Christian-Albrechts-Universität zu Kiel

Entwurf und Implementierung einer GUI zur unterstützenden Annotation von Bildsequenzen

Bachelorarbeit

Kolja Samuel Strohm
März 2018

Betreut durch Prof. Dr.-Ing. Reinhard Koch
und M. Sc. Claudius Zelenka

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Ort, Datum Unterschrift

Zusammenfassung

In dieser Arbeit wurde eine GUI zur Annotation von Bildsequenzen entworfen und implementiert. Die fertiggestellte GUI schnitt bei einer Untersuchung im Vergleich mit einem anderen Werkzeug zur Annotation von Bildsequenzen deutlich besser ab und konnte mit Hilfe der Egebnisse sogar noch verbessert werden. Die resultierende GUI kann für das effiziente Erstellen von Trainingsdaten für das Erkennen von Objekten in Bildern verwendet werden und ist somit in vielen Gebieten anwendbar.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In den vergangenen Jahren hat sich im Forschungsgebiet der Computer Vision viel getan. Im Groben geht es dabei darum, verschiedene Aufgaben, welche bis heute noch nur von Menschen erledigt werden können, auf Computer zu übertragen. Zum Beispiel das Fahren eines Autos kann vermutlich schon bald vollständig von einem Computer übernommen werden. Um diese Aufgaben erledigen zu können, müssen die Computer lernen zu sehen und müssen dabei genau wissen, was sie sehen.

In dem Bereich des maschinellen Lernens gibt es verschiedene Ansätze, um einen Computer für solche Aufgaben zu trainieren. Zum Beispiel können Deep Convolutional Neuronal Networks für die Klassifizierung von Bildern in verschiedene Kategorien und zur Detektion von Objekten in Bildern verwendet werden [1, 2]. Bei der Detektion von Objekten in Bildern geht es darum, verschiedene Kategorien von Objekten auf Bildern zu erkennen und zu unterscheiden.

So muss ein autonomes Auto beispielsweise andere Autos, Fußgänger und Radfahrer erkennen können und darf diese dabei nicht miteinander verwechseln. Um einen Computer darauf zu trainieren, wird eine spezielle Trainingsmethode namens Supervised Learning verwendet. Dabei wird das Training wie das Lernen mit einem Lehrer gestaltet: Dem Computer wird eine Frage gestellt, woraufhin der Computer eine Antwort errechnet. Dann wird dem Computer die korrekte Antwort auf die Frage gegeben, so dass er anhand der Differenz zwischen der korrekten Antwort und der errechneten Antwort seine Parameter so anpassen kann, dass beim nächsten Mal eine bessere Antwort errechnet werden kann. Um damit ein gutes Ergebnis zu erzielen, wird ein möglichst großer Datensatz an Trainingsbildern benötigt.

Bei der Detektion von Objekten in Bildern muss für die Bilder in dem zum Training verwendeten Datensatz bekannt sein, wo sich auf welchem Bild welches Objekt befindet.

Für einige häufiger verwendete Kategorien von Objekten, gibt es bereits große Datensätze, die für das Training eines Computers verwendet werden können. Zum Beispiel enthält der Datensatz Microsoft Common Objects in Context (Microsoft COCO) bis zu 91 verschiedene Objektklassen des Alltags wie Fahrzeuge, Tiere und Essbares [3]. Weitere bekannte Datensätze sind Pascal Visual Object Classes (Pascal VOC) [4] und ImageNet [5].

Wenn es aber darum geht, Objekte aus spezielleren Kategorien zu erkennen, reichen diese Datensätze nicht aus. In diesem Fall müssen die Datensätze erst noch erstellt werden, das heißt, es muss eine große Menge an Bilder von Hand annotiert werden.

In dieser Arbeit wird eine GUI, welche die Erstellung solcher Datensätze unterstützt, entworfen und implementiert. Die Arbeit entstand im Kontext eines Projektes, bei dem Pakete in einer Sortierstation mit Hilfe von Überwachungskameras live von einem Computer verfolgt wurden. Für diesen Zweck mussten die Trainingsbilder per Hand annotiert werden, da es keinen so großen Datensatz gab, der ausreichte um alle Arten von Paketen zu erkennen. Die GUI ist daher optimiert für die Annotation von Bildsequenzen, wie sie von Überwachungskameras erzeugt werden. Sie bietet aber auch komfortable Features zur Annotation von Bildern ohne zeitlichen Zusammenhang an.

Im folgenden Abschnitt wird das Annotieren von Bildsequenzen genauer erklärt und Unterschiede zwischen Bildsequenzen und einzelnen Bildern werden genannt. In Abschnitt 4 und 5 wird die in dieser Arbeit entworfene GUI vorgestellt und es werden Kernpunkte der Implementierung erläutert. In Abschnitt 6 werden die Ergebnisse einer Untersuchung zum Vergleich der GUI mit dem oben erwähnten internen Werkzeug analysiert. In Abschnitt 7 befindet sich eine kurze Zusammenfassung der Ergebnisse.

2 Verwandte Arbeiten

Im Internet gibt es bereits viele Anwendungen, welche zur Annotation von einzelnen Bildern oder Videos verwendet werden können. Zum Beispiel LabelMe [6] erlaubt einem das Annotieren direkt im Browser vorzunehmen. Dies hat jedoch den großen Nachteil, dass die Bilder hochgeladen werden müssen, was bei sehr vielen Bildern sehr viel Zeit in Anspruch nehmen würde. LabelMe erlaubt außerdem nur das Hochladen von 20 Bildern zurzeit.

Ein weiteres Werkzeug zur Annotation von Bildern ist LabelImg[1]. LabelImg unterstützt aber nur die Annotation von Boundingboxen und nicht die von Polygonen.

Außerdem unterstützen die online verfügbaren Werkzeuge nicht die Annotation von Bildsequenzen (genauer in Abschnitt 3), sondern nur die Annotation von einzelnen Bildern. Zu diesem Zweck wurde bisher ein internes Werkzeug verwendet, welches allerdings keine GUI beinhaltet. Dieses Werkzeug muss mit Tastatur bedient werden und die Bedeutung der einzelnen Tasten muss auswendig gelernt werden.

Die in dieser Arbeit vorgestellte GUI unterstützt sowohl das Annotieren von Bildsequenzen, als auch das Annotieren einzelner Bilder. Außerdem werden die Objekt Polygone und die Boundingboxen gespeichert. Daher ist diese GUI einzigartig und bietet neue Möglichkeiten Bilder zu annotieren.

1 https://github.com/tzutalin/labelImg.git

3 Annotation von Bildsequenzen

Wie oben bereits erwähnt, ist die Annotation von Bildern ein notwendiger Prozess, um einem Computer beizubringen, Objekte automatisch zu erkennen. Je nachdem welche Aufgabe der Computer später übernehmen soll, müssen dabei verschiedene Objekte annotiert werden. Wenn der Computer zum Beispiel lernen soll, Menschen und Tiere zu erkennen, dann würde es nichts bringen Autos zu annotieren. Die verschiedenen Objektkategorien, deren Objekte annotiert werden sollen, werden Klassen genannt. In diesem Beispiel würden die Klassen ‚Mensch' und ‚Tier' ausreichen. Je nach Anwendungsfall könnte es aber auch sinnvoll sein diese Klassen noch weiter aufzuteilen.

In der GUI, welche in dieser Arbeit vorgestellt wird, im nachfolgenden Text nur noch GUI genannt, geht es jedoch nicht nur um die Annotation von Objekten auf Bildern, sondern auch um die Annotation von Bildsequenzen. Eine Bildsequenz ist eine geordnete Abfolge von Bildern, die von verschiedenen Kameras aufgenommen wurden. Dabei sind die Bilder zunächst nach der Kamera und dann nach dem Aufnahmenzeitpunkt geordnet. Diese besondere Ordnung der Bilder erleichtert den Prozess des Annotierens, da sich die Objekte in den verschiedenen Bildern einer Kamera immer nur ein Stück in eine Richtung bewegen. Es ist für den Nutzer der GUI daher leicht, dieselben Objekte auf den Bildern wiederzuerkennen. Außerdem ändern sich die Objekte nur selten in ihrer Form, weshalb die Umrisse eines Objektes oft von einem Bild zum nächsten kopiert werden können.

Beispiel: Es sind zwei Kameras A und B vorhanden, welche alle 2 Sekunden ein Bild machen. Die Objekte, die von den Kameras erfasst werden, bewegen sich zunächst durch das Sichtfeld von Kamera A und dann durch das Sichtfeld von Kamera B. Eine Bildsequenz dieser beiden Kameras enthält nun zuerst alle Bilder von Kamera A, in zeitlich sortierter Reihenfolge, und dann alle Bilder von Kamera B, welche im gleichen Zeitfenster aufgenommen wurden.

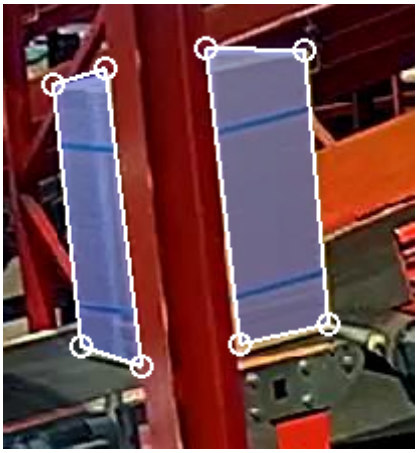


Abbildung 3.1: Mehrfaches Vorkommen desselben Objektes auf einem Bild

Der Nutzer kann nun das erste Bild von Kamera A annotieren. Dann kann der Nutzer jedes annotierte Objekt kopieren und auf dem nächsten Bild leicht verschoben oder gedreht wieder einfügen.

Bei der Annotierung von einzelnen Bildern reichen bei der Objekterkennung als Trainingsdaten die Daten aus, wo sich auf welchem Bild ein Objekt einer bestimmten Klasse befindet. Bei einer Sequenz von Bildern spielen jedoch auch andere Daten eine Rolle. Zum Beispiel ist es sinnvoll zu wissen, bei welchen Objekten auf unterschiedlichen Bildern es sich um dasselbe Objekt handelt, welches zu einem anderen Zeitpunkt oder aus einer anderen Blickrichtung aufgenommen wurde. Daher bekommt in der GUI jedes Objekt eine eindeutige ID, an welcher man das Objekt auf verschiedenen Bildern wiedererkennen kann. Jeder Objekt-ID wird dann eine Objektklasse zugeordnet. So ist sichergestellt, dass dasselbe Objekt nicht auf unterschiedlichen Bildern unterschiedlichen Klassen zugeordnet werden kann. Es kann sogar vorkommen, dass dasselbe Objekt an mehreren unterschiedlichen Stellen im gleichen Bild vorkommt. Das passiert genau dann, wenn die Sicht auf das Objekt durch einen Gegenstand versperrt wird und das Objekt nur teilweise sichtbar ist (siehe Abbildung 3.1). Außerdem gibt es bei einer Bildsequenz für eine Kamera oft Bereiche, in denen sich kein Objekt befinden kann. Solche Bereiche können in der GUI mit Masken festgelegt werden.

