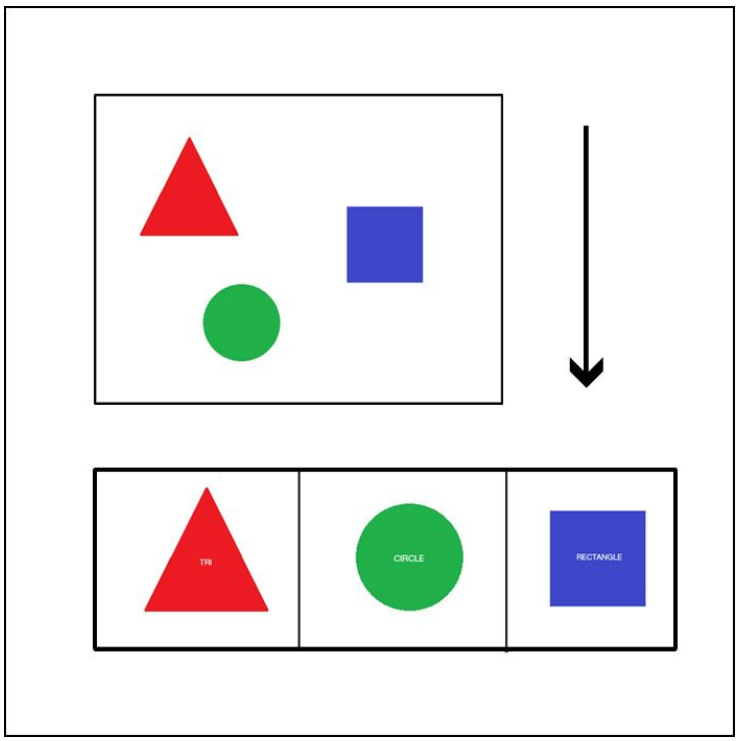
Tim Lukas Döring -2170147
Marvin Schwieger -2169489
Alexander Schiffke -

Technische Dokumentation - AVPRG WiSe 15/16

In diesem Dokument soll erläutert werden, wie Sound by Step Formen erfasst. Diese Funktion ist grundlegend für die Software. Die Formen, die vom User auf die Grundfläche gelegt wurden, bestimmen die Art des Samples, das abgespielt werden soll.

Zur allgemeinen Erfassung des Bildes verwenden wir openCV, eine freie Programmbibliothek mit Algorithmen für Bildverarbeitung. Da diese kostenlos zur Verfügung steht und Unterstützung für C++ bietet war sie unsere erste Wahl.

Wir setzen vorraus, dass das Bild erfolgreich eingelesen wurde. Über ein Objekt ColorSplitter erhalten wir drei Matrizen, die je einen unterschiedlichen Farbkanal (in unserem Beispiel Rot, Blau oder Grün) des Quellbildes enthalten. Als nächstes wird jede einzelne Matrix zur eigentlichen Form Erkennung an ein Objekt ShapeDetector übergeben. Die Methode getShapes(string, cv::Mat, int) ermittelt anhand erkannter Ecken und Winkel bis zu fünf verschiedene, von uns definierte Formen in einem Bild. Dazu wird das übergebene Farbbild zunächst in ein Graustufenbild via cv::cvtColor(Mat, Mat, int) konvertiert. Die noch nicht behandelten Konturen sollen über einen Gaußschen Weichzeichnungsfilter sauberer dargestellt werden. Dies vermeidet später fehlerhafte Erkennung der Kanten. Der in diesem resultierenden Bild abgedeckte Wertebereich von Grauwerten ist noch sehr hoch. Um diesen zu minimieren, wenden wir einen Schwellwert an und erhalten so ein Binärbild, das im Regelfall nur noch die Konturen der vorhandenen Formen enthält, da diese einen starken Kontrast zur weißen Grundfläche haben werden. Die Fläche der Formen ist an diesem Punkt nicht länger relevant und wird deshalb vernachlässigt. Den durch den Schwellwert erzeugten Kontrast zwischen den Flächen können wir nutzen um mit Hilfe des Canny-Algorithmus Kanten im Bild zu erkennen, denn der Algorithmus macht sich die Helligkeitschwankungen benachbarter Pixel zu nutze um Konturen zu erkennen. Da solche Schwankungen auch durch Rauschen verursacht werden können, benutzt der Canny dafür Normalverteilte, also gewichtete, ihn umgebende Pixelwerte. Schlussendlich erhalten wir ein binäres Kantenbild unseres Objektes.



