Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Design, Medien und Information
Department Medientechnik
Studiengang Media Systems
Wahlpflichtmodul Audio-Video
Audio-Video-Programmierung
WiSe 2014/15
Dozent: Prof. Dr. Andreas Plaß

Projektdokumentation Gesten steuern Musik „Sonic Guessture“

Autor: Robin Christopher Ladiges
Matrikelnummer: 1978865
Abgabedatum: 2015-01-25

Inhaltsverzeichnis

1. Projektidee

1.1 Exposé

Sonic (Schall, schnell, akustisch, Videospielfigur) Guessture (Kombination aus guess und gesture → mit Gesten raten) ist ein kleines akustisches Spiel. Die Spieler bekommen Ausschnitte aus (ihnen hoffentlich) bekannten Musikstücken zu hören und müssen selbstständig ohne Hinweise den Bezug zu einer thematisch zum Stück passenden Geste herstellen, welche bei erfolgreicher Ausführung das nächste musikalische Rätsel (Guessture) freischaltet.

Die Gestenerkennung soll über eine einfache Webcam mit Color Keying erfolgen. Vor dem eigentlichem Spiel erfolgt eine Kalibrierungsphase, bei welcher die Hautfarbe des Spielers, oder die Farbe seines Handschuhes, mit der Maus ausgewählt wird (Mittelwert von mehreren Pixeln).

Damit die Spieler sich dabei auch anstrengen und beeilen, werden nur kurze Stücke oder kurze Ausschnitte aus längeren Musikstücken abgespielt, welche, bei endloser Wiederholung, früher oder später anfangen werden zu nerven.

1.2 Gamification

Weil die verschiedenen Guesstures (akustische Rätsel) teilweise stark von den Vorkenntnissen des Spielers abhängen, wird es zwei Schwierigkeitsgrade geben, einen Casual- und einen Nerd-Modus.

Bei dem Casual-Modus werden nur einfache Guesstures gestellt (z.B. Beispielrätsel C) und bei dem Nerd-Modus werden Guesstures gestellt die Expertenwissen voraussetzen (z.B. Beispielrätsel D).

Nach jeden Guessture soll eine Bewertung des Spielers in Sternen angezeigt werden (z.B. von ein bis fünf Sternen), die abhängig davon ist, wie viel Zeit zur Lösung des Guesstures benötigt wurde. Im Nerd-Modus steht für Guesstures, die auch im Casual-Modus gestellt werden, weniger Zeit zur Verfügung.
Kommt ein Spieler gar nicht weiter kann er sich Tipps anzeigen lassen, die seine Bewertung um jeweils einen Stern verschlechtern.
Tipp 1: Musiktitel / Interpret / Bezeichnung
Tipp 2: Hinweis auf die Lösung.

1.3 Beispielrätsel

A) Es ist das Legend of Zelda Overworld Theme zu hören und als Geste muss grob mit den Fingern das Triforce-Symbol geformt werden.
Musik: https://youtu.be/lpEzYEoV9qY
Symbol: https://en.wikipedia.org/wiki/Triforce
Tipp 1: „Kōji Kondō – The Legend of Zelda Theme (Overworld Theme)“
Tipp 2: „Triforce“

B) Bei dem Lied We Will Rock You von Queen, muss im richtigen Moment geklatscht werden (zwei separate Handflächen verschmelzen für einen kurzen Moment zu einer Fläche).
Musik: http://vimeo.com/60898686
Tipp 1: „Queen – We Will Rock You“
Tipp 2: „If you're happy and you know it ...“

C) In Star Trek: The Original Series ist Mr. Spock bekannt für den vulkanischen Gruß, bei dem eine flache Hand zwischen Mittel- und Ringfinger gespreizt wird.
Musik: https://youtu.be/hdjL8WXjlGI
Geste: https://en.wikipedia.org/wiki/Vulcan_salute
Tipp 1: „Alexander Courage – Where No Man Has Gone Before (Raumschiff Enterprise)“
Tipp 2: „Live long and prosper / Lebe lang und in Frieden“

D) Für das Intro von Star Trek: The Next Generation muss um die Ecke gedacht werden, um auf die Facepalm-Geste von Jean-Luc Picard zu kommen.
Musik: https://youtu.be/XsxgcLf0TSY
Geste: http://knowyourmeme.com/memes/facepalm

Tipp 1: „Dennis McCarthy – Star Trek: The Next Generation Main Title“

Tipp 2: „Meme“

E) Weil bei weißem Rauschen die Rauschenergie stochastisch über alle Frequenzen verteilt ist, muss ebenso der Spieler, zum Weiterkommen die Arme so schnell bewegen, dass alle Pixel der Webcam innerhalb eines kurzen Zeitintervalls berührt werden.

