

**Industrial photogrammerty**

In this exercise we load a video, try to identify in the video the movement of a ball and mark the ball with cross and add its coordinates

-start IDE: Computer->O:\Software\Programmierung\QT\Tools\QtCreator\bin->qtcreator.exe

- create a new project motiondetect in folder opencvproj/***UE\_4***

- set mode down left from debug to **release**

- in the .pro file add the location of the ***include files*** and of the ***dlls*below**

**FORMS += \**

mainwindow.ui:

INCLUDEPATH += U:\opencv340\include

LIBS += U:\opencv340\bin\libopencv\_core340.dll

LIBS += U:\opencv340\bin\libopencv\_highgui340.dll

LIBS += U:\opencv340\bin\libopencv\_imgcodecs340.dll

LIBS += U:\opencv340\bin\libopencv\_imgproc340.dll

LIBS += U:\opencv340\bin\libopencv\_videoio340.dll

LIBS += U:\opencv340\bin\libopencv\_video340.dll

LIBS += U:\opencv340\bin\libopencv\_features2d340.dll

LIBS += U:\opencv340\bin\libopencv\_calib3d340.dll

----------------------------------------------------------------

- under sources open the file ***mainwindow.cpp***

- over line add:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMainWindow>

//opencv

#include "opencv2/imgcodecs.hpp"

#include "opencv2/imgproc.hpp"

#include "opencv2/videoio.hpp"

#include <opencv2/highgui.hpp>

#include <opencv2/video.hpp>

//C++

#include <iostream>

#include <sstream>

using namespace std;

using namespace cv;

//our sensitivity value to be used in the absdiff() function

const static int SENSITIVITY\_VALUE = 20;

//size of blur used to smooth the intensity image output from absdiff() function

const static int BLUR\_SIZE = 10;

//we'll have just one object to search for

//and keep track of its position.

int theObject[2] = {0,0};

//bounding rectangle of the object, we will use the center of this as its position.

Rect objectBoundingRectangle = Rect(0,0,0,0);

----------------------------

- under addthe function ***processVideo()***;

-under Headers ***mainwindow.h*** doubleclick and completely change content to:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

//opencv

#include "opencv2/imgcodecs.hpp"

#include "opencv2/imgproc.hpp"

#include "opencv2/videoio.hpp"

#include <opencv2/highgui.hpp>

#include <opencv2/video.hpp>

using namespace cv;

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~***MainWindow***();

void **processVideo**();

std::string **intToString**(int number);

void **searchForMovement**(Mat thresholdImage, Mat &cameraFeed);

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

----------------------------------------------------

- at the end of ***mainwindow.cpp*** add our function code for ***processVideo()***:

void MainWindow::**processVideo**(){

//some boolean variables for added functionality

//these two can be toggled by pressing 'd' or 't'

bool debugMode = false;

bool trackingEnabled = false;

//pause and resume code

bool pause = false;

//set up the matrices that we will need

//the two frames we will be comparing

Mat frame1,frame2;

//their grayscale images (needed for absdiff() function)

Mat grayImage1,grayImage2;

//resulting difference image

Mat differenceImage;

//thresholded difference image (for use in findContours() function)

Mat thresholdImage;

//video capture object.

VideoCapture capture;

while(1){

//we can loop the video by re-opening the capture every time the video reaches its last frame

capture.*open*("U://opencvProjEng//opencvProj//TRACKMOTIONBALL//bounce.avi");

if(!capture.*isOpened*()){

cout<<"ERROR ACQUIRING VIDEO FEED\n";

getchar();

return;

}

//check if the video has reach its last frame.

//we add '-1' because we are reading two frames from the video at a time.

//if this is not included, we get a memory error! while(capture.*get*(CV\_CAP\_PROP\_POS\_FRAMES)<capture.*get*(CV\_CAP\_PROP\_FRAME\_COUNT)-1){

//read first frame

capture.*read*(frame1);

//convert frame1 to gray scale for frame differencing

cv::cvtColor(frame1,grayImage1,COLOR\_BGR2GRAY);

//copy second frame

capture.*read*(frame2);

//convert frame2 to gray scale for frame differencing

cv::cvtColor(frame2,grayImage2,COLOR\_BGR2GRAY);

//perform frame differencing with the sequential images. This will output an "intensity image"

//do not confuse this with a threshold image, we will need to perform thresholding afterwards.

cv::absdiff(grayImage1,grayImage2,differenceImage);

//threshold intensity image at a given sensitivity value cv::threshold(differenceImage,thresholdImage,SENSITIVITY\_VALUE,

255,THRESH\_BINARY);

//show the difference image and threshold image

cv::imshow("Difference Image",differenceImage);

cv::imshow("Threshold Image", thresholdImage);

//blur the image to get rid of the noise. This will output an intensity image

cv::blur(thresholdImage,thresholdImage, cv::Size(BLUR\_SIZE,BLUR\_SIZE));

//threshold again to obtain binary image from blur output

cv::threshold(thresholdImage,thresholdImage,SENSITIVITY\_VALUE,

255,THRESH\_BINARY);

//show the threshold image after it's been "blurred"

imshow("Final Threshold Image",thresholdImage);

MainWindow::searchForMovement(thresholdImage,frame1);

//flip(frame1, frame1, 0); //turn image upside down

//show our captured frame

imshow("Frame1",frame1);

//check to see if a button has been pressed.

