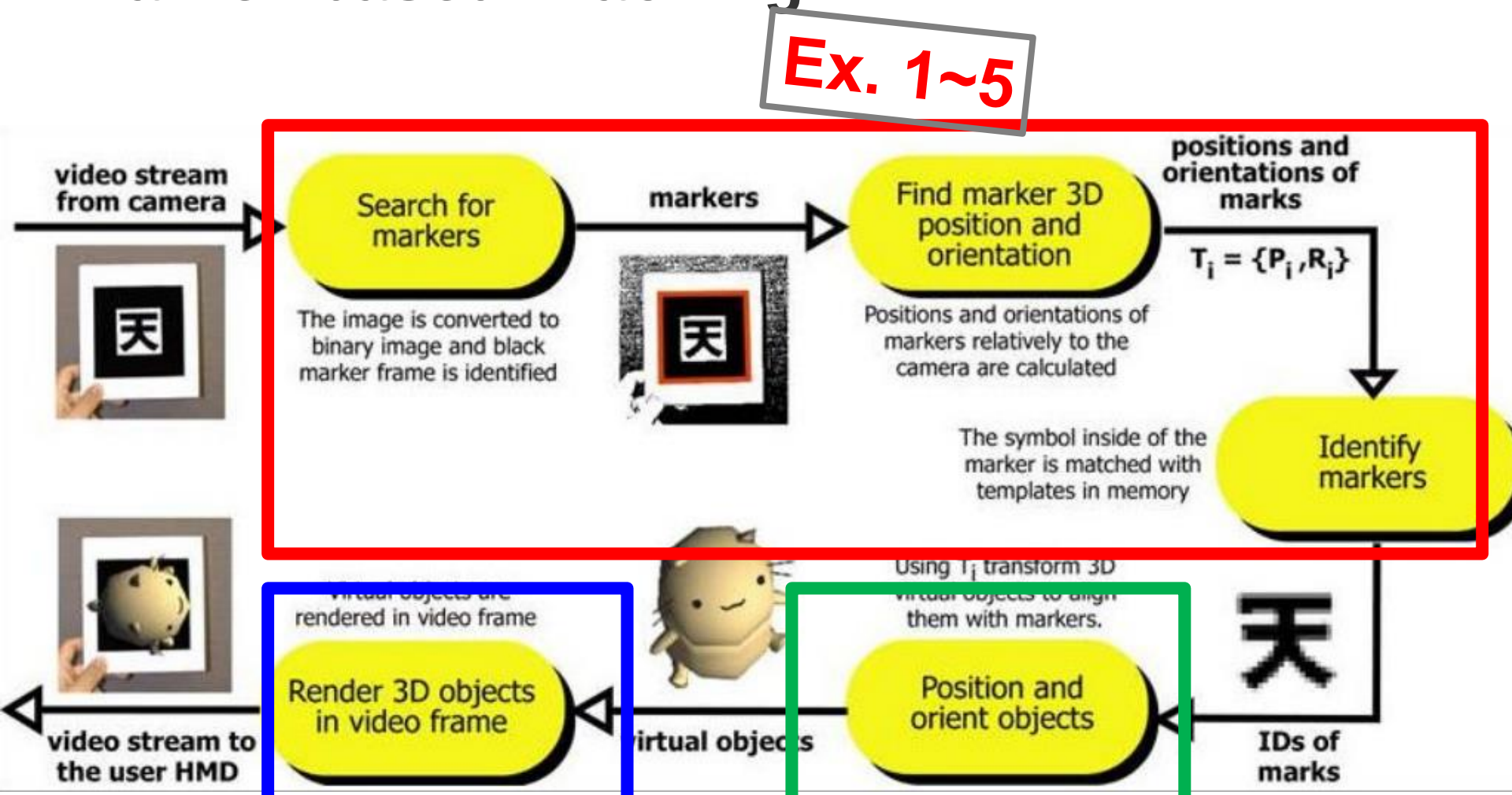
Introduction to Augmented Reality

Tutorial 2: Marker Tracking Part 2
Apr 25th 2016

Andreas Langbein, Adnane Jadid, David Plecher

