

**HTBLuVA Salzburg**

**Höhere Lehranstalt für**

**Elektronik und Technische Informatik**

HTL_Logo

**DIPLOMARBEIT**

Gesamtprojekt **GitCon**

**Entwicklung eines E-Gitarren zu MIDI-Converters**

Daniel Bräumann 5AHEL Betreuer: Prof. Dipl.-Ing. Siegbert

Schrempf

Simon Grundner 5AHEL

Laurenz Hölzl 5AHEL

ausgeführt im Schuljahr 2022/23

Abgabevermerk:

Datum: übernommen von:

**Eidesstattliche Erklärung**

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Salzburg, am TT.MM.JJJJ Verfasserinnen / Verfasser:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Daniel Bräumann

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Simon Grundner

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Laurenz Hölzl

**DIPLOMARBEIT**

**DOKUMENTATION**

|  |  |
| --- | --- |
| Namen der  Verfasserinnen / Verfasser | Daniel Bräumann  Simon Grundner  Laurenz Hölzl |
| Jahrgang  Schuljahr | 5AHEL  2022/23 |
| Thema der Diplomarbeit | GitCon – Entwicklung eines E-Gitarren zu MIDI-Converters |

|  |  |
| --- | --- |
| Aufgabenstellung | Das Vorliegende Projekt ermöglicht die Verwendung, einer E-Gitarre als MIDI-Device. Das Gerät soll einzelne Noten und Akkorde, mit möglichst geringer Latenz, zuverlässig in das MIDI-Format umwandeln. |

|  |  |
| --- | --- |
| Realisierung |  |

|  |  |
| --- | --- |
| Ergebnisse |  |

|  |  |
| --- | --- |
| Typische Grafik, Foto etc.  (mit Erläuterung) |  |

|  |  |
| --- | --- |
| Teilnahme an Wettbewerben,  Auszeichnungen |  |

|  |  |
| --- | --- |
| Möglichkeiten der Einsichtnahme in die Arbeit |  |

|  |  |  |
| --- | --- | --- |
| Approbation  (Datum / Unterschrift) | Prüferin / Prüfer | Direktorin / Direktor  Abteilungsvorständin / Abteilungsvorstand |

**DIPLOMA THESIS**

**Documentation**

|  |  |
| --- | --- |
| Author(s) | Daniel Bräumann  Simon Grundner  Laurenz Hölzl |
| Form  Academic year | 2022/23 |
| Topic | GitCon - Development of an electric guitar to MIDI converter |

|  |  |
| --- | --- |
| Assignment of Tasks | The present project enables the use of an electric guitar as a MIDI device. The device should reliably convert individual notes and chords into MIDI format with the lowest possible latency. |

|  |  |
| --- | --- |
| Realisation |  |

|  |  |
| --- | --- |
| Results |  |

|  |  |
| --- | --- |
| Illustrative Graph, Photo  (incl. explanation) |  |

|  |  |
| --- | --- |
| Participation in Competitions  Awards |  |

|  |  |
| --- | --- |
| Accessibility of  Diploma Thesis |  |

|  |  |  |
| --- | --- | --- |
| Approval  (Date / Sign) | Examiner | Head of College  Head of Department |

Inhaltsverzeichnis

[1 Danksagung 9](#_Toc5620140)

[2 Einleitung 10](#_Toc5620141)

[3 Aufgabenstellung 10](#_Toc5620142)

[4 Implantmed 11](#_Toc5620143)

[5 Individuelle Zielsetzung und Terminplan 12](#_Toc5620144)

[6 Renci SSH 15](#_Toc5620145)

[6.1 Diffie-Hellman key exchange 15](#_Toc5620146)

[7 SSH (Secure Shell) 16](#_Toc5620147)

[7.1 Verbindungsherstellung Version 1 17](#_Toc5620148)

[7.2 Verbindungsherstellung Version 2 18](#_Toc5620149)

[7.2.1 Implementierung in das Projekt 19](#_Toc5620150)

[7.2.2 Erstellen der Screenshots zur Datenverarbeitung 20](#_Toc5620151)

[8 SFTP (SSH File Transfer Protocol) 22](#_Toc5620152)

[8.1 Datentransfer 23](#_Toc5620153)

[8.2 Bild Rotation 24](#_Toc5620154)

[9 Benutzeroberfläche 25](#_Toc5620155)

[9.1 Layout 25](#_Toc5620156)

[9.2 Connect 27](#_Toc5620157)

[9.2.1 Threading 28](#_Toc5620158)

[9.3 Screenshot 29](#_Toc5620159)

[9.4 Download 29](#_Toc5620160)

[10 Theoretische Grundlagen der Bildverarbeitung 30](#_Toc5620161)

[10.1 Einleitung 30](#_Toc5620162)

[10.2 Definition 31](#_Toc5620163)

[10.3 Industrielle Bildverarbeitung 32](#_Toc5620164)

[10.4 Bildverarbeitungsalgorithmen 32](#_Toc5620165)

[10.4.1 Cannyalgorithmus 33](#_Toc5620166)

[10.4.2 Sobelalgorithmus 34](#_Toc5620167)

[10.4.3 Pixel Vergleich 35](#_Toc5620168)

[10.5 Segmentierung 36](#_Toc5620169)

[10.6 .NET Framework 36](#_Toc5620170)

[10.6.1 Sytem.Drawing 37](#_Toc5620171)

[10.6.2 Bitmap 37](#_Toc5620172)

[10.7 EmguCV 38](#_Toc5620173)

[10.7.1 EmguCV verwenden 39](#_Toc5620174)

[10.7.2 Emgu.CV.ML 39](#_Toc5620175)

[10.7.3 Emgu.CV.UI 39](#_Toc5620176)

[10.7.4 Emgu.CV.GPU 40](#_Toc5620177)

[10.7.5 Images 40](#_Toc5620178)

[10.7.6 Automatic Garbage Collection 41](#_Toc5620179)

[10.7.7 CvInvoke 41](#_Toc5620180)

[10.7.8 Mat/Map 42](#_Toc5620181)

[11 Aufbau der Software 43](#_Toc5620182)

[11.1 Erstellen des Projektes 43](#_Toc5620183)

[11.2 Konzept überlegen 44](#_Toc5620184)

[11.2.1 Erstes Konzept 44](#_Toc5620185)

[11.2.2 Zweites Konzept 45](#_Toc5620186)

[11.3 Umsetzung der Funktionen 46](#_Toc5620187)

[11.3.1 CompImage 47](#_Toc5620188)

[11.3.2 KordComp 47](#_Toc5620189)

[11.3.3 CompIcon 48](#_Toc5620190)

[11.3.4 SearchIcon 49](#_Toc5620191)

[11.3.5 GetGrayImage 50](#_Toc5620192)

[11.3.6 GetLuminanz 51](#_Toc5620193)

[11.4 Testen der Funktionen 52](#_Toc5620194)

# Danksagung

An dieser Stelle bedanken wir uns bei allen, die diese Diplomarbeit unterstützt haben. Zunächst bei unserem Projektbetreuer Prof. Dipl.-Ing. Siegbert Schrempf, der uns nicht nur mit seiner Expertise unterstützt, sondern auch immer wieder motiviert hat.

Für die Hilfe bei der Erstellung der Projektpräsentation, bedanken wir uns außerdem bei Prof. Mag. Paul Schwaiger, der mit seiner Kritik geholfen hat, die Präsentation weiter zu verbessern.

Eine große Hilfe waren auch die Softwarelehrer der Schule, die uns durch ihre Erfahrung und Wissen unterstützt haben. Hier bedanken wir uns besonders bei:

* Dipl.-Ing. Roman Schragl
* Prof. Dipl.-Ing Gerfried Susani-Etzerodt

# Einleitung

Das vorliegende Projekt ermöglicht die Verwendung einer E-Gitarre als MIDI-Device. Das Gerät soll einzelne Noten und Akkorde zuverlässig mit möglichst geringer Latenz in das MIDI-Format umwandeln. Die MIDI-Signale werden an die USB-Schnittstelle eines PCs übertragen, von welchem die Signale beliebig interpretiert werden können.

# Aufgabenstellung

# Implantmed

Das Gerät wird uns von der Firma W&H Dentalwerk Bürmoos zur Verfügung gestellt. Es ist eines der zurzeit neusten Geräte, die derzeit von der Firma in Entwicklung sind. Mit dem digitalen Display und den automatischen Funktionen soll es das Setzen von Implantaten und andere Aufgaben eines Zahnarzts erleichtern. Bei dem Gerät sind ein Bohrer sowie eine Messsonde inbegriffen. Die Messsonde kann feststellen ob ein Implantat richtig gesetzt wurde oder ob es sie noch zu locker sitzt. Dies reduziert den Arbeitsaufwand für den Arzt und das Problemrisiko für den Patienten, da dieser nicht nachträglich zum Festziehen des Implantats wieder in die Klinik muss. Das Gerät wurde bezüglich der Programmierung extra neu verkabelt, um uns einen Zugriff auf die Daten zu geben. Mit einer Ethernet- und USB-Schnittstelle wurde dies verwirklicht.

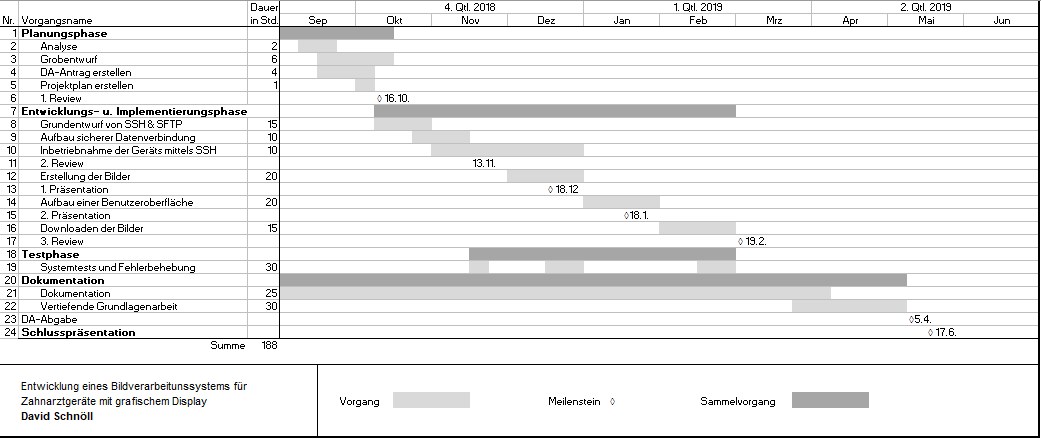
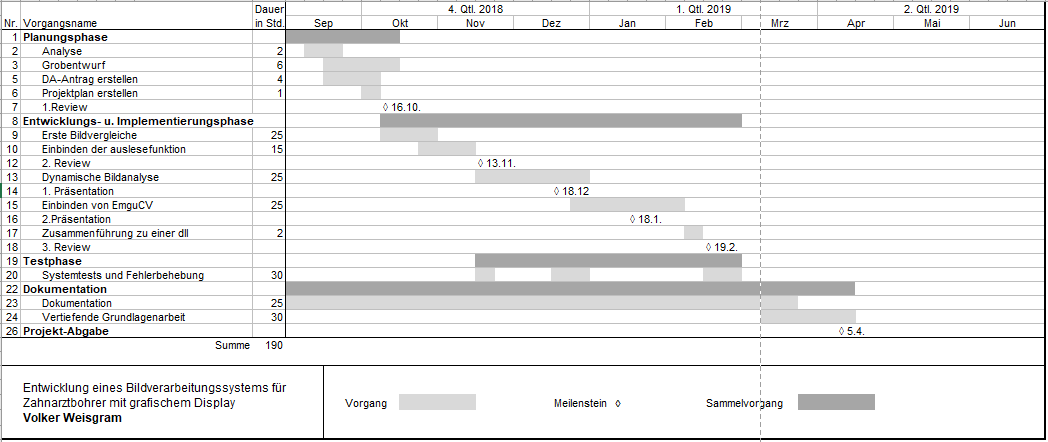
# Individuelle Zielsetzung und Terminplan

Die verschiedenen Aufgabenstellungen wurden auf uns verteilt. Natürlich haben wir uns auch gegenseitig unterstützt, womit die Arbeit gleichmäßiger verteilt wurde.

