Smart Scan Android Application

Dan Abigadol

Eitan Netzer

Supervisor: Amir Geva

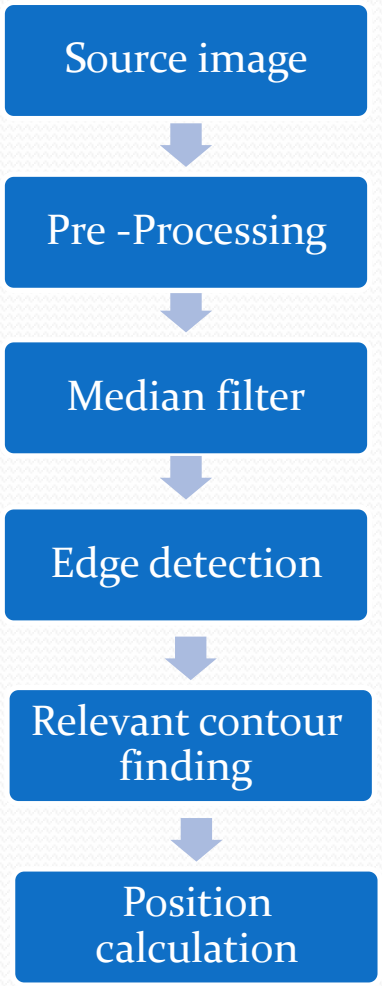
Intro

Developing and implementing an android application that assists the user in placing the camera in an optimal position for scanning a document.

Goals

- Image acquisition using the android API.
- Segmentation of the document inside the acquired image.
- Enabling scanning of the document in case of good positioning.

Algorithm overview



Pre-Processing methods choosing

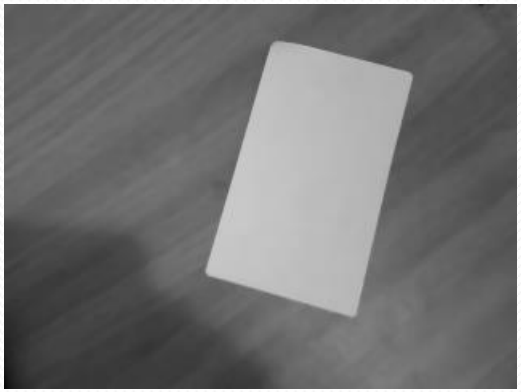
- 4 Pre-Processing methods:
  - Normal
  - Normal using equalizer
  - Inner mask
  - Outer mask

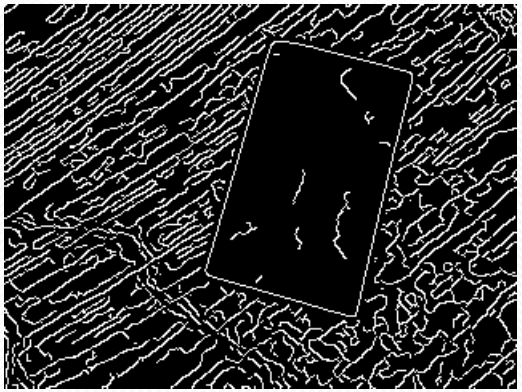
  Each frame gets pre-processed by only one method out of the four. In case that a rectangle is not found using the chosen method we switch to the next method in a cyclic manner until a rectangle is found.

Normal Method

- Our first optional method is using gray level image, that can be easily extracted from the android camera (ycbcr format).
- This method should deal with most of the cases.

Normal Example





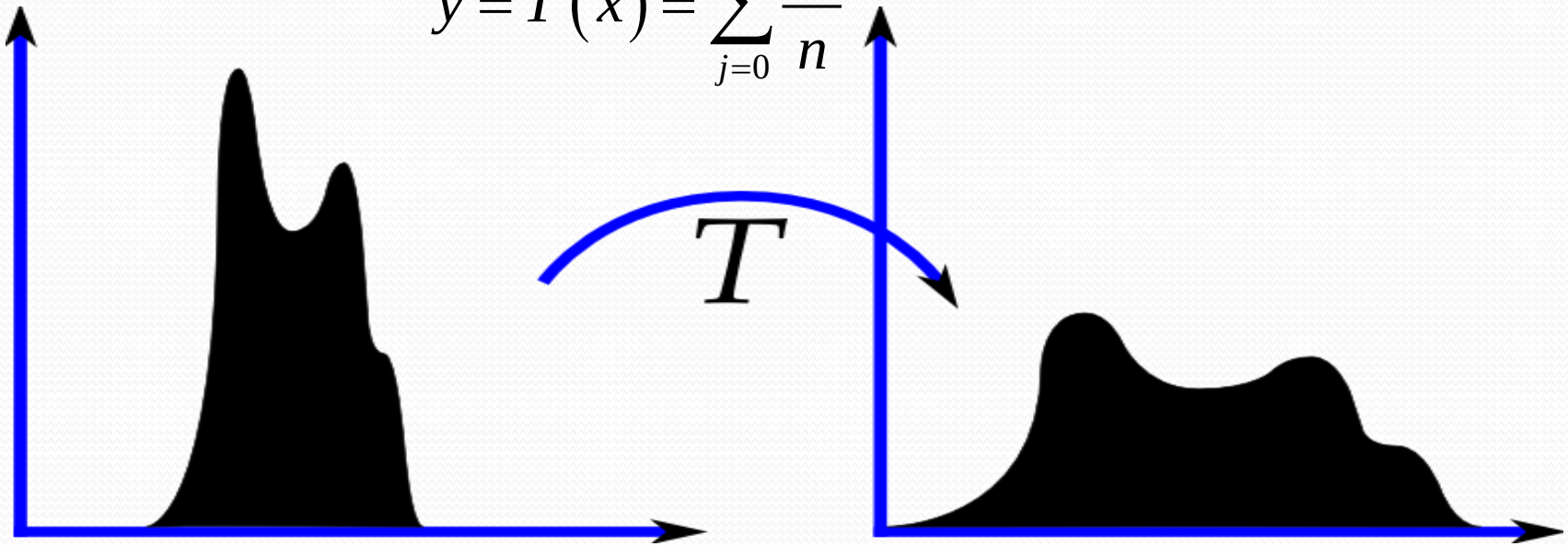
Normal with equalizer

- Using normal gray level image and filtering it using histogram equalizer. This path should deal with most of the cases where the image has a low contrast.

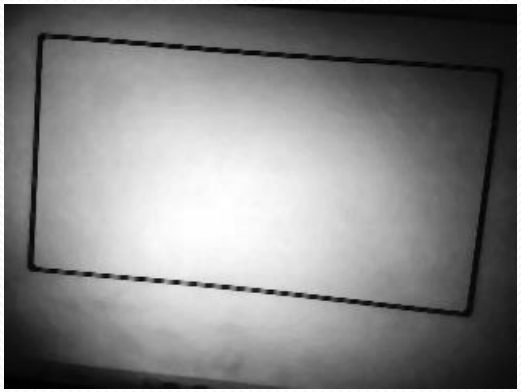
Histogram equalizer

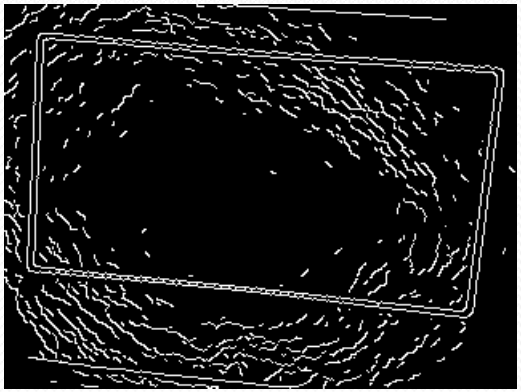
- The histogram equalizer is a transformation that convert the histogram to be as linear as possible using the whole dynamic domain
-

