Matlab 6.0

1. Einleitung

Matlab ist ein Programm, das es ermöglicht mit Hilfe einer High-Level-Programmiersprache mathematische Programme und Prozeduren kurz und kompakt zu erstellen. Aufgrund der Tatsache, dass von Matlab für viele spezielle Berechnungen bereits Befehle vordefiniert sind, kann der User sehr schnell, mit wenigen Codezeilen eine gewünschte Anwendung erstellen. Meist reicht es allerdings aus, eine Skriptdatei zu erstellen, welche die Befehle abarbeitet und anschließend terminiert. Mit dieser Arbeitsweise wurde auch im Rahmen unseres Projektes mit Matlab gearbeitet.

Für die „Wagner & Platzer – Anfangssequenz“ wurde ein Ausschnitt aus „Star Wars - Episode 1 – Die dunkle Bedrohung“ herangezogen und mit einem speziellen Effekt bearbeitet. Mit Hilfe der Image-Toolbox von Matlab stehen bereits umfangreiche Methoden zum Laden, Bearbeiten und Exportieren von Bildern zur Verfügung, weshalb wir auf Matlab zurückgegriffen haben, um den Effekt in kurzer Zeit auszuprogrammieren.

In den nächsten Abschnitten werden der Algorithmus und dessen Implementierung genauer beschrieben.

2. Algorithmus

Für die Bearbeitung des zugrunde liegenden Filmmaterials in Matlab musste dieses in einer Bildsequenz vorliegen. Wir verwendeten dafür eine Bitmap-Sequenz, weil dabei kein Qualitätsverlust auftritt. Der nun folgende Algorithmus lädt jeweils fünf Bilder dieser Sequenz und erstellt daraus ein neues Bild, indem für jede Pixelposition der Median der RGB-Werte der fünf Basisbilder verwendet wird. Man kann die Berechnung in folgenden Formeln angeben:

$$I_r(x,y) = \text{median}(A1_r(x,y), A2_r(x,y), A3_r(x,y), A4_r(x,y), A5_r(x,y))$$
$$I_g(x,y) = \text{median}(A1_g(x,y), A2_g(x,y), A3_g(x,y), A4_g(x,y), A5_g(x,y))$$
$$I_b(x,y) = \text{median}(A1_b(x,y), A2_b(x,y), A3_b(x,y), A4_b(x,y), A5_b(x,y))$$

Dabei sind A1, A2, A3, A4, A5 die Basisbilder und I das Ergebnisbild. x und y geben die jeweilige Pixelposition im Bild an. Die Subscripte r, g und b geben die drei Farbkanäle Rot, Grün und Blau an.

Dieses Verfahren wurde in der LU Bildverstehen verwendet, um statische Hintergründe aus Bildsequenzen zu extrahieren. In unserer Übungsaufgabe rekonstruierten wir aus einer Serie von Standbildern einer Autobahnwebcam das Bild einer leeren unbefahrenen Autobahn. Um das Problem zu vereinfachen, wurden aus den Webcam-Bildern Grauwertbilder erzeugt. Dies ermöglichte es, alle Bilder für die Generierung des Hintergrundes zu berücksichtigen. In *Abbildung 1(a)* ist ein Beispielbild von der Webcam zu sehen. In *Abbildung 1(b)* das generierte Bild.

(a) Original-Webcambild

(b) Ergebnisbild

Abbildung 1: Hintergrunddetektion

Ursprünglich war gedacht, den Algorithmus auf Kinofilme anzuwenden, um ebenfalls Hintergründe zu separieren. Da aber dafür eine statische Kameraeinstellung benötigt wird, war es zunächst überhaupt schwer eine geeignete Szene zu finden. Als wir dann bei dem Lichtschwertkampf aus „Starwars – Episode 1 – Die dunkle Bedrohung“ fündig wurden, erweiterten wir den Algorithmus, so dass er auf RGB-Bildern operieren konnte. Er schaffte es allerdings nicht, die Schauspieler komplett vom Hintergrund zu trennen. Wir testeten dabei auch mehr als fünf Bilder, wobei aber zu beachten ist, dass das Programm sehr viel Speicher benötigt, da die Bildsequenz in einem vierdimensionalen Array gespeichert und bearbeitet wird. (Die vier Dimensionen sind: die Bildnummer, der Farbkanal, die x- und die y-Position der Pixel) Dies hatte zur Folge, dass der Rechner mit maximal acht Bildern arbeiten konnte, um in annehmbarer Zeit ein Ergebnis zu erzeugen. Weil aber das Erhöhen der Bildzahl, das Ergebnis nicht wesentlich verbesserte, hätte man den Algorithmus modifizieren müssen. Dabei hätte man nicht mehr die Mediane der Bildintensitäten pro Kanal verwenden müssen, sondern eine vom User definierte Nummer der gereihten Intensitätswerte. Da man dies für jeden Farbkanal separat einstellen hätte können, wäre man flexibler geworden. Doch diese Herangehensweise hätte den Aufwand drastisch erhöht. Erstens liegt in Matlab keine Funktion vor, die dies bewerkstelligt. Zweitens hätte man, da die beiden Schauspieler unterschiedlich angezogen sind, zwei Bilder erstellen müssen, in denen jeweils nur ein Schauspieler entfernt worden ist. Diese Bilder hätten anschließend noch kombiniert werden müssen.

Da der Algorithmus aber interessante Bilder erzeugte, entschieden wir uns, eine Animation daraus zu erzeugen. Um die Rechenzeit zu vermindern, verwendeten wir fünf Bilder zur Generierung des Ergebnisses.