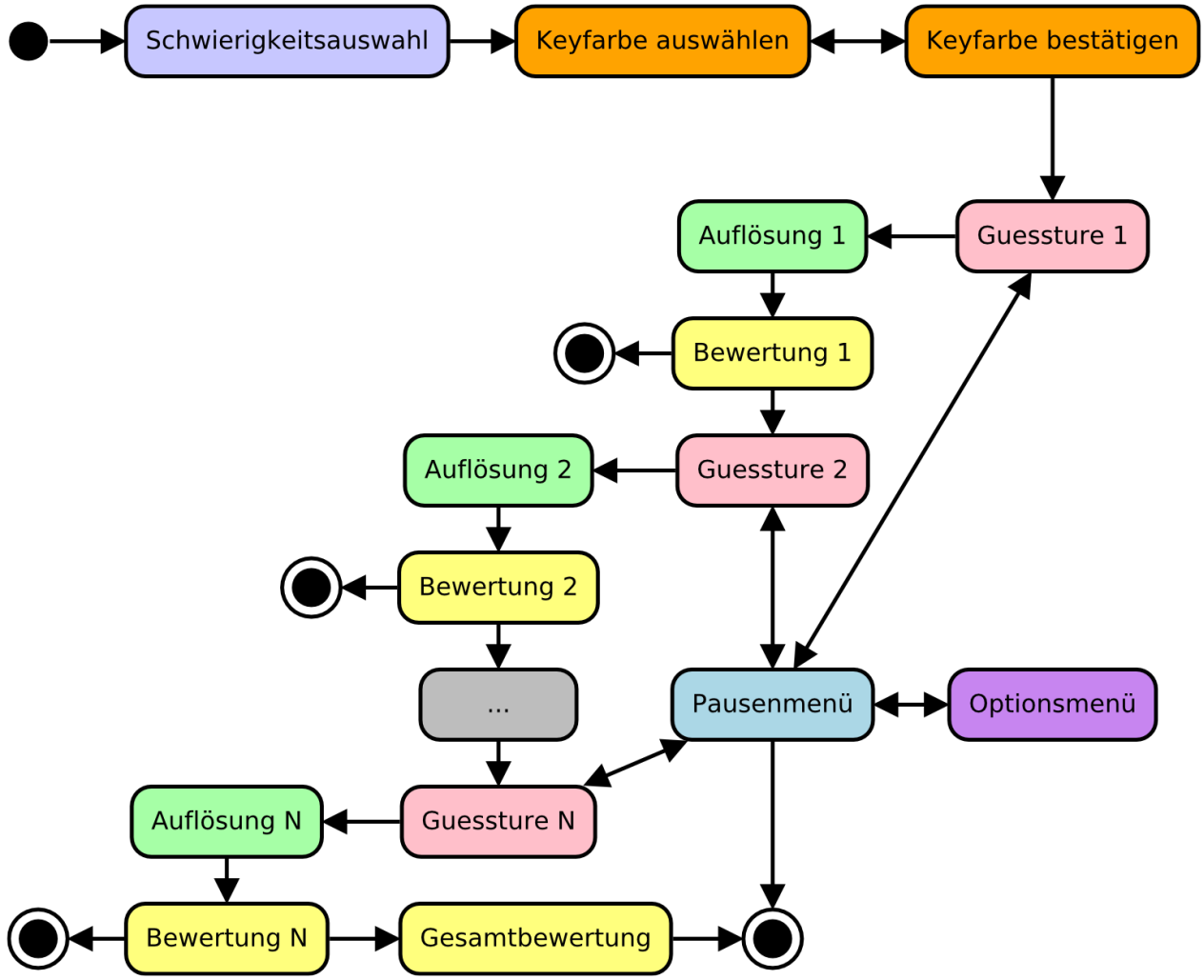
Geräusch: https://en.wikipedia.org/wiki/White_noise

Tipp 1: „Weißes Rauschen“

Tipp 2: „ALLE Frequenzen“

2. Nutzersicht

2.1 Spielablauf



Wenn der Spieler das Spiel startet, muss zunächst der Schwierigkeitsgrad ausgewählt werden.