1. **Daniel Bräumann**



1. **Simon Grundner**
2. **Laurenz Hölzl**

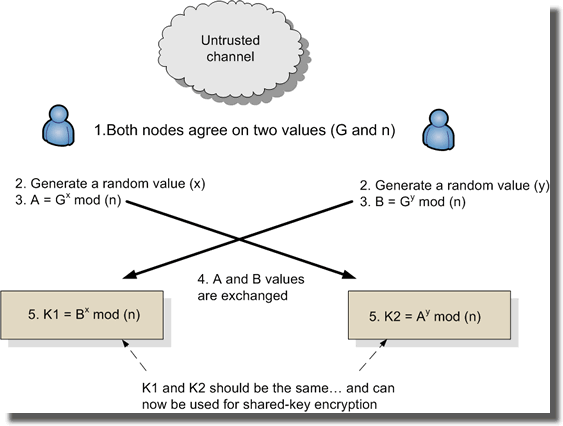


# Renci SSH

Renci SSH.Net ist eine Library in Visual Studio die dafür verwendet wird um Programme mit SSH und SFTP zu erstellen. Die Library ist frei zur Verfügung ohne zusätzlliche Programme oder Libaries zu verwenden. Es werden verschieden Methoden verwendet um die zu übertragenen Daten zu verschlüsseln. Eine der bekanntesten ist der „Diffie-Hellman key exchange“

## Diffie-Hellman key exchange

Der Diffie-Hellman key exchange ist eine sichere Methode um über öffentliche Channels einen Schlüssel auszutauschen ohne das ein Mithörer rausfinden kann was der Key ist. Nur die zwei am Austausch beteiligten Benutzer wissen am Ende was der Key ist.



# SSH (Secure Shell)

Die Secure Shell ist ein verschlüsseltes Netzwerk Protokoll. Dieses Protokoll soll ermöglichen, dass man eine verschlüsselte Übertragung über ein unverschlüsseltes und nicht sicheres Netzwerk übertragen kann[[1]](#footnote-1). Der Standard Port, der vom Protokoll verwendet wird, ist der Port Nummer 22. Verwendet wird die SSH hauptsächlich in UNIX System. Jedoch kann man sie, wie wir, genauso auch in Windows betreiben. Ursprünglich wurde die SSH entwickelt, um unsichere Shell Protokolle wie Telnet[[2]](#footnote-2) zu ersetzen. Diese Protokolle senden ihre Daten, wie Passwörter und andere wichtige Informationen, ohne Verschlüsselung und waren daher äußerst anfällig. Ein simples Programm wie „packet analyzer“ reichte aus, um Passwörter abzufangen.

## Verbindungsherstellung Version 1

Es gibt mehrere Varianten, um meine Verbindung zu einem Server herzustellen. Die erste wäre manuell ein „public-private key“ Paar zu generieren. In dieser Version müssen die verwendeten Keys auf jedem verwendeten Client installiert sein. Bei dieser Version wird nie der „private key“ übertragen und kann somit auch nicht abgefangen werden. Die SSH überprüft lediglich ob die „keys“ übereinstimmen. Es wird für die Verbindung kein Passwort verwendet. Es wird empfohlen jedem „public key“ eine Benutzer-Identität zuzuweisen, um mögliche Angreifer zu erkennen und den Zugriff zu erschweren.

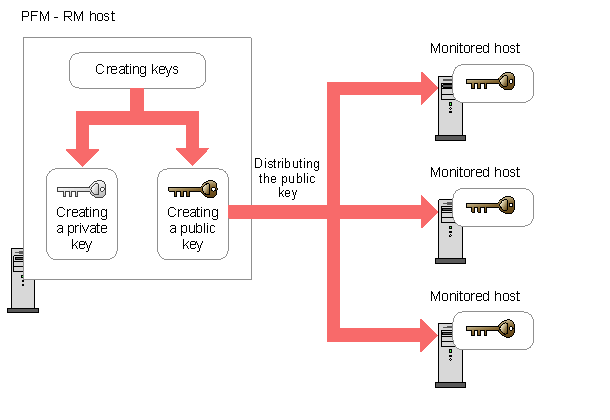


Abbildung 1 <http://itdoc.hitachi.co.jp/manuals/3020/30203R3900e/JRMP0035.HTM> Hitachi Ltd. Zugriff 26.03.19

## Verbindungsherstellung Version 2

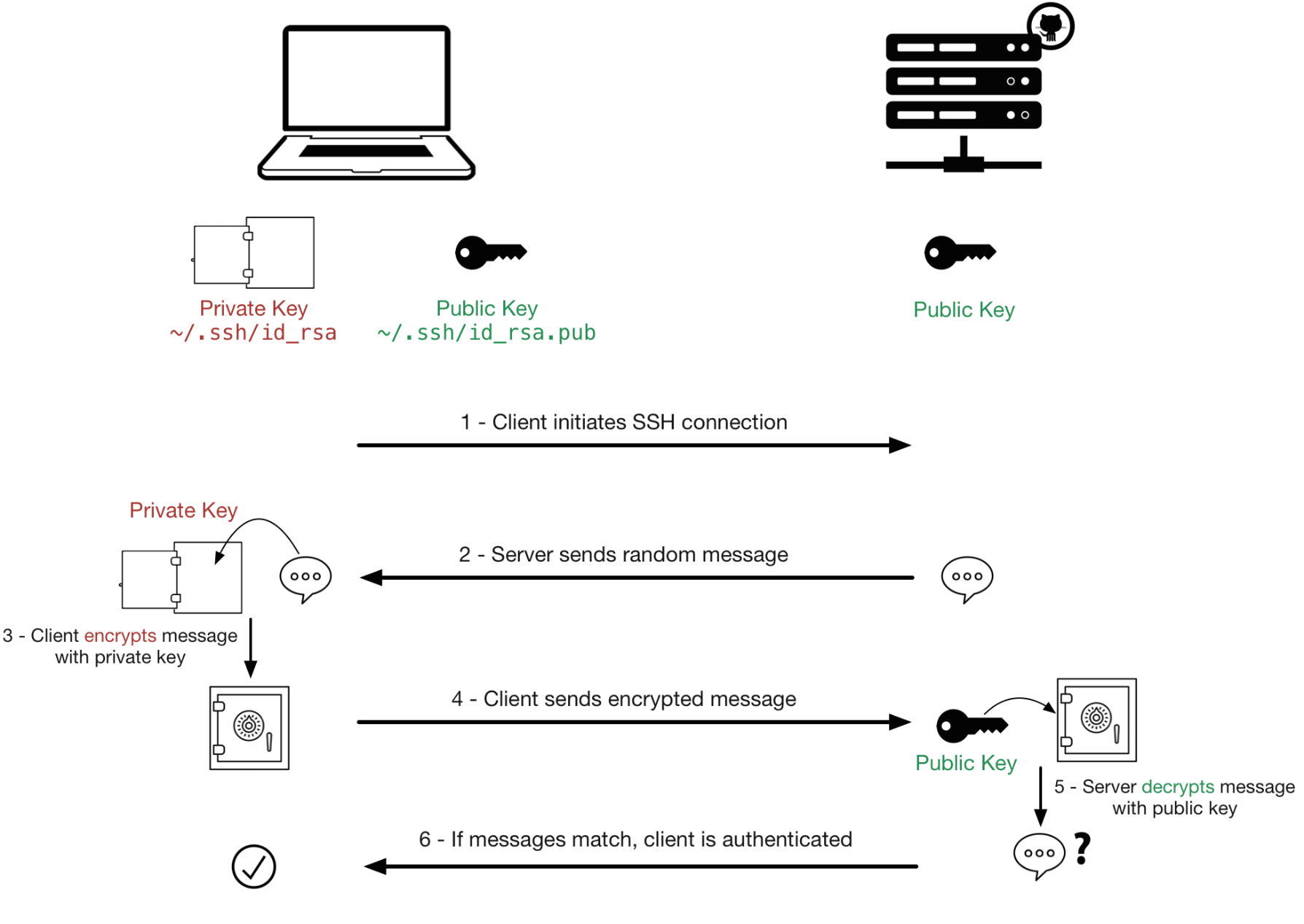
Die zweite Variante, die wir verwenden, ist das „public-private key“ Paar automatisch generieren zu lassen. Damit wird die Verbindung verschlüsselt und man benötigt ein Passwort, um dem Netzwerk beitreten zu können. Bei dieser Variante ist man anfällig gegen eine „man-in-the-middle-attack“[[3]](#footnote-3), da der Angreifer den Server imitieren könnte und damit an das Passwort gelangt. Dies ist jedoch nur möglich, wenn beide Seiten sich noch nie „getroffen“ haben. Die SSH merkt sich die Keys, mit denen sie bereits eine Verbindung hergestellt hat, und vergleicht diese mit dem bereits vorhandenen.

Abbildung 2 <https://sebastien.saunier.me/blog/2015/05/10/github-public-key-authentication.html> Sebastian Sauner, Zugriff 26.03.19

### Implementierung in das Projekt

Implementiert wurde SSH über das Visual Studio Nugget „SSH“ von Renci. Das Nugget Packet ist als Opensource-Library öffentlich zugänglich. Mit der Variante, die wir verwenden, hat das Gerät einen fest vorgeschriebenen Key. In diesem Fall ein Passwort und einen Username. Über eine Funktion werden diese übertragen und verwertet.

public bool Connect(string ip, int port, string un, string pw)

{

sshClientCon = new SshClient(ip, port, un, pw);

sshClientCon.ConnectionInfo.Timeout = ConnectionTimeout;

sshClientCon.KeepAliveInterval = KeepAlive;

Console.WriteLine("Trying to connect...");

try

{

sshClientCon.Connect();

}

catch (Exception e1)

{

MessageBox.Show(e1.Message, "Error detected", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

return true;

}

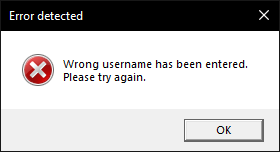
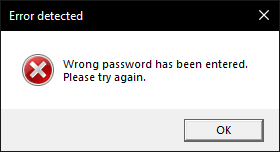
Codeauschnitt 1

#### Connect

Die Connect Funktion bringt das komplette Programm zum Laufen. Sie initalisiert den Ablauf. Über eine App.config oder einer Benutzeroberfläche werden die Daten wie Passwort und Username eingegeben. Diese Daten werden dann über SSH an das Gerät übertragen und verifziert. Sollte das Gerät die erkannten Daten akzeptieren und deren Richtigkeit bestätigen, wird eine feste Verbindung hergestellt. Der Ablauf verhält sich wie in dem bereits vorher erwähnten Diagram.

