Hochschule für Technik, Wirtschaft und Kultur Leipzig
Fakultät Informatik, Mathematik und Naturwissenschaften
Masterstudiengang Informatik

Masterprojekt

Entwicklung einer Trikoterkennung für den humanoiden Roboter NAO

B.Sc. Patrick Fleischhauer

31. März 2015

Gutachterin: Prof. Dr. rer. nat. Sibylle Schwarz

Inhaltsverzeichnis

1 Einleitung

Der NAO ist ein humanoider Roboter. Er ist der menschlichen Gestalt nachempfunden. Durch Kameras und Sensoren, z.B. für Ultraschall und Druck, werden die Sinne des Menschen nachgebildet. Wissenschaftler und Studenten nutzen ihn für die Forschung in der Robotik. In regelmäßig stattfindenden Wettbewerben werden die Fähigkeiten der Roboter in verschieden Disziplinen präsentiert. Einer dieser Wettbewerbe ist der RoboCup. Der NAO wird dort zum Roboterfußball in der RoboCup Standard Platform League verwendet. An der HTWK nimmt an diesem Wettbewerb die Forschergruppe „Nao-Team HTWK“ teil.

Bei solchen Fußballspielen ist die visuelle Erkennung der Roboter auf dem Spielfeld eine Voraussetzung für die Planung der Spielstrategie. Für die Roboter muss dabei die Teamzugehörigkeit bestimmt werden. In dieser Arbeit wird dazu ein Algorithmus entwickelt, der anhand der Trikots die Zugehörigkeit erkennt. Ein existierender Algorithmus des Nao-Teams diente hierfür als Vorlage und Vergleichsmöglichkeit. Der neu entwickelte Algorithmus konnte bei verschiedenen Testdurchläufen eine durchschnittliche Quote der korrekt erkannten Trikots von 87% erzielen.

Im nachfolgenden Kapitel werden zunächst die Anforderungen an Trikoterkennung erläutert. Zum Verständnis der Algorithmen wird nachfolgend das Grundwissen zum YCbCr-Farbmodell vermittelt, die einen zentralen Punkt in beiden Algorithmen spielen. Mit diesem Wissen wird der alte Algorithmus anschließend analysiert und dessen Schwächen betrachtet. Daraus erfolgt dann der Entwurf des neuen Algorithmus. Dazu werden die einzelnen Teile des Algorithmus ausführlich beleuchtet. Anschließend erfolgt die Evaluierung der erreichten Testergebnisse. Zuletzt wird das Ergebnis zusammengefasst und ein Ausblick auf Verbesserungen und andere Ansätze gemacht.

2 Anforderungen an die Trikoterkennung

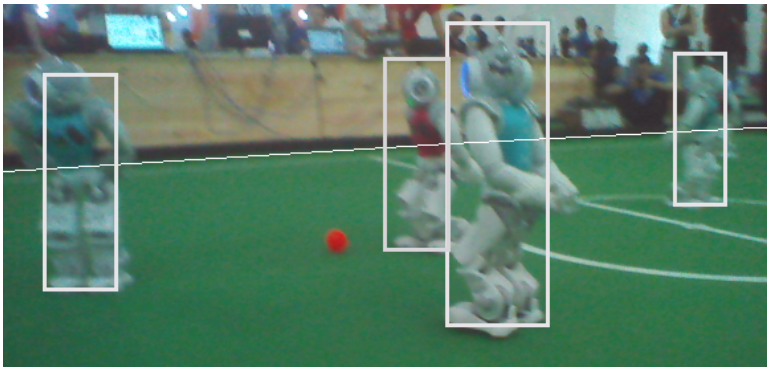

Abbildung 2.1: Detektion der Roboter und Spielfeldbegrenzung

Das NAO-Team hat bereits eine Roboterkennung realisiert. Sie bildet die Basis für die anschließende Trikoterkennung. Abbildung 2.1 veranschaulicht die Erkennung in einer Spielszene. Für jeden erkannten Roboter wird ein Bildausschnitt bestimmt, welcher diesen möglichst komplett enthält. Die Trikoterkennung ermittelt daraus die Teamzugehörigkeit der einzelnen Roboter.

Im RoboCup werden aktuell zwei verschiedene Teamfarben verwendet. Die Roboter tragen entweder ein rot- oder türkisfarbenes Trikot. Damit ist die Trikoterkennung auf eine binäre Entscheidung beschränkt. Das Ziel ist die Trikotfarben richtig zu bestimmen und eine optimale Quote zu erreichen. Dabei sollen die Lichtverhältnisse und mögliche Farbstiche berücksichtigt werden.