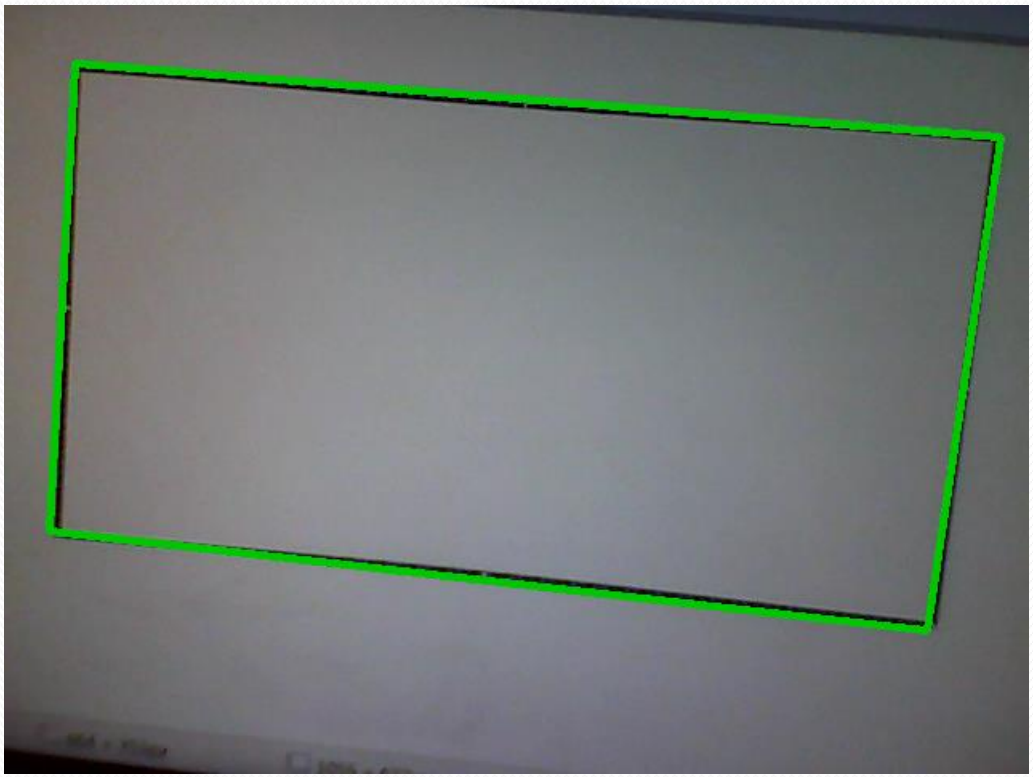
$$y = T(x) = \sum_{j=0}^{x} \frac{n_j}{n}$$



Normal with Equalizer Example







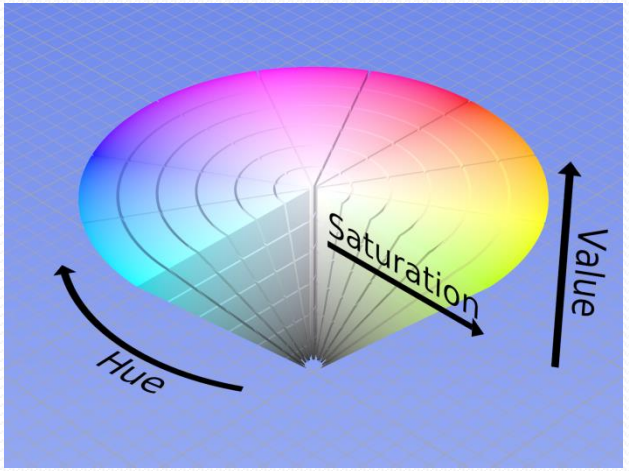
RGB to HSV conversion

$$V \leftarrow \max(R, G, B)$$

$$S \leftarrow \begin{cases} \frac{V - \min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} 60(G - B)/(V - \min(R, G, B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - \min(R, G, B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - \min(R, G, B)) & \text{if } V = B \end{cases}$$