#### Connection Timeout

Der ConnectionTimeout sorgt dafür, dass nach einer gewissen Zeit ohne erfolgreich aufgebaute Verbindung der Thread abgebrochen wird und nicht in eine Endlosschleife verfällt. Er wird jedoch nur aufgerufen, wenn der Server nicht zu erreichen ist. Wenn der Server die Verbindung ablehnt, springt der Client in eine Exception, die dann vom Benutzer verarbeitet werden muss. In unserem Fall wird eine MessageBox aufgerufen, welche die Art des Fehlers beschreibt. Die Fehlernachricht beschreibt ob der User, die IP oder das Passwort nicht stimmt.

### Erstellen der Screenshots zur Datenverarbeitung

Um sicherzustellen, dass vom voherigen Test keine Bilder zurückgeblieben sind, wird das Verzeichniss von jeglichen Bildern gereinigt, um keine falschen Ergebnisse zu erzeugen. Diese Funktion „Clear“ führt einen Linux Befehl aus welcher alle Dateien mit dem gewählten Dateinamen aus dem Verzeichniss entfernt.

public void Clear()

{

SshCommand cmd = sshClientCon.RunCommand("rm /data/test\*.png");

}

Codeauschnitt 2

Die Funktion „Capscreen“ erstellt dann die Bilder in einem zuvor gewählten lokalen Verzeichniss mit einem vorgegebenen Namen. Dieser Name ist frei vom Benutzer in der App.Config defninierbar.

public void Capscreen(int nr)

{

SshCommand cmd = sshClientCon.RunCommand($"/usr/local/bin/screenshot /data/test{nr}.png");

}

Codeauschnitt 3

Nachdem der SSH Client seine Aufgabe erledigt hat und nicht mehr benötigt wird, wird er vom Benutzer abgemeldet. Realisiert wird dies durch eine kurze Funktion. Der Benutzer wird nach einer erfolgreichen Abmeldung informiert. Diese Funktion kann eine Fehlermeldung aufrufen, wenn jemand versucht sich abzumelden ohne jemals eine Verbindung hergestellt zu haben.

public void DisconnectSSH()

{

if (sshClientCon.IsConnected == true)

{

sshClientCon.Disconnect();

Console.WriteLine("SFTP has been disconnected");

}

else

{

Console.WriteLine("Client is not connected");

}

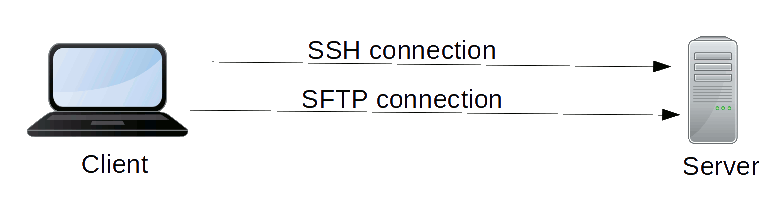
}

Codeauschnitt 4

Dies sind kurz zusammengefasst die hauptsächlichen Funktionen die für die Verbindung und die Erstellung der Bilder verantwortlich sind. Da nun die Bilder vorhanden sind können sie über SFTP an unseren Computer übertragen werden, um sie dort dann mit Referenzbilder zu vergleichen.

# SFTP (SSH File Transfer Protocol)

Das SFTP ist ein Netzwerk Protokoll, welches für eine sichere Daten-Übertragung, Zugriff und Daten Management sorgt. Es wurde als Erweiterung für das SSH (Secure Shell Protocol) entwickelt. An sich geht das Protokoll immer davon aus, dass es über einen sicheren Channel läuft und der Server, auf den zugegriffen wird, bereits den Client authentifiziert hat. Es wurde entwickelt um SCP zu ersetzen, da SCP nur einen File Transfer erlaubt im Gegensatz zum SFTP. Das SFTP ist jedoch, wie viele glauben, nicht einfach FTP über SSH, sondern ein von Grund auf neu entwickelt Protokoll der IETF Gruppe. Ebenso verwendet SFTP keine eigenen Sicherheits Maßnahmen sonder geht davon aus, dass die unterliegenden Protokoll sich darum kümmern. Bei unserem Projekt verwenden wir SFTP lediglich um die gewünschten Daten an unseren Computer zu übertragen, damit diese dann verwertet werden können.



public bool Connect(string ip, int port, string un, string pw)

{

sshClientCon = new SftpClient(ip, port, un, pw);

sshClientCon.ConnectionInfo.Timeout = ConnectionTimeout;

sshClientCon.KeepAliveInterval = KeepAlive;

Console.WriteLine("Trying to connect...");

try

{

sshClientCon.Connect();

}

catch (Exception e1)

{

MessageBox.Show(e1.Message, "Error detected", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

return true;

}

Codeauschnitt 5

Für die Verbindung des SFTP Client wird eine nahezu identische Funktion wie bei der SSH verwendet. Die Funktionen führen ebenso zu den gleichen Ergebnissen.

## Datentransfer

Um die erstellten Bilder auf den Computer zu laden, müssen diese auf dem Gerät gefunden werden. Mit einer ChangeDirectory Funktion gelangen wir an den Speicherort der erstellten Bilder. Zuerst bewegen wir uns zum Ursprungspunkt des Pfads, von welchem aus wir dann an unseren Speicherort gelangen. Von dem Speicherort werden die Bilder an den Computer gesendet.

public void ChangeDir()

{

clientSFTP.ChangeDirectory("..");

clientSFTP.ChangeDirectory("..");

clientSFTP.ChangeDirectory("data");

}

Codeauschnitt 6

Mit der Funktion Fetch laden wir die Bilder auf den Computer und können sie nach Belieben bennenen. Der Speicherort wird über die App.Config oder die Benutzerfläche ausgewählt. Falls dieser Pfad nicht existiert, wird die Funktion abgebrochen und man muss einen neuen Pfad auswählen. Dies passiert solange bis ein existierender Pfad gewählt wird. Der Filestream legt den neuen Namen und den Speicheort fest.

public void Fetch(string path, string imgname, int co)

{

if (!Directory.Exists(path))

{

throw new DirectoryNotFoundException("The selected Path is not available");

}

using (Stream fileStream = File.Create(Path.Combine(path, imgname)))

{

clientSFTP.DownloadFile("/data/test" + co + ".png", fileStream);

}

}

Codeauschnitt 7

## Bild Rotation

Um die Bilder mit den Referenzbildern zu vergleichen müssen diese vorher um 180° gedreht werden. Die Bilder werden am Anfang um 180° verdreht erstellt. Um die Bilder zu drehen werden, sie zuerst in eine Bitmap umgewandelt und dann um 180° gedreht. Nachdem die Bilder gedreht wurden, werden sie wieder in ein PNG umgewandelt und weiterverarbeitet.

public Image RotateImage(Image img)

{

var bmp = new Bitmap(img);

using (Graphics gfx = Graphics.FromImage(bmp))

{

gfx.Clear(Color.White);

gfx.DrawImage(img, 0, 0, img.Width, img.Height);

}

bmp.RotateFlip(RotateFlipType.Rotate270FlipNone);

return bmp;

}

Codeauschnitt 8

Um die Verbindung zwischen SFTP und dem Computer zu beenden wird dieser einfach wieder durch die selbe Funktion wie bei der SSH abgemeldet.

public void DisconnectSFTP()

{

if (clientSFTP.IsConnected == true)

{

clientSFTP.Disconnect();

Console.WriteLine("SFTP has been disconnected");

}

else

{

Console.WriteLine("Client is not connected");

}

}

Codeauschnitt 9

# Benutzeroberfläche

Um das gesamte Projekt zu visualisieren, haben wir in Visual Studio eine Benutzeroberfläche (GUI) entwickelt. Die GUI ist in WPF (Windows Form Presentation) entwickelt worden. Wir haben sie simple und übersichtlich aufgebaut, um eine fehlerfreie Eingabe zu ermöglichen.

## Layout

Das komplette Layout wurde in mehrere Abschnitte unterteilt. Zum einem hat man die Eingabe der Verbindungsdetails, das Passwort, die IP-Adresse und den Username welche mit dem des Servers übereinstimme müssen. Ansonsten wird keine Verbindung hergestellt. Nach einer fehlerhaften Eingabe muss das Passwort einfach wieder neu eingegeben werden. Sollte die Benutzerdaten stimmen, wird die Verbindung hergestellt und die Buttons für „Screenshot“ und „Download“ werden freigeschalten. Von dieser Oberfläche aus muss man ebenso den Speicherort bestimmen. Sollte keiner ausgewählt werden, lassen sich die Bilder nicht herunterladen.

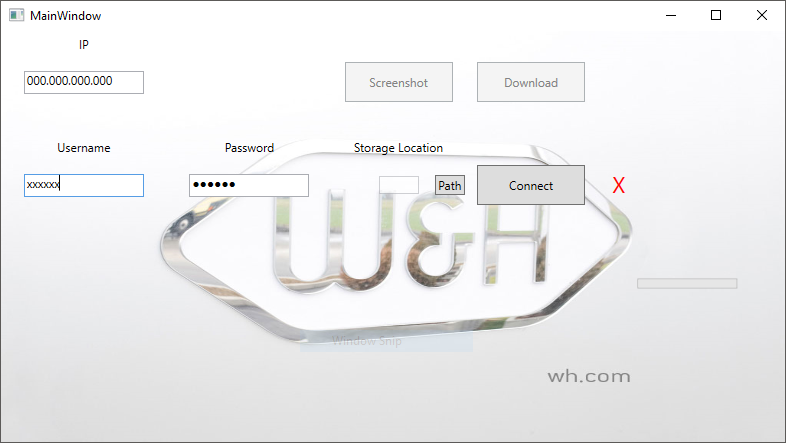


Abbildung 3 Ausschnitt aus der GUI

Um beim Aufbau oder Einfügen neuer Eigenschaften die GUI grafisch nicht zu verändern, wurde sie in einem Grid aufgebaut. Dadurch befindet sich jede Funktion in ihrem dafür festgelegten Abschnitt. Somit wird sichergestellt, dass kein Button sich durch Hinzufügen eines neuen Buttons ungewollt verschiebt. Das Grid wird durch Definitionen der Reihen und Spalten festgelegt. Die Größe der Reihe wird durch einen Koeffizienten festgelegt. Dieser multipliziert sich selbst mit den vorhandenen Reihen. Somit kann man genau definieren wie groß jede Reihe im Verhältnis zu den anderen sein soll.

