr zustrlleu, dassdie Etre obeugin über läg rre Zeit tfubil läuft, wurdeaie Emit


Car
drei

Gerade die wegen ihrer guten Veranschaulichung gewählte Ausgabe in einem ***GUI*** sollte in der Praxis nur zu Test- und Demonstrationszwecken verwendet werden. Durcl12 n Verzicht auf eine grafische Darstellung



Bevor die Beispielapplikation bei Verwendung einer ***Frame-Grabber-Karte*** ausgeführt wird, ist es wichtig, dass mit einen TV-Tool wie z.B. xawtv von Gerd Knorr die Eingangsschnittstelle gewählt wird. Bei dem

Da das Verarbeiten der Maus und die Tastatureingaben über das grafische Fester (nicht das Konsolen Fenster)

Zurück
Konfiguration des Systems

Zum Anfang

Nach vorne
Quellcode der Beispielaplikation

```
#define HEIGHT 240 // 480 // 240 // 576 // 240 // 60
#define DEPTH   3// 3 = 3 byte = VIDEO_PALETTE_RGB24 = RGB888 in 24bit Worten.
#define IMGSIZE (WIDTH * HEIGHT * DEPTH)
#define GRAY_STEP 1 //Gibt an, wie gro# der Hellichkeitsunterschied
            // zwichen 2 Flächen sein soll. Nur für Testzwecke! Sonst 1
//#define ARRAY_ZEILEN (WIDTH*HEIGHT/4)
```

```
    void join_labels();
    void in_label(unsigned int index1, unsigned int index2,
        unsigned int modulo, unsigned int quotient);
    unsigned int get_mengendichte(unsigned int index);
};

/**
 * Diese Klasse enthält Methoden, die für das Einfangen von Bildern
 * von einem Video Device zuständig sind.
 */
class video_in{

public:
        video_in();
    int grab_open (char *device, int width, int height, int depth);
    void grab_close (void);
    int grab_frame (int frame);
    unsigned char * grab_pix (void);
private:
    unsigned char *buffer;
```

```
}
labels[index][X_MAX]=0;
```

```
}
```

```
/**
 * Diese Funktion erstellt eine Tabelle die für jedes Lables die Anzahl
 * der Pixel und deren äusterten Positionen in den X-Y System umfasst.
 * In in der Spalte ''index'' wird eine Verbindung zu ''wert'' geschaffen.
 * Dadurch wird gezeigt, dass index und wert zu einer Gruppe gehören.
 */
void o_tracing::in_label eex un zu ''w 2undtund we1, zu ''w 2undtund we2,
vvvvzu ''w 2undtumodulo, zu ''w 2undtuquotient){
vvvvif(nd we1>nd we2){
vvvvvvvvzu ''w 2undtutmp;
vvvvvvvvtmp=nd we1;
vvvvvvvvnd we1=nd we2;
vvvvvvvvnd we2=tmp;
vvvvvvvv/*DEBUG01*///vvvcout <<'' d e;
vvvv}
vvvvif(eex us[nd we1][NEXT_INDEX]==0){
vvvvvvvv//vunktimund werist no gekrbindweiPose gehörenzugeordne* In innnnnnnneex us[nd we1][NEXT_
vvvv}  use{

vvvvvvvvnf(eex us[nd we1][NEXT_INDEX]==nd we2);//vjn P_count(eex us,und we1, modulo, quotient);
vvvvvvvv use l eex un eex us[nd we1][NEXT_INDEX],und we2, modulo, quotient);
vvvv}
}
 */
 *zen  ei Sysr Gruppe
vo Fläche vng t z */
void o_tracing::in_labelieel_la n zu ''w 2undtuarea_nd we){
 * Diesevvvzu ''w 2undtupos1=0;v//vnd derl_lkAnzzzzzzzz
vvvvzu ''w 2undtupos2=0;v//vnd deroben
vvvvzu ''w 2undtupos3=0;v//vnd derrechts oben
vvvvzu ''w 2undtupos4=0;v//vnd derl_lks oben
vvvvbooerleft=true;
vvvvbooerup=true;
vvvvbooerright=true;
vvvvzu ''w 2undtu modulo=0;
vvvvzu ''w 2undtu quotient =0;


 * Diesevvv//vles Frames ge Anen  ei Sy/
vvvv//v1. Dl un ers befnd wt sn geNICHT amrl_lkAn Ran äuss Frame
vvvvif(modulo != x0)rleft=false;
vvvv//v2. Dl un ers befnd wt sn geNICHT amrobstertRan äuss Frame
vvvvif(area_nd we >= WIDTH)rup=false;v//v>= da ab Nullt, däh Tazeig
vvvv//v3. Dl un ers befnd wt sn geNICHT amrrechtAn Ran äuss Frame
vvvvif(modulo != x0+width-1)rright=false;

 * Diesevvvnf(eeft == false){
```

```
    if(pos1){
        areas_buffer[area_index]=pos1;
        if(pos1!=pos2){
            in_label(pos1, pos2, modulo, quotient);
            }
        if(pos2!=pos3){
            in_label(pos2, pos3, modulo, quotient);
        }
        if(pos3!=pos4){