Danach wird die Farbe für das Color Keying ausgewählt, also die eigene Hautfarbe der Hände in der Ausgabe der angeschlossenen Webcam. Zum Bestätigen der Farbe werden die markierten Pixel im Webcambild hervorgehoben dargestellt. Die Farbauswahl soll nur einmalig am Anfang geschehen, weil sonst eventuell Rätsel durch das Verändern der Farbe einfacher werden könnten.
Ein Rätsel besteht aus drei Phasen, das eigentliche Guessture, bei dem mittels Gesten das akustische Rätsel gelöst werden muss, die Auflösung des Rätsels, bei dem ein Sound für das Bestehen abgespielt wird und ein bildliche Auflösung des Rätsels angezeigt wird, und die Bewertung, bei der ausgehend von der benötigten Zeit und der Anzahl der eingesetzten Hilfen Sterne als Belohnung vergeben werden.

In der Guessture-Phase kann das Spiel pausiert werden, um das Spiel zu beenden, das Guessture zu überspringen (ohne Auflösung des Rätsels und mit der schlechtest möglichen Bewertung von Null Sternen) oder Einstellungen, wie die Lautstärke, zu ändern.

Nach der Bewertung kann der Spieler entweder das nächste Guessture spielen oder das Spiel beenden.

Sind alle Guesstures gelöst wird abschließend noch ein letzter Bildschirm mit einer Gesamtbewertung angezeigt.

2.2 Interface

Es soll in gesamten Spiel immer das Bild der Webcam angezeigt werden. Auch in Menüs, ist die Webcam der Hintergrund, auf dem die Menüelemente liegen. Der Bildschirm eines Laptops wird dadurch zu einem Spiegel, in dem der Spieler die eigenen Gesten sieht. Damit das Bild der Webcam auch wie ein Spiegel wirkt und nicht irritiert muss es vertikal gespiegelt werden, weil es ansonsten Seitenverkehrt ist.