//this 10ms delay is necessary for proper operation of this program

//if removed, frames will not have enough time to referesh and a blank

//image will appear.

switch(waitKey(10)){

case 27: //'esc' key has been pressed, exit program.

return;

case 116: //'t' has been pressed. this will toggle tracking

trackingEnabled = !trackingEnabled;

if(trackingEnabled == false) cout<<"Tracking disabled."<<endl;

else cout<<"Tracking enabled."<<endl;

break;

case 100: //'d' has been pressed. this will debug mode

debugMode = !debugMode;

if(debugMode == false) cout<<"Debug mode disabled."<<endl;

else cout<<"Debug mode enabled."<<endl;

break;

case 112: //'p' has been pressed. this will pause/resume the code.

pause = !pause;

if(pause == true){ cout<<"Code paused, press 'p' again to resume"<<endl;

while (pause == true){

//stay in this loop until

switch (waitKey()){

//a switch statement inside a switch statement? Mind blown.

case 112:

//change pause back to false

pause = false;

cout<<"Code resumed."<<endl;

break;

}

}

}

}

}

//release the capture before re-opening and looping again.

capture.*release*();

}

return;

}

--------------------------------------------

- add below that a small helper function that converts a number into a string:

//int to string helper function

string MainWindow::**intToString**(int number){

//this function has a number input and string output

std::stringstream ss;

ss << number;

return ss.str();

}

--------------------------------------------

-add below a function that is looking for movement by subtracting the intensities per pixel of two subsequent images + blurring + thresholding:

void MainWindow::**searchForMovement**(Mat thresholdImage, Mat &cameraFeed){

//notice how we use the '&' operator for the cameraFeed. This is because we wish

//to take the values passed into the function and manipulate them, rather than just working with a copy.

//eg. we draw to the cameraFeed in this function which is then displayed in the main() function.

bool objectDetected=false;

Mat temp;

thresholdImage.copyTo(temp);

//these two vectors needed for output of findContours

vector< vector<Point> > contours;

vector<Vec4i> hierarchy;

//find contours of filtered image using openCV findContours function

//findContours(temp,contours,hierarchy,CV\_RETR\_CCOMP,CV\_CHAIN\_APPROX\_SIMPLE );// retrieves all contours

findContours(temp,contours,hierarchy,CV\_RETR\_EXTERNAL,CV\_CHAIN\_APPROX\_SIMPLE );// retrieves external contours

//if contours vector is not empty, we have found some objects

if(contours.size()>0)objectDetected=true;

else objectDetected = false;

if(objectDetected){

//the largest contour is found at the end of the contours vector

//we will simply assume that the biggest contour is the object we are looking for.

vector< vector<Point> > largestContourVec;

largestContourVec.push\_back(contours.at(contours.size()-1));

//make a bounding rectangle around the largest contour then find its centroid

//this will be the object's final estimated position.

objectBoundingRectangle = boundingRect(largestContourVec.at(0));

int xpos = objectBoundingRectangle.x+objectBoundingRectangle.width/2;

int ypos = objectBoundingRectangle.y+objectBoundingRectangle.height/2;

//update the objects positions by changing the 'theObject' array values

theObject[0] = xpos , theObject[1] = ypos;

}

//make some temp x and y variables so we dont have to type out so much

int x = theObject[0];

int y = theObject[1];

//draw some crosshairs on the object

circle(cameraFeed,Point(x,y),20,Scalar(0,255,0),2);

line(cameraFeed,Point(x,y),Point(x,y-25),Scalar(0,255,0),2);

line(cameraFeed,Point(x,y),Point(x,y+25),Scalar(0,255,0),2);

line(cameraFeed,Point(x,y),Point(x-25,y),Scalar(0,255,0),2);

line(cameraFeed,Point(x,y),Point(x+25,y),Scalar(0,255,0),2);

putText(cameraFeed,"Tracking object at (" + intToString(x) +","+intToString(y)+")",Point(0,0),1,1,Scalar(255,0,0),2);

}

---------------------------------------

- Datei → alles speichern → build (hammer)

***-copy all the dlls*** from the QT5dlls\_r230 folder into the newly built release folder build\_motiondetect\_Desktop...

execute your motiondetect.exe file

- four windows should appear:

