Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES
Näher dran.

MMT Fakultät für Maschinenbau
und Mechatronik

Anleitung zur Inbetriebnahme des Projektes Swimming Pixels – Kamera Ortung

Eine Ausarbeitung im Fach Informationstechnik
Betreuer: Prof. Walter

von
Christoph Gerard (61570)
Sammy Radhouani (60735)

Karlsruhe, 17. November 2020

Inhaltsverzeichnis

1. Bildverarbeitungs-Bibliothek OpenCV

1.1 Download

Um die Bibliothek OpenCV in eine Entwicklungsumgebung einzubinden muss diese zunächst gedownloadet werden. Es gibt eine pre-buildet Version für Windows unter folgendem Link:

https://opencv.org/releases/

Hier muss die .exe ausgeführt werden, um die Bibliothek zu entpacken. Dabei sollte nun ein Speicherort angegeben werden, an dem die Dateien von OpenCV dauerhaft liegen gelassen werden können, da später im Programm verschiedene Dateipfade an genau diesen Speicherort verknüpft werden müssen.

Bei unserem Projekt wurde die zu diesem Zeitpunkt aktuellste Version OpenCV 4_4_0 verwendet.

1.2 Einbindung in VisualStudio

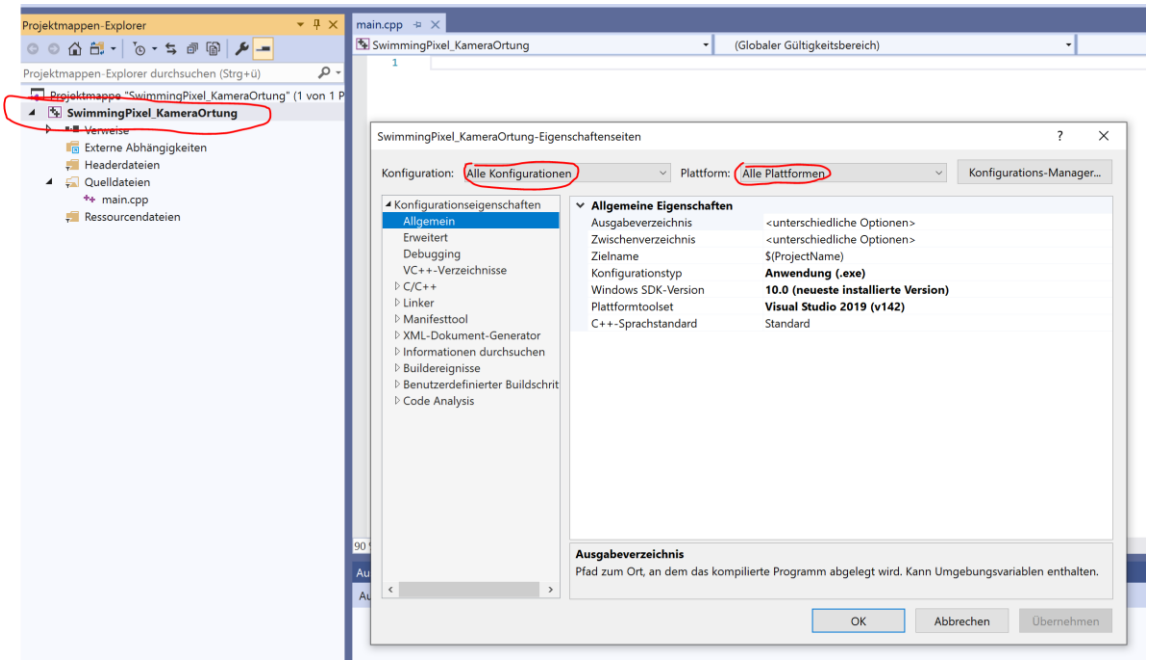
Nun da die Bibliothek an einem gewissen Speicherort untergebracht ist, muss der Entwicklungsumgebung gesagt werden wo dieser ist, um sie im Programm nutzen zu können.

In unserem Projekt verwendeten wir Visual Studio 2019 Community.