3. Implementierung

In diesem Kapitel werden nun die einzelnen Codezeilen der Matlab-Implementierung genauer erörtert. (Die Bilder der zu ladenden Bildsequenz haben den Basisnamen `obimaul`, was daraus resultiert, dass in dem besagten Videoausschnitt die beiden Charaktere Obi-wan Kenobi und Darth Maul zu sehen sind.)

Die äußere Schleife definiert welche Bilder gewählt und wie viele Ergebnisbilder erzeugt werden. In diesem Beispiel werden 10 Ergebnisbilder generiert. Die Laufvariable `j` durchläuft den Bereich 1 bis 10. `j` legt unter anderem fest welches das erste Basisbild ist.

Unserer Erfahrung nach ist es optimal maximal 10 Bilder in einem Durchlauf zu generieren. Ansonsten kann es vorkommen, dass Matlab abstürzt. Damit nicht immer dieselben Bilder generiert werden, muss der Bereich für `j` natürlich verändert werden.

```
for j = 1:10
```

Als erstes wird der Name des ersten Basisbildes in der Variable `picname` abgespeichert. Dabei wird die Laufvariable `j` mit dem Befehl `int2str()` in einen String konvertiert. Um einzelne Strings aneinander zu reihen, werden diese in eckigen Klammern hintereinander geschrieben. Letztlich sind einige if-Abfragen notwendig, um passend für die verschiedenen Anzahlen von Stellen von `j` den korrekten String zu erzeugen.

```
    if j < 10
        picname = ['obimaul000' int2str(j) '.bmp'];
    else
        if j < 100
            picname = ['obimaul00' int2str(j) '.bmp'];
        else
            if j < 1000
                picname = ['obimaul0' int2str(j) '.bmp'];
            else
                picname = ['obimaul' int2str(j) '.bmp'];
            end
        end
    end
```

Anschließend wird das Bild in die Variable `immatrix` mit dem Befehl `imread()` geladen. Dabei werden die Bildeinträge mit `im2double()` in das double Format konvertiert. `immatrix` ist zunächst ein dreidimensionaler Array mit den Dimensionen Farbkanal, x-Position und y-Position.

```
    immatrix = im2double(imread(picname));
```

In der nun folgenden inneren Schleife werden vier weitere Bilder geladen. Dabei wird aus der ursprünglichen Bildsequenz jeweils ein Bild ausgelassen, damit unterschiedlichere Bildinhalte in die Berechnung eingehen. Wurde also beispielsweise der Algorithmus mit `j`=1 gestartet, so gehen die Bilder 1, 3, 5, 7 und 9 in die Berechnung ein.

```
      for i = 1:4;

        if (j+2*i) < 10
            picname = ['obimaul000' int2str(j+2*i) '.bmp'];
        else
            if (j+2*i) < 100
                picname = ['obimaul00' int2str(j+2*i) '.bmp'];
            else
                if (j+2*i) < 1000
                    picname = ['obimaul0' int2str(j+2*i) '.bmp'];
                else
                    picname = ['obimaul' int2str(j+2*i) '.bmp'];
                end
            end
        end
```

Mit dem nächsten Befehl wird in die Variable `immatrix` ein weiteres Bild eingefügt. Beim ersten Schleifendurchlauf wird dabei automatisch die Dimension erhöht. Die vierte Dimension ist nun die Bildnummer.

```
        immatrix(:,:,:,i+1) = im2double(imread(picname));
    end
```

Mit dem Befehl `median(immatrix,4)` wird nun der Median über die 4. Dimension der Variable `immatrix` gebildet.

```
imean = median(immatrix,4);
```

Abschließend wird der Name des Ergebnisbildes festgelegt und die Variable `imean` mit dem Befehl `imwrite()` als Bild mit dem Namen, der `picname` gespeichert ist, exportiert.

```
picname = ['ergebnis' int2str(j) '.jpg'];
imwrite(imean, picname);

end
```

Dieser Code wird nun in einer Skriptdatei mit dem Dateityp .m gespeichert. Man kann nun in Matlab diese Datei aufrufen, in dem man den Dateinamen ohne dessen Dateiendung (Dateityp) eingibt.

4. Ergebnisse

In den nächsten 3 Abbildungen sind jeweils unter *(a)* die ersten Basisbilder für die unter *(b)* gezeigten, vom Algorithmus generierten Ergebnisbilder zu sehen. Vor allem in *Abbildung 4(b)* ist sehr gut zu erkennen, dass aufgrund der starken Bewegung der Schauspieler eine Extraktion des Hintergrundes mit dieser Methode möglich ist.

(a) Erstes Basisbild

(b) Ergebnisbild

***Abbildung 2*:** `j` = 212

(a) Erstes Basisbild

(b) Ergebnisbild

Abbildung 3: `j` = 300

(a) Erstes Basisbild

(b) Ergebnisbild

Abbildung 4: `j` = 700

5. Weitere Bearbeitungsschritte

Um die einleitende Animation fertig zu stellen, wurden noch Einblendungen von Wolkenanimationen und des „Wagner & Platzer“ Logos angefertigt. Die Beschreibung der Generierung von animierten Wolkenvideos ist in den Kapiteln „Eigene Programme: Wolkengenerator“ und „Arbeiten mit Adobe Aftereffects“ nachzulesen. Die Kombination der Wolkeneffekte mit dem in Matlab erstellten Video und die Anzeige des Wagner & Platzer – Logos ist im Kapitel „Arbeiten mit Corel Photopaint“ beschrieben.