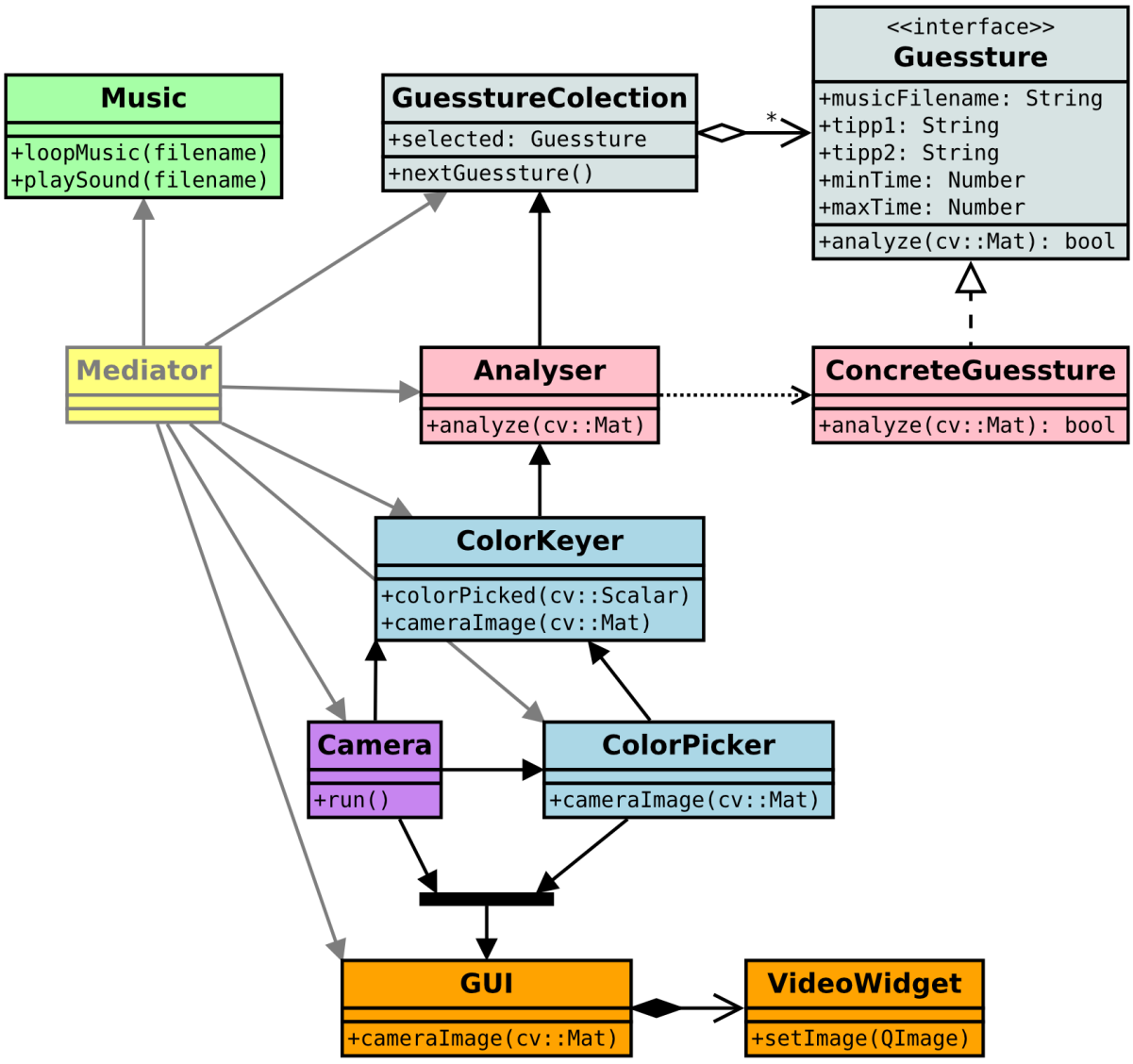
Die Auswahl der Keying-Farbe erfolgt mit der Maus, ebenso wie die Interaktion mit den Interfaceelementen. Alternativ kann aber auch die Tastatur zusätzlich verwendet werden (Bewegen der Maus oder der Auswahl mittels Pfeiltasten, Enter zum Bestätigen, Escape zum Abbrechen und Pausieren).

Es werden folgende Interfaces benötigt:

- Bestätigen der Farbauswahl
- Bewertung eines einzelnen Guesstures
- Gesamtbewertung

- Pausenmenü
- Optionsmenü
- Hauptmenü (optional)
- Hilfe (optional, beim Druck auf F1 um die Tastaturbelegung anzuzeigen)

3. Technisches Konzept



Sonic Guessture wird eine Qt 5 Anwendung. Qt GUI für das Interface und Qt Multimedia für das Abspielen von Musik (QMediaPlaylist) und Sounds (QMediaPlayer).

Die Aufnahme des Webcam-Bildes, das Color-Keying und die Bildanalyse erfolgt mit OpenCV.

Die verschiedenen Klassen (Mediator, Camera, GUI, ColorPicker, ColorKeying, Analyser, Guessture-

Collection) laufen jeweils in eigenen Threads (QThread) getrennt voneinander. Die Kommunikation zwischen den Klassen erfolgt nicht direkt mittels Methoden, sondern über Slots und Signals, was eine Implementierung des Observer-Patterns (GoF:293) darstellt und zusätzlich dafür sorgt, dass die Behandlung der Nachrichten anderer Komponenten in dem Thread der empfangenden Klasse läuft und nicht im aufrufenden Thread der Methode.

Kennen tun sich die verschiedenen Klassen nicht direkt, sondern welches Signal zu welchem Slot welcher Klasse gesendet wird, steuert der Mediator (GoF:273). Der Mediator erzeugt die anderen Klassen, erhält die wichtigsten Ablaufs-relevanten Nachrichten und befehligt dementsprechend die anderen Klassen, fügt neue Kommunikations-Verbindungen hinzu oder entfernt sie wieder wenn sie nicht mehr benötigt werden. Eventuell bekommt der Mediator noch einen Zustandsautomaten mit austauschbaren States (GoF:305), wenn er zu komplex und Umfangreich werden sollte.