Hinweis: Zur besseren Unterscheidung werden die Trikotfarben in der restlichen Arbeit als rot bzw. blau bezeichnet.

Zur Bewertung der Trikoterkennung wurden 676 blaue und 331 rote Testbilder ausgewählt. Für beide Trikotfarben wird jeweils eine eigene Quote bestimmt. Der alte Algorithmus erreichte hierbei eine Quote von 673/676 (99%) und 34/331 (10%).

3 Das YCbCr-Farbmodell

Dieses Farbmodell gehört zu den sogenannten Helligkeit-Farbigkeit-Modellen. Anders als beim RGB-Farbmodell, indem die Farbinformation durch die drei Grundfarben beschrieben wird, besteht YCbCr aus der Helligkeit Y und den beiden Farbkomponenten Cb und Cr. Dabei speichern die Komponenten Cb und Cr Informationen zu den Farben Blau bzw. Rot in Abhängigkeit zur Helligkeit Y. Cb und Cr werden als Chrominanzen bezeichnet.

YCbCr wurde ursprünglich für das Digitalfernsehen entwickelt. Es findet auch bei anderen Zwecken zur Speicherung oder Übertragung von Bild- und Videodaten Anwendung. Denn der Vorteil dieses Farbmodells ist, dass die Helligkeit explizit kodiert wird. Der menschliche Sehsinn nimmt Helligkeitsunterschiede besser wahr als Farbtonunterschiede. Dadurch können die Farbkomponenten Cb und Cr ohne sichtbare Qualitätsverluste in geringerer Auflösung gespeichert werden. Entsprechend wird der benötigte Speicherplatz reduziert.

Im Rahmen dieser Arbeit wird die YCbCr mit vollem Wertebereich verwendet. In anderen Anwendungsfällen werden i.d.R. Werte für Synchronisationszwecke reserviert. Die allgemeine Umrechnung von RGB nach YCbCr erfolgt nach folgenden Formeln:

$$
\begin{aligned}
Y &= K_R \cdot R + (1 - K_R - K_B) \cdot G + K_B \cdot B \\
C_B &= 2^{n-1} + \frac{1}{2} \cdot \frac{B - Y}{1 - K_B} \\
C_R &= 2^{n-1} + \frac{1}{2} \cdot \frac{R - Y}{1 - K_R}
\end{aligned}
$$

Die Konstanten K_B und K_R beschreiben den Anteil der Rotkomponente bzw. Blaukomponente an der Helligkeit. n legt die Auflösung der Komponenten in Bits fest.

In der digitalen Bildverarbeitung wird eine Auflösung $n = 8$ verwendet, womit ein Wertebereich von 0 bis 255 zur Verfügung steht. Bei den Farbanteilen werden i.d.R. die Konstanten für den analogen Fernsehempfang (SDTV) benutzt. Diese sind $K_B = 0,114$ und $K_R = 0,299$. Somit ergeben sich folgende Formeln:

$$
\begin{aligned}
Y &= 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B \\
C_B &= 128 + \frac{B - Y}{1,772} \\
C_R &= 128 + \frac{R - Y}{1,402}
\end{aligned}
$$

Die Algorithmen zur Trikoterkennung verwenden diese Umrechnungsformeln. Durch die Komponenten Cb und Cr können einfach Farbinformationen in Abhängigkeit der Helligkeit verarbeitet werden

4 Analyse des alten Algorithmus

In diesem Kapitel wird der alte Algorithmus vorgestellt, welcher auf Basis des YCbCr-Farbmodells arbeitet. Dazu wird zuerst die Funktionsweise erläutert, um anschließend die Schwächen des Algorithmus zu analysieren.

4.1 Funktionsweise

```
1  JerseyColor Scan(Image img) {
       Image bottomHalf = img.subImage(0, img.height * 0.5, img.width, img.height * 0.5)
3
       Histogram histoY = histogram(getY(bottomHalf.pixels))
5      int thresY = thresholdByRatio(histoY, 0.7)
       int avgBrightCb = avg(getCb(filterByY(bottomHalf.pixels, thresY)))
7      int avgBrightCr = avg(getCr(filterByY(bottomHalf.pixels, thresY)))

9      Image topSection = img.subImage(0, 0, img.width, img.height * 0.7)

11     Histogram histoCb = histogram(getCb(topSection.pixels))
       int thresCb = thresholdByRatio(histoCb, 0.15)
13     int avgMaxCb = avg(getCb(filterByCb(topSection.pixels, thresCb)))

15     Histogram histoCr = histogram(getCr(topSection.pixels))
       int thresCr = thresholdByRatio(histoCr, 0.15)
17     int avgMaxCr = avg(getCr(filterByCr(topSection.pixels, thresCr)))

19     int diffCb = avgMaxCb - avgBrightCb
       int diffCr = avgMaxCr - avgBrightCr
21
       if (diffCr  - diffCb > 10)
23         return RED
       else
25         return BLUE
   }
```

Listing 4.1: Pseudocode des alten Algorithmus

Der Algorithmus trifft seine Entscheidung auf Basis eines Weißabgleichs. Er führt dazu zwei Schritte in unterschiedlichen Bildabschnitten aus.

Im ersten Schritt werden die für den Abgleich notwendigen hellsten Pixel bestimmt. Dies wird in den Zeilen 2 bis 7 beschrieben. Dazu wird nur die untere Bildhälfte verwendet, da i.d.R. der Unterkörper des Roboters dort abgebildet ist. Der Unterkörper ist weiß und gehört damit zu den hellsten Bereichen im Bild. Im Algorithmus wird zuerst in diesen Bildabschnitt eine Häufigkeitsverteilung über die Helligkeit aufgestellt. Anschließend wird ein Schwellwert für die 70% hellsten Pixel ermittelt. Für die Trikoterkennung sind jedoch die Farbkomponenten Cb und Cr relevant. Dazu wird jeweils über diese Pixel die Durchschnittswerte der beiden Komponenten bestimmt und für die Weiterverarbeitung gespeichert.

Der zweite Schritt wird in den Zeilen 9 bis 17 dargestellt. Dort wird der obere Bildteil (70%) bearbeitet. In der Regel befindet sich dort das Trikot. Der Algorithmus sucht nach Pixeln mit den höchsten Werten in den Komponenten Cb und Cr. Dazu stellt er jeweils eine Häufigkeitsverteilung auf und berechnet die Durchschnittswerte für die 15% höchsten Pixel.

In den Zeilen 19 und 20 erfolgt der Weißabgleich. Es werden jeweils die Differenzen zwischen den Ergebnissen der beiden Schritte ermittelt. Dadurch erhält man an die Lichtverhältnisse angepasste Werte. Die Entscheidung über die Trikotfarbe wird durch die Differenz der Werte und einen festgelegten Schwellwert getroffen.

4.2 Schwächen

Der Algorithmus verarbeitet ganze Bildbereiche und filtert nicht gezielt nach dem Trikot. Dadurch können Pixel, die nicht zum Trikot gehören, das Ergebnis verfälschen. Ein Problem stellt dies bei der Durchschnittsberechnung dar, wenn der Anteil der Trikotpixel gering ist.

Die weitere Schwäche ist der Weißabgleich. Der Algorithmus arbeitet auf dem YCbCr-Farbmodell. Dadurch stehen Komponenten Cb und Cr bereits in Abhängigkeit zur Helligkeit. Das stellt die Notwendigkeit eines Weißabgleichs in Frage.

Als Entscheidungskriterium spielt die Komponente Cb eine Rolle. Die relevante Trikotfarbe ist jedoch türkis, dadurch kann nicht allein Cb herangezogen werden. Der Grünanteil trägt den größten Anteil der Helligkeit bei, wodurch Cb kleiner wird.

5 Konstruktion eines neuen Algorithmus

Der neue Algorithmus soll die Schwächen des alten Algorithmus vermeiden. Dazu wird er so konstruiert, dass er gezielt nach Trikotpixeln sucht und mithilfe dieser seine Entscheidung über die Trikotfarbe trifft. Da es nur zwei Trikotfarben gibt, ist es sinnvoll den Algorithmus auf eine der Trikotfarben zu optimieren. Ist die Suche des Algorithmus erfolglos, dann wird die andere Trikotfarbe angenommen.

In diesem Fall wird nach der roten Trikotfarbe gesucht. Der Algorithmus durchläuft dabei mehrere Schritte, die in den folgenden Abschnitten einzeln erläutert werden. Der letzte Abschnitt befasst sich mit der Implementierung des Algorithmus und vereint die zuvor erläuterten Abschnitte.

5.1 Pixelfilterung nach Chrominanzen

Die Idee ist Pixel, welche nicht zum Trikot gehören, herauszufiltern. In der Regel wird der Hauptteil des Bildhintergrunds vom Spielfeld belegt. Das Spielfeld ist grün. Ebenso sind auf den Bildern große Teile des Roboterkörpers abgebildet. Dieser ist weiß oder enthält Grautöne. Deswegen ist es sinnvoll zunächst das Spielfeld und die Körperteile auszuschließen. Dazu soll nach Grüntönen gefiltert werden. Das YCbCr-Farbmodell bietet dazu keine explizite Komponente. Jedoch ist eine Ableitung aus dem Schema der Formeln in Kapitel 3 möglich:

$$C_G = 2^{n-1} + \frac{1}{2} \cdot \frac{G - Y}{1 - (1 - K_R - K_B)} = 2^{n-1} + \frac{1}{2} \cdot \frac{G - Y}{K_R + K_B}$$

mit eingesetzten Werten:

$$C_G = 128 + \frac{G - Y}{0,826}$$

Der Algorithmus soll nur die Pixel weiterverarbeiten, welche keine Grüntöne enthalten. Entsprechend muss nach $C_G < 128$ gefiltert werden.

Um die rote Trikotfarbe zu finden, werden die verbleibenden Pixel nach Rottöne gefiltert. Dazu wird die Komponente Cr verwendet. Als Filterung ergibt sich dann $C_R > 128$.

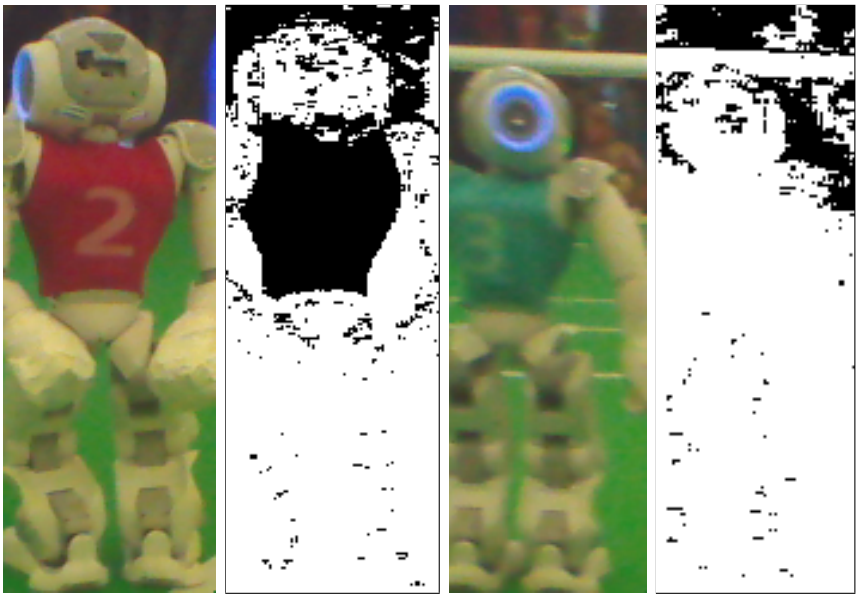

Abbildung 5.1: Beispiele der Pixelfilterung (schwarz = verbleibende Pixel)

Abbildung 5.1 zeigt, dass das Spielfeld komplett ausgefiltert wurde. Da im YCbCr-Farbmodell die Chrominanzen aus der Differenz der RGB-Komponente und der Helligkeit berechnet werden, entsteht bei weiß oder Grautönen eine Differenz von 0. Dadurch wird der Roboterkörper mit ausgefiltert. Beim Roboter mit dem türkisfarbenen Trikot wurde auch das Trikot selbst aussortiert, da sich der Grünanteil im Türkis auswirkt. Die verbleibenden Pixel sind das rote Trikot und der dunkle Bildhintergrund.

5.2 Binärisierung mit dynamischer Schwellwertberechnung

Hier wird für jedes Pixel entschieden, ob es ein Kandidat für das rote Trikot ist oder nicht. Der Schwellwert s wird dynamisch für das Bild berechnet. Dazu wird das Maximum $maxCr$ der Cr-Komponente aller Pixel ermittelt. Mit einem festgelegten Koeffizienten k wird der notwendige Anteil vom Maximum bestimmt, damit ein Pixel als Kandidat akzeptiert wird:

$$s = k \cdot maxCr,\ k \in [0, 1]$$

Durch die dynamische Berechnung werden rote Farbstiche kompensiert.

5.3 Flächenerkennung mittels Connected-component labeling

Um ein Trikot im Bild festzustellen, kann man nach zusammenhängenden Flächen suchen. Dazu existiert der Connected-component-labeling-Algorithmus. Dieser wird meist auf Binärbilder angewandt und ordnet jeder zusammenhängenden Fläche einen eindeutigen Bezeichner zu.

Eine Variante ist der „One component at a time“-Algorithmus:

```
Labelmap CCL(Bitmap img) {
    Labelmap result = create(img.size)
    int label = 1
    for each pixel in img.pixels {
        if (img.isSet(pixel) and not result.labeled(pixel)) {
            result.set(pixel, label)
            Queue q = create()
            q.add(pixel)
            while (not q.empty) {
                for each neighbour in q.next.neighbours {
                    if (img.isSet(neighbour) and not result.labeled(neighbour)) {
                        result.set(neighbour, label)
                        q.add(neighbour)
                    }
                }
            }
            label = label + 1
        }
    }
    return result
}
```

Listing 5.1: Pseudocode des „One component at a time“-Algorithmus

Das Bild wird Pixel für Pixel durchlaufen. Erreicht der Algorithmus ein gesetztes Pixel, dem noch kein Bezeichner zugeordnet wurde, so wird für das Pixel der aktuelle Bezeichner festgelegt. Anschließend werden von diesem Pixel aus die Nachbarn besucht. Sind diese gesetzt und ebenfalls ohne Bezeichner, so wird der gleiche Bezeichner vergeben und diese besuchen wiederum ihre Nachbarn. Dadurch ergibt sich eine Breitensuche, die mit Hilfe einer Warteschlange verwaltet werden. Sobald keine passenden Nachbarn mehr gefunden werden, wird ein neuer Bezeichner festgelegt und nach dem nächsten Pixel gesucht.

Die möglichen Nachbarn werden durch einen Verbindungsgrad („Pixel connectivity„) bestimmt. In der Regel werden die Varianten „4-connected“ und „8-connected“ verwendet. Beide betrachten die vertikalen und horizontalen Nachbarn. Bei „8-connected“ werden zusätzlich die diagonalen Nachbarn beachtet.

Für die Trikoterkennung ist die Bezeichnung der Flächen nicht relevant. Ziel ist es festzustellen, ob eine große zusammenhängende Fläche existiert. Deswegen soll eine Abwandlung des Algorithmus verwendet werden, welcher die Pixelanzahl der größten zusammenhängenden Fläche ermittelt:

```
1  int MaxCCL(Bitmap img) {
       int max = 0
3      int current = 0
       for each pixel in img.pixels {
5          if (img.isSet(pixel)) {
               img.unset(pixel)
7              Queue q = create()
               q.add(pixel)
9              while (not q.empty) {
                   current = current + 1
11                 for each neighbour in q.next.neighbours {
                       if (img.isSet(neighbour)) {
13                         img.unset(neighbour)
                           q.add(neighbour)
15                     }
                   }
17             }
               if (current > max) {
19                 max = current
               }
21             current = 0
           }
23     }
       return max
25 }
```

Listing 5.2: Pseudocode MaxCCL

5.4 Entscheidung über die Trikotfarbe

Dies ist der letzte Schritt in der Trikoterkennung. Die in den vorherigen Abschnitten beschriebenen Schritte wurden ausgeführt. Aus den Ergebnissen soll die Trikotfarbe bestimmt werden. Dazu soll in diesem Abschnitt die abschließende Entscheidung und Sonderfälle erläutert werden.

Nach einer Pixelfilterung und Binärisierung sollen im Idealfall bei einem blauen Bild alle Pixel ausgefiltert sein. Jedoch ist es wahrscheinlich, dass wenige Pixel übrig bleiben. Dies kann zum Beispiel eine kleine rote Fläche im Hintergrund sein. Um zu vermeiden, dass diese als rotes Trikot erkannt werden, ist es sinnvoll eine Mindestgrenze für die Pixelanzahl festzulegen. Wird diese nicht erreicht, wird das Bild als blaues Trikot gewertet.

Die abschließende Entscheidung wird durch das Ergebnis vom Algorithmus MaxCCL bestimmt. Die größte zusammenhängende Fläche muss dazu einen bestimmten Anteil der Anzahl der gefilterten Pixel enthalten. Dadurch soll verhindert werden, dass verstreute Pixel, welche die Mindestgrenze erreichen, als rotes Trikot interpretiert werden.

5.5 Implementierung

```
1  JerseyColor Scan(Image img) {
       Image topHalf = img.subImage(0, 0, img.width, img.height * 0.5)
3
       int maxCr = getMaxCr(topHalf.pixels)
5
       Bitmap filteredPixel = create(topHalf.size)
7      int hits = 0
       for each pixel in topHalf.pixels {
9          if (pixel.cg < 128 and pixel.cr > 128 and pixel.cr > 1/3 * maxCr) {
               filteredPixel.set(pixel)
11             hits = hits + 1
           }
13     }

15     if (hits < 1/9 * count(topHalf.pixels)) return BLUE

17     int max = MaxCCL(filteredPixel)

19     if (max > 1/3 * hits)
           return RED
21     else
           return BLUE
23 }
```

Listing 5.3: Pseudocode des neuen Algorithmus

Der fertige Algorithmus bearbeitet nur die obere Bildhälfte, da dort sich das Trikot in der Regel befindet. Es wird eine Pixelfilterung und anschließend eine Binärisierung durchgeführt. Dabei ist es sinnvoll, die Filterung und Binärisierung zu kombinieren. In Zeile 4 wird zuerst das für die Schwellwertberechnung notwendige maxCr berechnet. Die Zeilen 6 bis 13 führen eine Pixelfilterung durch und binärisieren die Pixel gleichzeitig. Für die Schwellwertberechnung wurde der Koeffizient $\frac{1}{3}$ verwendet.

Zeile 15 entspricht der in Abschnitt 5.4 beschriebenen Mindestgrenze. Die Anzahl der gefilterten Pixel muss mindestens $\frac{1}{9}$ der Gesamtpixelanzahl erreichen. Ist dies nicht erfüllt, wird das Bild als Trikotfarbe blau bewertet.

Anschließend wird der Algorithmus MaxCCL auf das Binärbild angewendet. Das Ergebnis ist die Pixelanzahl der größten zusammenhängenden Fläche. Befinden sich in dieser Fläche mehr als $\frac{1}{3}$ der gefilterten Pixel, so ist das Ergebnis die Trikotfarbe rot, sonst wird blau angenommen.

6 Evaluierung der Testergebnisse

Der neue Algorithmus erreichte auf den Testbildern eine Quote von 583/676 (86%) bei blau und 292/331 (88%) bei rot. Die Parameter (Mindestgrenze, Flächenanteil, etc.) wurden so gewählt, dass möglichst ein Gleichgewicht zwischen den Trikotfarben besteht. Im Gegensatz zum alten Algorithmus, der bei seiner Parametereinstellung eine starke Tendenz (99%/10%) zu blau hatte.

Eine Veränderung der Parameter bewirkt i.d.R. eine umgekehrt proportionale Veränderung zwischen den jeweiligen Quoten der Trikotfarben. Weißt der Algorithmus in einem Anwendungsfall eine Tendenz auf, dann kann über die Parameterwahl entsprechend ein Ausgleich erreicht werden.

7 Zusammenfassung und Ausblick

Der neue Algorithmus hat eine deutliche Steigerung gegenüber dem alten Algorithmus erreicht. Anstatt ganze Bildbereiche einzubeziehen, wurden die relevanten Pixel gefiltert und anschließend bewertet. Der Algorithmus ist jedoch genau für diesen Anwendungsfall konstruiert wurden. Er kann nur für die binäre Entscheidung zwischen zwei Trikotfarben verwendet werden.

Die Trikoterkennung entscheidet durch die beiden Algorithmen nur über die Farbdaten. Ergänzend wären auch z.B. Verfahren zur Objekterkennung möglich, um das Trikot durch seine Form festzustellen.

In dieser Arbeit hatte die Performance der Algorithmen wenig Relevanz. Es gibt jedoch schnellere Varianten des Connected-Component-labeling-Algorithmus als der vorgestellte Algorithmus. Hier besteht die Möglichkeit den Algorithmus weiter zu verbessern.

Literatur

Connected-component labeling: One component at a time. URL: http://en.wikipedia.org/wiki/Connected-component_labeling#One_component_at_a_time.

Studio encoding parameters of digital television for standard 4:3 and widescreen 16:9 aspect ratios. International Telecommunication Union, 2011. URL: http://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf.

YCbCr Farbmodell gegenüber dem RGB Farbmodell. URL: http://www.faqour.com/4854/ycbcr-farbmodell-gegenueber-dem-rgb-farbmodell.