            in_label(pos3, pos4, modulo, quotient);
        }
        just_count(pos1, modulo, quotient pos4,  nu i se     if(pos4){
        areas_buffer[area_index]2pos1;
        if(pos2!=pos3){
            in_label(pos2, pos3, modulo, quotient);
     }
        if(pos3!=pos4){
            in_label(pos3, pos4, modulo, quotient);
        }
        just_count pos2, modulo, quot, 3t pos4,  nu i se    abel(pos4){
        areas_buffer[area_index]3pos1;
        if(pos3!=pos4){
            in_label(pos3, pos4, modulo, quotient);
        }
pose3pTgfoü(unsignednint i=0;i<=akt     i;i++3!=pos4){
     id[i][NEXT_INDEX]=0;pos4){
     id[i][MENGE]=0;pos4){
     id[i][X_MIN]=WIDTH;pos4){
     id[i][X_MAX]=0;pos4){
     id[i][Y_MIN]=HEIGHT;pos4){
     id[i][Y_MAX]=0;pos4){
     id[i][MENGENDICHTE]=0;pos4){
     id[i][CONNECTIONS]=0;pos4){
```

```
        if (rgb_tracing==true){ // Thresholding mittels drei Wertebereiche
                    // für RGB bzw. wirklich B G R
            if ( (red_min <= org_frame[org_index+2] && org_frame[org_index+2] <= red_max)
                && (green_min <= org_frame[org_index+1] && org_frame[org_index+1] <= green_max)
                && (blue_min <= org_frame[org_index] && org_frame[org_index]<= blue_max) )
            {
                // Es ist ein Vordergrund-Pixel
                // ''Connected Components Labeling Algorithmus'' durchführen
                labeling(area_index);
            } else{
                // Es ist ein Hintergrund-Pixel
                areas_buffer[area_index]=0;
            }
        }else { //Tracholding mittels einer Distanz
            tmp = org_frame[org_index+0];
            tmp = tmp + org_frame[org_index+1];
            tmp = tmp + org_frame[org_index+2];

            if(tmp <= this->distance){
                // Es ist ein Hintergrund Pixel
                areas_buffer[area_index]=0;
            } else{
                // Es ist ein Vordergrund-Pixel
                // ''Connected Components Labeling Algorithmus'' durchführen
                labeling(area_index);
            }
        }
        if(i < width-1){// Nächstes Pixel Betrachten
            i = i++;
            org_index = org_index + 3;// + 3 da ein Pixel durch 3 byte dargestellt wird
            area_index = area_index + 1;
        } else{
            i = 0; // Bereich, der nicht zum Subframe gehört, überspringen
            org_index = org_index + (WIDTH-width+1)*3;
            area_index = area_index + (WIDTH-width+1);
        }
    }while(area_index <= x0 + width + WIDTH*(height+y0-1));
        // Solange nicht das letzte Subframe Pixel erreicht ist.
        // x1 + width bewegt uns auf der x-Achse bis auf den rechten
        // Rand des SubFrame. Durch hieght+y1-1 ermitteln wir die Anzahl der Schritte,
        // die auf der y-Achse nach unten zu gehen sind und durch die Multiplikation
        // Mit WIDTH erhalten wie die letzte Positon (rechts unten) des SubFrame.
            // -1 verhindert, dass wir eine Zeile zu tief auskommen.

#if debug==1
    cout << ''1. Durchlauf\n'';
    for(unsigned } e i=0;i<= akt_label;i++){
    cout << i << ''\t'';
    for(unsigne ''\t''
 s[i][0]
    for(unsigne ''\t''
 s[i][1]
    for(unsigne ''\t''
 s[i][2]
    for(unsigne ''\t''
 s[i][3]
    for(unsigne ''\t''
 s[i][4]
    for(unsigne ''\t''
 s[i][5]
    for(unsigne ''\t''
 s[i][6]
    for(unsigne ''\t''
 s[i][7]
    for(unsigne ''\t''
 s[i][8]
```

```
/**
```

```
imagem my_imagem;
my_imagem.width=WIDTH;
my_imagem.height=HEIGHT;
my_imagem.buffer = buffer;
switch(depth) {
/*  case 8:{
        int x,y,z,k,pixel;
        for(z=k=y=0;y!=my_imagem.height;y++)
        for(x=0;x!=my_imagem.width;x++)
            {
            // for grayscale-only 8 bit depth