Um die Annotation zu speichern, können verschiedene Formate benutzt werden. Dabei ist es sinnvoll sich für ein Format zu entscheiden, welches möglichst viele andere Personen auch benutzen, da dann möglichst viele von den annotierten Bildern profitieren können. Im folgenden Abschnitt wird das von der GUI verwendete Format zum Spei-

chern von Annotationen und die unterstützten Formate für das Erstellen neuer annotierter Bildsequenzen erklärt.

3.1 Annotationsformat

Damit möglichst viele Forscher die von der GUI erstellten Annotationen verwenden können, wird zur Speicherung der Annotationen ein Format verwendet, welches sich sehr stark an dem Format des PascalVOC 2007 Datensatzes orientiert [7]. Es werden lediglich zusätzliche Informationen wie zum Beispiel Kamera und Objekt-ID gespeichert. Zu jeder annotierten Bildsequenz gehören dabei die vier Ordner ‚Annotations', ‚SourceImages', ‚SourceMasks' und ‚JPEGImages'. Im Ordner ‚Annotations' wird für jedes Bild eine XML Datei erstellt, welche alle annotierten Objekte auf dem Bild enthält. Zusätzlich werden hier auch Informationen zu dem zugehörigen Bild wie zum Beispiel Größe und Aufnahmezeitpunkt gespeichert. Der Inhalt einer XML Datei könnte für ein Bild mit nur einem Objekt zum Beispiel wie folgt aussehen:

```
<?xml version="1.0" encoding="utf-8"?>
<annotation>
    <folder>VOC2007</folder>
    <filename>0001.jpg</filename>
    <source>
        <database>The VOC2007 Database</database>
        <annotation>PASCAL VOC 2007</annotation>
        <image>0001.jpg</image>
    </source>
    <camera_id>Scan E 01</camera_id>
    <size>
        <width>1920</width>
        <height>1080</height>
        <depth>32</depth>
    </size>
    <segmented>0</segmented>
    <timestamp>2017-02-07 21:07:47.275</timestamp>
    <object>
        <name>packingbox</name>
        <id>1</id>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>100</xmin>
            <ymin>100</ymin>
            <xmax>150</xmax>
            <ymax>150</ymax>
        </bndbox>
        <polygon>
            <point>
                <x>100</x>
                <y>100</y>
            </point>
            <point>
                <x>150</x>
                <y>100</y>
```

```
38              </point>
39              <point>
40                  <x>150</x>
41                  <y>100</y>
42              </point>
43          </polygon>
44      </object>
45  </annotation>
```

Dabei wird der Name der Kamera in dem camera_id Tag (im Beispiel Zeile 10) angegeben. Der Zeitpunkt, zudem das Bild aufgenommen wurde, wird durch timestamp (im Beispiel Zeile 17) angegeben. Die Objekt-ID, an der die Objekte auf unterschiedlichen Bildern wiedererkannt werden, wird im id Tag (im Beispiel Zeile 20) angegeben. Der Objektumriss wird im polygon Tag (im Beispiel Zeile 30-43) als Liste von Punkten angegeben, welche in Pixelkoordinaten angegeben werden. Falls das Objekt mehrmals auf dem Bild zu sehen ist, werden die Polygone als Liste hintereinander abgespeichert.

Im Ordner ‚SourceImages' werden die originalen Bilder der Sequenz gespeichert. Sie werden von der GUI benötigt, da der Nutzer auf diesen Bildern die Objekte annotiert.

Im Ordner ‚SourceMasks' wird für jede Kamera eine Maske gespeichert. Eine Maske ist ein schwarz-weiß Bild mit gleicher Größe wie das zugehörige originale Bild. Weiße Flächen in der Maske bedeuten, dass innerhalb dieser Fläche im originalen Bild ein Objekt vorkommen darf. Schwarze Flächen bedeuten, dass sich an diesen Stellen im originalen Bild kein Objekt befinden kann.

(a) Source Image (b) Source Mask (c) JPEG Image

Abbildung 3.2: Beispiel für eine Kameramaske

Im Ordner ‚JPEGImages' werden die Bilder gespeichert, welche man erhält, wenn man die Masken auf die originalen Bilder anwendet. Dabei werden die schwarzen Flächen der Maske auf das originale Bild übertragen. Die Bilder werden von der GUI generiert, sobald die Maske einer Kamera durch den Nutzer verändert wird. Die Bilder in diesem Ordner

werden später für das Training verwendet, da hier keine Objekte außerhalb der Maske mehr vorkommen.

3.2 Format neuer Annotationen

Wie oben bereits erwähnt, entstand diese GUI im Kontext einer Arbeit, bei der es darum ging ein neuronales Netzwerk darauf zu trainieren, automatisch Pakete durch Überwachungskameras zu verfolgen. Die GUI hat daher eine Funktion, die es dem Nutzer erlaubt eine Bildsequenz, wie sie von den Überwachungskameras erzeugt wird zu importieren. Damit die GUI eine solche Sequenz öffnen kann, müssen die Bilder in folgendem Format vorliegen:

1. Alle Bilder befinden sich in einem Ordner
2. Die Bilder sind aufsteigend nummeriert und im JPG Format gespeichert.
3. Es existiert eine CSV Datei mit Informationen über Aufnahmezeitpunkt und verwendeter Kamera für jedes Bild im Ordner.

Falls die Bilder im obigen Format vorliegen, kann die GUI daraus automatisch eine annotierte Bildsequenz im Format aus Abschnitt 3.1 generieren. Natürlich soll die GUI auch für andere Zwecke verwendet werden können. Sobald in dem Ordner keine CSV Datei gefunden wird, wird der Ordner von der GUI rekursiv nach Bildern durchsucht. Dabei werden die Namen der Unterordner als Kameranamen und das Datum der Dateierstellung als Aufnahmezeitpunkt übernommen.

4 GUI Funktionen

In diesem Abschnitt der Arbeit werden die Hauptfunktionen der GUI vorgestellt und erklärt. Beim Design der GUI wurde besonders darauf geachtet, dass die GUI übersichtlich und leicht zu benutzen ist. Daher können verschiedene Funktionen sowohl mit der Tastatur, als auch über Buttons aktiviert werden, je nachdem wie es dem Nutzer am besten gefällt.

Die GUI wurde zudem so entworfen, dass die Annotation möglichst automatisch erfolgen kann. Der Nutzer muss die automatischen Annotationen dann nur noch leicht anpassen. Das ist möglich, da im Prozess des maschinellen Lernens das verwendete unfertig trainierte System recht schnell schon Vorhersagen darüber treffen kann, wo sich auf einem neuen Bild vermutlich Objekte befinden. Diese Vorhersagen werden in dieser Arbeit als Vorabannotationen bezeichnet. Für diesen Zweck enthält die GUI einen Client, der über ein Netzwerk (z.B. Internet) nach Vorabannotationen fragen kann. Dabei wird das Bild an einen Server gesendet, welcher mit dem lernenden System kommuniziert. Der Server antwortet schließlich mit einem segmentierten Bild, auf dem alle Pixel mit der gleichen Farbe zu einem erkannten Objekt gehören. Aus diesem Bild extrahiert der Client dann die Eckpunkte der Objekte. Auf diese Weise muss der Nutzer der GUI nur noch die erkannten Objekte anpassen. Dafür benötigt die GUI also nicht nur Funktionen zur Erstellung von Annotationen, sondern auch Funktionen zur nachträglichen Veränderung von Annotationen.

Um eine leicht benutzbare GUI zu entwerfen, müssen die verschiedenen grafischen Oberflächen gut strukturiert und durchdacht sein. Wenn es zum Beispiel zu viele verschiedene Knöpfe oder zu viele verschiedene Oberflächen gibt, dann verliert der Nutzer sehr schnell den Überblick. Wenn es aber zu wenig verschiedene Oberflächen gibt, dann ist die Steuerung oft zu kompliziert, da alle Funktionen auf einer Oberfläche angeboten werden müssen. Die im Rahmen dieser Arbeit entworfene GUI bietet für folgende Funktionen eigene Oberflächen:

(i) Das Erstellen und Verändern von annotierten Objekten (Hauptoberfläche 4.1)
(ii) Das Erstellen von Masken für die Kameras (Maskenoberfläche 4.6)
(iii) Das Zuweisen von Objekt-IDs für dieselben Objekte auf unterschiedlichen Bildern (Objekt-Zuweisungsoberfläche 4.7)
(iv) Das Verwalten von verschiedenen Objektklassen, welche in einer Sequenz annotiert werden sollen (Klassenoberfläche 4.8)

In den folgenden Abschnitten wird auf das Design dieser Oberflächen näher eingegangen und es werden ihre Funktionalitäten erläutert.

4.1 Hauptoberfläche

Dies ist die Oberfläche, die der Nutzer beim Starten der Anwendung zu Gesicht bekommt. Ihre volle Funktionalität erhält die Oberfläche jedoch erst, sobald der Nutzer eine Sequenz zum Annotieren geladen hat. Im oberen Bereich befindet sich dafür eine Leiste mit Befehlen, wie zum Beispiel das Laden und Speichern einer Sequenz. Auch einige Anzeigeoptionen können über diese Leiste gesteuert werden. Die Leiste wurde dort platziert, weil es bei vielen anderen GUIs genauso gemacht wird (zum Beispiel bei den OpenOffice Anwendungen). Der Nutzer ist es also schon durch andere Anwendungen gewohnt in diesem Bereich eine solche Leiste vorzufinden. Die Oberfläche darunter ist in drei weitere Oberflächen aufgeteilt:

(i) Eine Arbeitsfläche 4.2
(ii) Einen Navigationsbaum 4.3
(iii) Ein Hilfstext 4.4
(iv) Eine Werkzeugpalette 4.5

Die Arbeitsfläche nimmt den größten Teil der Hauptoberfläche ein, da hier die Bilder angezeigt werden, welche annotiert werden sollen. Dabei sollen auch große originale Bilder noch gut erkennbar sein. Links oben befindet sich die Werkzeugpalette, damit der Nutzer immer sehen kann welches Werkzeug momentan aktiviert ist und welche Werkzeuge er zur Verfügung hat. Links unten befindet sich der Navigationsbaum, mit dem der Nutzer eine Übersicht über alle Bilder und Kameras der geöffneten Sequenz erhält. Rechts unten befindet sich ein Hilfstext, der dem Nutzer eine Hilfestellung zum ausgewählten Werkzeug gibt. Ein Bild hiervon befindet sich in Abbildung 4.1.