Choosing saturation



Inner mask method

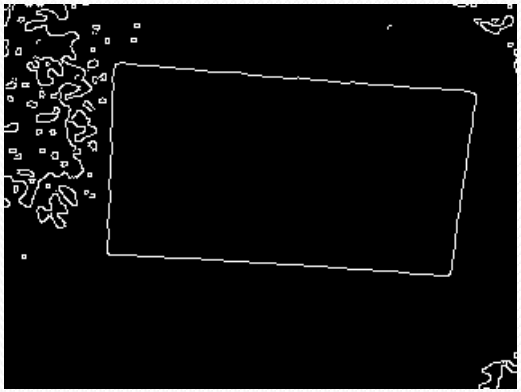
- colored mask HSV image instead of the gray-level one:
  - Conversion of RGB image to HSV image.
  - Histogram calculation of the central region of the image.
  - Finding max value
  - Setting pixels according to the value found
  - The result is a mask of the original image
- Noisy cases

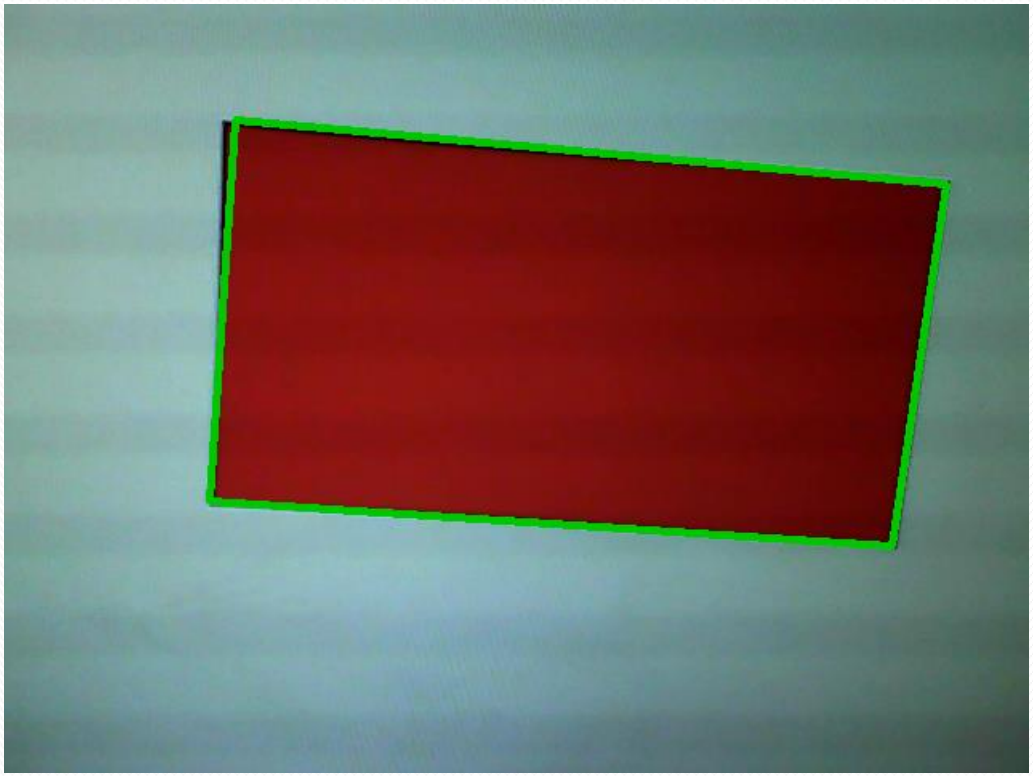
Inner Mask Example

Outer mask method

- This method is identical to the previous one except for calculating the histogram according to the surrounding part of the image.