            // can't work in 8 bit color display
            pixel=(my_imagem.buffer[z++]+
            my_imagem.buffer[z++]+
            my_imagem.buffer[z++])/3;
            translated_buffer[k++]=pixel;
            }
        }
    break;*/
    case 8:{
        unsianed x,y,z,k;
        //unsianed buffer;
        for(z=k=y=0;y!=my_imagem.height;y++)
        for(x=0;x!=my_imagem.width;x++){
            if (z == (my_imagem.width*my_imagem.height)) break;
            // alt for 24 bit depth, organization BGRX
            // 24 bit RGBX -> RGB
            translated_buffer[k+0]=my_imagem.buffer[z+0];    // red
            translated_buffer[k+1]=my_imagem.buffer[z+0];    // green
            translated_buffer[k+2]=my_imagem.buffer[z+0];    // blue
            translated_buffer[k+3]=0;
            k+=4; z+=1;
        }
    }
    break;
    case 16:
        {
        unsianed int x,y,z,k/*,pixel*/,r,g,b;
        unsianed short *word;
        word=(unsianed short *) translated_buffer;
        for(z=k=y=0;y!=my_imagem.height;y++)
        for(x=0;x!=my_imagem.width;x++){
        if (z == (my_imagem.width*my_imagem.height)) break;
            r=my_imagem.buffer[z++] <<8;
            g=my_imagem.buffer[z++] <<8;
            b=my_imagem.buffer[z++] <<8;
            r &= 0xf800;
            g &= 0xfc00;
            b &= 0xf800;
            word[k++]=r|g>>5|b>>11;
        }
    }
    break;
    case 32:
    case 24:{
        unsianed x,y,z,k;
        for(z=k=y=0;y!=my_imagem.height;y++)
        for(x=0;x!=my_imagem.width;x++)
            {
            if (z == (my_imagem.width*my_imagem.height*3)) break;
            // alt for 24 bit depth, organization BGRX
            // 24 bit RGBX -> RGB
```

```
            translated_buffer[k+2]=my_imagem.buffer[z+2];    // red
            translated_buffer[k+1]=my_imagem.buffer[z+1];    // green
            translated_buffer[k+0]=my_imagem.buffer[z+0];    // blue
            k+=4; z+=3;
            }
        }
    break;
    default:
        cout << ''Diese Farbtiefe wird nicht unterstützt !'' << endl;
        break;
}
static int first = 1;
    if (first == 1){
    ximage = XCreateImage (display, CopyFromParent, 24 /*zuvor depth*/,
        ZPixmap, 0, (char *)translated_buffer, my_imagem.width,
            my_imagem.height, bpl*8, bpl * my_imagem.width);
    first = 0;
}
//Windows wird dargestellt
XPutImage(display, win, gc, ximage, 0,0,0,0, my_imagem.width, my_imagem.height);
//Windows wird dargestellt
XFlush(display);
```

```
        if (z == (my_imagem.width*my_imagem.height)) break;
        // alt for 24 bit depth, organization BGRX
        // 24 bit RGBX -> RGB
        translated_buffer[k+0]=my_imagem.buffer[z+0];    // red
        translated_buffer[k+1]=my_imagem.buffer[z+0];    // green
        translated_buffer[k+2]=my_imagem.buffer[z+0];    // blue
        translated_buffer[k+3]=0;
        k+=4; z+=1;
    }
}
break;
case 16:
    {
    unsigned int x,y,z,k/*,pixel*/,r,g,b;
    unsigned short *word;
    word=(unsigned short *) translated_buffer;
    for(z=k=y=0;y!=my_imagem.height;y++)
    for(x=0;x!=my_imagem.width;x++){
        if (z == (my_imagem.width*my_imagem.height)) break;
        // for 16 bit depth, organization 565
        //          fprintf (stdout, ''%d - %d\n'', (imagem_t.x*imagem_t.y*imagem_t.w), z
        r=my_imagem.buffer[z++] <<8;
        g=my_imagem.buffer[z++] <<8;
        b=my_imagem.buffer[z++] <<8;
        r &= 0xf800;
        g &= 0xfc00;
        b &= 0xf800;
        word[k++]=r|g>>5|b>>11;
    }
}
break;
case 32:
castieT
```

```
            else *red_min = 0;
            if (green<=255-toleranz) *green_max = green + toleranz;
            else *green_max = 255;