Zum Beispiel wird während der Farbauswahl das Bild der Camera nicht direkt an die GUI gesendet, sondern geht erst über die ColorPicker-Klasse, wo es, wenn eine Farbe ausgewählt wurde, verändert wird. Nach der Farbauswahl wird das Bild wieder direkt ohne Umweg an die GUI gesendet aber wird dann auch noch zusätzlich zur ColorKeyer-Klasse gesendet, wo Gesten erkannt werden sollen.

Die Auswahl des nächsten Guesstures erfolgt über die GuesstureCollection-Klasse, in der alle Guesstures gespeichert werden. Von der momentan aktiven Guessture (selected) der GuesstureCollection wird vom Analyzer die analyze-Funktion aufgerufen, welche gestenspezifisch, anhand der Bitmaps vom ColorKeying, feststellt, ob die konkrete Geste konkret ausgeführt wurde. Durch das Austauschen der Guessture-Objekte ändert sich zur Laufzeit dynamisch der Algorithmus der Bildanalyse, es handelt sich um eine Anwendung des Strategy Patterns (GoF:315).

Da die meisten Klassen nur einmalig erzeugt und verwendet werden, bietet es sich an, sie als Singletons (GoF:127) zu implementieren.

4. Dokumentation

4.1 Änderungen vom ursprünglichen Konzept

Die verschiedenen Schwierigkeiten Casual und Nerd wurden nicht umgesetzt, da eine Aufteilung bei vier bis fünf implementierten Beispiel-Guesstures in einem Prototyp nicht viel Sinn macht, auch weil dieses Spiel nie mehr als ein Prototyp sein wird, weil es, aufgrund von fehlenden Musiknutzungsrechten, nicht veröffentlicht werden kann.

Die Beispiel-Guesstures B, C, E wurden implementiert, sowie ein neues Viertes.

Die Architektur wurde größtenteils so umgesetzt wie es geplant war. Die größten Probleme bzw. Änderungen gab es bei der Musik und bei der GUI, weil die Musik zwingend vom Main-Thread ausgeführt werden muss, auf dem auch die GUI läuft. Damit es beim ein und Ausfaden nicht zu Problemen mit mehreren Threads kommt wurde ein binärer QSemaphore zur Threadsynchronisation verwendet, weil bei einem QMutex, nur der sperrende Thread auch wieder entsperren darf und nicht der Subthread (QtConcurrent). Die GUI selbst besteht aus mehreren Qt Widgets, die zur Laufzeit ausgetauscht werden.

4.2 Gestenerkennung

4.2.1 weißes Rauschen

Um zu Erkennen, dass der Spieler mit seinen Händen (fast) alle Pixel berührt, hat die Beispielgeste E einen eigenen Bildspeicher. Wenn ein neues Bitmap vom ColorKeyer kommt, wird dieses mit dem aus dem Speicher verodert (cv::bitwise_or).

Dividiert man die Anzahl der weißen Pixel im Bild durch die Gesamtanzahl an Pixeln, erhält man das Verhältnis zwischen gelöster und ungelöster Pixel. Überschreitet dieses Verhältnis einen gewissen Schwellwert (90%), gilt das Guessture als gelöst.

Um zu verhindern, dass der Spieler das Guessture zufällig löst, wird eine zeitliche Begrenzung dadurch umgesetzt, dass alle n Pixel (z.B. n = 62) der lokale Bildspeicher zurückgesetzt wird (etwa alle 5 Sekunden).

4.2.2 Klatschen

Klatschen für Beispielrätsel B kann dadurch erkannt werden, dass man die größten Regionen im Bild betrachtet. Wird gerade nicht geklatscht, existieren zwei etwa gleich große größte Regionen. Verändert sich das Bild, so dass eine noch größere Region entsteht und die zweitgrößte Region ziemlich klein ist, berühren sich beide Hände – es wird geklatscht.

Für dieses Rätsel ist aber auch die zeitliche Komponente relevant. Zum einen muss verhindert werden, dass ein einmaliges zufälliges Klatschen erkannt wird – es muss also mehrmals geklatscht werden - und andererseits muss im Takt der Musik geklatscht werden.