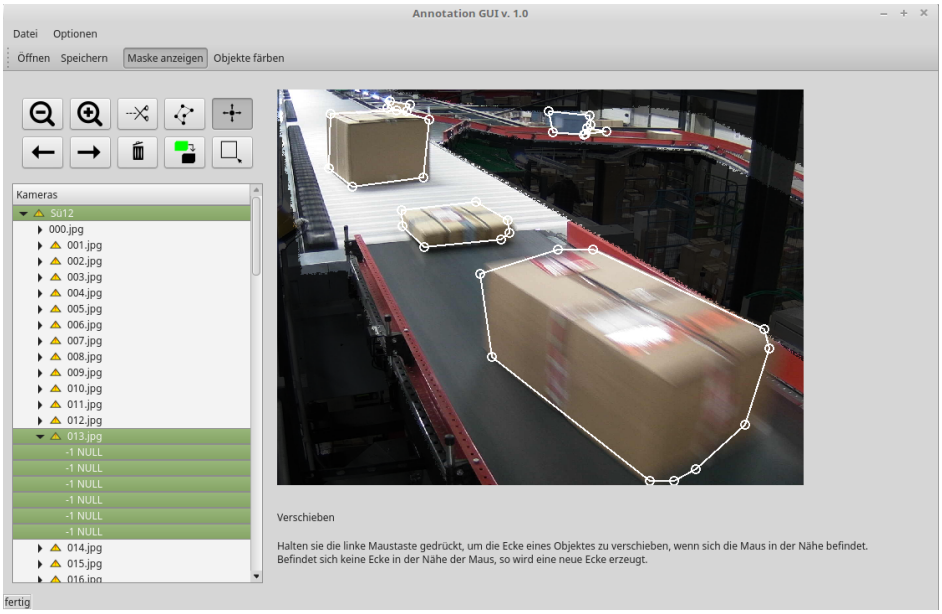


Abbildung 4.1: Die Hauptansicht der GUI

4.2 Arbeitsfläche

Die Arbeitsfläche ist die Fläche, mit der der Nutzer am meisten interagieren muss, da hier viele Annotationsprozesse stattfinden. Je nach dem welches Werkzeug aus der Werkzeugpalette ausgewählt ist, kann der Nutzer hier neue Objekte erstellen, einzelne Eckpunkte verschieben oder löschen oder ganze Objekte kopieren. Damit der Nutzer das korrekt tun kann, wird hier im Hintergrund das Bild aus der Sequenz angezeigt, welches gerade annotiert wird. Auf dem Bild werden die Umrisse und Eckpunkte annotierter Objekte angezeigt. Dabei ist es möglich die Eckpunkte einzelner Objekte auf dem Bild auszublenden, um einen besseren Überblick über die anderen Objekte zu erhalten.

Die Interaktion mit der Arbeitsfläche findet überwiegend mit der Maus statt. Dabei wird je nach ausgewähltem Werkzeug in grün eine Vorschau angezeigt, damit der Nutzer sehen kann, was er ändern würde. Auf diese Weise können viele Fehler, die beim Annotieren auftreten könnten, verhindert werden, wodurch viel Zeit erspart werden kann. In Abbildung 4.2 ist die Arbeitsfläche mit der Vorschau für das Einfügen eines Objektes zu sehen.

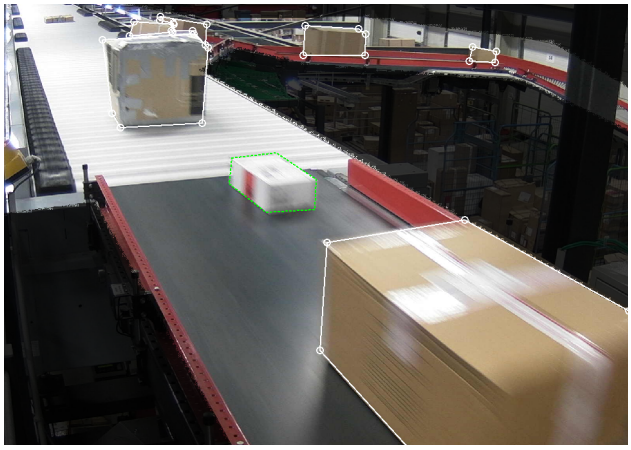


Abbildung 4.2: Die Arbeitsfläche der GUI

4.3 Navigationsbaum

Der Navigationsbaum dient dazu, dem Nutzer einen Überblick über die gesamte Sequenz zu verschaffen, die er momentan annotiert. Dafür wird die Sequenz als aufklappbarer Baum angezeigt. Dadurch weiß der Nutzer, welche Bilder zu welcher Kamera gehöhren. Durch einen Klick auf ein Bild kann der Nutzer das Bild in der Arbeitsfläche anzeigen lassen und es anschließend annotieren. Das momentan ausgewählte Bild ist in dem Baum markiert und die Objekte auf dem Bild werden in dem Baum angezeigt.

Des Weiteren zeigt der Objektbaum dem Nutzer an, welche Bilder noch nicht fertig annotiert wurden, und welche Bilder noch nie betrachtet worden sind. Bilder, die noch nicht betrachtet wurden, werden mit dunkelgrauem Hintergrund angezeigt. Bilder, die noch nicht fertig annotiert sind bekommen auf der linken Seite ein Warndreieck. Dabei wird ein Bild von der GUI genau dann als unfertig annotiert betrachtet, wenn noch keine Objekt-IDs vergeben wurden, oder wenn die gleiche ID ungewollt mehreren Objekten zugewiesen wurde. In Abbildung 4.3 ist ein Beispiel mit noch nicht annotierten Bildern und unfertig annotierten Bildern zu sehen.

Mit Hilfe der rechten Maustaste kann der Nutzer auf dem Navigationsbaum ein Menü erscheinen lassen. Dieses Menü zeigt dem Nutzer, je nach dem auf welchem Objekt der Klick erfolgt ist, spezifische Operationen. Beim Klicken auf eine Kamera kann der Nutzer die Maskenoberfläche 4.6 aufrufen und so die Maske der Kamera verändern. Beim Klicken auf ein Bild kann der Nutzer manuell Vorabannotationen vom Annotationsserver abfragen.

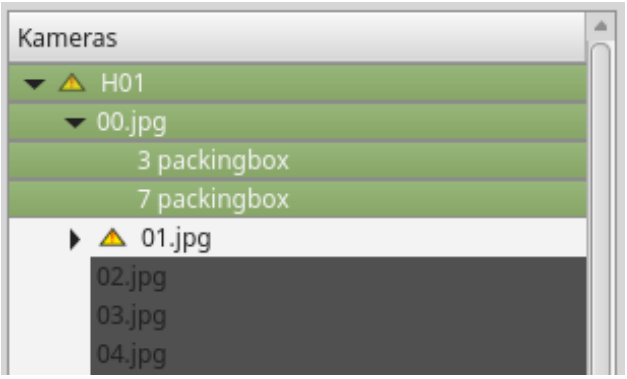


Abbildung 4.3: Der Navigationsbaum der GUI

Beim Klicken auf ein Objekt kann der Nutzer die Objekt-ID und Objektklasse festlegen oder das Objekt löschen.

4.4 Hilfstext

Unter der Arbeitsfläche wird in der GUI immer ein Hilfstext eingeblendet, der den Nutzer darüber informiert, auf welche Weise er die Arbeitsfläche mit dem ausgewählten Werkzeug bedienen kann. Dies erleichtert die Bedienung der GUI für unerfahrene Nutzer. Auch kann dadurch Zeit eingespart werden, da die unerfahrenen Nutzer einfach nur lesen müssen und nicht erst einen erfahrenen Nutzer fragen müssen.

4.5 Werkzeugpalette

Die Werkzeugpalette dient dem Nutzer zur Auswahl der verschiedenen Werkzeuge. Dabei bietet jedes Werkzeug dem Nutzer eine andere Möglichkeit die Annotation einer Bildsequenz zu bearbeiten order zu erstellen. Das momentan ausgewählte Werkzeug wird dabei mit grauem Hintergrund markiert. Jedes Werkzeug hat ein Icon, mit dem es dargestellt wird. Außerdem hat jedes Werkzeug einen Namen, der erscheint, falls der Nutzer über einen längeren Zeitraum mit der Maus darüber verweilt. In Abbildung 4.4 ist ein Bild der Wergzeugpalette. Im Folgenden werden die verschiedenen Werkzeuge kurz vorgestellt.

Abbildung 4.4: Werkzeuge der GUI

4.5.1 Vergrößern

Dieses Werkzeug ermöglicht es dem Nutzer das momentan angezeigte Bild in der Arbeitsansicht zu vergrößern, so dass bestimmte Details besser erkennbar werden. Dadurch können die Objekte dann viel genauer annotiert werden. Der Nutzer kann sich den Bereich, den er vergrößert sehen möchte, dabei selbst aussuchen, indem er diesen Bereich in der Arbeitsfläche mit der linken Maustaste markiert.

4.5.2 Verkleinern

Dies ist das Gegenstück zum Vergrößern. Wenn man das Bild mit dem Vergrößern-Werkzeug vergrößert hat, kann man es mit diesem Werkzeug wieder verkleinern. Dabei wird es mit nur einem Klick auf das Werkzeug so verkleinert, dass wieder alles komplett sichtbar ist. Dieses Werkzeug bleibt nach der Benutzung nicht aktiviert, da der Nutzer das Bild nur einmal verkleinern braucht, damit wieder alles sichtbar ist.

4.5.3 Nächstes Bild

Dieses Werkzeug erlaubt es, zum nachfolgenden Bild in der Sequenz zu wechseln. Genau wie beim Verkleinern-Werkzeug ist dies eine einmalige Operation, so dass das Werkzeug nicht aktiviert bleibt und direkt nach der Benutzung wieder das vorherige Werkzeug aktiviert ist.

4.5.4 Vorheriges Bild

Mit diesem Werkzeug ist es möglich, zum vorherigen Bild in der Sequenz zu wechseln. Die Werkzeuge Nächstes Bild und Vorheriges Bild können auch über die Pfeiltasten aktiviert werden. Auch dieses Werkzeug ist eine einmalige Operation.

4.5.5 Objekt zerteilen

Durch die Vorabannotationen kann es vorkommen, dass zwei verschiedene Objekte fälschlicherweise als ein einzelnes Objekt erkannt werden. In diesem Fall ist es hilfreich, wenn ein Objekt in zwei Objekte zerteilt werden kann. Dafür gibt es dieses Werkzeug. Ist dieses Werkzeug ausgewählt, so kann der Nutzer in der Arbeitsfläche zwei Eckpunkte eines Objektes auswählen, an denen es zerteilt werden soll.

4.5.6 Löschen

Mit diesem Werkzeug ist es möglich einzelne Eckpunkte oder ganze Objekte zu löschen. Dafür kann man in der Arbeitsfläche einen Bereich markieren, in dem alle Eckpunkte gelöscht werden sollen. Entstehen dadurch Objekte mit weniger als 3 Eckpunkten, werden diese Objekte entfernt, da ein Objekt mit nur zwei Ecken kein sinnvolles Objekt darstellt.

4.5.7 Verschieben

Mit diesem Werkzeug ist es möglich die Eckpunkte der Objekte in der Arbeitsansicht zu verschieben, oder neue Eckpunkte einzufügen. Dabei wird der Eckpunkt, der verschoben wird, grün markiert. Befindet sich die Maus nicht nah genug an einem Eckpunkt, so wird an dem nächstgelegenen Randpunkt eines Objektes ein neuer Eckpunkt hinzugefügt, der dann gleich verschoben werden kann.

