Self-parking System Based on General Hough Transformation

肖书奇　　刘瑞铭

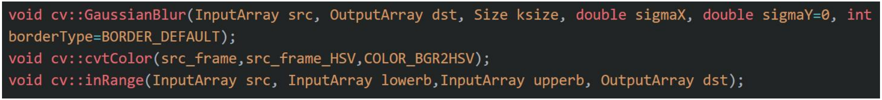
December 12, 2020

Overview

1. Color Segmentation
2. IPM(Inverse Perspective Mapping)
   - Homograph Matrix
   - Functions in OpenCV
3. GHT(General Hough Transformation)
   - ϕ-table
   - Vote
4. Locomotion
   - P and Bang-bang Controllers
5. Summary

HSV Color Segmentation

1. Denoising.
2. RGB to HSV.
3. Color segmentation.

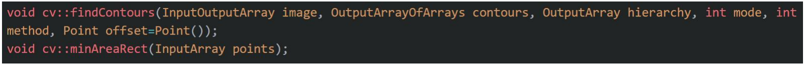


```
void cv::GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int
borderType=BORDER_DEFAULT);
void cv::cvtColor(src_frame,src_frame_HSV,COLOR_BGR2HSV);
void cv::inRange(InputArray src, InputArray lowerb,InputArray upperb, OutputArray dst);
```

When there is no distractions...

1. Binarize
2. Extract contours
3. Cover the area

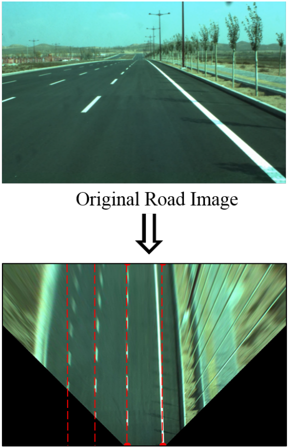
Just move to the center of the area.



```
void cv::findContours(InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int
method, Point offset=Point());
void cv::minAreaRect(InputArray points);
```

But what if your shoes have the same color?

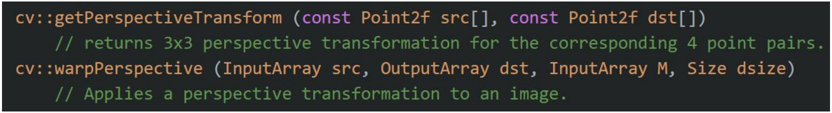
IPM: Homograph Matrix



Original Road Image

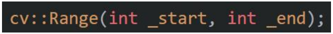
$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \qquad \text{DoF} = 8$$

IPM: Functions in OpenCV



```
cv::getPerspectiveTransform (const Point2f src[], const Point2f dst[])
    // returns 3x3 perspective transformation for the corresponding 4 point pairs.
cv::warpPerspective (InputArray src, OutputArray dst, InputArray M, Size dsize)
    // Applies a perspective transformation to an image.
```

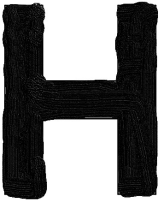
Perspective Transform



```
cv::Range(int _start, int _end);
```

Crop the ROI

GHT: Construct the ϕ-table



Template

$\longrightarrow$

ϕ	r	α
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

ϕ-table

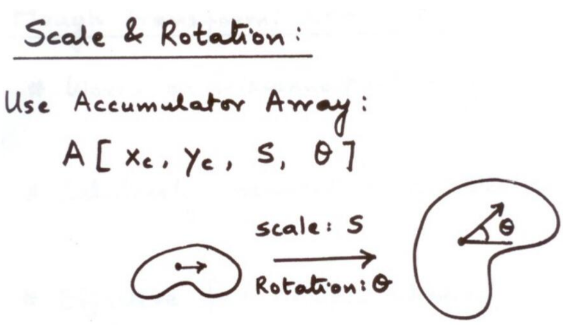
GHT: Vote

1. Check the qualifications for voters (1st column of ϕ-table).
2. Let's vote. (2nd, 3rd column of ϕ-table).

$$\begin{cases} x_c = x_i + r_k^i \cos(\alpha_k^i) \\ y_c = y_i + r_k^i \sin(\alpha_k^i) \end{cases} \quad , \quad A[x_c, y_c] + +$$

But this can't detect scaled or rotated pattern!

GHT: Vote



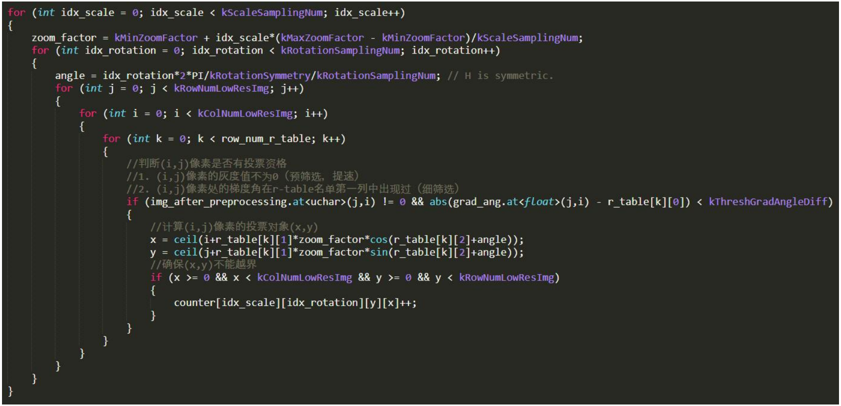
Scale & Rotation:

Use Accumulator Array:

A[x_c, y_c, S, θ]

$$\begin{cases} x_c = x_i + r_k^i s \cos(\alpha_k^i + \theta) \\ y_c = y_i + r_k^i s \sin(\alpha_k^i + \theta) \end{cases} , \quad A[x_c, y_c, s, \theta] + +$$

GHT: Vote



```
for (int idx_scale = 0; idx_scale < kScaleSamplingNum; idx_scale++)
{
    zoom_factor = kMinZoomFactor + idx_scale*(kMaxZoomFactor - kMinZoomFactor)/kScaleSamplingNum;
    for (int idx_rotation = 0; idx_rotation < kRotationSamplingNum; idx_rotation++)
    {
        angle = idx_rotation*2*PI/kRotationSymmetry/kRotationSamplingNum; // H is symmetric.
        for (int j = 0; j < kRowNumLowResImg; j++)
        {
            for (int i = 0; i < kColNumLowResImg; i++)
            {
                for (int k = 0; k < row_num_r_table; k++)
                {
                    //判断(i,j)像素是否有投票资格
                    //1. (i,j)像素的灰度值不为0（预筛选，提速）
                    //2. (i,j)像素处的梯度角在r-table名单第一列中出现过（细筛选）
                    if (img_after_preprocessing.at<uchar>(j,i) != 0 && abs(grad_ang.at<float>(j,i) - r_table[k][0]) < kThreshGradAngleDiff)
                    {
                        //计算(i,j)像素的投票对象(x,y)
                        x = ceil(i+r_table[k][1]*zoom_factor*cos(r_table[k][2]+angle));
                        y = ceil(j+r_table[k][1]*zoom_factor*sin(r_table[k][2]+angle));
                        //确保(x,y)不能越界
                        if (x >= 0 && x < kColNumLowResImg && y >= 0 && y < kRowNumLowResImg)
                        {
                            counter[idx_scale][idx_rotation][y][x]++;
                        }
                    }
                }
            }
        }
    }
}
```

5-layer Loop

Locomotion

1. P controller for angular velocity.
2. Bang-bang controller for linear velocity.
3. Stop trigger.

References

Ioannis Gkioulekas (2020)
CMU 16-385 Computer Vision §5.4
General Hough Transform

Appraise Ourselves

Pros

- Adapable.
- Deals with occlusion well.
- Detects multiple instances.
- Real DIP, not ML or DL.
- 100% original codes(GHT).

Cons

- Tiny FoV due to rigid IPM.
- High computational complexity.
- Hard to to set parameters.

The End