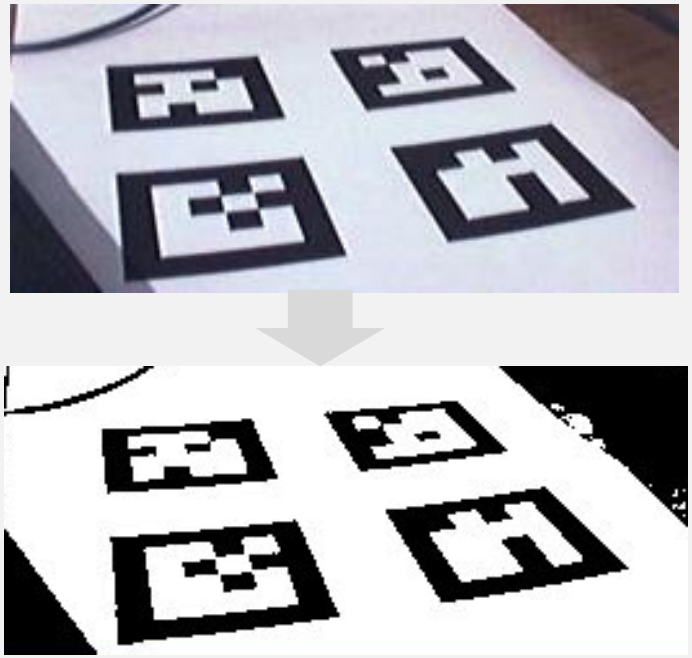
Marker-based Tracking



Solution for the Previous Tutorial

Ex. 1

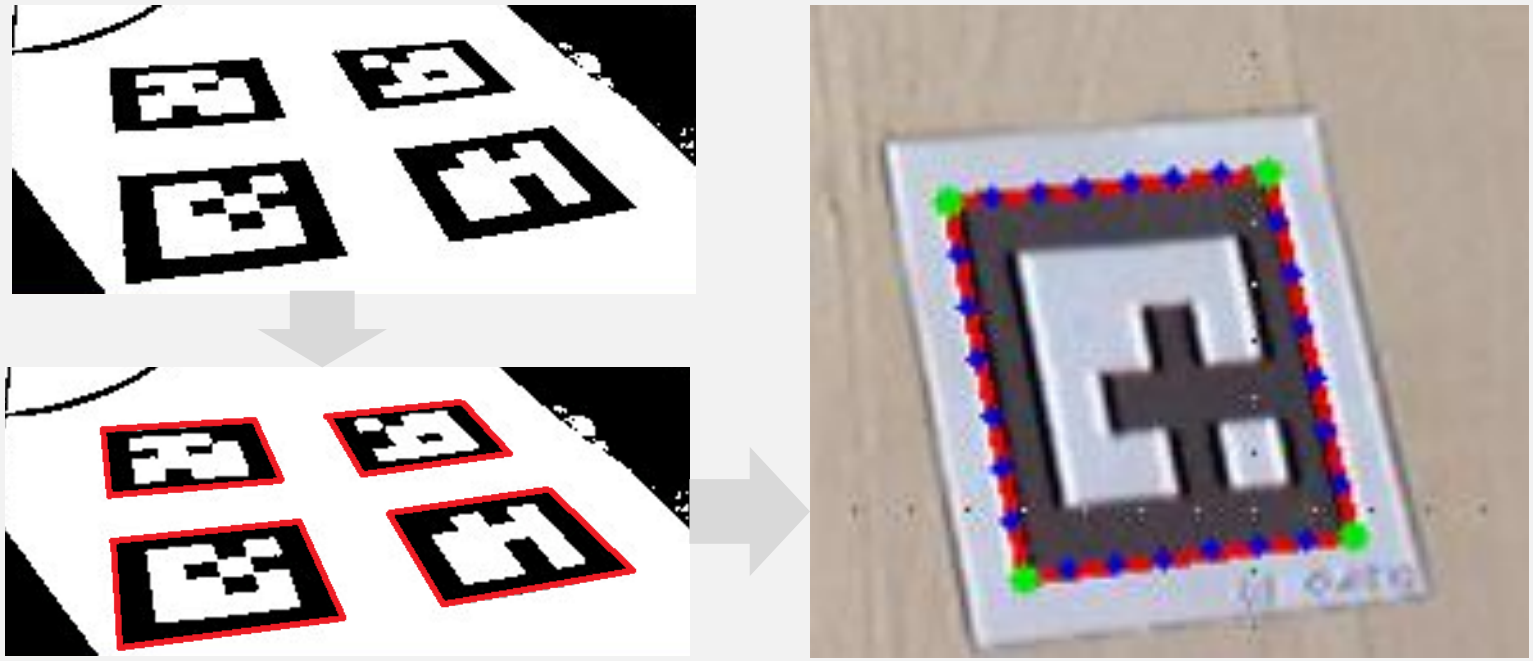
Preprocess image (thresholding)