4.5.8 Neues Objekt

Dieses Werkzeug dient dazu neue Objekte zu annotieren. Dafür kann der Nutzer jeden Eckpunkt eines Objektes der Reihenfolge nach anklicken. Um das neue Objekt zu vollenden kann der Nutzer einmal auf den Startpunkt klicken. Während dieses Prozesses wird eine Vorschau des Polygons des neuen Objektes mit grün gestrichelten Linien angezeigt, damit der Nutzer einen Fehler schnell erkennen kann. Falls ein Eckpunkt unabsichtlich gesetzt wurde, kann er mit der rechten Maustaste wieder entfernt werden.

4.5.9 Kopieren und Einfügen

Mit diesem Werkzeug können Objekteigenschaften bildübergreifend kopiert und eingefügt werden. Dafür kann der Nutzer in der Arbeitsansicht zunächst ein Objekt auswählen, welches kopiert werden soll. Auf einem anderen Bild der Sequenz kann der Nutzer dann andere Objekte auswählen, die die Eigenschaften des kopierten Objektes übernehmen sollen. So lässt sich einfach die Objekt-ID und die Klasse der Objekte auf nachfolgenden Bildern festlegen. Der Nutzer kann auch das gesamte kopierte Objekt in ein anderes Bild einfügen, indem er dort auf eine leere Stelle klickt, wo sich das Objekt befinden soll. Während dieses Vorgangs kann der Nutzer auch das kopierte Objekt mit der rechten Maustaste drehen und es so den Bewegungen anpassen, die das Objekt in der Bildsequenz macht. Durch dieses Werkzeug kann der Nutzer viel Zeit sparen, da nicht jedes Objekt auf jedem Bild neu annotiert werden muss.

4.5.10 Verstecken

Mit diesem Werkzeug kann man Objekte verstecken, so dass sie von den anderen Werkzeugen ignoriert werden. Außerdem können versteckte Objekte wieder sichtbar gemacht werden. Versteckte Objekte werden in der Arbeitsansicht ausgegraut angezeigt. Die Eckpunkte von versteckten Objekten werden nicht angezeigt. Dieses Werkzeug ist vor allem dann nützlich, wenn sich mehrere Objekte direkt nebeneinander befinden, so dass es schwierig ist die Eckpunkte auseinanderzuhalten.

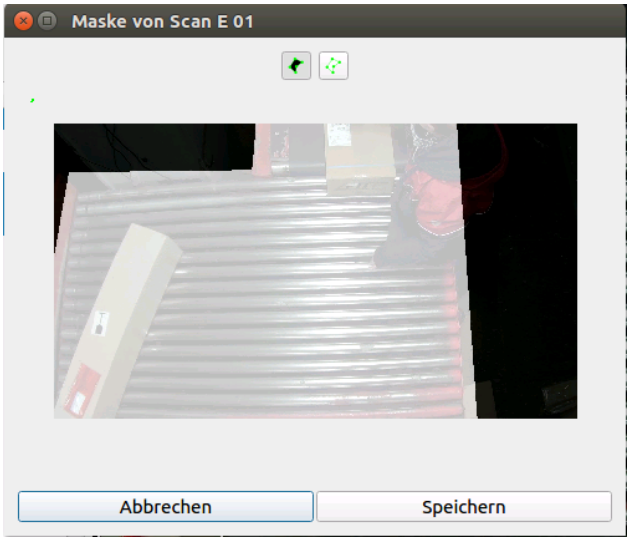


Abbildung 4.5: Maskenoberfläche der GUI

4.6 Maskenoberfläche

Diese Oberfläche dient dem Nutzer dazu Masken für Kameras zu erstellen. Sie kann über den Objektbaum für eine bestimmte Kamera aufgerufen werden. In dieser Oberfläche kann der Nutzer schwarze oder weiße Polygone zeichnen, wobei in den schwarzen Bereichen später keine Objekte vorkommen dürfen.

Beim Öffnen der Ansicht, wird die aktuelle Maske der Kamera geladen und angezeigt. Existiert noch keine Maske für die ausgewählte Kamera, wird eine komplett weiße Maske erstellt, da standardmäßig überall auf einem Bild Objekte sein könnten. Um dem Nutzer das Zeichnen der Maske zu vereinfachen, wird im Hintergrund der Ansicht das erste Bild der Kamera aus der Sequenz angezeigt. Der Nutzer braucht also nur noch den Bereich des Bildes mit schwarzen Polygonen zu markieren, auf dem sich keine Objekte befinden können.

Die Angabe von Masken ist vor allem wegen den Vorabannotationen hilfreich. Ohne eine Maske müssten auf vielen Bildern immer wieder die gleichen fälschlicherweise erkannten Objekte per Hand gelöscht werden. Mit einer Maske geschieht dies meist automatisch und es kann viel Zeit erspart werden. In Abbildung 4.5 befindet sich ein Bild der Maskenoberfläche.

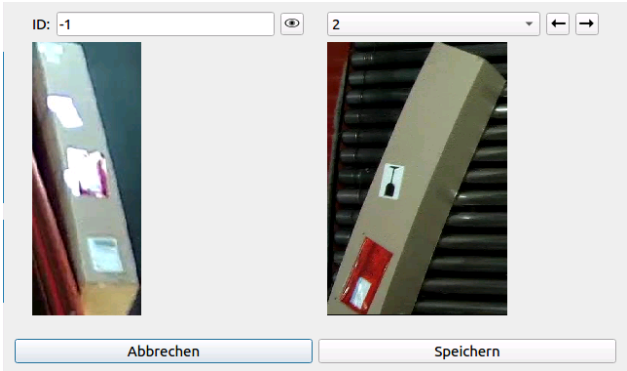


Abbildung 4.6: Objekt-Zuweisungsoberfläche der GUI

4.7 Objekt-Zuweisungsoberfläche

Neben dem ‚Kopieren und Einfügen' Werkzeug, kann man die Objekt-ID eines Objektes auch mit der Objekt-Zuweisungsoberfläche anpassen. Diese Oberfläche ist notwendig, da neu vorkommende Objekte zu Beginn eine ID zugewiesen bekommen müssen und diese nicht von anderen Objekten kopiert werden kann. Außerdem erleichtert die Oberfläche das Zuweisen der Objekt-IDs nach einem Kamerawechsel, da hier weit auseinanderliegende Bilder einer Sequenz miteinander verglichen werden müssen. Die Objekt-Zuweisungoberfläche bietet dem Nutzer daher die Möglichkeit das Objekt mit allen bereits bekannten Objekten zu vergleichen und so das passende Objekt zu finden.

Für diesen Zweck ist die Ansicht in zwei Teile geteilt. Links befindet sich das Objekt, dem die richtige ID zugewiesen werden soll und rechts befinden sich andere Objekte aus vorherigen Bildern zum Vergleich. Der Nutzer kann rechts ein Objekt auswählen, welches er vergleichen will. Dazu gibt es eine Liste mit allen Objekt-IDs. Wird eine dieser Objekt-IDs ausgewählt, so wird das Objekt auf dem letzten Bild mit der ID zum Vergleich angezeigt.

Da die Bilder einer Bildsequenz unscharf sein können, kann es vorkommen, dass das Objekt, welches als Vergleich angezeigt wird, nicht gut zu erkennen ist. Für diesen Fall gibt es zwei Knöpfe, um durch die Sequenz zu navigieren. Der Eine bewirkt dabei, dass das ausgewählte Objekt aus einem Bild weiter vorne in der Sequenz zum Vergleich angezeigt wird. Der Andere zeigt das Objekt auf einem Bild weiter hinten in der Sequenz an. In Abbildung 4.6 befindet sich ein Bild der Objekt-Zuweisungsoberfläche.

Abbildung 4.7: Klassenoberfläche der GUI

4.8 Klassenoberfläche

Diese Oberfläche dient zur Übersicht und zur Bearbeitung der verschiedenen Annotationsklassen, deren Objekte in der Sequenz annotiert werden sollen. Sie kann vom Nutzer im Menü der GUI unter der Kategorie Optionen geöffnet werden. Die Oberfläche zeigt eine Liste mit allen Klassen an, die in der Sequenz vorkommen. Der Nutzer kann eine Klasse umbenennen, neue Klassen hinzufügen oder Klassen löschen. Beim Löschen einer Klasse erhalten alle Objekte dieser Klasse automatisch die Klasse zugewiesen, welche sich in der Liste ganz oben befindet.

5 Implementierung

In diesem Abschnitt der Arbeit wird die GUI aus Implementationssicht betrachtet und es wird erläutert, wie genau die einzelnen Features, die in Abschnitt 4 erwähnt wurden, umgesetzt worden sind. Bei der Implementation einer Anwendung ist es üblich für bestimmte Bereiche fertige Bibliotheken zu verwenden. Auf diese Weise muss nicht jeder Entwickler immer wieder den gleichen Code schreiben und es wird viel Zeit gespart. Des Weiteren ist es üblich den Code gut zu strukturieren, damit er auch später noch gut verstanden und verändert werden kann. Hierfür verwendet man bestimmte Architektur- und Entwurfsmuster, welche sich im Laufe der Jahre als vorteilhaft etabliert haben. In den folgenden Abschnitten werden die verwendeten Bibliotheken und Entwurfs- und Architekturmuster vorgestellt.

5.1 Verwendete Bibliotheken

Für die Implementierung der GUI wurde die objektorientierte Programmiersprache C++ [8] gewählt. Objektorientierte Programmiersprachen bieten den Vorteil, dass das Programm besser strukturiert werden kann, da man es in verschiedene Klassen aufteilen kann.

Für die Gestaltung der GUI Oberflächen wurde das Qt-Framework[1] verwendet. Dies bietet den Vorteil, dass es plattformübergreifend ist und daher nicht für jedes Betriebssystem neuer Code geschrieben werden muss. Das Qt-Framework bietet außerdem viele praktische Funktionen um eine GUI zu entwerfen. Als Entwicklungsumgebung wurde der in Qt enthaltene QtCreator verwendet. Dieser bietet speziell für das Qt-Framework ausgelegte Funktionen, die es einem Entwickler erheblich erleichtern eine Anwendung

1 http://doc.qt.io/qt-5/index.html

mit dem Qt-Framework zu entwickeln. Der QtCreator erleichtert den Entwurf einer GUI vor allem durch die Möglichkeit diese grafisch zu erstellen. Es muss also nicht jedes GUI Element erst aufwendig im Quellcode erstellt werden, sondern kann in einem grafischen GUI Editor platziert werden.

Für die Kommunikation mit dem oben erwähnten Server, welcher die Vorabannotationen bereitstellt, wurde die ZeroMQ Bibliothek[2] verwendet. ZeroMQ ist eine bekannte Bibliothek für die Kommunikation zwischen verschiedenen Anwendungen. ZeroMQ läuft auf den meisten modernen Betriebssystemen und ist zudem einfach und schnell zu benutzen [9].