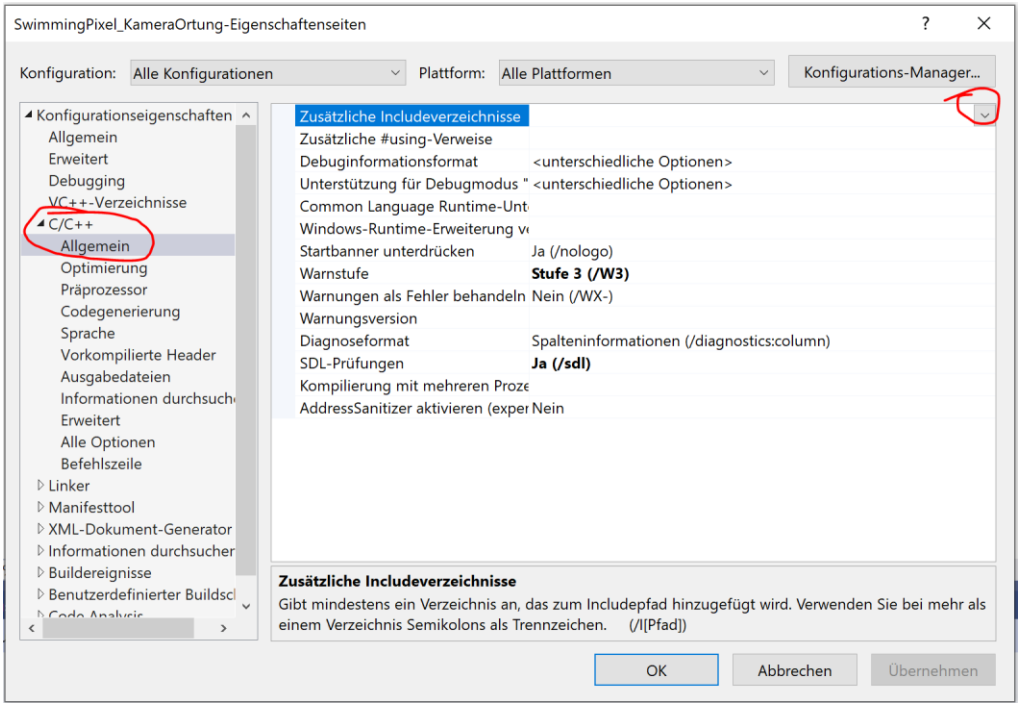
Der nachfolgend beschriebene Weg der Verknüpfung setzt ein bereits erstelltes (wenn auch leeres) Projekt voraus. Die Schritte müssen für jedes Projekt separat durchgeführt werden!

1. Da in C++ programmiert werden soll, sollte zumindest eine main.cpp Datei im Projekt erstellt sein, da man ansonsten nachfolgend nicht an die benötigte Konfiguration rankommt.