Today's Tutorial

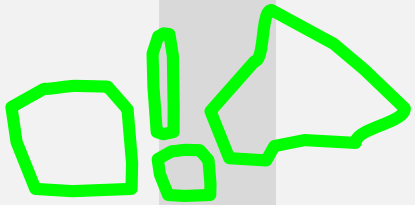
Ex. 2

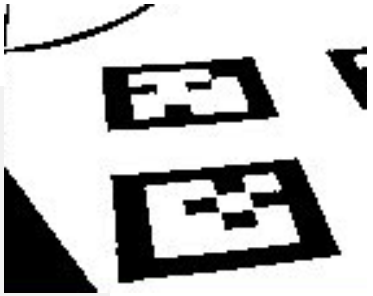
Find marker in 2D



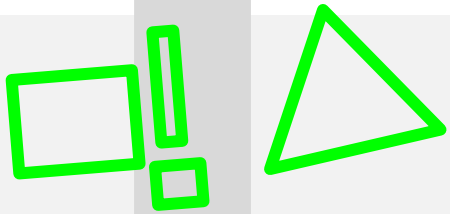
Detecting Connected Components

1. Find **contours**
   - `cvFindContours`

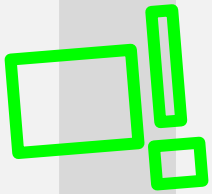




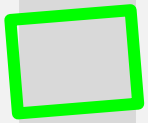
2. Polygonal approximation
   - `cvApproxPoly`

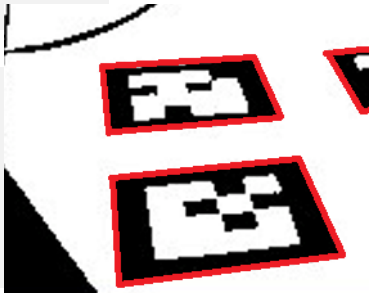


3. Selecting only those with four corners



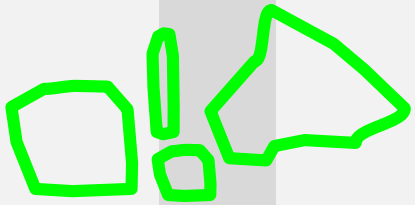
4. Filter tiny ones (noise)



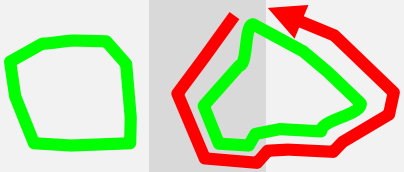


Detecting Connected Components (Option 2)

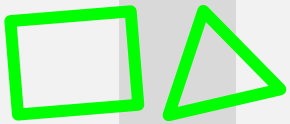
1. Find **contours**
   - `cvFindContours`



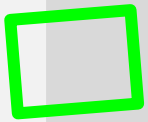
2. Filter tiny ones (noise)

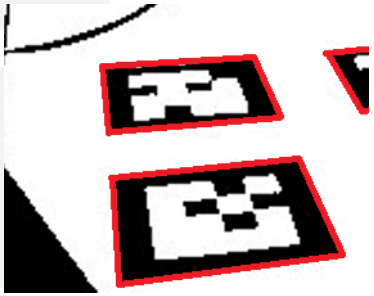


3. Polygonal approximation
   - `cvApproxPoly`



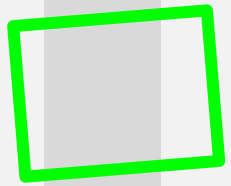
4. Selecting only those with four corners



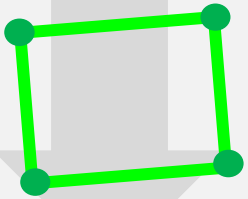


Detecting Connected Components

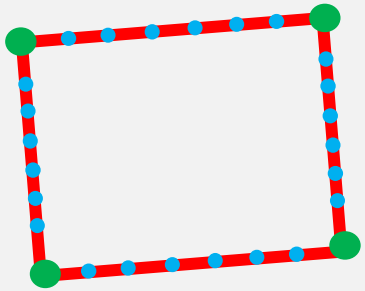
4. Selecting only those with four corners



5. Mark corner points in picture (green circle)



6. Divide edges into seven intervals and mark six delimiters



Sketch Solution for: 1. Find contours

OpenCV (1.x API) has its own heap management

- Create heap (on startup)

```
CvMemStorage* memStorage = cvCreateMemStorage();
```

- Re-initialize heap (at end of processing loop)

```
cvClearMemStorage (memStorage);
```

- Release heap (program exit)

```
cvReleaseMemStorage (&memStorage);
```

i.e. dynamically growing data structures

Why not 2.x APIs?

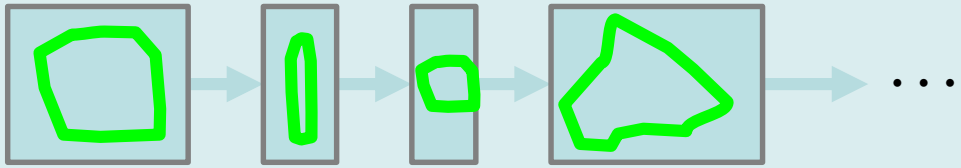
- An equivalent conversion has a performance issue:

```
std::vector< std::vector<cv::Point> > mem;
```

Sketch Solution for Exercise 2

Contour detection

```
CvSeq* contours;
cvFindContours( iplThreshold, memStorage, &contours,
          sizeof(CvContour), CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE);
for ( ; contours; contours = contours->h_next ){
```




```
}
```

- Use `cvFindContours`

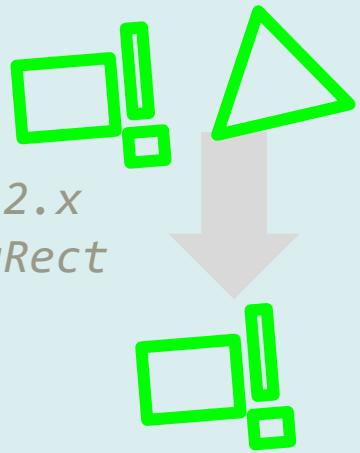
Why not 2.x APIs?

- `cv::findContours`
  takes `std::vector< std::vector<cv::Point> >`
  and internally use cvFindContours....

Sketch Solution for Exercise 2

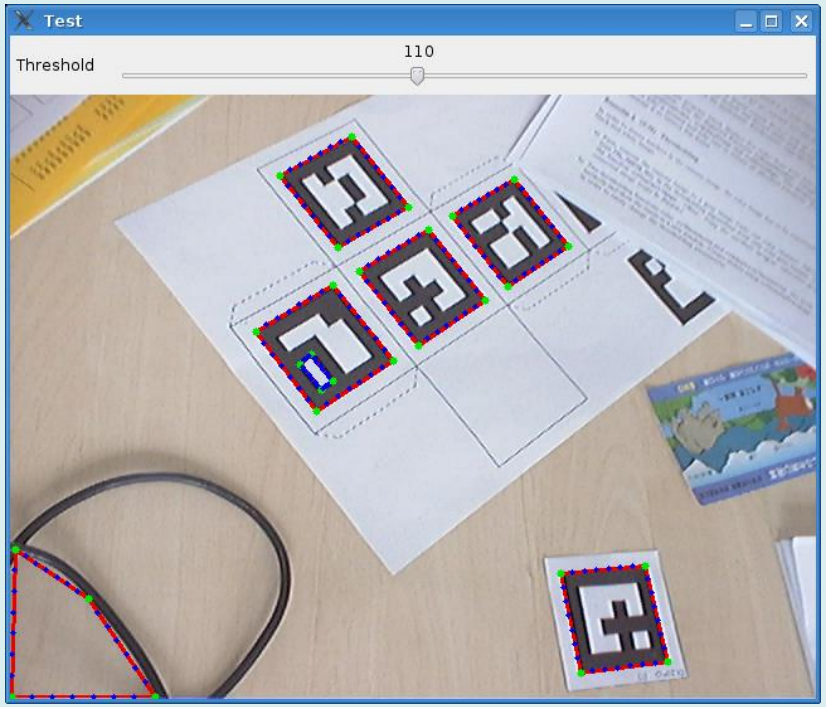
Marker detection

```
#include <opencv2/imgproc/imgproc_c.h>

CvSeq* result = cvApproxPoly( contours, sizeof(CvContour), memStorage,
            CV_POLY_APPROX_DP, cvContourPerimeter (contours)*0.02, 0 );


// Check size
cv::Mat result_ = cv::cvarrToMat(result); /// API 1.X to 2.x
cv::Rect r = cv::boundingRect(result_); // or cv::minAreaRect
…

// then – next slide
```