Outer Mask Example





Median filter

This filter replaces the pixel value with the median value of all the pixels that surrounds it.

This filter is common tool for reducing noise, can be useful for eliminating the text from the picture and fixing mistake in the masking process.

Edge detection

Generally the edge detection is done using HPF

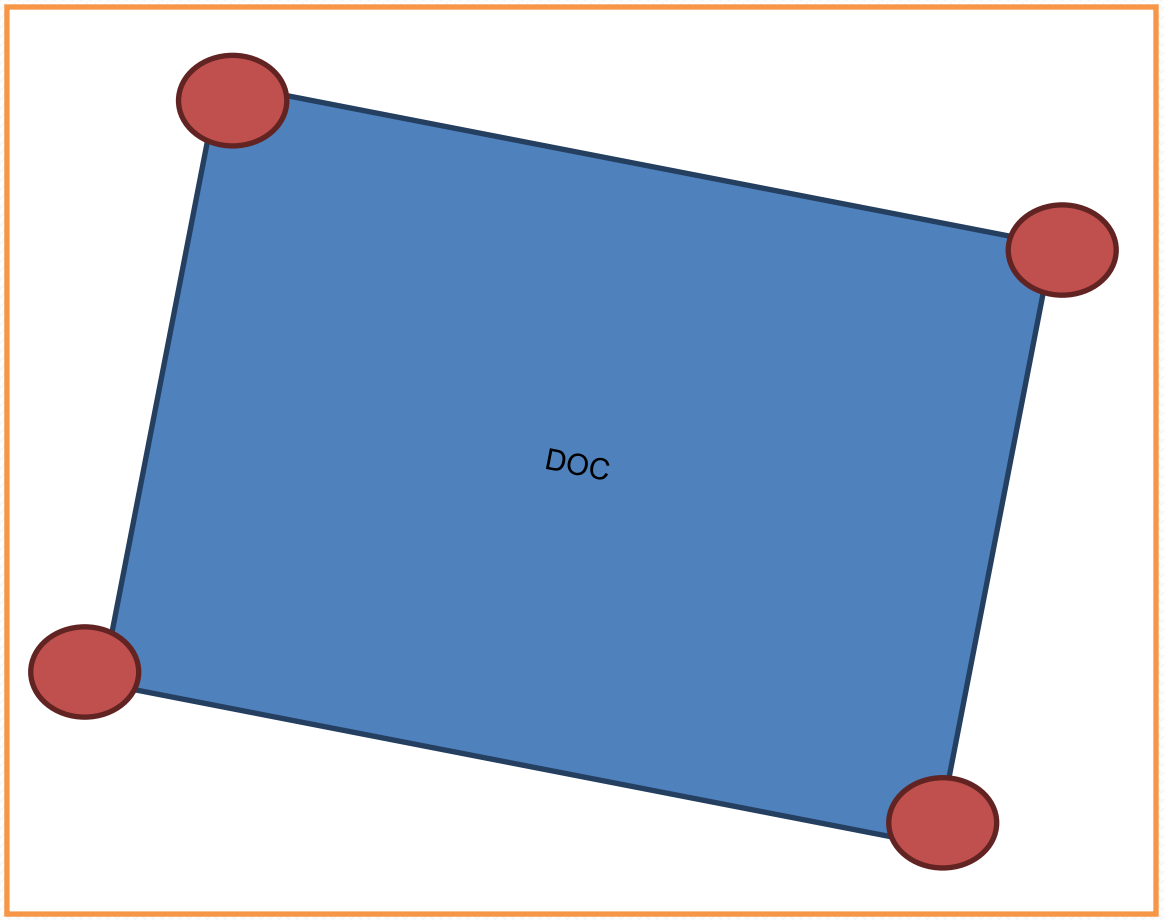
Because of noises on the image we use Canny algorithm:

- Phase 1: noises filtering using Gaussian filter.
- Phase 2: gradient calculation for X, Y axes .
- Phase 3: selecting all edges above the upper threshold.
- Phase 4: selecting all neighboring edges which are above the lower threshold.