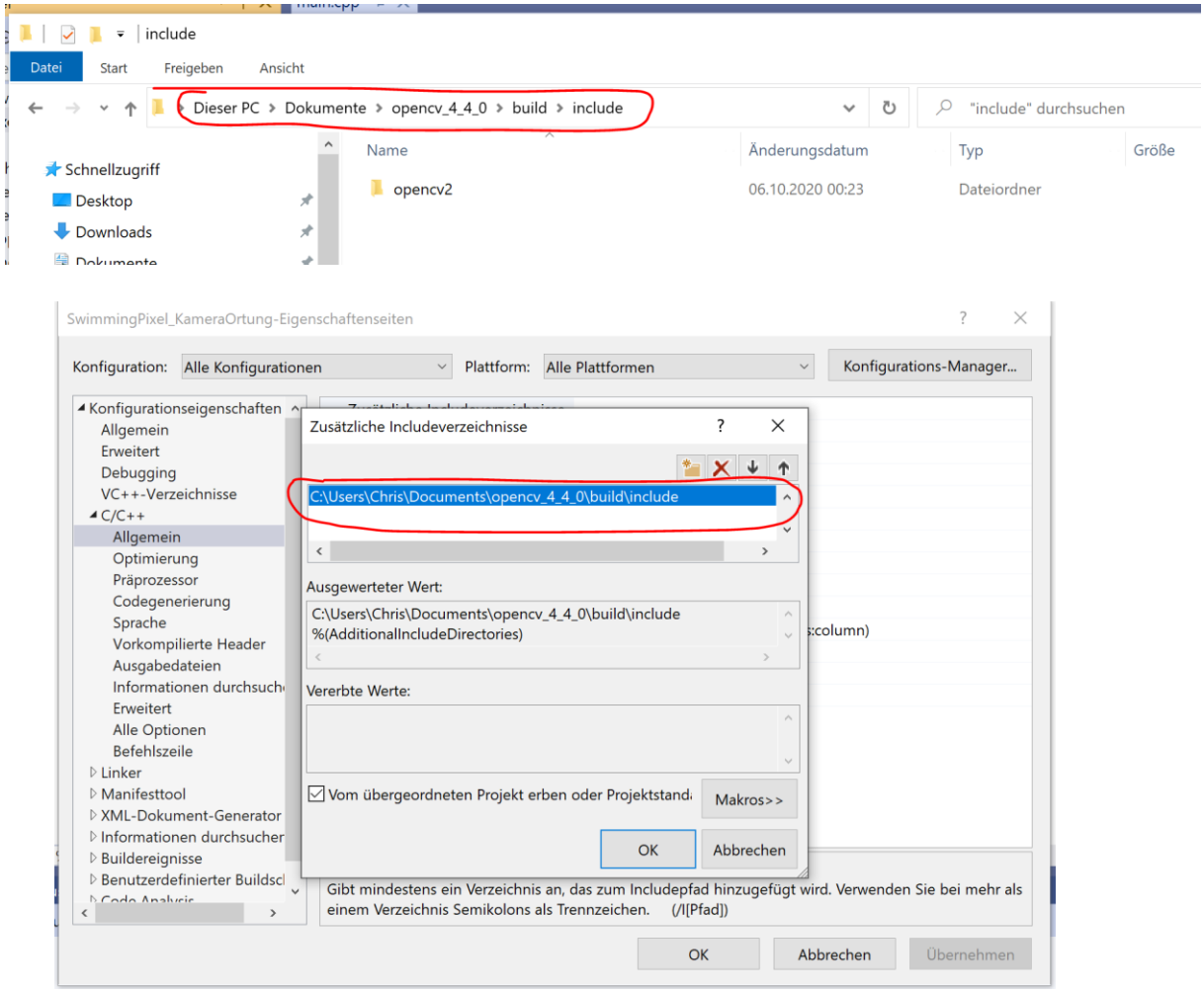
2. Einstellungen öffnen
   Im Projektmappen-Explorer auf den Projekt-Namen mit rechtsklick klicken und ganz unten die Einstellungen öffnen. Anschließend zunächst oben im Fenster auf „Alle Konfigurationen“ und „Alle Plattformen“ stellen.



3. Im Reiter C/C++ (dieser würde ohne vorhandene .cpp Datei u.U. fehlen) – Allgemein – zusätzliche Includeverzeichnisse ganz rechts anwählen und auf „Bearbeiten“ drücken.