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150\*" />

<ColumnDefinition Width="150\*" />

<ColumnDefinition Width="120\*" />

<ColumnDefinition Width="120\*" />

<ColumnDefinition Width="170\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

</Grid.RowDefinitions>

Codeauschnitt 10

Für eine direkte Übergabe der eingegebenen Benutzerdaten, werden Data-Bindings verwendet. Data-Bindings verbinden den Anbieter der Daten mit dem Verwender und synchronisiert die vorhandenen Daten. Dies wird normalerweise verwendet, um zwei Datensätze aus unterschiedlichen Programmiersprachen zu verbinden. In unserem Fall werden die Daten einer XAML-Datei mit einer C# CS-Datei verbunden. Abhängig von den Anforderungen, kann das Binding „OneWay“ oder „TwoWay“ festgelegt werden. Da beim Speicherort die Textbox und der Pfad geändert werden, wird dieser als „TwoWay“ festgelegt.

{Binding Path=DirectoryPath, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=Window},Mode=TwoWay}

Codeauschnitt 11

## Connect

Der Verbindungsaufbau wird über einen Switch-Button aufgebaut. Sollte dieser betätigt werden, wird die Funktion aufgerufen und es wird überprüft, ob alle Benutzerdaten vorhanden sind. Sind alle vorhanden und die grundlegende Struktur stimmt (IP-Adresse besteht aus Zahlen) wird der Verbindungsaufbau gestartet. Der Aufbau wird an einen neuen Thread abgegeben, um die Benutzeroberfläche nicht zu beeinträchtigen.

private void HandleCheck(object sender, RoutedEventArgs e)

{

if (textbox\_ip.Text.Trim() == string.Empty ||

textbox\_user.Text.Trim() == string.Empty)

{

MessageBox.Show("Please enter something in the textbox");

return;

}

else

{

Cursor = Cursors.AppStarting;

t1 = new Thread(MyThread);

t1.Start(textbox\_ip.Text +";"+ textbox\_user.Text + ";" + textbox\_pw.Password);

if (isConnected == true)

{

btn\_con.Content = "Disconnect";

}

}

}

Codeauschnitt 12

### Aufbau des Threads

Der Verbindungsaufbau wird in den neuen Thread übergeben, welcher sich im Hintergrund befindet, und durch einem Ladezeichen beim Mauszeiger signalisiert wird. Das Ladezeichen ist notwendig, dass der Benutzer erfährt, wann die Verbindung stabil ist.

void MyThread(object input)

{

string str = (string)input;

Console.WriteLine(str.Split(';')[0]);

if (capt.Connect(str.Split(';')[0], 22, str.Split(';')[1], str.Split(';')[2]) == false)

{

this.Dispatcher.Invoke(() =>

{

btn\_con.IsChecked = false;

Cursor = Cursors.Arrow;

}

);

t1.Abort();

}//connecting to ssh

if(fetch.Connect(str.Split(';')[0], 22, str.Split(';')[1], str.Split(';')[2]) == false)

{

this.Dispatcher.Invoke(() =>

{

btn\_con.IsChecked = false;

Cursor = Cursors.Arrow;

}

);

t1.Abort();

}//connecting to sftp

IsConnected = true;

this.Dispatcher.Invoke(() =>

{

Img\_con.Visibility = Visibility.Visible;

Cursor = Cursors.Arrow;

}

);

}

Codeauschnitt 13

## Screenshot

Die Funktion zum Erstellen der Screenshots wird abhängig von der gewünschten Anzahl n-mal aufgerufen. Die Anzahl wird in der App.Config festgelegt. Nachdem die Anzahl bestimmt wurde, wird die vorhandene Funktion zum Erstellen aufgerufen. Da das Erstellen eines Screenshots eine gewisse Zeit in Anspruch nimmt, haben wir eine Verzögerung eingebaut, die verhindern soll, dass zu viele Bilder zur selben Zeit erstellt werden.

private void Btn\_cap\_click(object sender, RoutedEventArgs e)

{

capt.Clear();

for (int i = 1; i <= pics; i++)

{

capt.Capscreen(i);

Thread.Sleep(200);

}

}

Codeauschnitt 14

## Download

Nach dem Erstellen der Screenshots, wird die Download-Funktion freigeschalten. Diese überprüft vorerst, ob der vorliegende Speicherort vorhanden ist oder nicht. Sollte diese vorhanden sein, werden die Bilder in diesem abgespeichert. Der Name ist dem Benutzer frei überlassen und kann ebenso in der App.Config geändert werden.

private void Btn\_load\_click(object sender, RoutedEventArgs e)

{

fetch.ChangeDir();

try

{

for (int y = 1; y <= pics; y++)

{

fetch.Fetch(DirectoryPath, $"test{y}.png", y);

}

MessageBox.Show("All files have been downloaded", "Success", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Download failed", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

Codeauschnitt 15

# Theoretische Grundlagen der Bildverarbeitung

In den folgenden Kapiteln wird die Bildverarbeitung theoretisch abgeabreitet. Dies haben wir, verknüpft mit dem Wissen, welches im Informatik-Unterricht vermittelt wurde, verwendet, um die Software praktisch umzusetzen.

## Einleitung

Bildverarbeitung ist in den meisten Bereichen der Wirtschaft bereits weit verbreitet und spielt immer mehr eine wichtige Rolle. Hier wird sie meist in großen Fertigungsanlagen verwendet, um autonom arbeitende Maschinen zu steuern. Ein weiterer Bereich, in dem die Bildverarbeitung bereits oft verwendet wird, ist die Qualitätssicherung, bei der Stichproben optisch überprüft werden. Aber auch in der Medizin hat sie bereits Einsatzgebiete. Hier werden meist Ärzte unterstützt bessere Diagnosen zu erstellen oder um Röntgenbilder einfacher zu verstehen. Die Wissenschaft nutzt natürlich auch Bildverarbeitung immer häufiger. Ein Beispiel wäre hier die Astronomie, um Planeten zu finden oder frühzeitig Meteoriten zu finden, welche der Erde gefährlich werden können.

## Definition

*„Bildverarbeitung umfasst generell alle Themengebiete,  
 die sich mit der Erzeugung, Transformation, Auswertung und Darstellung von Bildern oder Bildinhalten befassen.“  
(Ralf Bruder, 2002)*

Nach dieser Definition kann man sagen, dass Bildverarbeitung viel mehr ist als man zuerst vermutet, da es auch die Erstellung von Bildern und deren Ausgabe beinhaltet. Das zeigt auch, dass Bildverarbeitung nicht nur in der Wirtschaft von großem Nutzen ist, sondern von jeder Person verwendet wird, wenn man nur ein einfaches Foto macht oder wenn man Bilder ausdruckt.

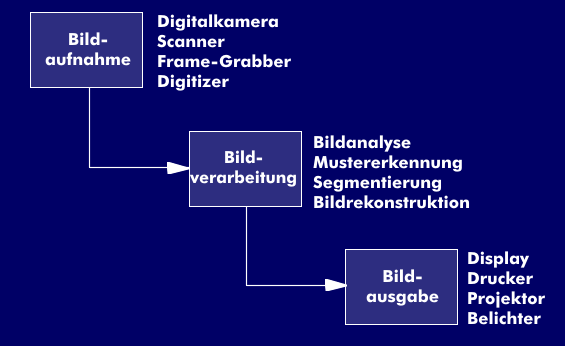


Abbildung 1: Bildverarbeitung

Häufig denkt man jedoch bei dem Begriff „Bildverarbeitung“ nur an die Bildanalyse oder die Erkennung von Mustern, Gegenständen oder auch Personen und Tieren in Bildern.

## Industrielle Bildverarbeitung

Die industrielle Bildverarbeitung wird oft in automatisierten Fertigungsanlagen angewendet. Hierbei wird oft eine Mischung aus Sensoren und Kameras verwendet. Es muss aber in der Industrie nicht zwingend eine Kamera eingesetzt werden. Bei der Qualitätssicherung von Software, die mit graphischen Oberflächen arbeitet, kann zum Beispiel kontrolliert werden, ob es bei Softwareupdates oder schon in der Entwicklung zu Fehler gekommen ist und dadurch etwas falsches am Bildschirm angezeigt wird. Bei dieser Form der Bildverarbeitung wird ein Screenshot gemacht und dieser dann mit einem Referenzbild verglichen. Es kann aber auch überprüft werden ob bestimmte Merkmale am Bildschirm zu sehen sind.

## Bildverarbeitungsalgorithmen

Es gibt in der Bildverarbeitung viele verschiedene Algorithmen, die für verschiedene Zwecke entwickelt wurden und dort auch angewendet werden. Hierbei gibt es die Möglichkeiten einfach nur Pixel für Pixel zu vergleichen oder aber auch über Kantenerkennung Abstände zu messen.

### Cannyalgorithmus

Der Cannyalgorithmus ist ein Algorithmus, der für „Edge-Detection“, oder Kantendetektion, verwendet wird. Dieser wurde von John Canny im Jahr 1986 veröffentlicht. Er ist ein weitverbreiteter und oft eingesetzter Algorithmus und hat als Ziel die Kanten in einem Bild zu erkennen. Hierbei sollten möglichst alle Kanten erkannt und angezeigt werden, aber keine falschen Kanten. Zusätzlich sollen diese auch noch an der richtigen Position erkannt werden. Daher eignet sich dieser auch besonders gut, wenn man auf Bildern die Breite von Gegenständen messen möchte, die man mit der Hand nicht so leicht messen könnte, ohne dass man das Messergebnis stark beeinflussen würde.