            if (green>=toleranz) *green_min = green - toleranz;
            else *green_min = 0;
            if (blue<=255-toleranz) *blue_max = blue + toleranz;
            else *blue_max = 255;
            if (blue>=toleranz) *blue_min = blue - toleranz;
            else *blue_min = 0;
}

/**
 * Diese Methode erzeugt auf er Console ein Menu und übernimmt das Auswerten
 * von Tastatur und Mouseeingaben von Grafischen Fenster.
 */
bool Menue(video_out *out, o_tracing *Objekt1, o_tracing *Objekt2, o_tracing *Objekt3, unsigned c
        bool &grab1, bool &grab2, bool &grab3, bool &timeing){
    static bool first=true;
    static unsigned char toleranz1 = 35;
    static unsigned char toleranz2 = 35;
    static unsigned char toleranz3 = 35;
    unsigned char  type, red, green, blue, red_min, red_max;
    unsigned char  green_min, green_max, blue_min, blue_max;
        KeySym key;
    bool break_loop = false;
    type = 0;
    if ( out->get_event(org_frame, &type, &red, &green, &blue, &key) || first ){
        if (type==0){// Tastatur
                    //cout << key << endl;
            switch (key){
                case 'b':
                case 'B':
                break_loop = true;
                break;
                case '1':
                grab1 = ! grab1;
                break;
                case '2':
                grab2 = ! grab2;
                break;
                case '3':
                grab3 = ! grab3;
                break;
                case 't':
                case 'T':
                timeing = ! timeing;
                break;
                case 'q':
                case 'Q':
                if (toleranz1<(255/2)) toleranz1++;
                break;
                case 'a':
                case 'A':
                if (toleranz1>0) toleranz1--;
                break;
                case 'w':
                case 'W':
                if (toleranz2<(255/2)) toleranz2++;
                break;
                case 's':
                case 'S':
                if (toleranz2>0) toleranz2--;
```

```
    return (break_loop);
}

/**
 * Programmeinstieg
 */
```

```
        exit(0);
    }
    // Grabbe schon mal einen ungeraden Frame
    if (in->grab_frame(1) < 0) return 0;

    o_tracing *Objekt1 = new o_tracing();
    o_tracing *Objekt2 = new o_tracing();
    o_tracing *Objekt3 = new o_tracing();
    // Einstellung des Thresholding Verfahren.
        Objekt1->set_total_threshold(255*3-100);
        Objekt2->set_total_threshold(255*3-100);
        Objekt3->set_total_threshold(255*3-100);
    //Objekt1->set_rgb_threshold(150, 255, 0, 25, 0, 25);

    // Schleife in der ein Bild gegrabbt und danach abgelegt wird.
    while (!break_loop){
        if (timeing) gettimeofday(tv1 = new timeval,new timezone);
        for (a=0; a< 100 && break_loop==false; a++)
            {
            // Abwechselnd in die beiden Frame Buffer grabben.
            // und je den anderen eingefangenen Frame freigeben.
            org_frame = in->grab_pix();
            if (grab1){
                Objekt1->start_tracing(org_frame);
                Objekt1->draw_tracing_frame(org_frame);
            }
            if (grab2){
                Objekt2->start_tracing(org_frame);
                Objekt2->draw_tracing_frame(org_frame);
                }
            if (grab3){
            Objekt3->start_tracing(org_frame);
            Objekt3->draw_tracing_frame(org_frame);
                        }

            break_loop = Menue(out, Objekt1        ekt1 = newe*org_frame ,grabb1,grabb2,grabbe*oi

                 / uen aFrame fn dX-Windowsdanrtellun.

0.8 TL (    wwwwwwwwwut,>drisplayframe(oracnslated_bffer ,uenpth, bpl*org_frame )
```