Eine Synchronisation mit der im Hintergrund ablaufenden Musik wurde nicht vorgenommen. Dies

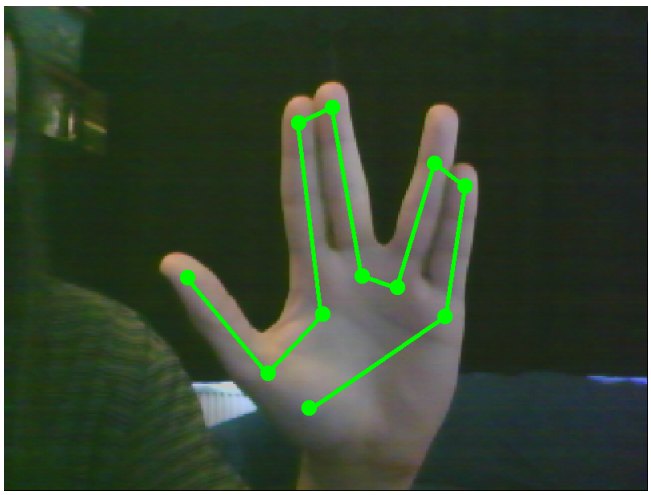
hätte sonst zu ziemlich unschönen Quellcode geführt, weil die Musik in einem eigenem Thread und einer eigenen Komponente getrennt und abgeschottet vom Analyser läuft. Stattdessen werden die zeitlichen Abstände zwischen den Klatschern grob verglichen.

Dazu wird ein boolesches Array als Gedächtnis angelegt, in dem für die letzten n Bilder (z.B. n = 20) gespeichert wird, ob geklatscht wurde oder nicht. Bei einem neuem Bild wird der älteste Wert gelöscht, die anderen Werte verschoben und der neue Wert ins Array geschrieben.

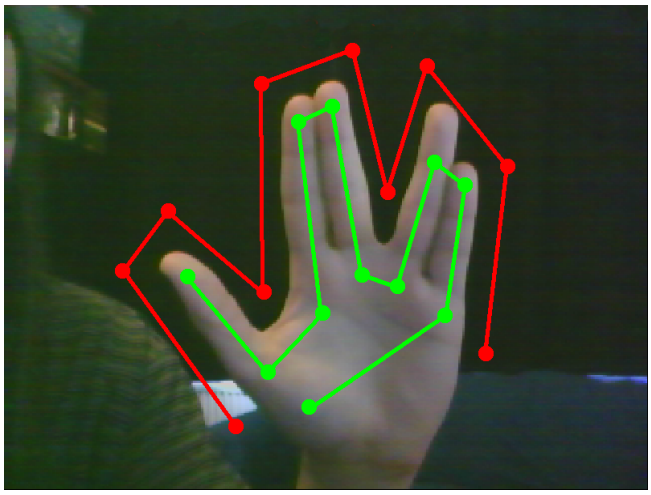
An diesem Array lässt sich nun grob der Abstand zwischen den Klatschern feststellen. Situationen die man ausschließen will sind einerseits wenn zu wenig Klatscher erfolgen (zählen der true-Werte), zu lange Klatscher, bei denen z.B. mehr als zwei true-Werte direkt aufeinanderfolgen, und ausbleibende Klatscher, bei denen z.B. auf einem Klatscher innerhalb von 4, 5 oder 6 Arraypositionen kein weiterer Klatschen folgt. Sind diese Situationen ausgeschlossen, erfolgten im beobachteten Zeitraum die gemachten Klatscher zeitlich grob im Takt relativ zueinander.

4.2.3 Handgesten

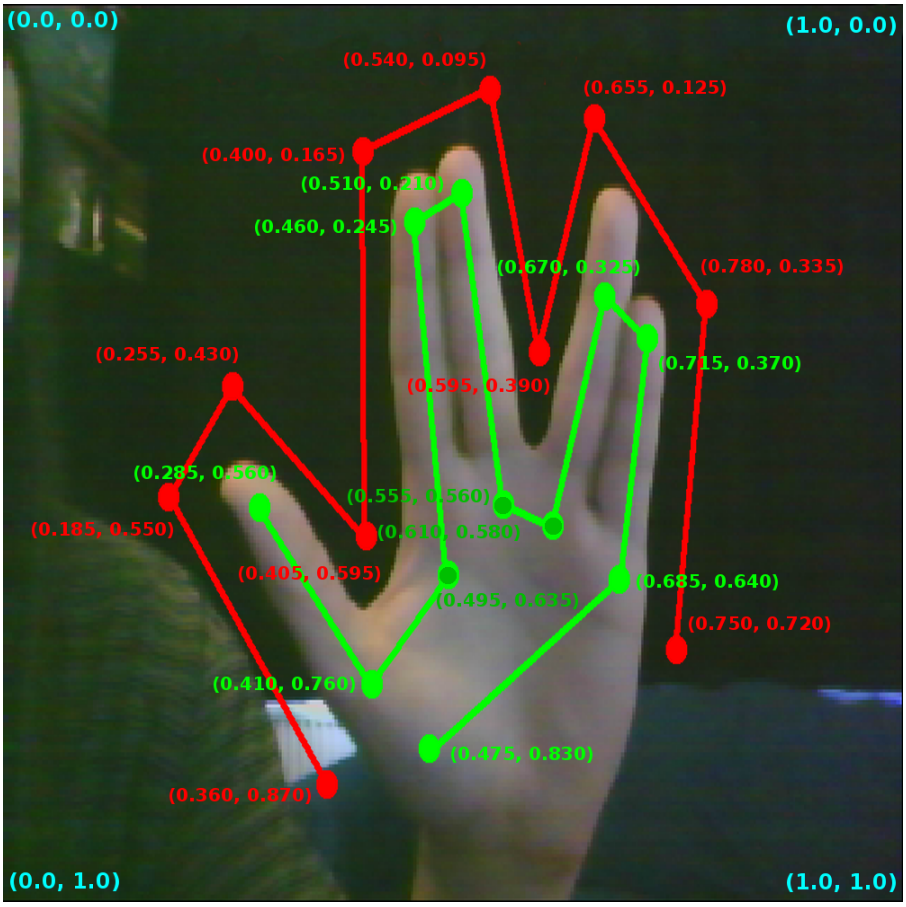
Um Handgesten, wie für die Beispielgesten A und C, im Bitmap zu erkennen, werden Punkte und Linien verwendet. Betrachtet werden alle Pixel zwischen zwei Punkten einer Linie, die Anzahl Pixel die weiß sind dividiert durch die Gesamtanzahl Punkte auf der Linie ergeben das Verhältnis, das zur Entscheidung verwendet werden kann.



Es reicht aber nicht aus nur positive Punkte zu betrachten, weil theoretisch ja auch das ganze Bitmap weiß sein könnte. Deshalb müssen auch negative Punkte betrachtet werden. Dabei ist es wichtig auf ein einigermaßen ausgeglichenes Verhältnis zwischen positiven und negativen Linien zu achten (hier: 10 zu 9), damit ein leeres Bild zu etwa 50% als richtig erkannt wird.



Um die Koordinaten der Punkte nicht als absolute Pixel anzugeben, die man ständig auf die konkreten Bildkoordinaten der Webcam umrechnen müsste, werden sie normiert zu Koordinaten zwischen 0 und 1. Der Koordinatenursprung liegt in OpenCV in der linken oberen Bildschirmecke, so wie auch in den meisten Bildbearbeitungsprogrammen, wodurch man, wenn man einen Screenshot auf 1000x1000 Pixel skaliert, die Koordinaten direkt ablesen kann.



Implementiert wird dies durch eigene Shape-Klassen. Eine für einzelne Linien, eine für Linienzüge (geschlossen oder ungeschlossen), eine Klasse, um mehrere Shapes zu einem vereinigen zu können, und letztendlich ein Complement-Shape, dass ein anderes Shape invertiert. Dabei handelt es sich um das Composite-Entwurfsmuster (GoF:163).

Alle Shapes besitzen eine Funktion test, die ein Bitmap erhält und als Ergebnis ein std::pair zweier Integer zurückgibt (abgekürzt als int2). ein Integer für die korrekten Punkte im Bild dieses Shapes und eines für die Gesamtanzahl Punkte dieses Shapes.

Bei einer einfachen Linie zwischen zwei Punkten ergeben sich die zu prüfenden Pixel einerseits durch alle ganzzahligen X-Werte zwischen den beiden x-Komponenten der Punkte und andererseits durch die Y-Werte die sich aus der Steigungsformel f(x) = m * x + b für die X-Werte berechnen lassen.