Für das Arbeiten mit Bildern, wie zum Beispiel für die Extraktion der Objekt-Polygone aus einem Bild, wurde die OpenCV Bibliothek[3] verwendet. OpenCV steht für Open Source Computer Vision und bietet viele nützliche Algorithmen für Bild- und Videoanalyse an [10].

Für das Lesen und Speichern der Annotationen im XML Format wurde TinyXML-2[4] verwendet. TinyXML-2 ist eine kleine Bibliothek zum Laden und Speichern von XML Dateien in C++, welche einfach einzubinden ist.

5.2 Architekturmuster

Bei der Implementierung der GUI wurde besonders darauf geachtet, dass die Programmstruktur gut wartbar und veränderbar ist. Hierfür wurde das Model-View-Controller Architekturmuster [11] verwendet. Dabei wird zwischen der Datenhaltung (Model), der Darstellung (View) und den Schnittstellen für die Nutzereingaben (Controller) unterschieden. Abbildung 5.1 zeigt eine grafische Darstellung des Model-View-Controller Architekturmusters. Hierbei sind die Klassen für Model, View und Controller abstrakt, so dass es verschiedene Umsetzungen von ihnen in einer Anwendung geben kann. Das hat den Vorteil, dass man ohne viel Aufwand verschiedene Views implementieren kann, welche die gleichen Daten unterschiedlich darstellen. Auch kann man verschiedene Controller implementieren, welche die Nutzereingaben unterschiedlich verarbeiten und die Daten

2 http://zguide.zeromq.org/page:all
3 https://opencv.org/
4 http://leethomason.github.io/tinyxml2/

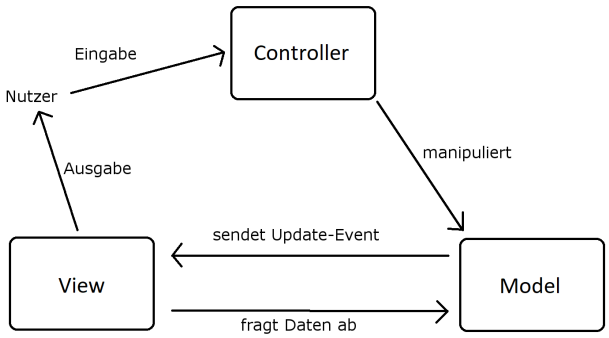


Abbildung 5.1: Model-View-Controller

auf andere Weise ändern. Auf diese Weise erhält man eine hohe Flexibilität des Codes. Durch die Aufteilung des Codes in die drei Bereiche ist der Code außerdem gut strukturiert und kann leicht außeinandergehalten werden. Wenn zum Beispiel die Oberfläche der Anwendung angepasst werden soll, muss auch nur Code in diesem Bereich verändert werden.

Im Folgenden wird beschrieben wie das Model-View-Controller Architekturmuster zur Implementierung der GUI eingesetzt wurde.

5.2.1 Datenhaltung

Im Wesentlichen gibt es in der GUI zwei unterschiedliche Model-Klassen. Die Eine, das Arbeitsmodel, enthält lediglich Daten, die für die korrekte Darstellung der Arbeitsfläche notwendig sind, aber nicht zur geöffneten Bildsequenz gehören. Dazu gehört beispielsweise ein unvollendetes Polygon, welches vom Nutzer erst noch fertig erstellt wird, bis es schließlich zu einem neuen Objekt in der Sequenz wird (mit dem ‚Neues Objekt' Werkzeug aus Abschnitt 4.5.8). Auch wird dort ein Objekt gespeichert, welches vom Nutzer kopiert wurde, aber noch nicht eingefügt wurde (mit dem ‚Kopieren und Einfügen' Werkzeug aus Abschnitt 4.5.9).

Die andere Model-Klasse, die Sequenz Klasse, enthält alle Daten der geöffneten Bildsequenz. Dafür benutzt sie wegen der Vielzahl an verschiedenen Daten in einer Sequenz einige weitere Hilfsklassen. In Abbildung 5.2 befindet sich dazu ein gekürztes UML (Unified Modeling Language) Klassendiagramm.

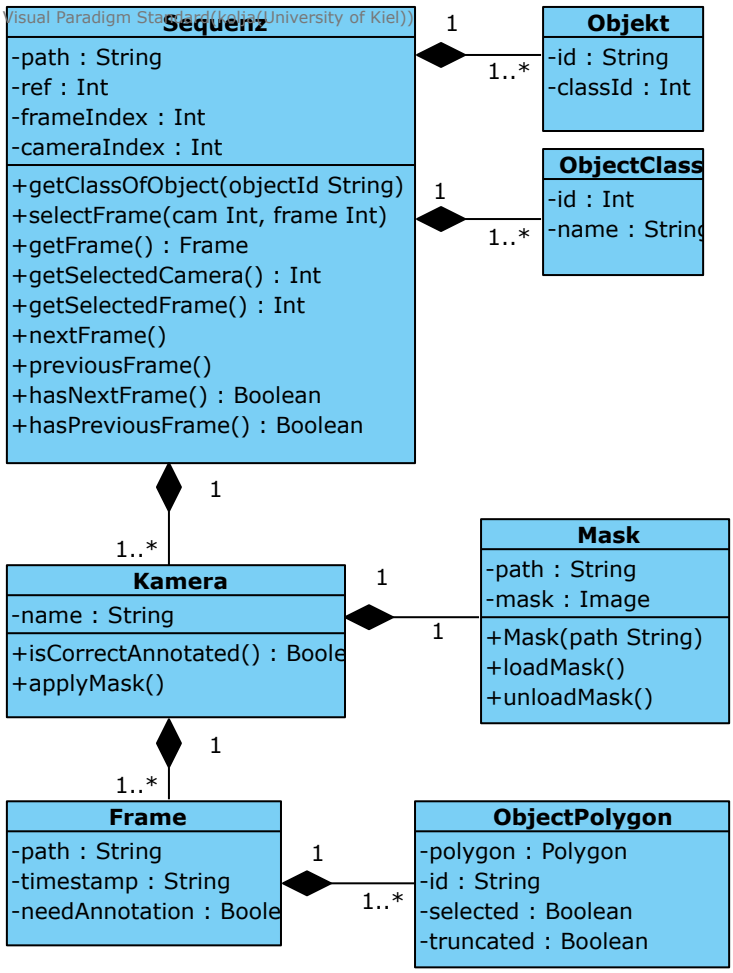


Abbildung 5.2: Klassendiagramm der Datenhaltung-Klassen für eine Bildsequenz

Da die GUI dazu verwendet werden soll große Bildsequenzen mit mehreren tausend Bildern zu annotieren, ist es nicht möglich alle Bilder gleichzeitig in den Arbeitsspeicher zu laden. Daher lädt die GUI zunächst nur die Pfade zu den Bild-Dateien, und lädt die Bilder nur, wenn sie gebraucht werden. Dazu gibt es eine Frame Klasse, welche für das Laden von Bildern zuständig ist. Außerdem enthält sie eine Liste mit Objekt-Polygonen zu einem Bild. Des Weiteren gibt es eine Mask Klasse, welche für das Laden und Speichern der Masken Bilder zuständig ist. Zur Verwaltung dieser Klassen gibt es eine Kamera Klasse, welche eine Liste von Frame Objekten, ein Mask Objekt und den Namen einer Kamera enthält. Die verschiedenen Kamera Objekte werden von der Sequenz Klasse verwaltet.

Zusätzlich enthält die Sequenz Klasse auch noch eine Liste mit allen Objektklassen und

eine Liste mit allen in der Sequenz vorkommenden Objekten. Letzteres wurde dabei aus Performance Gründen hinzugefügt. Wenn festgestellt werden soll, ob ein bestimmtes Objekt existiert, müssen so nicht immer alle Kameras und Bilder durchsucht werden. Das wird zum Beispiel von der Objekt-Zuweisungsoberfläche aus Abschnitt 4.7 benötigt, in der eine Liste mit allen vorhandenen Objekten angezeigt wird. Zwei Views, welche die Sequenz Klasse als Model verwenden, sind der Navigationsbaum aus Abschnitt 4.3 und die Arbeitsfläche aus Abschnitt 4.2. Letztere greift dabei hauptsächlich auf die Frame Klasse zu, welche das aktuell ausgewählte Bild enthält.

5.2.2 Controller

Im Wesentlichen ist in der GUI immer ein Haupt-Controller aktiv, welcher für die Auswahl der Werkzeuge und für Aktionen wie zum Beispiel das Laden einer Sequenz in der Hauptansicht 4.1 zuständig ist. Dieser Controller aktiviert dann je nach ausgewähltem Werkzeug andere Controller, die nur für die Interaktion des Nutzers mit der Arbeitsfläche aus Abschnitt 4.2 zuständig sind. Dafür erben diese Controller von einer ArbeitsController Klasse, die bereits über grundlegende Funktionalitäten verfügt, welche bei allen Controllern gleich sind. Dazu gehört zum Beispiel die Möglichkeit mit dem Mausrad näher ran oder weiter weg zu zoomen. Die erbenden Controller erweitern diese Funktionen, so dass sie die Funktionen von jeweils einem Werkzeug anbieten. Eine Ausnahmen ist dabei das ‚Kopieren und Einfügen' Werkzeug, welches in zwei Controller, dem CopyController und dem PasteController, aufgespalten ist. Dies ist als UML Klassendiagramm in Abbildung 5.3 verkürzt dargestellt.

5.2.3 Entwurfsmuster

Bei der Implementierung der GUI wurden neben dem Model-View-Controller Architekturmuster auch noch einige Entwurfsmuster eingesetzt. Entwurfsmuster werden dafür eingesetzt, wiederkehrende Probleme beim Design von Software zu lösen. Dabei wird beim Entwurf nach einem bestimmten Muster vorgegangen, welches sich für eine bestimmte Problemstellung bewährt hat.