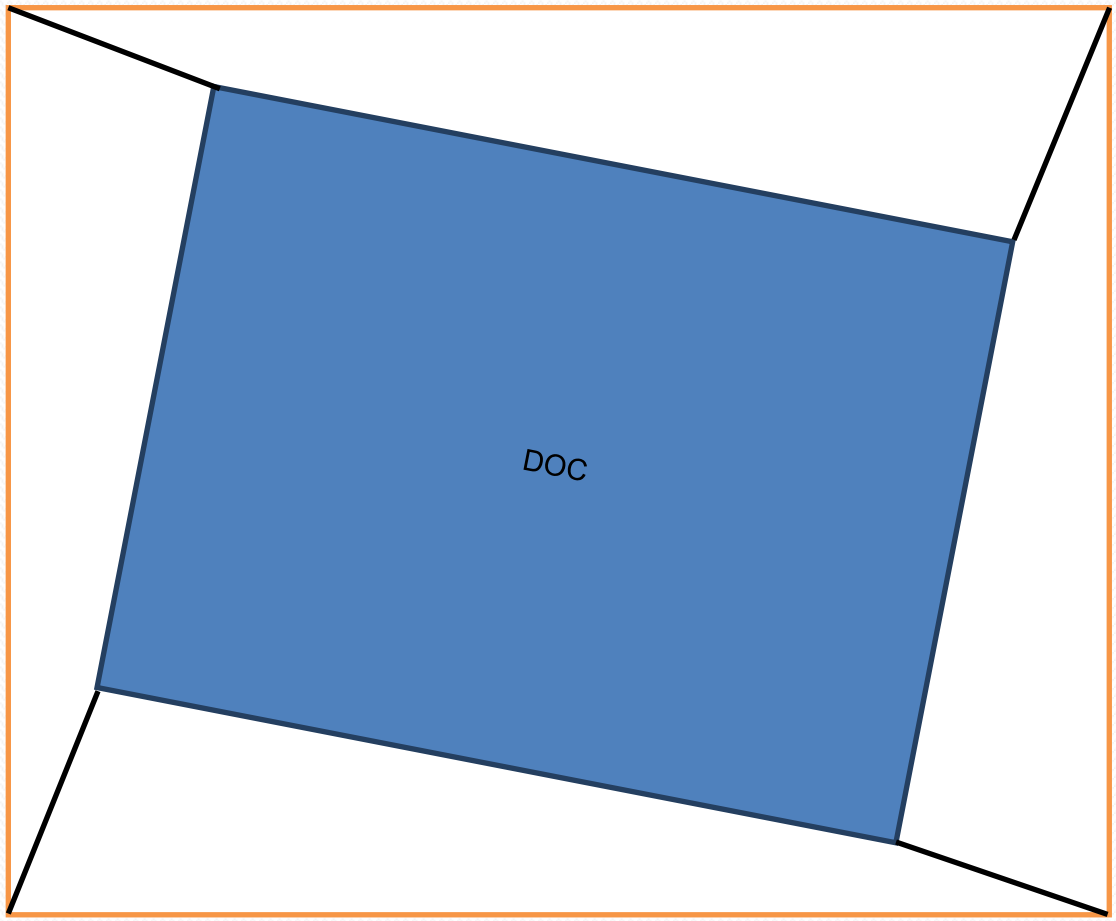
Contour finding

- Choosing biggest contour
- Finding four corners:
  - Choosing the extreme points
  - Choosing points according to the distance from the image frame and candidates for corners according to angle

Choosing the extreme points



Choosing points according to the distance from the image frame



perspective projection

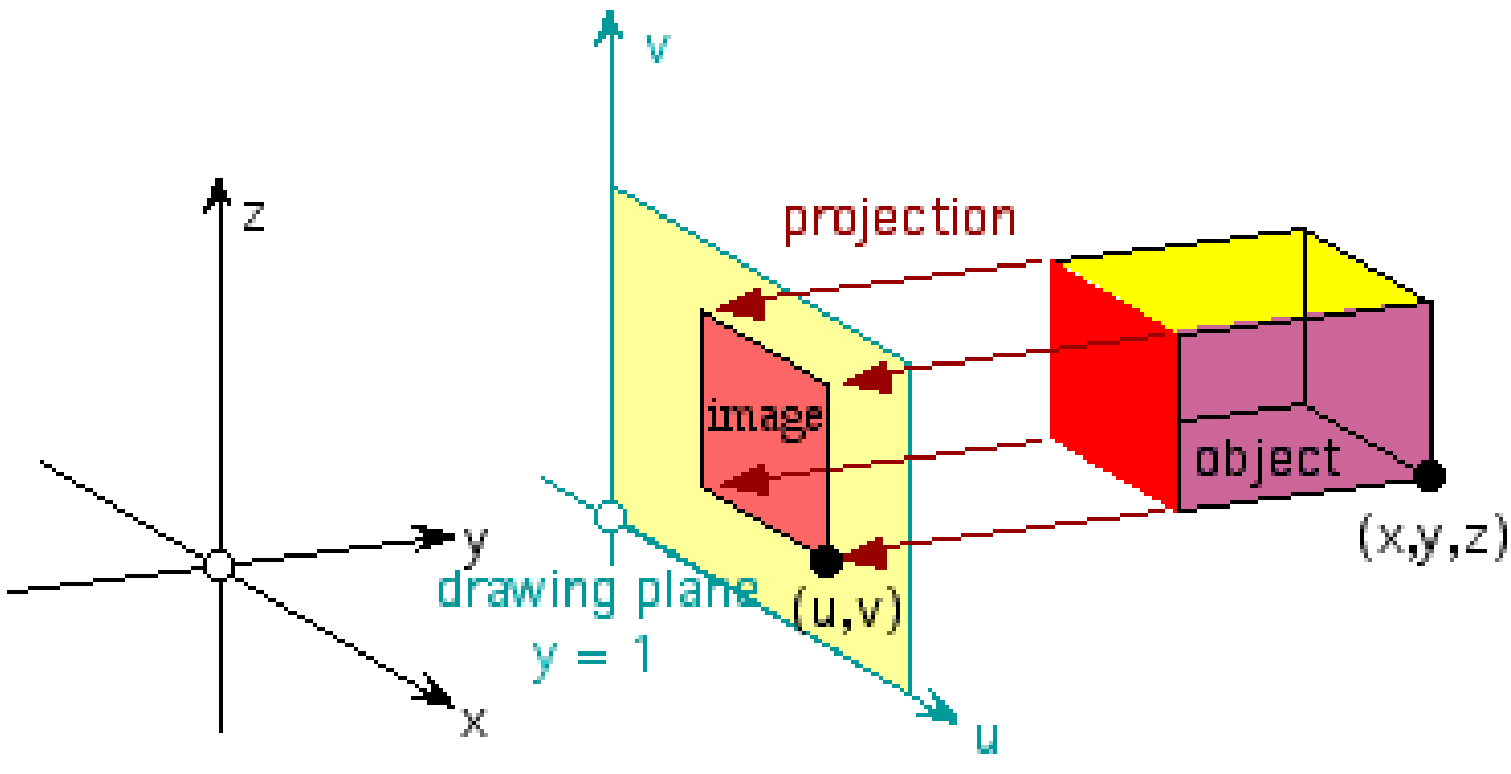
Simple model projecting from 3D coordinates to 2D coordinates

In inhomogeneous

$$\bar{x} = P_z(p) = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

Position calculation (1)

Perspective projection – projection of 3D object on a 2D canvas.



