Freie Universität Berlin
Institut für Informatik
SoSe 2013

Softwareprojekt Telematik:

Portierung von OpenCV auf RIOT OS am Beispiel von BackgroundSubtractorMOG

Projektbericht

Daniel Akrap <daniel.akrap@fu-berlin.de>
Lidia Krus <ruskrus@gmail.com>

Betreuerin: Dipl.-Inform. Alexandra Danilkina
<alexandra.danilkina@fu-berlin.de>

30. Juli 2013

Abstract

Das Ziel des Softwareprojektes ist die Erschließung der Möglichkeiten für die Portierung von OpenCV auf das Betriebssystem RIOT mit Bezug auf Projekt SAFEST, dessen Schwerpunkt auf der Verbesserung der Sicherheit an Flughäfen mittels Überwachung von Menschenansammlungen liegt. Die Portierung von OpenCV auf RIOT soll Bildverarbeitung in Echtzeit auf der Plattform RIOT ermöglichen und damit die Möglichkeiten im Bereich des Internet of Things erweitern. Als Beispielanwendung wird ein Programm entwickelt, das Hintergrundentfernung bei der Bildverarbeitung macht.

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument ist der Schlussbericht des Softwareprojekts „Portierung von OpenCV auf RIOT OS“. Das Projekt wurde vom 23.04.2013 bis 09.07.2013 von Daniel Akrap und Lidia Krus unter Betreuung von Dipl.-Inform. Alexandra Danilkina im Rahmen der Lehrveranstaltung Softwareprojekt Telematik am Institut für Informatik an der FU Berlin durchgeführt.

Der Projektbericht beschreibt die Aufgabenstellung, die Planung, das Umfeld sowie den Ablauf vom Projekt. Des Weiteren werden die erzielten Ergebnisse ausführlich beschrieben sowie Einsatzempfehlungen für die entwickelte Lösung gegeben. Schließlich werden der Nutzen und die Verwertbarkeit der Ergebnisse ausgeführt.

1.1 Aufgabenstellung

Der Schwerpunkt dieses Softwareprojekts liegt auf der Erschließung der Möglichkeiten zur Portierung von OpenCV auf das Betriebssystem RIOT.

Die vorliegende Arbeit hat Bezug zum Projekt SAFEST (Social-Area Framework for Early Security Triggers at Airports). Das ist ein deutsch-französisches Gemeinschaftsprojekt zur Überwachung von Menschenansammlungen, die der Verbesserung der Sicherheit an Flughäfen dient [1]. Die Daten zur Überwachung von von Menschenansammlungen werden mit Videokameras aufgenommen und sollen mit der Graphikbibliothek OpenCV (Open Source Computer Vision Library) verarbeitet werden. Diese Bibliothek enthält Algorithmen für Bildverarbeitung und Maschinelles Sehen mit Fokus auf Echtzeitfähigkeit [2].

Im Rahmen des Softwareprojekts Telematik wird ein Versuch unternommen, diese Bibliothek auf das Betriebssystem RIOT OS zu portieren. Das Mikrokernel-Betriebssystem RIOT soll unterschiedlichsten Software-Anforderungen an die Geräte im Internet of Things (IoT) entsprechen [4]. RIOT OS ist modular aufgebaut, unterstützt Multithreading und ist echtzeitfähig [3].

Die Portierung von OpenCV auf RIOT ermöglicht Bildverarbeitung in Echtzeit auf der Plattform RIOT und erweitert damit die Möglichkeiten im IoT-Bereich.

Da im Projekt SAFEST der erste Bildverarbeitungsschritt auf die Vorverarbeitung auf der Node mittels Separierung von Personen und Menschenansammlungen von dem Hintergrund zwecks Verringerung des Datenumfangs ausgerichtet ist, haben wir uns für den folgenden Anwendungsfall als Vorgabe für unsere Testanwendung entschieden:

- Das Programm bekommt eine Reihe von Bildern als Eingabe.
- Diese Bildern werden an die Funktion zur Hintergrundentfernung übergeben.
- Die Funktion wendet das MOG-Modell auf die Bilder an und speichert das letzte Ergebnis in einer Bilddatei.

Dieser Anwendungsfall setzt die Portierung der Funktionalität zur Hintergrundentfernung aus dem für das Projekt SAFEST relevanten OpenCV-Modul *video* voraus.

