#include <iostream>

#include <cstring>

#include <vector>

#include <list>

#include <sys/time.h>

#ifndef \_\_APPLE\_\_

#define EXPOSURE\_CONTROL // only works in Linux

#endif

#ifdef EXPOSURE\_CONTROL

#include <libv4l2.h>

#include <linux/videodev2.h>

#include <fcntl.h>

#include <errno.h>

#endif

#include "opencv2/opencv.hpp" // OpenCV library for easy access to USB camera and drawing of images on screen

#include "AprilTags/TagDetector.h"

#include "AprilTags/Tag16h5.h"

#include "AprilTags/Tag25h7.h"

#include "AprilTags/Tag25h9.h"

#include "AprilTags/Tag36h9.h"

#include "AprilTags/Tag36h11.h"

#include <unistd.h> // Needed for getopt / command line options processing

#include "Serial.h" // For Arduino: locally defined serial port access class

// utility function to provide current system time (used below in determining frame rate at which images are being processed).

**double tic()**

// Normalize angle to be within the interval [-pi,pi].

**inline double standardRad(double t)**

// Convert rotation matrix to Euler angles

**void wRo\_to\_euler(const Eigen::Matrix3d& wRo, double& yaw, double& pitch, double& roll)**

**class Demo :**

**attributs:**

AprilTags::TagDetector\* m\_tagDetector;

AprilTags::TagCodes m\_tagCodes;

bool m\_draw; // draw image and April tag detections?

bool m\_arduino; // send tag detections to serial port?

bool m\_timing; // print timing information for each tag extraction call

int m\_width; // image size in pixels

int m\_height;

double m\_tagSize; // April tag side length in meters of square black frame

double m\_fx; // camera focal length in pixels

double m\_fy;

double m\_px; // camera principal point

double m\_py;

int m\_deviceId; // camera id (in case of multiple cameras)

list<string> m\_imgNames;

cv::VideoCapture m\_cap;

int m\_exposure;

int m\_gain;

int m\_brightness;

Serial m\_serial;

**fonctions:**

// default constructor

**Demo()**

// changing the tag family

**void setTagCodes(string s)**

// parse command line options to change default behavior

**void parseOptions(int argc, char\* argv[])**

//初始化TagDetector，初始化图像显示窗口，打开串口与Arduino通信

**void setup()**

// 获取视频流

**void setupVideo()**

// ？？？ recovering the relative pose of a tag, TagDetection.getRelativeTranslationRotation，cv::solvePnP

**void print\_detection(AprilTags::TagDetection& detection) const**

detection.getRelativeTranslationRotation()

**//** detect April tags (requires a gray scale image) TagDetector->extractTags

**void processImage(cv::Mat& image, cv::Mat& image\_gray)**

m\_tagDetector->extractTags;

print\_detection(detections[i]);

detections[i].draw(image);

**//** Load and process a single image

**void loadImages()**

**//** Video or image processing?

**bool isVideo()**

// The processing loop where images are retrieved, tags detected, and information about detections generated

**void loop()**

while(ture)

processImage()

**main():**

Demo demo;

demo.parseOptions(argc, argv);

demo.setup();

if (demo.isVideo()) {

cout << "Processing video" << endl;

// setup image source, window for drawing, serial port...

demo.setupVideo();

// the actual processing loop where tags are detected and visualized

demo.loop();

}