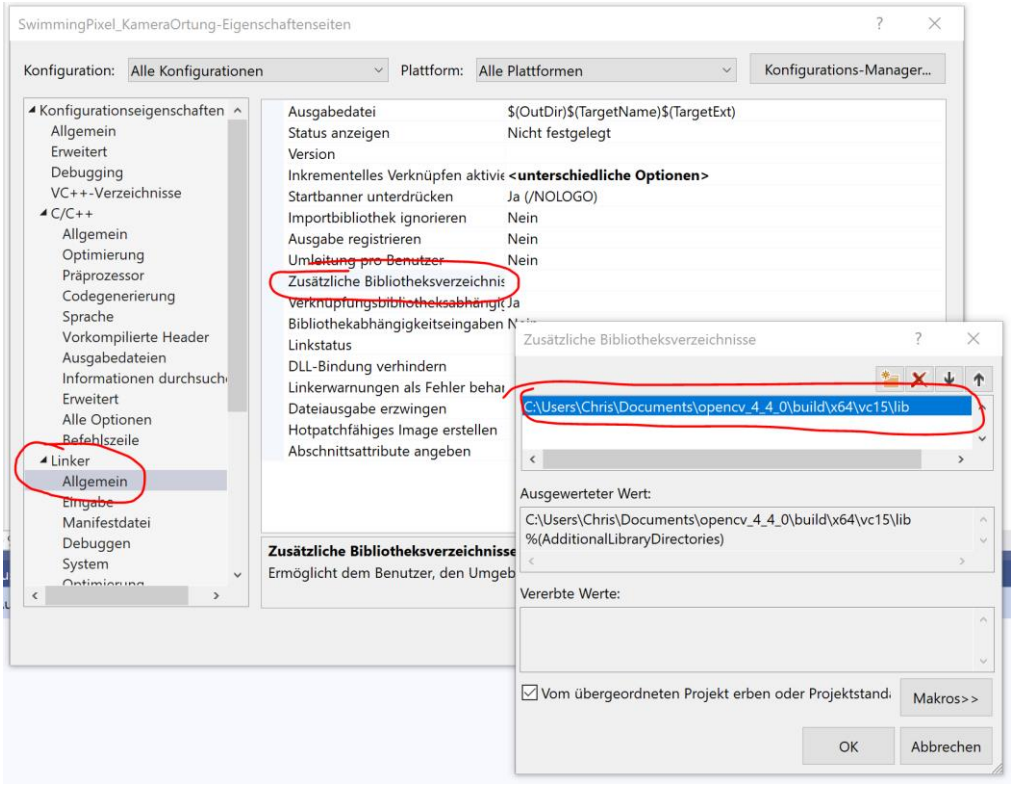


4. Nun muss im Windows Explorer der OpenCV Ordner geöffnet werden. Anschließend build – include. Diesen Dateipfad kopieren und in die Einstellung „Zusätzliche Includeverzeichnisse“ einfügen, bestätigen und übernehmen.

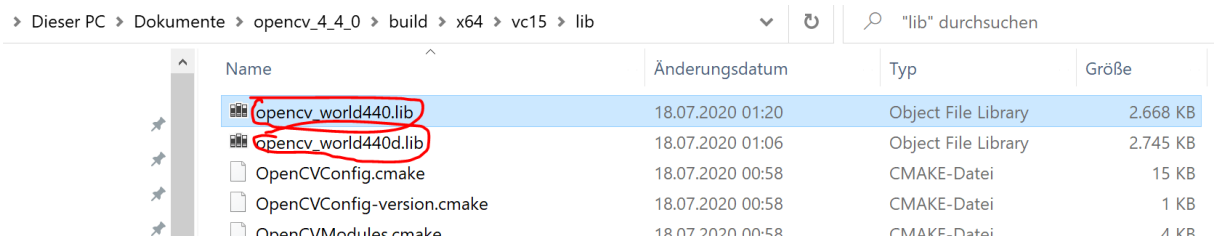


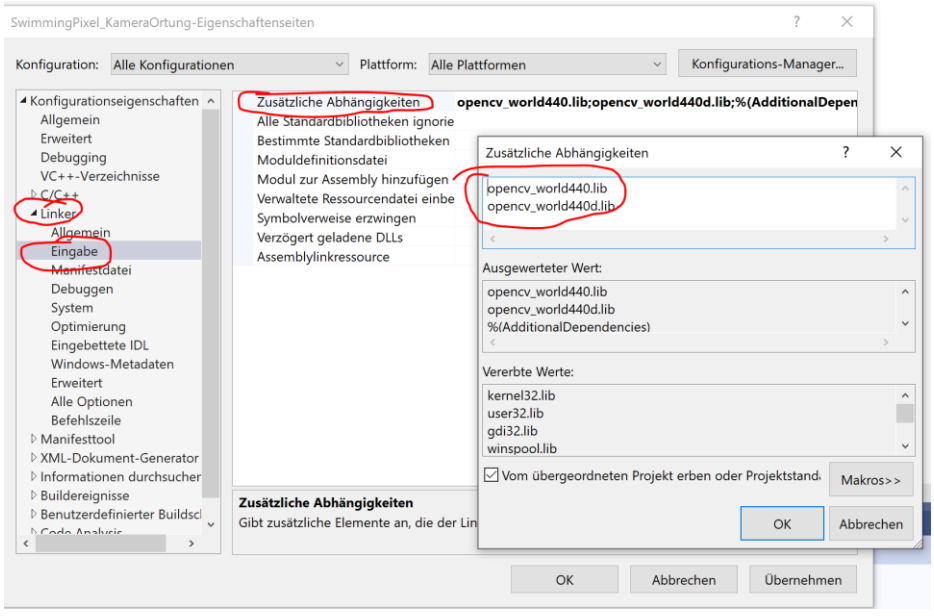
5. Der nächste Schritt findet in den Einstellungen im Reiter „Linker“ – „Allgemein“ statt. Hier wird genau wie im Schritt 3. und 4. ein Dateipfad in die Einstellung „Zusätzliches Bibliotheksverzeichnis“ eingefügt. Der Pfad ist nun jedoch ein etwas anderer:

   C:\Users\Chris\Documents\opencv_4_4_0\build\x64\vc15\lib

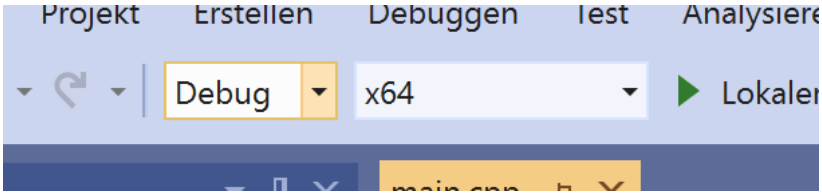


6. Unter der Einstellung „Linker“ – „Eingabe“ – „Zusätzliche Abhängigkeiten“ müssen nun zwei komplette Dateinamen (inkl. .lib Endung) eingefügt werden. Die Dateien finden sich im Windows-Explorer genau in dem Bibliotheksverzeichnis, welches im Schritt zuvor verwendet wurde:



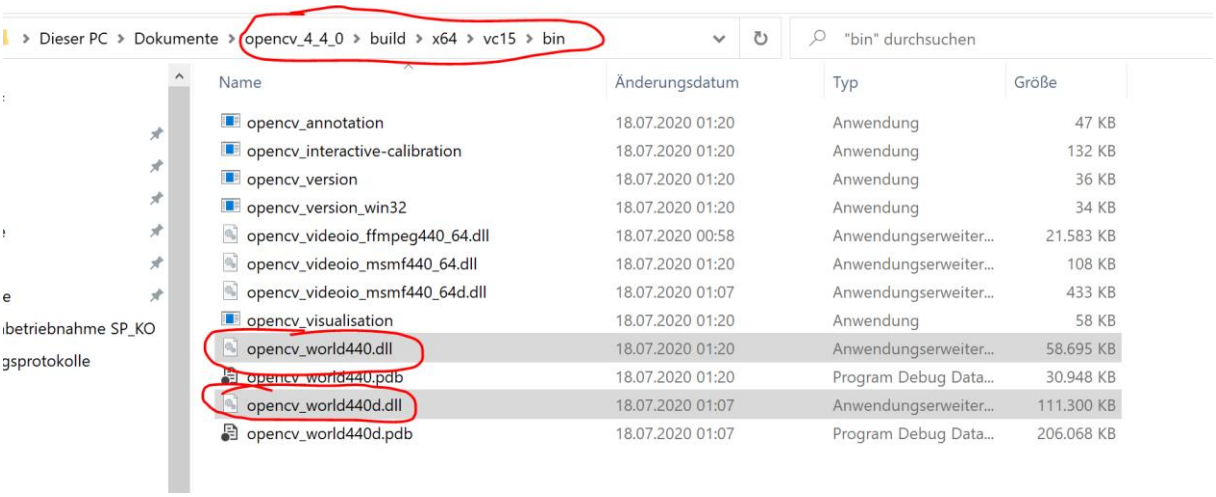


7. Anschließend kann man die Projekt-Einstellungen wieder schließen. Es muss darauf geachtet werden, dass der Debugger bzw. der Relase-Modus entsprechend auf der richtigen Bit-Version eingestellt ist. Bei uns ist das x64 (Betriebssystem und OpenCV sind so installiert)