1.2 Stand der Technik

In unserem Projekt haben wir mit OpenCV (Version 2.4.5) und RIOT OS (Stand April-Mai 2013) gearbeitet.

Programmentwicklung unter RIOT OS erfolgt mit den Standardentwicklungswerkzeugen wie gcc, gdb in den Programmiersprachen C oder C++. Das Betriebssystem wird aktiv weiter entwickelt und wird regelmäßig aktualisiert.

OpenCV ist in C und (ab Version 2.0) in C++ implementiert.

1.3 Projektplanung

Die Arbeit an dem Projekt umfasste die Tätigkeiten zum Management, der Dokumentation und der Außendarstellung des Projekts mittels 3 Präsentationen am Anfang, in der Mitte und am Ende des Projekts. Um den erfolgreichen Ablauf zu sichern, wurden in der Anfangsphase Meilensteine definiert und ein Arbeitsplan erarbeitet, der sich jedoch während der Projektentwicklung erheblich geändert hat.
Wir haben das Projekt in vier Teilprobleme gegliedert:

1. Einrichtung der Arbeitsumgebung. Einarbeitung in RIOT und OpenCV.
2. Test auf Machbarkeit der Portierung von OpenCV.
3. Entwicklung einer Beispielanwendung für RIOT OS.
4. Erstellung der Dokumentation.

Für die Zusammenarbeit am Projekt und die Versionsverwaltung haben wir das Plattform Github benutzt. Außerdem fanden wöchentliche Treffen der Teammitglieder statt.

2 Entwicklung der Aufgabenstellung

Wir haben den Schwerpunkt unseres Projektes darauf gelegt, die potentiellen Möglichkeiten für das Zusammenspiel von OpenCV und RIOT OS zu untersuchen. Wir sind davon ausgegangen, dass die Portierung erfolgreich sein würde, da zumindest theoretische Voraussetzungen dafür gegeben waren: Einerseits, ist OpenCV eine modulare Graphik-Bibliothek mit Fokus auf C++; andererseits, unterstützt RIOT OS die Entwicklung sowohl in C, als auch in C++. Wir haben uns zum Ziel gesetzt, diese theoretische Annahme in der Praxis zu überprüfen.

Während der Einarbeitungszeit hatten wir viele Schwierigkeiten aufgrund mangelhafter Dokumentation und öfteren Veränderungen am RIOT. Viel Zeit hat der Versuch gekostet, ein einfaches C++-Programm unter RIOT zum Laufen zu bringen bzw. den Grund für Probleme damit herauszufinden. Letztendlich hat sich herausgestellt, dass die Unterstützung vieler grundlegender C++-Bibliotheken fehlt (wie *fstream*, *iostream* usw.), die z.B. mit Systemzugriff arbeiten, was für Portierbarkeit von OpenCV entscheidend wäre.

Diese Tatsache hat zur erheblichen Änderung unserer Aufgabenstellung geführt.

2.1 Herausforderungen

a) Portierungsprobleme seitens RIOT

Wie es schon oben erwähnt wurde, haben wir eine Reihe der Faktoren, die eine erfolgreiche Portierung von OpenCV auf RIOT OS verhindern, festgestellt:

- Als IST-Zustand zu Projektbeginn war die C++-Unterstützung seitens RIOT OS mangelhaft.
- Im Laufe der Entwicklungen zeitgleich zum Projekt wurde Verbesserung der Unterstützung von Nativem Port, viele Fehlerbehebungen usw. beobachtet. Außerdem wurde die Unterstützung von C++ seitens RIOT verbessert (der RIOT-Compiler hat angefangen, nicht nur nach einem main.c, sondern auch nach einem main.cpp Programm zu suchen).
- Unterstützung von der Standardbibliothek von C++ blieb aber unkomplett (z.B. wird Input/Output Library nicht eingebunden).
- Ein gewisses Problem stellten auch verschiedene Toolchain- und Bibliotheksversionen für die Zielplattformen dar.

Für eine erfolgreiche Portierung von OpenCV auf RIOT OS wäre eine kompletttere Unterstützung von C++ nötig. Diese Behauptung wird durch die folgenden Ueberlegungen erlaeutert.