Abbildung 4

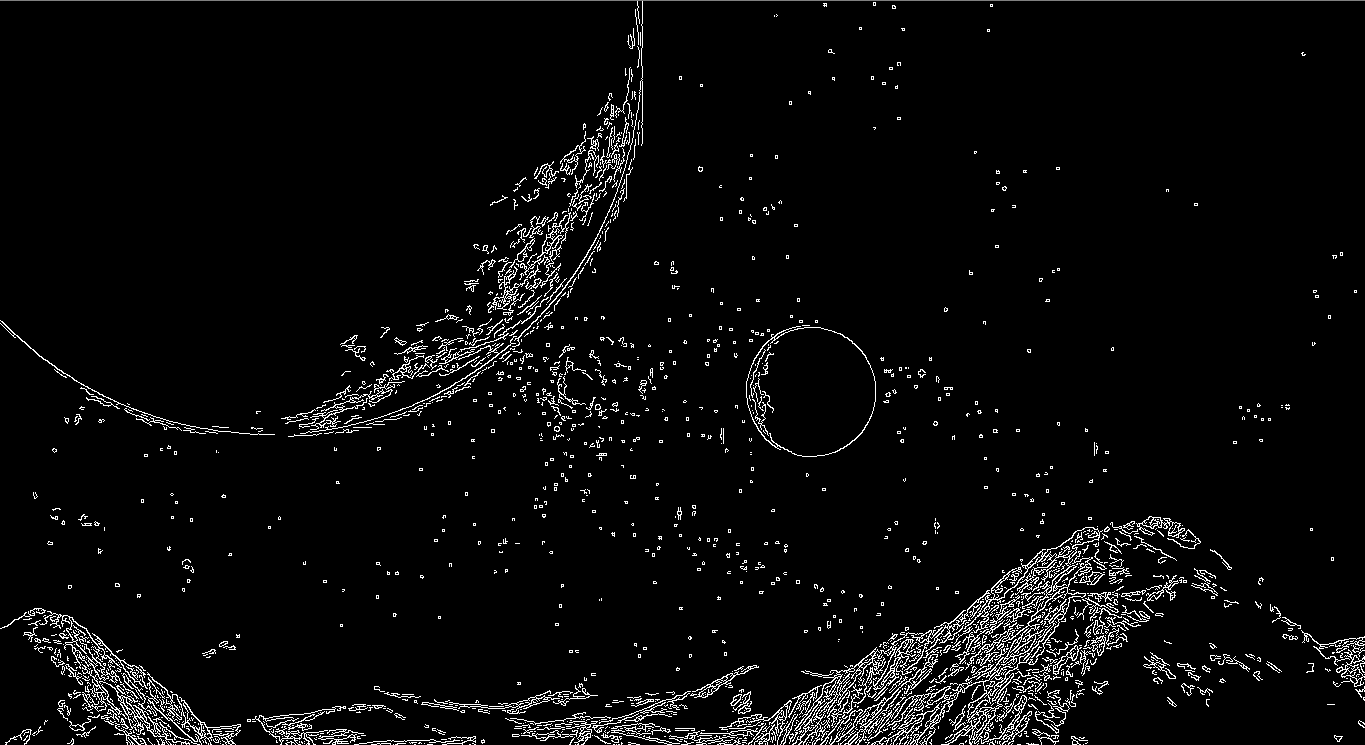


Abbildung 5

Auf Abbildung 4 wurde dieser Algorithmus angewendet. Danach wurde dem Benutzer das vom Programm verarbeitete Bild wieder ausgegeben, welches man in Abbildung 5 sehen kann.

Um zu diesem Ergebnis zu kommen, muss man sich vorerst um das Rauschen kümmern, da die Kantendetektion besonders anfällig hierbei reagiert. Um dies zu bewerkstelligen wird ein Gauß-Filter angewendet. Dieser muss dann je nach Situation angepasst werden. Als nächstes wird eine Art Ableitung über das gesamte Bild gemacht, wobei man hier einen sogenannten Sobel-Operator verwenden kann. Zum Schluss stellt man mit sogenanntem Threshholding ein, wie der Algorithmus die kannten erkennt. Dabei ist wichtig zu wissen, dass je höher der Wert, den man für Threshholding verwendet, die Kanten eindeutiger sein müssen.

### Sobelalgorithmus

Der Sobelalgorithmus ist auch, wie der Cannyalgorithmus, ein Kantendetektionsalgorithmus. Dieser wird auch oft als Sobel-Filter bezeichnet und er findet auch häufigen Gebrauch in der Industrie. Ebenso wie der Cannyalhorithmus hat der Sobel-Filter das Ziel, die Kanten in einem Bild so gut wie möglich und an den richtigen Stellen zu finden. Dieser ist jedoch nicht immer die beste Wahl, da er ein sehr einfacher Filter ist.

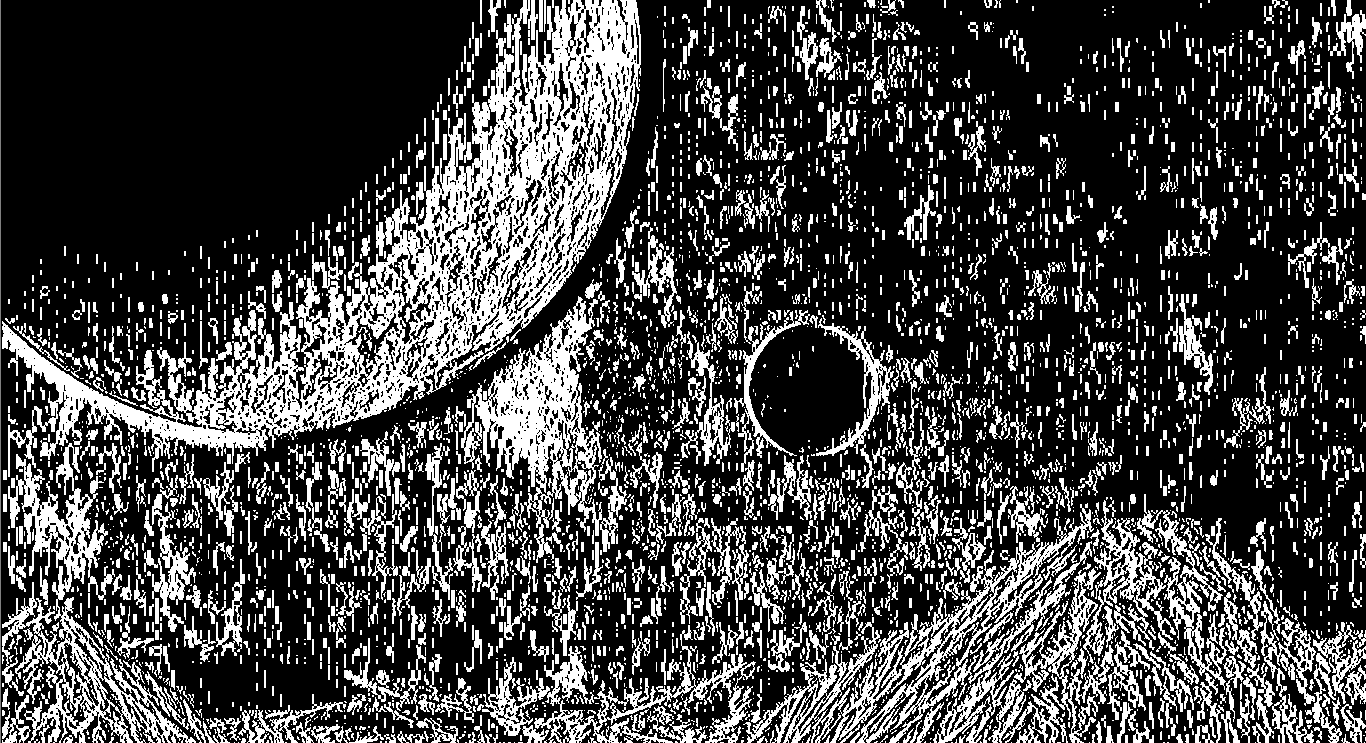


Abbildung 6

Das in Abbildung 6 gezeigte Bild ist das Bild aus Abbildung 4, bei dem der Sobel-Filter angewendet wurde. Hier kann man bereits erkennen, dass der Sobel-Filter nicht immer die beste Wahl ist, da er auch bei leichten Farbtonveränderungen bereits Kanten erkennt. Hierbei werden mithilfe von 3x3-Matrixen eine Faltung des Bildesdurchgeführt, um so die Kannten in einem Bild auf sehr einfache Weise zu erkennen.

### Pixel Vergleich

Einer der einfachsten Vergleichsalgorithmen ist der Vergleich von Pixeln. Hierbei wird von einem Bild jeder Pixel mit einem Pixel auf einem Referenzbild überprüft. Ein Nachteil dieser Variante des Bildvergleichs ist jedoch, dass je größer ein Bild ist, die benötigte Rechenleistung stark ansteigt, wodurch das Programm an Geschwindigkeit verliert. Jedoch kann dieser Algorithmus leicht implementiert werden, da er nur aus zwei einfachen „for-Schleifen“ besteht. Hierbei kann man auch bereits einige Verbesserungen machen, um die benötigte Rechenleistung zu verringern und die Performanz des Programmes zu steigern.

for (int i = 0; i<image.Width; i++)

{

for (int j = 0; j<image.Height; j++)

{

imagePixel = image.GetPixel(i, j).ToString();

refPixel = refImage.GetPixel(i, j).ToString();

if (imagePixel != refPixel)

{

help = false;

break;

}

}

}

Codeauschnitt 16

Es werden dann einfach die Farbwerte der Pixel verglichen und am Ende wird dem Benutzer mitgeteilt, ob die Bilder gleich sind oder nicht.

## Segmentierung

Das Ziel der Segmentierung ist es das Bild in relevante und nicht relevante Regionen zu unterteilen, beziehungsweise das Bild in verschiedene Segmente zu trennen. Hierbei wird das Bild oft vorher zu einem Bild mit Graustufen umgewandelt, da diese Informationen einfacher verarbeitet werden können.

Für die Segmentierung gibt es einige Verfahren. Eines der einfachsten stellt hierbei das Schwellwertverfahren dar. Hier wird geschaut, ob ein Pixel einen bestimmten Schwellwert überschreitet oder nicht. Wenn dieser Schwellwert überschritten wird, wird dieser Pixel als ein Objekt gezählt, ansonsten wird er als Hintergrund gezählt. Dieses Verfahren kann man zum Beispiel dazu anwenden, um Objekte in einem Bild einfacher zu finden.

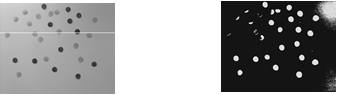


Abbildung 7 original Bild und Bild nach Segmentierung

## .NET Framework

Das .NET Framework stellt eines der wichtigsten Grundlagen in C# dar. In ihm finden sich viele wichtige Klassenbibliotheken, oder auch Libraries, die für den einfachen Bildvergleich bereits eine wichtige Rolle spielen. Aber nicht nur in der Bildverarbeitung wird dieses Framework verwendet, sondern auch in fast jedem C#-Programm. Es wurde von Microsoft entwickelt und veröffentlicht. Es sollte möglichst auf allen Plattformen funktionieren und wurde nach dem Vorbild von Java erstellt und nach den Wünschen von Microsoft angepasst, was auch zu einigen Rechtsproblemen zwischen Sun und Microsoft führte.

### Sytem.Drawing

Die Klassenbibliothek „System.Drawing“ ist ein Teil des .NET Frameworks und wird verwendet, um grundlegend mit Bildern zu arbeiten. In ihr finden sich einige Klassen, wie zum Beispiel die Bitmap Klasse, um die Daten von Bildern zu speichern oder aber auch um Zugriff auf die einzelnen Pixel in einem Bild zu erhalten. So kann man auch einzelne Farbwerte erfassen, um mit diesen dann zu arbeiten. Außerdem ist es mit den Klassen von System.Drawing möglich neue Bilder zu erstellen. Diese können dann auch als jpeg oder png, sowie auch als weitere gängige Bildformate abgespeichert werden.

### Bitmap

Bitmap ist eine Klasse, die für Bilder verwendet wird und sie kommt aus der Klassenbibliothek „System.Drawing“. Bei einem Bitmap werden die Daten von einem Pixel in einem Bild, also die drei Farbwerte Rot, Blau und Grün, sowie der Alphawert, welcher bestimmt ob etwas transparent ist oder nicht, in einem Array abgespeichert. Diese Informationen über einen Pixel können so in weiterer Folge gut verwendet werden, um Pixel untereinander zu vergleichen. Aber es gibt auch bereits Funktionen, welche den Vergleich zweier Bitmaps bereits übernehmen. Daher eignet sich diese Klasse besonders gut, um einen Pixelvergleich durchzuführen. Auch ist ein Bitmap recht einfach zu erstellen und es kann auch aus bereits bestehenden Bildern erzeugt werden, was die Verwendung von Bitmaps um einiges leichter macht.

