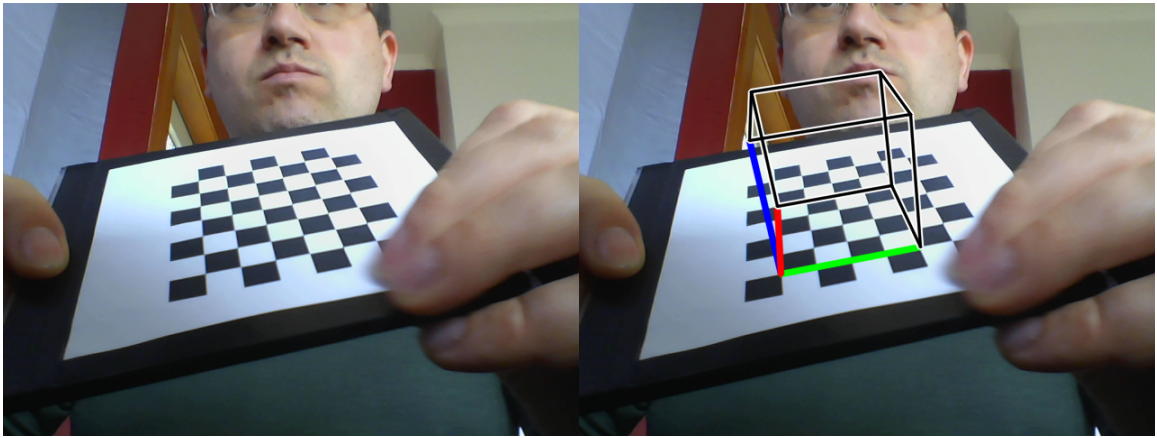
A Basic AR App

COSC450

Lecture 1

Augmented Reality Example



Code – Setup

```
#include <opencv2/opencv.hpp>

int main(int argc, char *argv[1]) {

  cv::VideoCapture cap;
  cap.open(argv[1]);

  CameraCalibration camCal;
  camCal.read(argv[2]);

  std::unique_ptr<Tracker> tracker;
  makeTracker(argv[4], tracker)) {

  cv::Mat frame;
  cap >> frame;

  CameraPose pose;
  pose = tracker->initialise(argv[3], frame, camCal);

  OpenCVBoxRenderer renderer(tracker->targetSize());
```

Code – Main Loop

```
bool done = false;
while (!done) {
  cap >> frame;

  pose = tracker->update(frame, camCal);

  done = renderer.render(camCal, pose, frame) == Event::QUIT;
}
```

Code – Tracker Class

```
class Tracker {
 public:

  Tracker() = default;
  Tracker(const Tracker&) = delete;
  virtual ~Tracker() = default;

  Tracker& operator=(const Tracker&) = delete;

  virtual CameraPose initialise(const std::string& targetFile,
                                const cv::Mat& frame,
                                const CameraCalibration& calibration) = 0;

  virtual CameraPose update(const cv::Mat& frame,
                            const CameraCalibration& calibration) = 0;

  virtual cv::Size targetSize() = 0;

};
```

Code – Chessboard Tracker

```
CameraPose ChessboardTracker::initialise(...) {

  // Set up a grid of points as our target model

  return update(frame, calibration);
}

CameraPose ChessboardTracker::update(...) {

  CameraPose cp;

  std::vector<cv::Point2f> corners;
  cp.tracked = cv::findChessboardCorners(frame, chessboardSize, corners);

  if (cp.tracked) {
    cv::solvePnP(chessboardPoints, corners,
                 calibration.K, calibration.d,
                 cp.rvec, cp.tvec);
  }

  return cp;
}
```

Code – Image-Based Tracker

```
SIFT_BF_Tracker::SIFT_BF_Tracker() :
  Tracker(), detector(cv::xfeatures2d::SIFT::create()),
  matcher(cv::BFMatcher::create()),
  targetKeyPoints(), targetDescriptors() {

}

CameraPose SIFT_BF_Tracker::initialise(...) {

  cv::Mat target = cv::imread(targetFile);

  detector->detectAndCompute(target, cv::noArray(), targetKeyPoints,
      targetDescriptors);

  matcher->add(targetDescriptors);
  matcher->train();

  return update(frame, calibration);
}
```

Code – Image-Based Tracker

```
CameraPose SIFT_BF_Tracker::update(...) {
  detector->detectAndCompute(frame, ..., frameKeyPoints, frameDescriptors);
  matcher->knnMatch(frameDescriptors, rawMatches, 2);

  for (auto& match : rawMatches) {
    if (match[0].distance < 0.8*match[1].distance) {
      targetPoints.push_back(targetKeyPoints[match[0].trainIdx].pt);
      framePoints.push_back(frameKeyPoints[match[0].queryIdx].pt);
    }
  }

  cv::Mat H = cv::findHomography(targetPoints, framePoints, mask, cv::RANSAC);
  for (size_t i = 0; i < targetPoints.size(); ++i) {
    if (mask[i]) {
      objectPoints.push_back(cv::Point3f(targetPoints[i].x, targetPoints[i].y, 0));
      imagePoints.push_back(framePoints[i]);
    }
 }

  CameraPose cp;
  cv::solvePnP(objectPoints, imagePoints, K, d, cp.rvec, cp.tvec);
  return cp;
}
```

So, what's there to do in COSC450?

There's quite a lot of 'magic' in that code...

- `cv::findChessboardCorners(...)`
- `cv::solvePnP(...)`
- `cv::projectPoints(...)`
- `detector->detectAndCompute(...)`
- `matcher->knnMatch(...)`
- `cv::findHomography(...)`

...we'll demystify as much of this as we can.

- We'll look at how to improve the performance of the tracker
- We'll look at better ways to render objects for AR
- We've got some time at the end to explore other topics