Wie oben bereits erwähnt, ist die Datenhaltung in der GUI ein komplexes System von Klassen, die miteinander interagieren. Der Navigationsbaum der GUI, welcher mit Hilfe

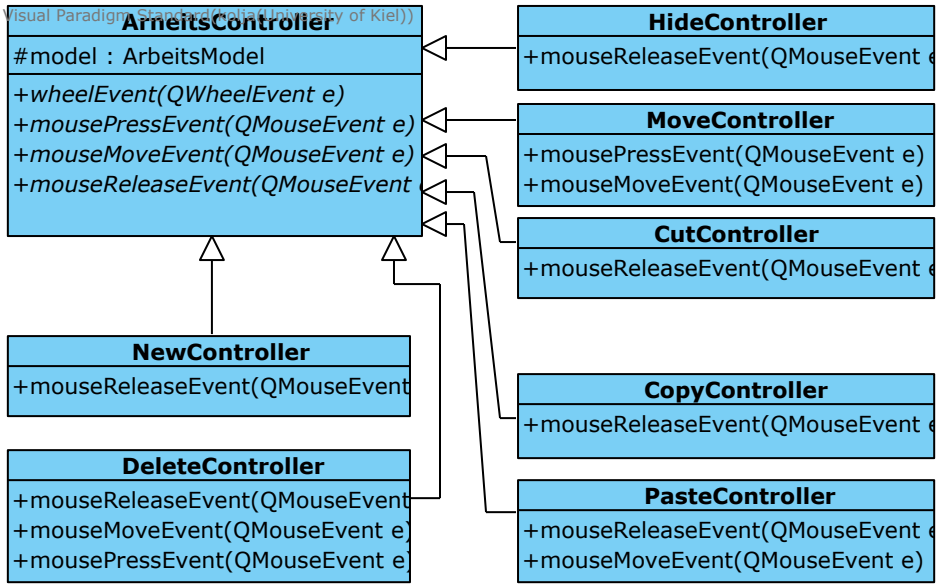


Abbildung 5.3: Klassendiagramm der Controller-Klassen

des Qt-Frameworks implementiert wurde, braucht Zugriff auf dieses komplexe System, damit er die Daten korrekt darstellen kann. Um dies effizient zu gewährleisten, wurde das Fassade Entwurfsmuster eingesetzt. Bei diesem wird eine Klasse als Schnittstelle für ein komplexes System eingesetzt [12].

Das Fassade Entwurfsmuster sorgt für eine gute Strukturierung des Codes, da die Anwendung einfacher in ihre verschiedenen Komponenten zerlegt werden kann.

Ein weiteres verwendetes Entwurfsmuster ist das Singleton Entwurfsmuster. Bei diesem Entwurfsmuster geht es darum sicherzustellen, dass zur Laufzeit von einer Klasse immer nur ein einziges Objekt existiert [12]. Dieses Entwurfsmuster wurde bei der GUI für die Controller-Klassen aus Abbildung 5.3 verwendet. Durch die Verwendung des Singleton Entwurfsmusters an dieser Stelle wird sichergestellt, dass immer nur ein einzelnes Controller Objekt von einer Klasse existiert. Dies ist sinnvoll, weil die Nutzereingaben nicht mehrfach verarbeitet werden sollen.

6 Untersuchung

Um den Nutzen der oben vorgestellten GUI zu belegen, wurde im Verlauf dieser Arbeit eine Untersuchung durchgeführt. Die Untersuchung verglich die GUI mit einem internen Werkzeug zur Annotation von Bildsequenzen. Das interne Werkzeug wurde bis zur Fertigstellung dieser Arbeit für die Annotation von Bildsequenzen verwendet, und soll nun durch die vorgestellte GUI abgelöst werden.

Das interne Werkzeug besteht aus verschiedenen Python Scripten, welche aufwendig per Hand über die Konsole gestartet werden müssen. Es ist dabei darauf beschränkt Packete zu annotieren und lässt keine weiteren Objektklassen zu. Außerdem ist es nicht möglich bereits annotierte Objekte nachträglich zu verändern. Dieses interne Werkzeug zeigt dem Nutzer immer ein Bild, welches er in verschiedenen Modis bearbeiten kann. Dabei muss der Nutzer viele Tastenbefehle auswendig kennen, da keine GUI mit Knöpfen vorhanden ist.

In diesem Kapitel werden der Ablauf und die Auswertung der Tests, sowie die Reaktion auf die Tests beschrieben.

6.1 Ablauf der Tests

Um zu belegen, dass mit der neuen GUI bessere Ergebnisse erzielt werden können, wurden 12 freiwillige Testpersonen (Studenten) herangezogen. Diese wurden zunächst in beide Anwendungen eingeführt, dann annotierten sie zehn Minuten lang eine Bildsequenz mit dem internen Werkzeug. Daraufhin annotierten sie dieselbe Bildsequenz zehn Minuten lang mit der in dieser Arbeit vorgestellten GUI. Die Bildsequenzen, die für die Tests verwendet wurden, bestanden jeweils aus einer Kamera mit 50 Bildern. Dies wurde so gewählt, weil für das Annotieren einer Sequenz mit mehreren Kameras mehr Zeit benötigt

wird als für das Annotieren einer Sequenz mit nur einer Kamera. Die Tests sollten aber nicht zu viel Zeit in Anspruch nehmen, da sich sonst weniger Testpersonen freiwillig gemeldet hätten.

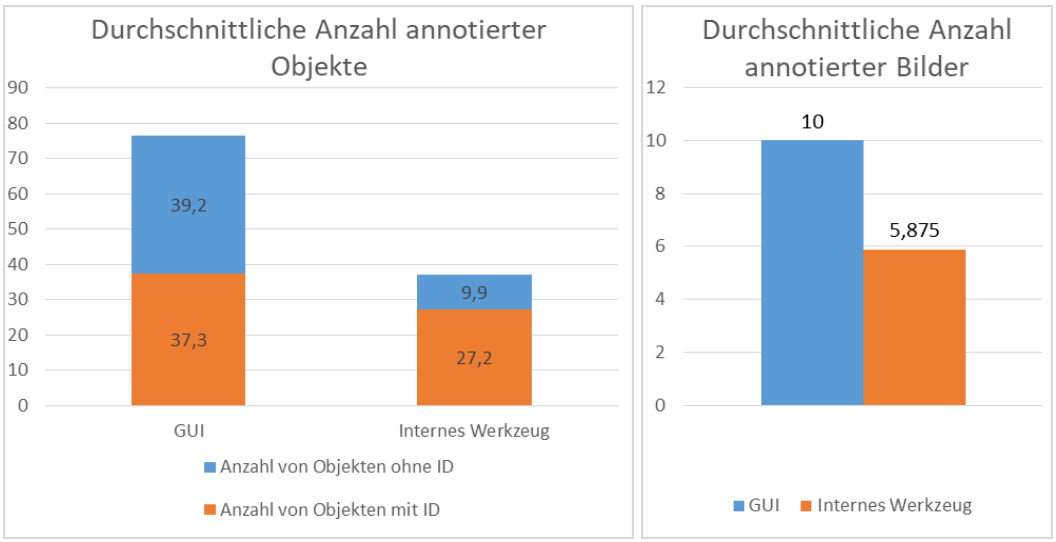
Am Ende erhielten die Testpersonen einen Fragebogen zur Bewertung der beiden Anwendungen. Während die Testpersonen die Bildsequenzen annotierten wurde gemessen, wie viele Objekte sie mit der GUI und wie viele sie mit dem internen Tool in der Zeit annotieren konnten. Bei der GUI wurde auch die Nutzung der unterschiedlichen Werkzeuge protokolliert, so dass sich feststellen lässt welche Werkzeuge am häufigsten genutzt wurden.

6.2 Fragebogen

Mit Hilfe eines Fragebogens sollte bei der Untersuchung die persönliche Meinung der Testpersonen ermittelt werden. Dies war notwendig, weil die persönliche Meinung nicht automatisch gemessen werden kann, wie zum Beispiel die Annotationsgeschwindigkeit. Der Fragebogen enthielt sowohl Fragen zum direkten Vergleich der beiden Anwendungen, als auch Fragen zur Bewertung von Funktionen, welche nur in der hier vorgestellten GUI vorhanden waren. Dazu gehört zum Beispiel das Benutzen von verschiedenen Objektklassen. Des Weiteren gab es Freitextfelder, wo die Testpersonen die vorhandenen Features loben oder kritisieren konnten und Vorschläge für weitere Features liefern konnten. Der vollständige Fragebogen befindet sich im Anhang.

6.3 Auswertung

Im Folgenden werden die Untersuchungsergebnisse beschrieben. Außerdem werden die Ergebnisse, die mit der GUI erzielt wurden, mit denen des internen Werkzeugs verglichen.



(a) Anzahl der annotierten Objekte (b) Anzahl der annotierten Bilder

Abbildung 6.1: Vergleich der annotierten Bilder und Objekte zwischen GUI und internem Werkzeug

6.3.1 Anzahl annotierter Objekte

In Abbildung 6.1 wird die Anzahl der im Durchschnitt von den Testpersonen annotierten Objekte verglichen. Dabei wird zwischen annotierten Objekten mit zugewiesener Objekt-ID und zwischen annotierten Objekten ohne Objekt-ID unterschieden. Es hat sich herausgestellt, dass mit der GUI insgesamt etwa doppelt so viele Objekte annotiert wurden wie mit dem internen Werkzeug. Dies lässt sich auch bei der durchschnittlichen Anzahl annotierter Bilder beobachten. Es wurde aber nur etwa der Hälfte der in der GUI annotierten Objekte eine ID zugewiesen. Die Anzahl der annotierten Objekte ohne ID ist bei dem internen Werkzeug geringer, da hier alle Objekte per Hand annotiert werden mussten. Bei der GUI wurden die Objekte als Vorabannotation automatisch erkannt, so dass nur noch eine ID zugewiesen werden musste.

Aus dem Ergebnis lässt sich schließen, dass das Zuweisen der Objekt-ID in der GUI noch verbessert werden kann, da es vermutlich mehr Zeit in Anspruch genommen hat als das Zuweisen der Objekt-ID im internen Werkzeug. Dennoch wurden in der GUI auch mehr Objekte mit ID annotiert als mit dem internen Werkzeug, da die Nutzer hier mehr Zeit für das Zuweisen der Objekt-IDs hatten als beim internen Werkzeug. Bei dem Annotieren einer großen Sequenz lohnt es sich also die GUI zu benutzen.

6.3.2 Bewertung durch die Testpersonen

In dem Fragebogen wurden die Testpersonen gebeten, das Annotieren von neuen Objekten und das Zuweisen der Objekt-ID auf einer Skala von 1 bis 10 zu bewerten. Dabei stand 1 für sehr umständlich und 10 für überhaupt nicht umständlich. Ein Vergleich der durchschnittlichen Bewertung für die GUI und für das interne Werkzeug befindet sich in Abbildung 6.2. In diesem Vergleich schneidet die GUI insgesamt besser ab als das interne Werkzeug.

Bei der Annotation neuer Objekte schneidet die GUI mit einer Bewertung von 8,5 am besten ab und liegt hier mit 2,5 Punkten vor der Bewertung des internen Werkzeug. Dies liegt vermutlich daran, dass den Testpersonen hier durch die Vorabannotationen viel Arbeit erspart wurde. Auch das Zuweisen der Objekt-ID hat mit 8,1 um 2,7 Punkte besser abgeschnitten als das interne Werkzeug.

Zu der GUI sollten die Testpersonen auch noch das Korrigieren von Vorabannotationen und das Zuweisen der Objektklasse bewerten. Bei den Tests hat sich dann jedoch herausgestellt, dass das Zuweisen der Objektklasse in den für die Tests verwendeten Sequenzen nicht benötigt wurde. Daher wurde die Frage von den Testpersonen nicht beantwortet und kann somit auch nicht ausgewertet werden. Das Korrigieren der Vorabannotationen bekam von den Testpersonen eine durchschnittliche Bewertung von 8,4.