Nicht unterstützte C++ Bibliotheken
algorithm; array; atomic; bitset; condition_variable; complex; forward_list; fstream; functional; future; iomanip; ios; iostream; istream; iterator; locale; map; memory; mutex; ostream; random; regex; sstream; stdexcept; streambuf; string; system_error; thread; tuple; unordered_map; unordered_set; valarray

Es gibt zahlreiche Beispiele für Abhängigkeiten zwischen den OpenCV-Modulen und den obengenannten Bibliotheken. Das verdeutlicht die Tatsache, dass eine komplettere Unterstützung von C++ ein wichtiges Kriterium für die Portierung wäre, bzw. dass unter den gegebenen Bedingungen OpenCV sich nicht auf RIOT OS portieren lässt.

Nach dieser Feststellung mussten wir uns zwischen 2 Wegen für die Weiterarbeit an dem Projekt entscheiden:
1) Entweder wird das Betriebssystem RIOT OS dazu gebracht, das erforderliche Maß an C++-Unterstützung zu erreichen;
2) Oder wir versuchen, einzelne Module oder Algorithmen zu portieren bzw. sie anzupassen. Eine solche Anpassung wäre z.B. durch das Lösen der Probleme mit Systemaufrufen möglich, indem die benutzten C++-Aufrufe durch die entsprechenden Funktionalitäten in C ersetzt werden.

Wir haben uns aufgrund des zeitlichen Projektrahmens für die zweite Alternative entschieden.

b) Portierungsprobleme seitens OpenCV

Als einen weiteren einschränkenden Faktor haben wir eine große Anzahl an OpenCV-eigenen Datentypen festgestellt, die im Modul *base* definiert und beim Importieren von OpenCV-Modulen durchaus erwünscht sind, weil sie intern für Effizienz und Abgestimmtheit sorgen; bei dem Extrahieren und Anpassen einzelner Algorithmen aber ein gewisses Problem darstellen, da ihre Äquivalente und Abhängigkeiten untereinander explizit programmiert werden müssen.

2.2 Folgeentscheidungen

Wir haben festgestellt, dass man für die Portierung von einzelnen Modulen das ganze Modul *base* portieren müsste. Das wäre aber nicht möglich, weil die Ein- und Ausgabe in C++ von RIOT nicht unterstützt wird. In dem Modul *base* sind alle Grunddatentypen von OpenCV enthalten. Das würde die Portierung weiterer Module ohne das Modul *base* erschweren.

Hätte man sich für die Portierung der Funktionalität zur Hintergrundentfernung aus dem Modul *video* entschieden, bräuchte man das ganze Modul *video*, das Frames aus Streams filtern kann und noch vieles mehr. Das wäre aber aus den gleichen Gründen nicht nutzbar.

Als Ergebnis der obenbeschriebenen Untersuchungen haben sich die technischen Anforderungen unseres Projektes geändert, und der vorgenommene Portierungsumfang war auf das Extrahieren und Anpassen des Algorithmus BackgroundSubtractorMOG reduziert. Außerdem sollte eine Hilfsdokumentation für die RIOT-Wiki erstellt werden.

3 Beispielanwendung für RIOT OS

3.1 Hintergrundentfernung: Motivation

Im Prozess des maschinellen Sehens ist Bildsegmentierung üblicherweise der erste Schritt der Bildanalyse. Unter Segmentierung versteht man die Erzeugung von ihnaltlich zusammengehörenden Regionen durch Zusammenfassung benachbarter Pixel entsprechend einem Homogenitätskritärium [6]. Ein typischer Anwendungsfall dafür ist Szenenanalyse in der Videoüberwachung, wobei Segmentierung in Echtzeit erfolgt. Eine der grundlegenden Methoden ist dabei Subtraktion des Hintergrunds. Es wurden viele Hintergrundmodelle entwickelt, die zur Lösung verschiedener Problemstellungen dienen. Ihnen ist gemeinsam, dass dabei angenommen wird, dass die Szenen bzw. Bildregionen ohne eindringende Objekte ein bestimmtes regelmäßiges Verhalten aufweisen, das mit einem statistischen Modell beschrieben werden kann [6]. Eindringende Objekte können dementsprechend durch die Erkundung der in dieses statistische Modell nicht reinpassenden Bildteile bestimmt werden.