Position calculation (2)

- Projected lines distortion
- Calculating rectangle angles
- Calculating opposite edges ratio

Perspective projection- proof (1)

$$Given:$$

$$\text{Rectangle};p_i;i\in\{0,1,2,3\}$$

$$P(p_i)\ \text{is Rectangle IFF}$$

$$\vec{A}=P(p_0)-P(p_1)$$

$$\vec{B}=P(p_0)-P(p_2)$$

$$\vec{A}\perp\vec{B}\Leftrightarrow\vec{A}\cdot\vec{B}=0$$

$$\Leftrightarrow\overbrace{(x_0-x_1)(x_0-x_2)+(y_0-x_1)(y_0-y_2)}^{=0}+(z_0-z_1)(z_0-z_2)=0$$

$$\ldots\Rightarrow$$

$$(z_0-z_1)(z_0-z_2)=0\wedge(z_1-z_0)(z_1-z_2)=0$$

$$\wedge(z_2-z_3)(z_2-z_0)=0\wedge(z_3-z_2)(z_3-z_1)=0$$

$$\Rightarrow\left\{\begin{array}{c}\{\{z_0=z_1\}\vee\{z_0=z_2\}\}\wedge\{\{z_0=z_1\}\vee\{z_1=z_2\}\}\\ \wedge\\ \{\{z_2=z_3\}\vee\{z_0=z_2\}\}\wedge\{\{z_1=z_3\}\vee\{z_3=z_2\}\}\end{array}\right\}$$

Perspective projection- proof (2)

$$A)\{z_0 = z_2\} \wedge \{z_1 = z_2\} \Rightarrow \{z_1 = z_3\} \vee \{z_3 = z_2\} \Rightarrow \boxed{z_0 = z_1 = z_2 = z_3}$$

$$B)\{z_0 = z_1\} \wedge \{z_2 = z_3\} \Rightarrow \boxed{z_0 = z_1, z_2 = z_3}$$

$$\left(\frac{x_0 - x_1}{z_0}\right)^2 + \left(\frac{y_0 - y_1}{z_0}\right)^2 = \left(\frac{x_2 - x_3}{z_2}\right)^2 + \left(\frac{y_2 - y_3}{z_2}\right)^2$$

$$\Rightarrow z_0 = z_2$$

$$\Rightarrow \boxed{z_0 = z_1 = z_2 = z_3}$$

$$Q.E.D$$

Time complexity

- Speed is important for user experience:
  - Image size reduction
  - Simple filters
  - Canny efficiency (choosing thresholds)
  - Finding biggest contour

CPU (Android) time table

Method	RetrieveContours			FindBigestClosed Contour	match points	Total time
	Preprocessing	Canny	findContours			
Normal	0.331023			0.00033	0.000931	0.332994
	0.172901	0.146605	0.011193			
Equalizer	0.299037			0.000275	0.002039	0.301602
	0.173725	0.111399	0.011954			
Inner Mask	0.578197			0.000515	0.004095	0.584809
	0.421027	0.12335	0.01686			
Outer Mask	0.429816			0.000637	0.006028	0.436936
	0.314266	0.104561	0.010025			

*time is given in seconds

Skills gained

- Design and implementation of computer vision algorithms, under time constraints
- Android programming
- Using OpenCV libraries and combining managed code with native code. (Java, JNI, C++)

Future research

- The option of masking according to the Hue instead or along the saturation
- Working with PDF files and adding Optical-character-recognition (OCR)
- Using transformation on the result image to improve alignment as post processing
- Using post processing to improve image quality and readability
- Using learning algorithms to classify the documents into different types

Bibliography

[1] Matlab documentation

[2] Opencv documentation

[3] Android developer

[4] Computer Vision: Algorithms and Applications; Richard Szeliski

Fin

Thanks to Amir