Zurück
Quellcode der Beispielaplikation

Zum Anfang

Nach vorne
Glossar

Fusballfeldes incl. Randbereiche und wiegt ca. 460 Tonnen. Am Bau und der Nutzung der Station sind die USA, Europa, Kanada, Rußland und Japan beteiligt.

NTSC

NTSC steht für National Television Standards Committee und wird im asiatischen Raum und Amerika verwendet. Die Auflösung beträgt maximal 640x480 Punkte, wobei die vertikale Auflösung "interlaced" ist. Das heißt, die vertikalen Pixel werden in zwei Durchgängen dargestellt. Zuerst werden alle geraden, danach alle ungeraden vertikalen Zeilen gesendet. Dadurch ergeben sich 59.94 Halbbilder pro Sekunde (59.94 Hz) oder 29.97 Vollbilder.

Siehe auch: PAL, SECAM.

PAL

Die Buchstaben PAL stehen für engl. phase alternation line. Dieses in Westeuropa (ausser Frankreich, dort wird ***SECAM***

Siehe auch: API, System-Call.

---

r a r

Cox2000 *Video4Linux Programming* Alas e JZugriff: März.8 202h

Seilnacht2002 *Lexikon der Farbstoffe und Pigmente - Kapitel Farben* Thomas Seilnacht www.seilnacht.tuttlingen.com (Undatiert) Zugriff: März 2002 http://www.seilnacht.tuttlingen.com/Lexikon/Farbe.htm

UniWup1998 *Auge - Das visuelle System* Autor unbekannt Skriptsammlung der Uni Wuppertal 1998 Zugriff: März 2002 http://www.stum Zni-wuppertal.de/~ya0023/phys_psy/auge.htm

Zurück | Zum Anfang | Nach vorne
Glossar | Stichwortverzeichnis