Von den Features, die von der GUI mit dem Fragebogen bewertet wurden, hat das Zuweisen der Objekt-ID etwas schlechter abgeschnitten als das Annotieren neuer Objekte oder das Korrigieren von Vorabannotationen. Das könnte daran liegen, dass für das Zuweisen der Objekt-ID mehrere Bilder angesehen werden mussten, um festzustellen, welche ID ein Objekt hat. Außerdem werden die Objekt-IDs in der GUI erst angezeigt, wenn der Nutzer mit der Maus auf das Objekt zeigt, was vermutlich als zu umständlich angesehen wurde.

6.3.3 Bevorzugte Anwendung

In dem Fragebogen haben die Testpersonen außerdem angegeben, welche Anwendung sie für eine große Sequenz von Bildern lieber verwenden würden. Das Ergebnis ist in Abbildung 6.3 dargestellt. Von den 12 Testpersonen würden 11 Personen lieber die hier

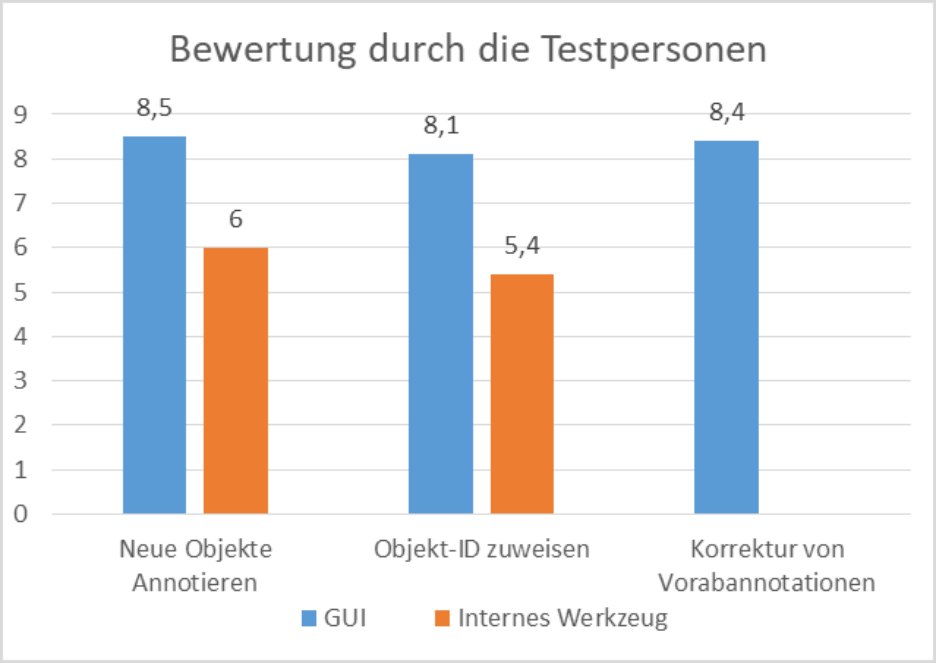


Abbildung 6.2: Auswertung der ersten beiden Fragen des Fragebogens

vorgestellte GUI verwenden. Dies wurde von den Testpersonen in einem Freitextfeld des Fragebogens unterschiedlich begründet. Die am häufigsten vorkommenden Begründungen waren, dass durch die Grafische Benutzeroberfläche das Bedienen einfacher ist und dass durch die Vorabannotationen und durch die Kopierfunktion von Objekten viel Zeit gespart werden kann.

Eine Testperson gab an lieber das interne Werkzeug zu benutzen, mit der Begründung, dass es einfacher zu bedienen ist, weil es weniger Features hat. Dies ist aber vermutlich eine Sache der Gewöhnung. Hinzu kommt, dass den Testpersonen für jede Anwendung nur zehn Minuten Zeit zur Verfügung standen und sie sich nicht länger damit beschäftigen konnten. Für ein simples Programm wie das interne Werkzeug reicht diese Zeit aus, um alle Funktionen zu verstehen und zu benutzen. Bei der GUI wird mehr Zeit benötigt um ein Gefühl dafür zu bekommen, wie man am effizientesten damit arbeiten kann.

Dennoch zeigt das Ergebnis, dass die GUI aufgrund der vielen Features für unerfahrene Benutzer nicht so leicht zu bedienen ist und eventuell mehr Zeit benötigt wird um sich einzuarbeiten.

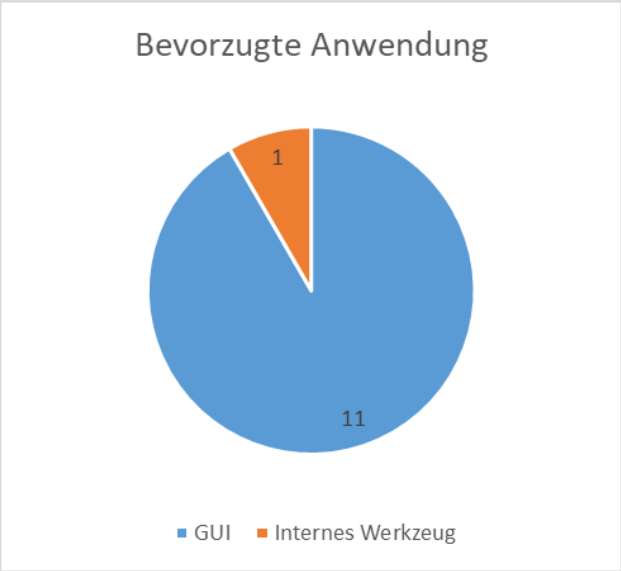


Abbildung 6.3: Bevorzugte Anwendung der Testpersonen

6.3.4 Weitere Angaben

Die Testpersonen wurden im Fragebogen auch nach fehlenden oder besonders hilfreichen Features gefragt. An dieser Stelle wurde bemängelt, dass bei der GUI die Vergabe von neuen Objekt-IDs nicht komfortabel genug sei und dass die Objekt-IDs in der Arbeitsfläche nicht gut genug sichtbar seien.

Es wurde vorgeschlagen die Objekt-IDs in der Arbeitsfläche anzeigen zu lassen, so dass es für den Nutzer einfacher ist die ID eines Objektes zu sehen. Dieser Vorschlag wurde nachträglich in die GUI integriert, wie es in Abschnitt 6.4 beschrieben wird. Des Weiteren gab es den Vorschlag das Kopieren von mehreren Objekten gleichzeitig zuzulassen, so dass der Nutzer nicht so oft zwischen den Bildern springen muss. Dieser Vorschlag wurde mangels Zeit noch nicht umgesetzt, da eine vorschnelle Umsetzung eine Verringerung der Intuitivität zufolge haben könnte.

Bei den besonders hilfreichen Features wurde noch einmal betont, was als Begründung für die bevorzugte Nutzung der GUI angegeben wurde:

1. die Grafische Benutzeroberfläche
2. die Vorabannotationen
3. die Kopierfunktion von Objekten

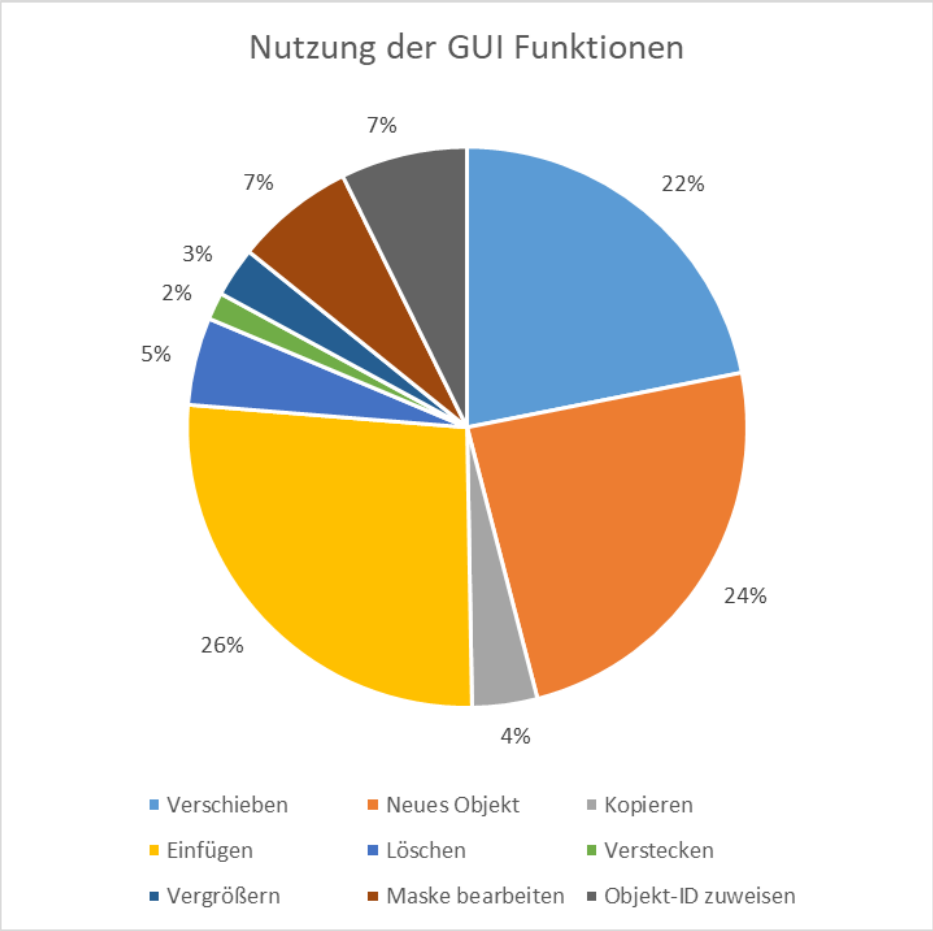


Abbildung 6.4: Prozentuale Nutzung der einzelnen GUI Funktionen nach Zeit

6.3.5 Nutzung einzelner GUI Features

Wie oben bereits erwähnt, wurde während der Tests analysiert, welche Werkzeuge und Funktionen der GUI von den Testpersonen hauptsächlich benutzt wurden. Mit den Ergebnissen kann analysiert werden, welche Funktionen für eine GUI zur Annotation von Bildsequenzen am wichtigsten sind. In Abbildung 6.4 wird die Zeit, die die Testpersonen mit den einzelnen Features verbracht haben, verglichen. Das ‚Objekt zerteilen' Werkzeug wird hier nicht betrachtet, da es nur in dem Sonderfall benötigt wird, wo in den Vorabannotationen zwei Objekte als eins erkannt werden. Dieser Sonderfall ist während der Tests nur sehr selten aufgetreten.

Insgesamt haben die Testpersonen 30% der Zeit mit dem ‚Kopieren und Einfügen' Werkzeug gearbeitet. Zusammen mit den obigen Daten lässt sich hiermit der hohe Nutzen

dieses Werkzeugs beim Annotieren von Bildsequenzen belegen. Bemerkenswert ist außerdem, dass die Testpersonen über fünf mal mehr Zeit damit verbracht haben Objekte einzufügen, als sie Zeit benötigten, diese zu kopieren. Hieraus lässt sich folgern, dass ein kopiertes Objekt viele Mahle eingefügt wurde.