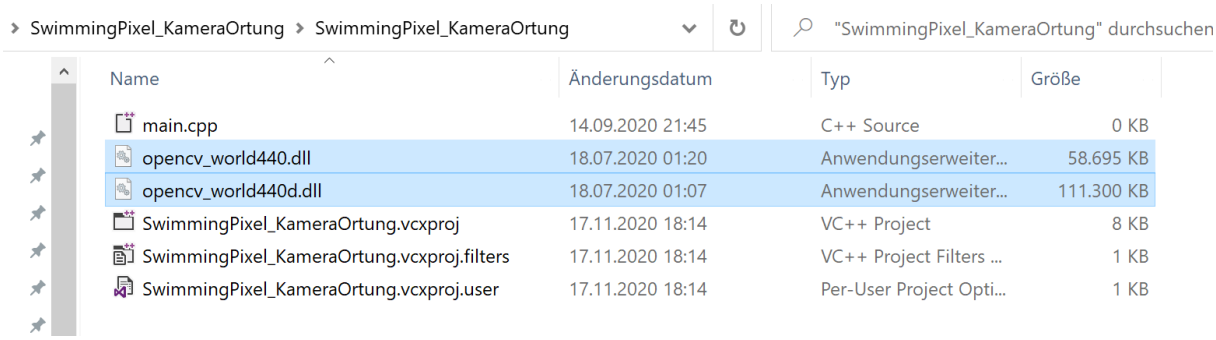


8. Der letzte Schritt zum Einbinden der Bibliothek OpenCV ist nun folgender:

   Kopieren von zwei .dll Dateien in den Projektordner, in welchem z.B. auch die main.cpp gespeichert ist. Gefunden werden diese Dateien hier:



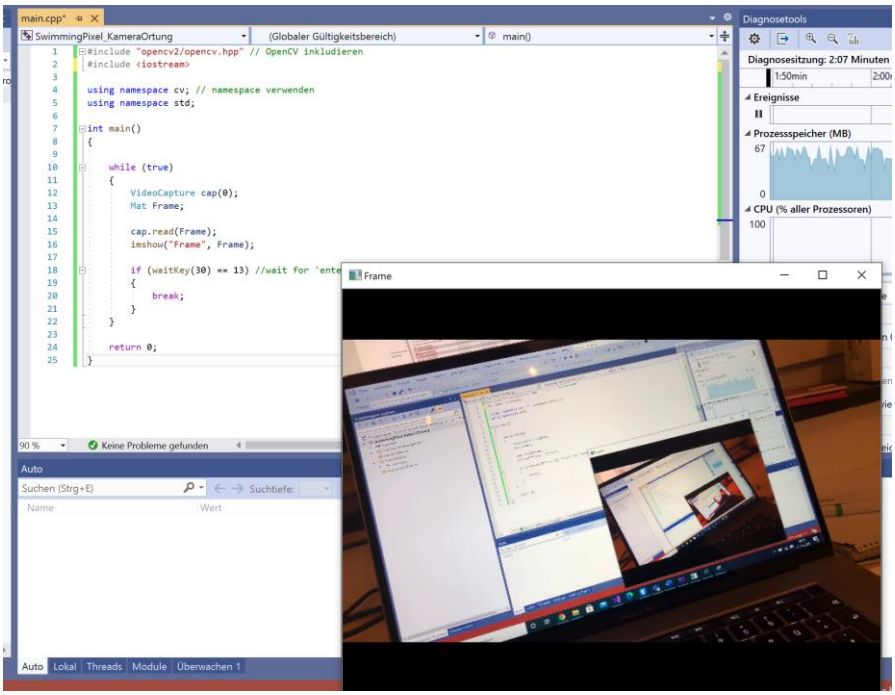
   Einfügen in:



1.3 Testprogramm OpenCV

Mit dem folgenden Programm kann die korrekte Einbindung leicht überprüft werden:

```
#include "opencv2/opencv.hpp" // OpenCV inkludieren
#include <iostream>

using namespace cv; // namespace verwenden
using namespace std;

int main()
{

	while (true)
	{
		VideoCapture cap(0); // index je nach Kamera. Vgl. COM1, COM2,…
		Mat Frame;

		cap.read(Frame);
		imshow("Frame", Frame);

		if (waitKey(30) == 13) //wait for 'enter' key press for 30ms. If 'enter'
key is pressed, break loop
		{
			break;
		}
	}

	return 0;
}
```



2. SwimmingPixel-KameraOrtung Code einbinden

Der erarbeitete Code ist so gestaltet, dass man diesen im Wesentlichen nur als zusätzliche Klasse einbinden muss und somit sehr einfach schon bestehenden Code mit der Möglichkeit der Kamera-Ortung erweitert werden kann.

Dazu müssen die folgenden Dateien eingebunden werden:

- **mouseKlick.h**
- **CCameraDetection.h**
- **CCameraDetection.cpp**

Die zusätzlich gegebene **main.cpp** dient nur zu Testzwecken. Der darin gegebene Grundaufbau muss jedoch übernommen werden, um die Funktionen sinnvoll zu nutzen.

************************Wichtig: **********************
„Release“ Modus verwenden! (anstatt Debug) sonst läuft das Programm nicht. Die Eingabe (z.B. „ENTER“) wird im Debug-Modus nicht erkannt.
****

3. iVCam als Kamera Schnittstelle

Um eine mobile Kamera mit hoher Auflösung zu nutzen, bietet sich ein Smartphone sehr gut an. Hierfür benötigt man jedoch die Möglichkeit, dieses per Zwischeninstanz als WebCam an den Laptop anzuschließen.

Hierbei verwendeten wir die App „iVCam“. Diese Lösung funktioniert auf der Smartphone Seite sowohl für iOS als auch für Android, auf der Notebook Seite für Windows.

Man muss zum einen die App auf das Smartphone installieren (erhältlich kostenlos im Store), zum anderen für Windows von der Herstellerseite downloaden:

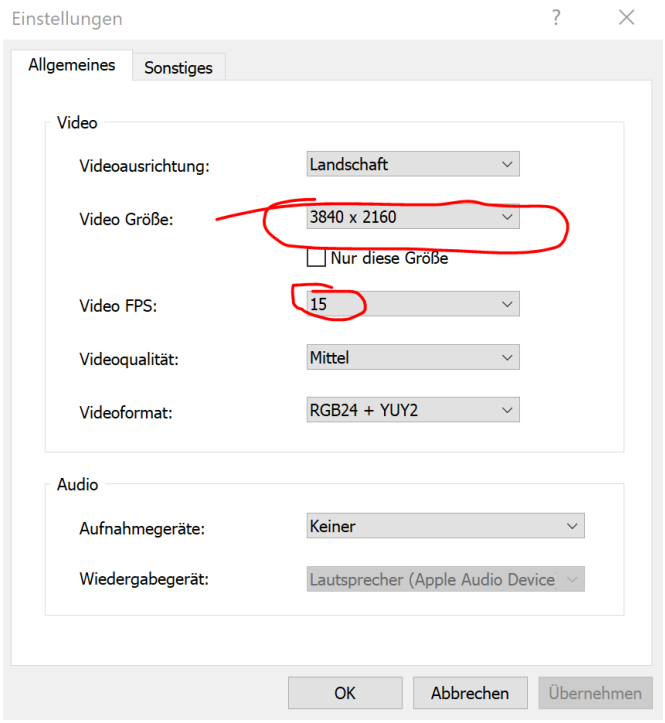
https://www.e2esoft.com/ivcam/

Nachdem beides installiert ist, hat man zwei Möglichkeiten:

1. Laptop und Smartphone müssen im selben Netzwerk sein -> kabellose Übertragung
2. Übertragung via USB Verbindung

In den Einstellungen hat man die Möglichkeit die Auflösung des Eingangs einzustellen. Ggf. müssen die Smartphone Einstellungen entsprechend angepasst werden.

Es hat sich herausgestellt, dass eine niedrige Framerate (z.B. 15FPS) völlig ausreicht und zusätzlich das c++ Programm schneller läuft.



Um die richtige Kamera im Programm zu nutzen muss nun lediglich der „CamIndex“ (siehe main.cpp) geändert werden. Damit kann man zwischen allen angeschlossenen Kameras umschalten.

Welche App letztlich eingesetzt wird ist nicht relevant für Visual Studio. Hier darf gern variiert werden.