Da in dem für uns relevanten Projekt SAFEST Hintergrundentfernung einen großen Vorteil der Reduzierung des Speicherverbrauchs bei der Verarbeitung der Daten der Videoüberwachung bringen könnte; und aufgrund der mangelhaften Unterstützung vieler C++-Bibliotheken unter RIOT, was die komplette Portierung erheblich erschwerte, - haben wir für das Herausnehmen und Anpassen des OpenCV-Algorithmus zur Entfernung des Hintergrunds entschieden. Dafür war das Kennenlernen dieses Algorithmus erforderlich.

3.2 Algorithmus zur Hintergrundentfernung

Das von uns gewählte Algorithmus BackgroundSubtractorMOG von OpenCV basiert auf dem theoretischen Modell, das in der Arbeit *An Improved Adaptive Background Mixture Model for Real-Time Tracking with Shadow Detection* von KaewTraKulPong und Bowden in [5] vorgestellt ist.

Dieses Modell nutzt den pixel-basierten Ansatz zur Hintergrundentfernung: für jedes einzelne Pixel wird eine Wahrscheinlichkeitsdichtefunktion erstellt, die zur Entscheidung herangezogen wird, ob der gegebene Pixel zum Hinter- oder zum Vordergrund gehört.

Die Implementierung von OpenCV unterscheidet sich von diesem theoretischen Modell dadurch, dass BackgroundSubtractorMOG keine Schattenerkennung macht [2].

3.3 Unsere Implementierung

Unser Quellcode umfasst 5 Dateien: *image.h, image.cpp, bgfg_gaussmix.h, bgfg_gaussmix.cpp, main.cpp* sowie Makefiles für das Testen unter Linux bzw. unter RIOT OS. Wir importieren eine geringe Anzahl an den seitens RIOT unterstützten Bibliotheken und zwar: *stdio.h, stdlib.h, float.h, math.h, jpeglib.h, list, algorithm.*

a) Implementierung nötiger Operationen mit Dateien und Bildern

In der Datei image.cpp werden die grundlegenden Funktionalitäten zum Lesen und Schreiben von .jpeg-Dateien; zum Erstellen, Belegen und Befreien der Bildmatrizen; zum Konvertieren zwischen Bildern und Bildmatrizen; sowie zum Erfassen von Bilddimensionen implementiert. Image.h liefert Schnittstelle dafür.

b) Implementierung der Funktionalität zur Hintergrundentfernung

Hier wird vor allem die Funktion *process8uC1()* aus dem BackgroundSubtractorMOG an unsere Anwendung angepasst. Sie stellt die Möglichkeit zur Hintergrundentfernung an den schwarz-weißen 8-bit Bildern bereit. Entfernung des Hintergrunds geschieht analog zu den Anweisungen im OpenCV-Algorithmus, die vorgenommenen Änderungen betreffen nur die Realisierung und nicht die Logik des Algorithmus.

c) Testfall

Die Datei main.cpp mit dem entsprechenden Makefile wurden zum Testen des Ablaufs der Hintergrundentfernung auf dem Plattform RIOT entwickelt. Entsprechend dem am Anfang postulierten Anwendungsfall werden beim Testen nach dem Dateieinlesen 6 Bildmatritzen erzeugt, die einen grauen Viereck mit einem schwarzen Streifen darstellen. Der Streifen befindet sich auf jedem Bild an einer anderen Stelle und soll als dynamischer Vordergrund interpretiert werden. Die graue Fläche stellt Hintergrund dar. Diese Bildmatritzen und sonstige nötigen Parameter wie Hintergrundmodell, Lernrate, Bilddimensionen, Varianz-Schwellenwert u.a. werden von der Funktion *process8uC1()* abgearbeiten und das Ergebnis der Hintergrundentfernung auf dem letzten Bild wird in der Datei *fg_image.jpg* gespeichert. Das Ergebnisbild ist schwarz und hat einen weißen Streifen an der Position des schwarzen Streifens aus dem letzten interpretierten Bild. Das zeugt davon, dass unser Programm die Hintergrundentfernung richtig gemacht hat: der Schwarzbereich gehört zum Hintergrund, das Objekt - der Streifen - gehört zum Vordergrund und wird weiß markiert.