Bitmap Image = new Bitmap(Image);

Codeauschnitt 17

Zusätzlich können die Daten von einem Pixel in einem Bild verändert werden oder auch neue Bilder mit der Bitmap Klasse erstellt werden. Das macht die Klasse auch vielseitig einsetzbar und dadurch eignet sie sich sehr gut, um gerade einfache Bildvergleiche durchzuführen.

## EmguCV

EmguCV ist einen von OpenCV abgeleitete Bildverarbeitungsbibliothek, welche mit allen Sprachen kompatibel ist, welche die .Net Klassenbibliothek verwenden können. Dies hat den Vorteil, dass die Funktionen von EmguCV auf fast allen Plattformen verwendet werden können. Ein weiterer Vorteil ist auch, dass Programme, die mit dieser Bibliothek arbeiten, in einigen Entwicklungsumgebungen kompiliert werden können. Funktionen und Klassen, die aus OpenCV kommen, können auch bei EmguCV verwendet werden, was die Kompatibilität der Programme fördert.

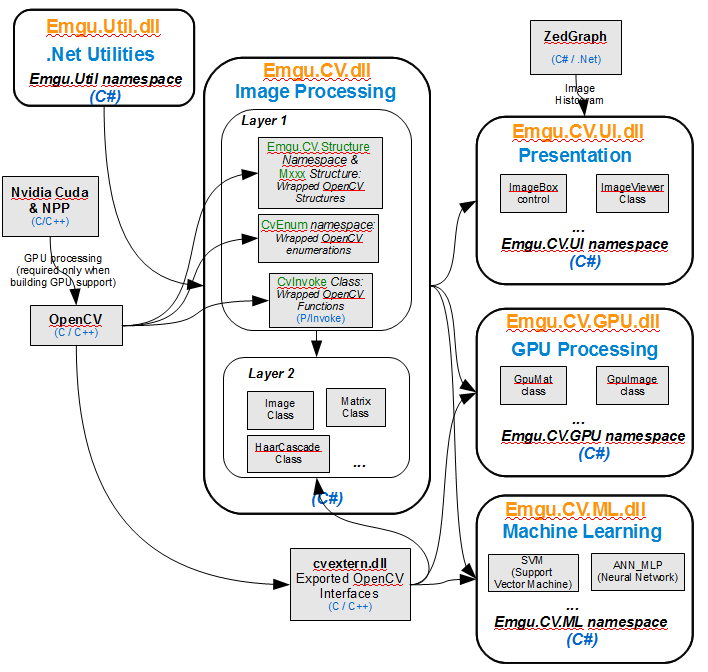


Abbildung 8 Aufbau von EmguCV

1 <http://www.emgu.com/wiki/index.php/Main_Page> (16.10.2018)

In Abbildung 8 kann man gut sehen, dass EmguCV aus 2 Layer besteht. Hierbei ist es wichtig zu wissen, dass im Layer 1 nur die Basisfunktionen enthalten sind, welche direkt von OpenCV abgeleitet wurden. Im Layer 2 sind nur Klassen, die auch mit .NET Klassen kompatibel sind, was ein großer Vorteil ist. Zusätzlich kann man in der Grafik gut erkennen, dass EmguCV einige Erweiterungen besitzt, um es noch besser in verschiedenen Bereichen der Bildverarbeitung zu verwenden.

### EmguCV verwenden

EmguCV ist nicht standardmäßig in Entwicklungsumgebungen inkludiert. Um es zu verwenden muss es zuerst heruntergeladen und installiert werden. Zusätzlich wird es dann noch in das Projekt, in dem EmguCV benötigt wird, eingebunden. Das wird von den meisten Entwicklungsumgebungen, wie zum Beispiel Visual Studio, unterstützt.

### Emgu.CV.ML

Emgu.CV.ML ist die Machine Learning Library von EmguCV und wurde von OpenCV abgeleitet. Machine Learning ist in der heutigen Zeit bereits ein sehr wichtiger und großer Bereich in der Bildverarbeitung. Hierbei bringt man seinem Programm bei, wie bestimmte Objekte aussehen. Dies wird häufig bei Überwachungskameras angewendet. Es kann aber auch in der Industrie dafür verwendet werden, um zu überprüfen, ob alle Bauteile richtig platziert wurden oder ob alles an seiner richtigen Stelle ist.

Aber es muss nicht unbedingt verwendet werden, um Dinge auf einem Bild zu finden. So kann man zum Beispiel auch dem Programm beibringen das Bild nach bestimmten Merkmalen in verschiedene Sektionen zu unterteilen. Ein weiterer Bereich in dem Machine Learning eingesetzt wird, ist die Entwicklung von Künstlichen Intelligenzen und deren neuronalen Netzwerken.

### Emgu.CV.UI

Emgu.CV.UI wird verwendet, um Benutzeroberflächen zu erstellen. Dadurch kann man auch dem Benutzer direkt mitteilen, was berechnet wird, ohne vorerst auf eine andere Library zurückgreifen zu müssen, um Fenster oder ähnliches am Bildschirm zu öffnen. Dadurch hat man den Vorteil, dass Bilder, die mit Funktionen und Klassen von EmguCV bearbeitet oder erstellt wurden, einfacher am Bildschirm wieder angezeigt werden können, da diese dann nicht erst wieder in einen anderen Datentyp, wie zum Beispiel Bitmap, oder ein anders Format umgewandelt werden müssen.

### Emgu.CV.GPU

Eine weitere Möglichkeit, die einem EmguCV gibt, ist es den Prozessor zu entlasten. Dafür werden einige Aufgaben an den GPU übertragen. Der GPU, Graphics Processing Unit, ist ein Prozessor, der auf die Berechnung von Pixel und Bildern spezialisiert wurde, das heißt er übernimmt alle Berechnungen, die mit Grafiken zu tun haben. In den heutigen Prozessoren ist bereits ein eigener GPU eingebaut, was das Nutzen dieser Funktionen von EmguCV deutlich vereinfacht. Auch werden so die Programme um einiges schneller, da nicht alle Berechnungen an die CPU fallen.

### Images

EmguCV hat auch den großen Vorteil, dass man direkt mit den Bildern arbeiten kann, da sie nicht erst in irgendeiner Form umgewandelt werden müssen. Dafür gibt es die Image Klasse, welche für Bilder verwendet wird. Wenn man jedoch mit einem Bitmap arbeitet, können diese einfach in ein Bild der Klasse Image konvertiert werden oder ein Image in ein Bitmap. Ein Nachteil bei der Image Klasse ist jedoch, dass bereits beim erstellen des Bildes bestimmt werden muss, welche Größe das Bild hat und ob es ein „RGB“ oder ein Graustufen-Bild ist. Wichtig ist hierbei nur zu wissen, das man zum einsetllen der Farbe entweder „Bgr“ für ein Farbbild oder „Gray “ für ein Graustufen Bild schreiben muss.

Image<Bgr, Byte> image = new Image<Bgr, Byte>("MyImage.jpg");

Codeauschnitt 18

Jedoch kann man dies auch gut ausnutzen, wenn man Farbbilder in Bilder mit Graustufen umwandeln möchte. Hierbei muss man dann nur den Befehl „Convert“ aufrufen und bestimmen ob man ein Bild in Graustufen möchte. Man muss dabei nur beachten, dass die Bilder beide dieselbe „Tiefe“ haben, wie in der Abbildung 10 zu sehen ist. Als „Tiefe“ wird die Größe des Bildes bezeichnet, was in diesem Fall „byte“ wäre.

Image<Gray, byte> grayImage = image.Convert<Gray, byte>();

Codeauschnitt 19

Zusätzlich kann man ein Bild mit dieser Klasse selbst erstellen. Dabei kann man dem Image seine genaue Größe mitteilen und wie der Hintergrund ausschauen könnte. Weiters gibt es auch, wie bei der Bitmap Klasse von .NET, die Möglichkeit, auf die einzelnen Pixel zuzugreifen, um deren Wert auszulesen, beziehungsweise ihn verändern.

### Automatic Garbage Collection

In der Programmierung bekommt der Begriff „Automatic Garbage Collection“ eine immer größere Bedeutung und wird auch oft als Grund verwendet, um festzustellen, welche Sprache in den Projekten verwendet werden soll. Dabei wird geschaut, welche Bereiche im Speicher nicht mehr gebraucht werden, nachdem die darin abgespeicherten Daten fertig bearbeitet wurden. Dann wird dieser Bereich des Speichers freigegeben, damit dieser erneut verwendet werden kann. Sozusagen räumt der Computer hinter einem auf, was die Performanz von Programmen deutlich steigern kann. Jedoch ist dies nicht eine Erfindung von EmguCV. So hängt es dann auch oft davon ab ob der Garbage Collctor richtig eingestellt wurde. Zum Beispiel kann man dies gut bei der Image Klasse sehen, da hier viel Platz im Speicher verwendet wird, um mit den Bildern zu arbeiten. Da ist es dann auch von großem Vorteil, wenn nach der Ausführung der Funktionen, die mit den Bildern arbeiten, der Platz im Speicher wieder frei gemacht wird. Es kann hier aber auch sein, dass die Daten, die ein Bild hat, so viel Speicher benötigen, dass der Garbage Collector diese Daten nicht entfernt, weshalb man immer auch selbst sicherstellen sollte, dass die Bilder von der Image Klasse auch wirklich entfernt werden, nachdem sie verwendet wurden.

### CvInvoke

Da es einige Funktionen in EmguCV gibt, die von OpenCV abgeleitet sind, kann man diese über den Befehl CvInvoke in .NET verständlicher Sprache aufrufen. Funktionen, die mit diesem Befehl aufgerufen werden, haben meist denselben Namen wie in OpenCV. Daher können auch oft Programme, die vorher nur mit OpenCV gearbeitet haben, auf EmguCV ohne große Probleme umgestellt werden. Auch können Klassen, die es in OpenCV gibt, in .NET verständliche Sprache umgeformt werden und dann verwendet werden.

### Mat/Map

Die Klassen Mat und Map sind so ähnlich wie die Image Klasse. Diese sind jedoch erst in den Versionen 3.x verfügbar. Sie haben gegenüber der Image Klasse den Vorteil, dass sie die Werte der Pixel direkt vom Bitmap übernehmen, beziehungsweise „teilen“ sie sich die Daten mit der Bitmap. Das heißt, dass wenn sich Pixelwerte im Bitmap verändern, ändern sich diese auch automatisch bei einem Mat/Map. Jedoch hat Mat/Map gegenüber der Image Klasse den entscheidenden Vorteil, dass erst zur Laufzeit des Programmes der Speicherplatz für das Bild festgelegt wird. Dadurch ist es um einiges vielseitiger einsetzbar.

Image<Bgr, byte> imageCV = new Image<Bgr, byte>(test);

Mat img = imageCV.Mat;

Codeauschnitt 20

Es können auch bereits verwendete Bilder der Image Klasse in ein Mat/Map Bild umgewandelt werden. Auch können die Bilder, die mit dieser Klasse bearbeitet oder erstellt wurden abgespeichert werden und zusätzlich kann man diese dem Benutzer sehr einfach anzeigen.

# Aufbau der Software

## Erstellen des Projektes

Das Ziel in dem Projekt ist es, eine Library, also eine DLL (Dynamic-Link Library). Datei zu erstellen, damit diese dann in der Testsoftware der Firma W&H Dentalwerk Bürmoos GmbH eingesetzt werden kann. Zur Entwicklung dieser wurde Visual Studio verwendet. Hier gibt es, beim Erstellen eines Projektes, die Möglichkeit zwischen verschiedenen Projekttypen zu wählen. Hierbei muss man dann bei der Erstellung einer dll den Typ Klassenbibliothek auswählen. Ein wichtiger Punkt, der bereits bei der Erstellung des Projektes bekannt sein sollte, ist, dass man bereits wissen sollte, wie man sein Projekt in etwa aufbauen möchte. Dementsprechend sollte einem bereits klar sein, welche Klassen man erstellen möchte und mit welchen Libraries diese arbeiten sollen. Da in dieser dll alles zum Testen des Bildschirmes eines Gerätes vorhanden sein sollte, wurde die Klassenbibliothek EmguCV gleich am Anfang eingebunden. Da diese aber nicht in Visual Studio vorhanden ist, wenn man ein Projekt erstellt hat, muss man diese erst downloaden und dann installieren. Hierbei hilft einem die Entwicklungsumgebung und macht es einem um einiges leichter, da man nur ein so genanntes „NuGet-Paket“ herunterladen und ins Projekt einbinden muss, indem man es in den Verweisen hinzufügt.

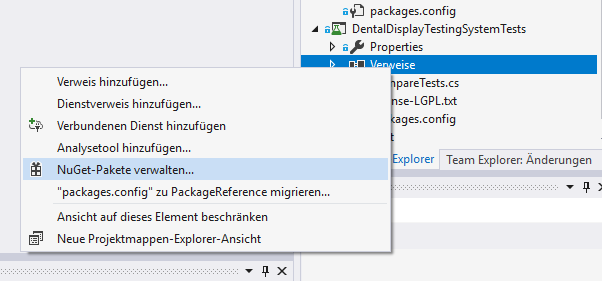


Abbildung 9

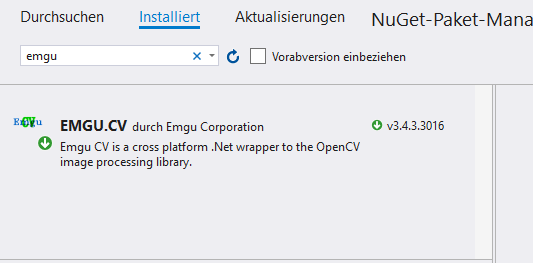


Abbildung 10

Wenn man dies erledigt hat, kann man sich genauer damit beschäftigen, welche Funktionen in dem Projekt vorhanden sein sollten und auf welche Funktionen der Benutzer von außerhalb zugreifen darf.

## Konzept überlegen

Bei diesem Schritt entscheidet sich grundlegend in welche Richtung sich das Projekt entwickeln wird und wie viel Zeit die verschiedenen Schritte in Anspruch nehmen werden. Hier wird außerdem geplant wie die Struktur des Projektes aussehen wird und welche Funktionen untereinander arbeiten und sich gegenseitig aufrufen. Hier haben wir uns zuerst überlegt was die dll alles können soll und diese dann auf Funktionen aufgeteilt. Hierbei gab es dann zwei Versionen, wobei die zweite Version auf der Ersten aufbaute und erst im Laufe des Projektes entstand.

### Erstes Konzept

Bei der ersten Version des Konzeptes wurden alle Funktionen nur in einer Klasse untergebracht. Das ergibt so auch keine größeren Probleme, könnte aber in der Benutzung der Klasse zu Schwierigkeiten führen, da der Benutzer einer dll im Normalfall nicht weiß, wie diese aufgebaut wurde. Weiters sollten gut verständliche Klassen und Funktionsnamen verwendet werden.

Compare:

* Compare
* CompIcon
* SearchIcon

Hier kann man auch bereits das erste Problem bei diesem Konzept erkennen. Es gibt keine Funktion, mit der man farbunabhängig arbeiten könnte und manche Namen sind nicht besonders aussagekräftig, was bei der Anwendung zu Problemen führen könnte. Trotz all dieser Problem wurde angefangen, die Software mit diesem Konzept zu entwickeln. Als weiteres Problem stellte sich auch gleich am Anfang heraus, welche Rückgabewerte die Funktionen haben und mit welchen Werten die Funktionen arbeiten sollten. Um diese Probleme zu beheben wurde die erste Version überarbeitet und mit dieser wurde dann weitergearbeitet.

Compare:

* CompImage
  + Berechnet, ob der Bildschirm mit der Referenz übereinstimmt oder nicht
* KordComp
  + Berechnet, ob der Bildschirm an der zu überprüfenden Stelle mit der Referenz übereinstimmt oder nicht
* CompIcon
  + Berechnet, ob ein Icon an der richtigen Position am Bildschirm ist
* SearchIcon
  + Berechnet, ob und wo ein Icon am Bildschirm ist, wenn die genaue Position am Bildschirm nicht bekannt ist

Mit diesem Grundkonzept wurde es um einiges einfacher die Software zu schreiben. Auch gab es einem eine bessere Sicht darauf, wie sich das Projekt entwickeln lassen würde. Ebenso konnte man einfacher die Funktionen erstellen, da es um einiges leichter war sich zu überlegen, welche Werte eine Funktion braucht, um den gewünschten Rückgabewert zu erhalten.

### Zweites Konzept

Die zweite Version ergab sich erst im späteren Verlauf des Projektes. Hierbei wurden Funktionen angepasst und hinzugefügt. Zusätzlich wurde eine zweite Klasse erstellt und die bereits vorhandenen Funktionen besser aufgeteilt, wodurch sich die Verwendung der dll später vereinfachen soll. Ein weiterer Punkt der hierbei zu tragen kam war, dass man das bereits vorhandene Projekt einfach umbauen konnte, ohne dass von vorne angefangen werden musste.

ImgCompare:

* CompImage
  + Berechnet, ob der Bildschirm mit der Referenz übereinstimmt oder nicht
* KordComp
  + Berechnet, ob der Bildschirm an der zu überprüfenden Stelle mit der Referenz übereinstimmt oder nicht

IconCompare:

* CompIcon
  + Berechnet, ob ein Icon an der richtigen Position am Bildschirm ist
* SearchIcon
  + Berechnet, ob und wo ein Icon am Bildschirm ist, wenn die genaue Position am Bildschirm nicht bekannt ist
* GetGrayImage
  + Soll das benötigte Bild in ein Schwarzweiß-Bild umwandeln
  + Soll nur innerhalb der Klasse verwendet werden
* GetLuminanz
  + Soll den Luminanzwert eines Pixels ausrechnen und diesen zurückgeben
  + Soll nur innerhalb der Klasse verwendet werden

Bei diesem Konzept wurde bereits daran gedacht, dass der Vergleich der Bilder möglichst farbunabhängig sein sollte, da es auch sein kann, dass sich die verwendeten Farben bei dem Bildschirm verändern können, wenn ein Kunde sich eine andere Farbe wünscht. Zusätzlich kann man in diesem Konzept auch sehen, dass Funktionen eingefügt wurden, welche der Benutzer nicht verwenden soll. Das hat den Grund, dass die Funktionen, die dem Benutzer zugänglich sind, mit diesen arbeiten und man so vermeiden möchte, dass ein Fehler durch falsches nutzen der dll auftritt.

## Umsetzung der Funktionen

Bevor die Funktionen umgesetzt wurden, haben wir die „Prototypen“ erstellt. Hier haben wir vorerst nur die Funktion mit ihrem Rückgabewert und ihren Parametern aufgebaut. Was sich bei den meisten Funktionen als relativ einfach erwies, wenn man nach dem oben vorgestellten Konzept vorging. Nur eine Funktion bereitete anfangs einige Probleme, wobei diese sich dann auch relativ einfach lösen ließen. Ein weiteres Problem, das vor dem Programmieren geklärt werden musste, war dass wir uns überlegen mussten, wie der Bildschirm überprüft werden sollte. Hier entschieden wir uns dann für den Vergleich mit Referenzbildern, da sich dieser für dieses Problem am besten eignete und es so auch keinen großen Aufwand gab eine passende Kamera zu finden, welche dann auch entsprechend platziert werden müsste.

### CompImage

Die CompImage-Funktion war die erste Funktion, welche programmiert wurde. Diese stellte auch die Grundlage für später folgende Funktionen dar. In ihr wird ein einfacher Pixelvergleich durchgeführt, was mit zwei „for-Schleifen“ realisiert wurde. In ihrer ersten Version überprüfte sie jedoch nicht ob die Größen der beiden Bilder überhaupt gleich waren, was so schon am Anfang zu großen Performanz Problemen führte, weshalb dieses auch gleich behoben werden musste, da es bei der Testsoftware auch auf die Geschwindigkeit ankommt.

public static bool CompImage(Bitmap actualImage, Bitmap expectedImage)

{

...

}

Codeauschnitt 21

### KordComp

Bei dieser Funktion wird überprüft, ob an den angegebenen Koordinaten der Screenshot mit dem Referenzbild auch übereinstimmt. Man kann also sehr einfach feststellen, dass es sich hierbei um eine Erweiterung der ersten Funktion handelt. In diesem Fall werden nur zusätzlich zu den Bildern, welche für den Vergleich benötigt werden, auch die Koordinaten mitgegeben, an denen der Vergleich durchgeführt werden soll.

public static bool KordComp(Bitmap actualImage, Bitmap expectedImage,

int ya, int ye, int xa, int xe)

{

...

}

Codeauschnitt 22

Diese Funktion wurde auch vorerst verwendet, um zu überprüfen, ob Icons am Display auf ihrer richtigen Position sind.

### CompIcon

Wie der Name der Funktion bereits vermuten lässt, ist diese Funktion zum Vergleich der Icons vorhanden. Dies wird benötigt, da am Bildschirm die richtigen Icons vorhanden sein sollten und diese auch zusätzlich an der richtigen Stelle gefunden werden sollten. Um dies zu ermöglichen, muss man der Funktion den Screenshot und das Icon mitgeben. Zusätzlich werden die x- und y-Koordinate, an dem das Icon am Bildschirm anfängt, benötigt. Bei dieser Funktion hat man gegenüber der Funktion KordComp den Vorteil, dass man nicht wissen muss, welche Größe das Icon hat, um zu überprüfen, ob es an der richtigen Stelle liegt, da dies von der Funktion übernommen wird. Dabei hat man dann auch den Vorteil, dass die Funktion um einiges einfacher einsetzbar ist.

public static bool CompIcon(Bitmap actualImage, Bitmap icon,

int coordinateX, int coordinateY)

{

...

for (int i = coordinateX; i < actualImage.Width; i++)//x-coordinate

{

iconY = 0;

iconX++;

if (iconX == icon.Width )

{

break;

}

for (int j = coordinateY; j < actualImage.Height; j++)//y-coordinate

{

if (iconY == icon.Height )

{

break;

}

...