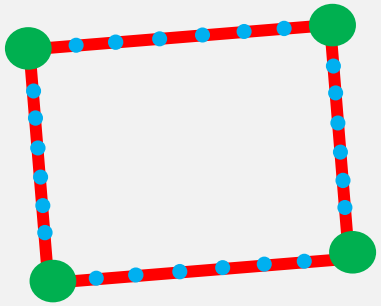
Sketch Solution for Exercise 2

```
// Only act on rectangles
if (result->total != 4) continue;

const cv::Point *rect
     = (const cv::Point*) result_.data;
int npts = result_.rows;
// draw the polygon
cv::polylines(...);
for (int i=0; i<4; ++i) {
    cv::circle (...);
    ...
    for (int j=1; j<7; ++j) {
        ...
        cv::circle(...);
    }
}
```



Homework

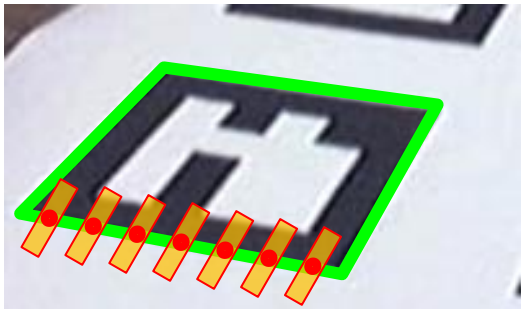
1. Programming part:
   Find and mark rectangles

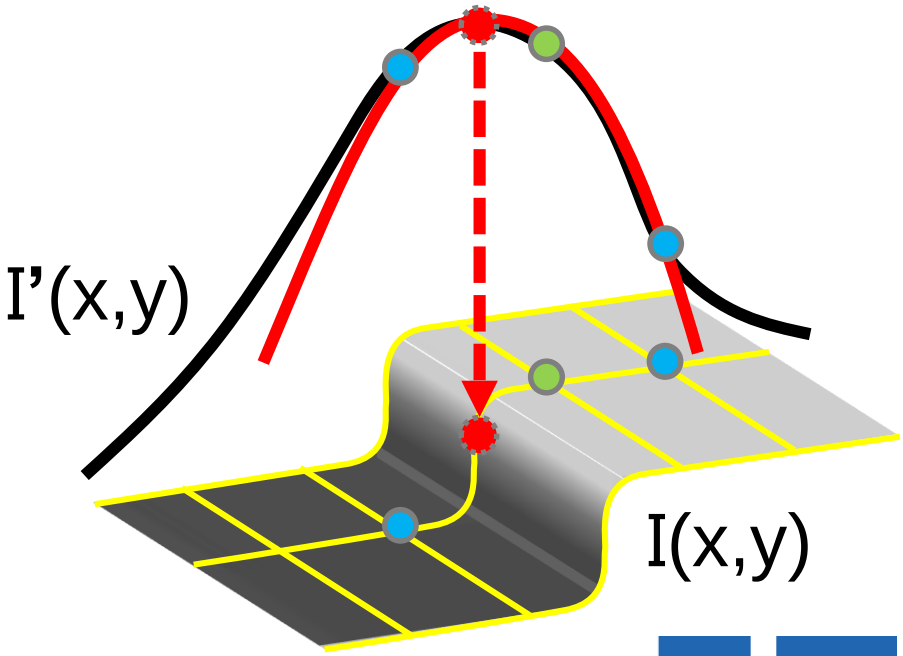


2. Theoretical part:
   Please find those questions on the Homework sheet on Moodle

Spoiler of the next tutorial

1. Make deeper inspection of the six interval points (Sobel operator; extract stripes; detect maximum at subpixel accuracy)
2. Fit lines through all points and mark in picture (green lines)
3. Compute corner points
4. Rectify contained image





That‘s it…

- Questions?