4 Zusammenfassung und Ausblick

Im Laufe der Projektentwicklung haben wir folgende Ergebnisse bekommen:

- C++ wird unter RIOT (Stand: Mai 2013) nicht komplett unterstützt. Die aufgedeckten Probleme haben wir in einem Beitrag zur RIOT-Wiki beschrieben.
- Dadurch sind aktuell die Möglichkeiten für Zusammenarbeit von OpenCV und RIOT OS als ziemlich begrenzt zu bezeichnen. Die Annahme, dass die Modularität von OpenCV zur erfolgreichen Portierung führen wird, hat sich nicht bestätigt.
- Ein entscheidender Schritt in Richtung der Portierbarkeit wäre eine vollständigere Unterstützung von C++ seitens RIOT (insbesondere von Systemzugriffsbibliotheken), was durchaus möglich scheint, da RIOT OS ständig weiterentwickelt und aktualisiert wird.
- Mittlerweile ist Extraktion einzelner OpenCV-Algorithmen möglich, was allerdings ein ziemlich aufwendiger Prozess ist.
- Wir haben eine Beispielanwendung entwickelt, das Hintergrundentfernung auf schwarz-weißen *.jpg Bildern unter RIOT OS erlaubt.

Für das weitere Vorgehen sehen wir folgende Alternativen:

- Für kleinere (Test)Anwendungen könnte die von uns bereitgestellte Funktionalität zur Hintergrundentfernung benutzt werden. Nach Bedarf könnte sie um Unterstützung farbiger Bilder, anderer Bildformate oder Videosequenzen erweitert werden. Außerdem könnte sie als ein Muster für Extraktion weiterer Funktionalitäten von OpenCV benutzt werden.
- Für größere Projekte wie SAFEST wäre ein solcher Ansatz unserer Meinung nach nicht erwünscht, das er den Effizienzanforderungen wenig entspricht. Dafür wäre eine erfolgreiche Portierung von OpenCV-Modulen besser geeignet, die wir für möglich halten, wenn die Unterstützung von C++ seitens RIOT OS verbessert wird. Dafür könnte z.B. unsere Beschreibung der entdeckten Probleme im Beitrag zur RIOT-Wiki herangezogen werden.

Im Allgemeinen hat für die Erfahrung der Arbeit an einem Softwareprojekt als positiv und wertvoll erwiesen. Wir haben wichtige softwaretechnische Aspekte kennengelernt: Anforderungenermittlung und -änderungen, Arbeit im Team und Zeitmanagement (mit Gantt-Diagrammen), Projektentwurf und -vorstellung, Versionsverwaltung (mit Github) und Dokumentation. Außerdem haben wir unsere Erfahrungen in der Programmierung in ANSI C und C++ deutlich erweitern können und haben neue Programmierwerkzeuge wie z.B. CMake kennen gelernt. Wir haben Kenntnisse in Bereich der Bildverarbeitung (insbesondere Segmentierung von Vordergrund und Hintergrund) bekommen und ein neues Betriebssystem für IoT kennengelernt.

5 Anhang

5.1 Quellcode & Doku

- Unser Quellcode und die Projektdokumentation online :

  SWP 2013: Portierung von OpenCV auf RIOT OS
  (<https://github.com/swp2013riot/riot-opencv-port>)

- Unser Beitrag zu RIOT-wiki :

  Using C++ on RIOT OS
  (<https://github.com/RIOT-OS/RIOT/wiki>)

5.2 Glossar

Die folgenden Definitionen stammen, falls nicht explizit angegeben, aus den entsprechenden Wikipedia-Artikeln.

OpenCV - Open Source Computer Vision Library

RIOT - One OS to Rule Them All in the IoT

SAFEST - Social-Aria Framework for Security Triggers at Airports

Internet der Dinge (IoT, Internet of Things) - die Verknüpfung eindeutig identifizierbarer physischer Objekte (things) mit einer virtuellen Repräsentation in einer Internet-ähnlichen Struktur.

Gantt-Diagramm (Gantt chart) - Instrument des Projektmanagements, das die zeitliche Abfolge von Aktivitäten grafisch in Form von Balken auf einer Zeitachse darstellt.

Maschinelles Sehen (Bildverstehen, Computer Vision) - beschreibt im Allgemeinen die computergestützte Lösung von Aufgabenstellungen, die sich an den Fähigkeiten des menschlichen visuellen Systems orientieren

Wahrscheinlichkeitsverteilung (probability distribution) - gibt an, wie sich die Wahrscheinlichkeiten auf die möglichen Zufallsergebnisse, insbesondere die möglichen Werte einer Zufallsvariablen, verteilen. Erfasst bzw. quantifiziert den Zufall in einem stochastischen Vorgang.

Stetige Wahrscheinlichkeitsverteilung (continuous probability distribution) - erstreckt sich auf größere Bereiche. Einzelne Punkte können dabei die Wahrscheinlichkeit 0 haben.

Normalverteilung (Gauß-Verteilung, normal distribution, Gaussian distribution) - ein prototypischer Vertreter von stetigen Verteilungen. Ihre Wahrscheinlichkeitsdichte bzw. deren Graph wird auch Gauß-Funktion, Gauß-Kurve, Gauß-Glocke oder Glockenkurve genannt. Mittels Normalverteilung lassen sich viele reale Situationen näherungsweise beschreiben.

Wahrscheinlichkteisdichtefunktion (probability density function, p.d.f.) - ein Hilfsmittel zur Beschreibung einer stetigen Wahrscheinlichkeitsverteilung. Mit Hilfe dieser Funktion lässt sich die Wahrscheinlichkeit für ein beliebiges Intervall bestimmen.

Erwartungswert (expected value) - beschreibt die Zahl, die die Zufallsvariable im Mittel annimmt.

Varianz (variance) - Maß für die Abweichung einer Zufallsvariablen von ihrem Erwartungswert.

Mischverteilung (mixture distribution, mixture model) - Wahrscheinlichkeitsverteilung einer Zufallsvariablen, die sich aus der Mischung von anderen Wahrscheinlichkeitsverteilungen ergibt.

Gaußsche Mischverteilung (gaussian mixture model, GMM) - Approximiert Verteilung durch gewichtete Summe von Normalverteilungen.

Hintergrundmodell (background mixture model) - basiert auf Annahme, dass Hintergrundfarbwert eines Pixels in einer Bildfolge normal verteilt ist (kann also als eine Gaußverteilung mit Dichtefunktion p(x) modelliert werden). Damit reduziert sich das Problem der Schätzung der aktuellen Hintergrundfarbverteilung auf das Schätzen des Erwartungsvektors (geschätzter Hintergrundfarbwert) und der Kovarianzmatrix (geschätzte Schwankung des Farbwertes), die eine Gaußverteilung eindeutig charakterisieren. Die Pixel, die keinen der (mittels einer einfachen Heuristik bestimmten) höchstwahrscheinlichen Hintergrundfarbwerte annehmen, werden als Vordergrundpixel bezeichnet. Die Vordergrundpixel werden dann zu 2D Objekten zusammengesfasst. Um den Algorithmus zu initialisieren, muss zuerst ein Hintergrundbild aufgenommen werden. (Beschreibung aus [7]).

adaptives Hintergrundmodell (adaptive background mixture model) - ein Verfahren, wobei anfangs eine l̈ereS̈zene als Hintergrundbild aufgenommen wird und dieses dann immer wieder adaptiert wird. Die ständige Anpassung des Hintergrundbildes ist nötig, um Änderungen der Lichtverhältnisse oder des Hintergrunds selbst zu berücksichtigen. (Beschreibung aus [7]).

6 Quellen

1. Projekt SAFEST Homepage (<http://safest.realmv6.org/>)
2. OpenCV Homepage (<http://opencv.org>)
3. OS RIOT Homepage (<http://www.riot-os.org>)
4. Baccelli E., Hahm O., Wählisch M., Günes M., Schmidt T., RIOT: One OS to Rule Them All in the IoT. INRIA, Research Report, No. RR-8176, Dec. 2012.
(<http://hal.inria.fr/hal-00768685/>)
5. KaewTraKulPong P., Bowden R., An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection. In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01, Sept 2001.
(<http://personal.ee.surrey.ac.uk/Personal/R.Bowden/publications/avbs01/avbs01.pdf>)
6. Segmentierung (Bildverarbeitung). Wikipedia.
(<http://de.wikipedia.org/wiki/Segmentierung_(Bildverarbeitung)>)
7. Focken D. Adaptive Hintergrundmodelle zur Personenverfolgung. Studienarbeit. Universität Karlsruhe, Fakultät für Informatik, Institut für Komplexität und Logik, Jul. 2000.
(<https://cvhci.anthropomatik.kit.edu/ stiefel/studienarbeiten/dirk_sa.pdf>)