}

Codeauschnitt 23

Diese Funktion war dann auch die erste Funktion, bei der an den farbunabhängigen Vergleich gedacht wurde, weshalb sich dann das zweite Konzept ergab. Dadurch wird die Funktion und die dll auch um einiges vielseitiger einsetzbar, als sie vorher gewesen wäre.

### SearchIcon

Bei dieser Funktion war der Grundgedanke der, dass man nicht unbedingt wissen muss, wo ein bestimmtes Icon liegt. So kann auf recht einfache Weise am Screenshot oder auf dem Referenzbild das Icon finden kann. Die Position kann man dann mit der Funktion CompIcon überprüfen. Für die Suche des Icons wurde ein sehr einfaches Verfahren angewendet, bei dem die Funktion CompIcon so oft aufgerufen und ausgeführt wird, bis das Icon gefunden wurde. Auch wird einem mitgeteilt, wenn es nicht gefunden wurde. Jedoch hat man bei dieser Funktion das Problem, dass man mehr als nur einen Rückgabewert hat. Um dieses Problem zu umgehen wurde hier ein sogenanntes „Tuple“ angewendet. Dies ist eine Art Datentyp, bei dem man mehrere Werte auf einmal zurückgeben kann. Um diesen jedoch so effektiv wie möglich zu verwenden, muss man wieder ein NuGet-Paket, System.ValueTuple, herunterladen und installieren. Danach kann man seinen Rückgabeparametern auch einen Namen geben, was die Verwendung der Funktion so erheblich vereinfacht.

public static (int x, int y, bool found) SearchIcon(Bitmap picture, Bitmap icon)

{

...

if (CompIcon(picture, icon, i, j) == true)

{

x = i;

y = j;

help = true;

break;

}

...

}

Codeauschnitt 24

Bei dieser Funktion stellte sich jedoch schnell heraus, dass die Performanz nicht gut war und je größer ein Bild wurde, umso länger dauerte dann der Suchvorgang. Um dieses Problem zu beheben, wurden einige Optimierungsmaßnahmen getroffen. Eine dieser war es, zu überprüfen ob das gesuchte Icon überhaupt noch an den angenommenen Koordinaten Platz haben würde.

### GetGrayImage

Diese Funktion ist eine Funktion, die dem Benutzer nicht zugänglich ist und mit der nur die Funktionen innerhalb der Klasse arbeiten können. Der erste Gedanke bei dieser Funktion war es jedoch, nicht nur ein Schwarzweiß-Bild zu erhalten, sondern ein Bild, in dem nur die Kanten innerhalb des Bildes vorhanden sind. Dies stellte sich aber schnell als nicht besonders zuverlässig heraus, da die Kantenerkennungsalgorithmen nicht immer dieselben Kanten gefunden haben. Also entschieden wir uns, dass die Funktion nur ein Schwarzweiß-Bild zurückgeben sollte, um so die Vergleichsalgorithmen zu vereinfachen. Hierbei wurde dann auch das erste Mal die Klassenbibliothek EmguCV verwendet, da es mit ihr sehr einfach war dies zu ermöglichen.

private static Bitmap GetGrayImage(Bitmap image)

{

Bitmap gray;

Image<Rgba, byte> rgbImage = new Image<Rgba, byte>(image);

Image<Gray, byte> grayImage = rgbImage.Convert<Gray, byte>();

gray = grayImage.ToBitmap();

return gray;

}

Codeauschnitt 25

Jedoch reicht ein Schwarzweiß-Bild immer noch nicht aus, um farbunabhängig Bilder zu vergleichen. Aber es macht diese Aufgabe um einiges leichter, da man mit diesen Werten um einiges leichter rechnen kann.

### GetLuminanz

Da man bei der Entwicklung der GetGrayImage Funktion auf das Problem gestoßen ist, dass die Kanten nicht immer gleich erkannt wurden, wurde diese Funktion entwickelt, um sich selbst die Kanten des Bildes zu berechnen, bei der man eine bestimmte Fehlerquote zulassen kann. Um dies zu realisieren werden die Luminanzwerte eines Pixels berechnet und diese dann an die Funktion zurückgegeben. Da es sich bei Luminanzwerten um Helligkeitswerte handelt, kann man sich so auch sehr einfach die Kanten berechnen, da sich hierbei der Luminanzwert ändern müsste.

private static double GetLuminaz(Bitmap image, int x, int y)

{

double luminanz = 0;

Color colorImage = image.GetPixel(x, y);

luminanz = colorImage.R \* 0.299 + colorImage.G \* 0.587

+ colorImage.B \* 0.114;

return luminanz;

}

Codeauschnitt 26

Da es jedoch anfänglich noch Probleme gab, bei dieser Methode zu erkennen, ob ein Icon an der richtigen Position war oder nicht, wurde eine neue Funktion erstellt, welche wie die Funktion CompIcon arbeitete, nur dass hier die Kanten selbst berechnet wurden und ein bestimmter Fehlerquotient bestimmt wurde, welcher sich bei ca. 1% Abweichung befand. Dabei mussten immer zwei Werte überprüft werden, da sich auf beiden Bildern der Luminanzwert immer gleich verändern sollte und dieser auch in den meisten Fällen in etwa denselben Wert haben sollte, da hier die ursprüngliche Farbe kaum einen Einfluss auf ihn hat.

## Testen der Funktionen

Nach der Ausprogrammierung der einzelnen Funktionen mussten diese auch getestet werden, um sicherzustellen, dass sie auch wirklich den Anforderungen entsprechen und die Arbeit so durchführen wie es gedacht ist. Dies stellte sich jedoch anfänglich als besonders schwierig heraus, da man eine dll nicht so leicht überprüfen kann, weil man zum Starten eines Programmes immer eine ausführbare Funktion benötigt und diese ist in einer dll nicht vorhanden. Um am Anfang der Entwicklungsphase die Funktionen zu testen, wurde deshalb ein neues Projekt gemacht, welche mit der dll arbeitete. Dies stellte sich jedoch nicht als sehr effektiv heraus, da es so sehr zeitaufwendig war alle Fälle zu testen, bei denen es zu Fehlern kommen könnte. Durch einige Umwege kamen wir dann zum Entschluss, dass in diesem Projekt sogenannte „UnitTests“ die einfachste und beste Lösung waren die einzelnen Funktionen zu testen. Für diese musste man jedoch wieder ein bestimmtes NuGet-Paket, MSTest.TestAdapter, herunterladen und installieren. Es gibt zwar die Möglichkeit direkt ein UnitTest Projekt zu erstellen, wobei diese nicht immer funktionieren, weshalb man zusätzlich dieses NuGet-Paket benötigt.

[TestMethod()]

public void CompIconTest\_01()

{

bool test;

test = Compare.CompIcon(image, icon, 271, 24);

Assert.AreEqual(true, test);

}

[TestMethod()]

public void CompIconTest\_02()

{

bool test;

Bitmap icon = new Bitmap(@"MyImage.png");

test = Compare.CompIcon(image, icon, 130, 202);

Assert.AreEqual(true, test);

}

Codeauschnitt 27

In Codeabschnitt 27 sind zwei Beispiel Tests angeführt. Hier ist gut erkennbar, dass ein UnitTest recht einfach zu schreiben ist. Dies hat den Vorteil, dass mehr Tests durchgeführt werden können, um potenzielle Fehler zu entdecken und diese dann zu beheben. Dadurch kann man sich sehr viel Zeit sparen, bevor man das Projekt veröffentlicht. Auch ist die benötigte Zeit, um einen solchen UnitTest zu schreiben, bei weitem geringer

# Quellenverzeichnis

Network Working Group of the IETF, The Secure Shell Architecture, Zugriff am 26.03.19  
<https://tools.ietf.org/html/rfc4251#page-3>,

Sean Micheal Kerner, SSH Hardens the Secure Shell, Zugriff am 26.03.2019  
<https://web.archive.org/web/20090706123457/http://www.serverwatch.com/news/article.php/3551081>

SSH Communications Security, Man-in-the-middle attack, Zugriff am 26.03.2019  
<https://www.ssh.com/attack/man-in-the-middle>

Klaus Lipinski, Bildverarbeitung, Zugriff am 23.02.2019  
<https://www.itwissen.info/Bildverarbeitung-image-processing.html>

Ralf Bruder, Bildverarbeitung, Zugriff am 23.02.2019  
<http://www2.in.tu-clausthal.de/~reuter/ausarbeitung/Ralf_Bruder_-_Digitale_Bildverarbeitung.pdf>

Stefan Kuppelwieser, Cannyalgorithmus, Zugriff am 23.02.2019  
<https://www.kuppelwieser.net/index.php/technik/15-bildverarbeitung/40-canny-algorithmus>

Microsoft .Net, Zugriff am 24.02.2019  
<https://docs.microsoft.com/en-us/dotnet/api/system.drawing.bitmap?view=netframework-4.7.2>

EmguCV, Zugriff am 24.02.2019  
<http://www.emgu.com/wiki/index.php/Main_Page>

# Abbildungsverzeichnis

[Abbildung 1 http://itdoc.hitachi.co.jp/manuals/3020/30203R3900e/JRMP0035.HTM Hitachi Ltd. Zugriff 26.03.19 17](file:///C:/Users/Anwender/Desktop/DIPLOMARBEIT_DDTS_Final_Final.docx#_Toc8814745)

[Abbildung 2 https://sebastien.saunier.me/blog/2015/05/10/github-public-key-authentication.html Sebastian Sauner, Zugriff 26.03.19 18](file:///C:/Users/Anwender/Desktop/DIPLOMARBEIT_DDTS_Final_Final.docx#_Toc8814746)

[Abbildung 3 Ausschnitt aus der GUI 25](#_Toc8814747)

[Abbildung 4 33](#_Toc8814748)

[Abbildung 5 33](#_Toc8814749)

[Abbildung 6 34](#_Toc8814750)

[Abbildung 7 original Bild und Bild nach Segmentierung 36](#_Toc8814751)

[Abbildung 8 Aufbau von EmguCV 38](#_Toc8814752)

[Abbildung 9 43](#_Toc8814753)

[Abbildung 10 43](#_Toc8814754)

1. Vgl. Network Working Group of the IETF, The Secure Shell Architecture, Zugriff am 26.03.19  
   <https://tools.ietf.org/html/rfc4251#page-3>, [↑](#footnote-ref-1)
2. Vgl. Sean Micheal Kerner, SSH Hardens the Secure Shell, Zugriff am 26.03.2019  
   <https://web.archive.org/web/20090706123457/http://www.serverwatch.com/news/article.php/3551081> [↑](#footnote-ref-2)
3. Vgl. SSH Communications Security, Man-in-the-middle attack, Zugriff am 26.03.2019  
   <https://www.ssh.com/attack/man-in-the-middle> [↑](#footnote-ref-3)