Die hohe Nutzung des ‚Kopieren und Einfügen' Werkzeugs lässt sich vermutlich darauf zurückführen, dass mit dem Werkzeug, neben dem Kopieren und Einfügen von Objekten, auch das Zuweisen der Objekt-IDs bewältigt werden kann. Dabei kopiert der Nutzer ein Objekt mit ID und weist diese dann beim Einfügen anderen Objekten mit nur einem Klick zu.

24% der Zeit wurde von den Testpersonen dafür verwendet neue Objekte manuell zu annotieren, welche von den Vorabannotationen nicht automatisch erkannt wurden. Dies zeigt, dass die Vorabannotationen zusammen mit dem ‚Kopieren und Einfügen' Werkzeug alleine noch nicht für eine voll funktionsfähige GUI zum Annotieren von Bildsequenzen ausreichen würden.

22% der Zeit verbrachten die Testpersonen damit Eckpunkte von Objekten zu verschieben. Dabei handelte es sich mit großer Wahrscheinlichkeit um nicht ganz korrekte Vorabannotationen, welche durch das Verschieben von Ecken angepasst werden können. Dies zeigt, dass durch die Vorabannotationen zwar viel Zeit gespart werden kann, die sonst benötigt werden würde um die Objekte einzeln zu annotieren. Andererseits muss aber auch etwas Zeit dafür verwendet werden um die Vorabannotationen zu korrigieren.

Das Bearbeiten der Maske benötigt nur wenig Zeit, da es nur einmal zu Beginn gemacht werden muss. Daher mussten die Testpersonen nur 7% der Zeit dafür verwenden.

Ebenfalls 7% der Zeit hat das Zuweisen von Objekt-IDs über die Objekt-Zuweisungsoberfläche aus Abschnitt 4.7 gekostet. Hierbei muss beachtet werden, dass dies nur ein Teil der Zeit ist, die mit dem Zuweisen von Objekt-IDs verbracht wurde, da man die Objekt-IDs auch mit dem ‚Kopieren und Einfügen' Werkzeug zuweisen kann.

Die Werkzeuge Vergrößern, Löschen und Verstecken wurden nur wenig verwendet. Dies zeigt, dass sie nur in Sonderfällen benötigt werden und für eine voll funktionsfähige GUI zum Annotieren von Bildsequenzen nicht unbedingt notwendig sind. Dabei muss

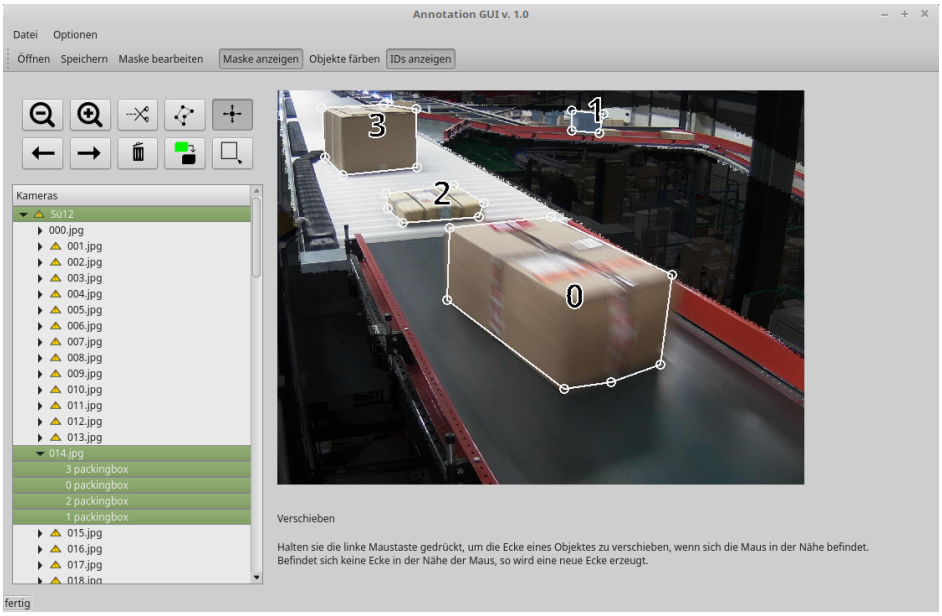


Abbildung 6.5: Die angepasste GUI mit angezeigten Objekt-IDs

beachtet werden, dass in der GUI neben dem Löschen Werkzeug ein Objekt auch mittels Rechtsklick gelöscht werden kann und das Löschen Werkzeug nur für das Löschen einzelner Eckpunkte benötigt wird.

6.4 Integration der Testergebnisse in die GUI

Als Reaktion auf die Testergebnisse wurden einzelne GUI Funktionen etwas angepasst, so dass die Hauptkritikpunkte der Testpersonen behoben sind.

Ein Hauptkritikpunkt an der GUI war, dass die Objekt-IDs in der Arbeitsfläche nicht gut genug sichtbar waren. Um dieses Problem zu beheben, wurde die Option hinzugefügt, sich die Objekt-IDs in der Arbeitsfläche anzeigen zu lassen. Ein Bild hiervon befindet sich in Abbildung 6.5.

Ein weiterer Kritikpunkt war das Zuweisen von neuen Objekt-IDs über die Objekt-Zuweisungsoberfläche. Diese war den Testpersonen zu umständlich gestaltet. Hier wurde die Bedienung vereinfacht, indem zum Beispiel die Objekt-ID des ausgewählten Vergleichsobjektes direkt im ID Textfeld übernommen wird. Sobald der Nutzer das richtige Vergleichobjekt gefunden hat, braucht er also nur noch auf Speichern zu klicken. Auch wird

nun bei der Eingabe einer ID direkt ein Vergleichsobjekt mit dieser ID angezeigt. Des Weiteren lässt sich die ID jetzt auch mit den Pfeiltasten um eins erhöhen oder verringern. Je nach Nutzervorliebe kann das Vergleichsobjekt und somit die korrekte Objekt-ID also jetzt auch nur mit der Tastatur gefunden und zugewiesen werden.

7 Schluss

In dieser Arbeit wurde eine neue GUI für die Annotation von Bildsequenzen entworfen und implementiert. Die entworfene GUI unterstützt den Nutzer bei der Annotation, indem sie Vorabannotationen bereitstellt, welche den Prozess des Annotierens beschleunigen.

Mit Hilfe der GUI können Datensätze erstellt werden, welche für das Training eines Computers zur Erkennung von Objekten auf Bildern verwendet werden können. Es werden dabei sowohl die Boundingboxen, als auch die Polygone der Objekte in dem Datensatz gespeichert. Es kann also später noch entschieden werden, welche Daten für das Training verwendet werden sollen. Auf diese Weise kann die GUI für viele unterschiedliche Forschungsprojekte verwendet werden, bei denen eine automatische Objekterkennung eine Rolle spielt.

Durch eine Untersuchung mit Testpersonen wurde belegt, dass mit der GUI einfacher und effizienter gearbeitet werden kann, als mit einem internen Werkzeug, welches zuvor für die Annotation von Bildsequenzen verwendet wurde. Die in dieser Arbeit vorgestellte GUI hat dabei im direkten Vergleich deutlich besser abgeschnitten als das interne Werkzeug. Es wurde also bewiesen, dass die GUI zur effizienten Annotation von Bildsequenzen geeignet ist und andere Anwendungen ablösen kann.

8 Anhang

Der Quelltext der GUI befindet sich Anbei auf einer CD oder kann im Internet unter http://gogs.koljastrohm-games.com/kolja/AnnotationGUI eingesehen werden.

Fragebogen zur Studie

Wie umständlich war das Annotieren eines neuen Objektes?

a) Bei der ersten Anwendung

1 (sehr umständlich)	2	3	4	5	6	7	8	9	10 (nicht umständlich)

b) Bei der zweiten Anwendung

1 (sehr umständlich)	2	3	4	5	6	7	8	9	10 (nicht umständlich)

Wie umständlich war das Festlegen der Objekt-ID?

a) Bei der ersten Anwendung

1 (sehr umständlich)	2	3	4	5	6	7	8	9	10 (nicht umständlich)

b) Bei der zweiten Anwendung

1 (sehr umständlich)	2	3	4	5	6	7	8	9	10 (nicht umständlich)

Wie umständlich war das Zuweisen der Objektklasse?

a) Bei der zweiten Anwendung

1 (sehr umständlich)	2	3	4	5	6	7	8	9	10 (nicht umständlich)

Wie umständlich war das Korrigieren von Vorabannotationen?

a) Bei der zweiten Anwendung

1 (sehr umständlich)	2	3	4	5	6	7	8	9	10 (nicht umständlich)

Welche Anwendung würden Sie bevorzugen, wenn eine große Menge (z.B. 1000) an Bildern annotiert werden soll?

☐ Erste Anwendung

☐ Zweite Anwendung

Begründung für die Entscheidung bei der vorherigen Frage:

Welche Features haben beim Annotieren gefehlt?

a) Bei der ersten Anwendung

b) Bei der zweiten Anwendung

Welche Features waren besonders hilfreich?

a) Bei der ersten Anwendung

b) Bei der zweiten Anwendung

Literaturverzeichnis

[1] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," *arXiv:1312.6229 [cs]*, Dec. 2013, arXiv: 1312.6229. [Online]. Available: http://arxiv.org/abs/1312.6229

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, vol. 8693, pp. 740–755, dOI: 10.1007/978-3-319-10602-1_48. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10602-1_48

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010. [Online]. Available: http://link.springer.com/10.1007/s11263-009-0275-4

[5] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database." IEEE, Jun. 2009, pp. 248–255. [Online]. Available: http://ieeexplore.ieee.org/document/5206848/

[6] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: A Database and Web-Based Tool for Image Annotation," *International Journal of*

Computer Vision, vol. 77, no. 1-3, pp. 157–173, May 2008. [Online]. Available: http://link.springer.com/10.1007/s11263-007-0090-8

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[8] J. E. Perry and H. D. Levin, *An introduction to object-oriented design in C++*. Addison-Wesley, 1996.

[9] A. Dworak, P. Charrue, F. Ehm, W. Sliwinski, and M. Sobczak, "Middleware Trends And Market Leaders 2011," *Conf. Proc.*, vol. C111010, no. CERN-ATS-2011-196, p. FRBHMULT05. 4 p, Oct 2011. [Online]. Available: http://cds.cern.ch/record/1391410

[10] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, "A brief introduction to opencv," in *MIPRO, 2012 proceedings of the 35th international convention*. IEEE, 2012, pp. 1725–1730.

[11] J. Deacon, "Model-view-controller (mvc) architecture," http://www.jdl.co.uk/briefings/index.html.

[12] E. Gamma, R. Johnson, R. Helm, and J. Vlissides, *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Pearson Deutschland GmbH, 2011.