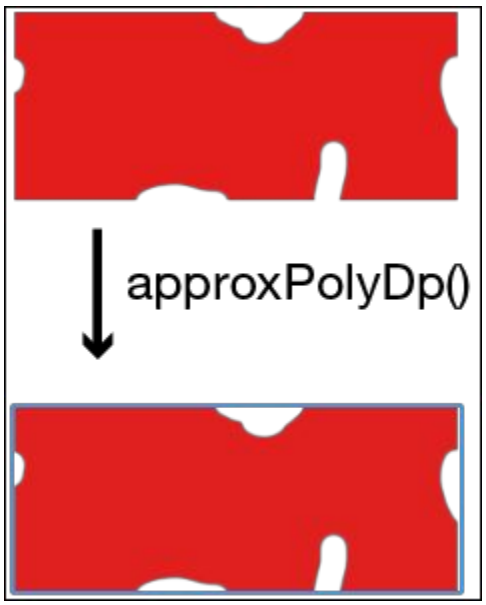
10. Januar 16
Betreuer: Prof. Dr. Andreas Plaß
Studiengang Media Systems (B.Sc.) Fach: Audio- Videoprogrammierung
Hamburg University of Applied Sciences| Department Medientechnik| Fakultät Design, Medien und Information

Tim Lukas Döring -2170147
Marvin Schwieger -2169489
Alexander Schiffke -

Verschiedene Video-Aufnahmegeräte liefern ein unterschiedlich genaues und feingranulares Bild. Im Worst-Case-Szenario enthalten die erkannten Regionen an dieser Stelle Flecken und sind deshalb nicht zusammenhängend, je nach Qualität des eingespeisten Bildes. Um diese Bildfehler zu korrigieren, wird das aktuelle Bild um einen konstanten Wert zunächst dilatiert. Dadurch werden die vorhandenen Areal-Kanten expandiert und bedecken damit die Lücken. Anschließend, um die eingeschlossene Größe der Regionen wiederherzustellen, wird eine Erosion angewendet: die hellen Bereiche im Bild (die erkannten Konturen) werden verkleinert, die dunkle Restfläche vergrößert. Angewendete Dilatation, gefolgt von einer Erosion nennt man Closing. Nach diesem Teil sind verhältnismäßig kleine Bildfehler korrigiert, bei schlechter Belichtung, Unreinheiten auf der Kameralinse, ect. können aber Abstände zwischen eigentlich zusammenhängenden Punkten auftauchen, die durch diese Operationen nicht korrigiert werden. Dieser Fehlerfall wird im ersten Release der Software nicht behandelt, da die Verwendung einer ausreichenden Kamera vorrausgesetzt wird.

Im nächsten Schritt wird die openCV Funktion findContours ansgeführt. Diese macht aus den, durch den Canny vorbereiteten, Kanten (welche wahrscheinlich noch viele einzelne Punkte enthalten) durchgehende Konturen. Dadurch ist gewährleistet, dass garantiert keine Lücken oder einzelne Punkte mehr vorhanden sind.
Durch die Liste an Konturen vector<vector<Point>> wird nun iteriert und die eigentliche Formerkennung wird angewendet.

Der Algorithmus approxPolyDP versucht nun eine Kontur, welche bis jetzt noch aus vielen, einzelnen Punkten besteht, ein Polygon reinzulegen, welches weniger Punkte hat als die eigentliche Kontur. Dies wird fortgeführt, bis später möglichst wenig Ecken und Kanten übrig bleiben. Dadurch werden zwei Dinge erreicht: Erstens, fallen pro Kontur viele Punkte weg, die auf der Kontur liegen und macht sie so ressourceneffizienter und für die folgenden Operationen einfacher zu handhaben. Zweitens, werden durch die Schätzung von Kurven Unebenheiten der unter der Kamera platzierten Form selbst korrigiert. Beschädigte Formen oder solche von schlechter Qualität werden für die Software keine mathematisch perfekten Kurven oder Linien als Konturen haben (die Software tut das in diesem Sinne ebenfalls nicht, aber sie gibt einen deutlich präziseren Näherungswert an.) Beispielsweise werden die Dellen eines Rechtecks irrelevant, da durch die Approximation der Kurven ein exaktes Rechteck entsteht (siehe auch Abbildung links). Für die Bestimmung der Formen können wir deshalb also von geraden Linien ausgehen.



Tim Lukas Döring -2170147
Marvin Schwieger -2169489
Alexander Schiffke -

Nun werden dieberechneten Polygone gefiltert. Die Software filtert folgende Formen heraus: Dreieck, Rechteck, Pentagon, Sechseck und Kreis. Zuvor wird noch kurz geprüft, ob sich unter den erkannten Polygone zu kleine oder konkave Formen befinden. Diese werden aussortiert. Bei einer Anzahl von drei Ecken können wir nun mit Sicherheit von einem Dreieck ausgehen; es ist kein weiterer Schritt erforderlich. Besitzt das Polygon zwischen vier und sechs Ecken, wird eine Fallunterscheidung abhängig von den Winkeln des Polygons durchgeführt, um alle Formen mit vier, fünf oder sechs Ecken herauszufiltern, die keine der zugelassenen Formen sind. Hat ein Polygon mehr als Sechs Ecken führt die Methode nur noch eine Prüfung durch, um festzustellen, ob es sich dabei um einen Kreis handelt. Alle anderen Formen sind nicht relevant.

Abschließend wird über den Mittelpunkt des Polygons die Position aus dem Grid ermittelt und zusammen mit den bekannten Farb- und Forminformation ein Objekt instanziiert, in einer Collection gesammelt und dem Aufrufer zurückgegeben. Dadurch können alle zugelassenen Formen eines Farbkanals pro Methodenaufruf gesammelt und in verwertbare Objekte konvertiert werden.