**INFOIP Image Processing - Assignment 3**

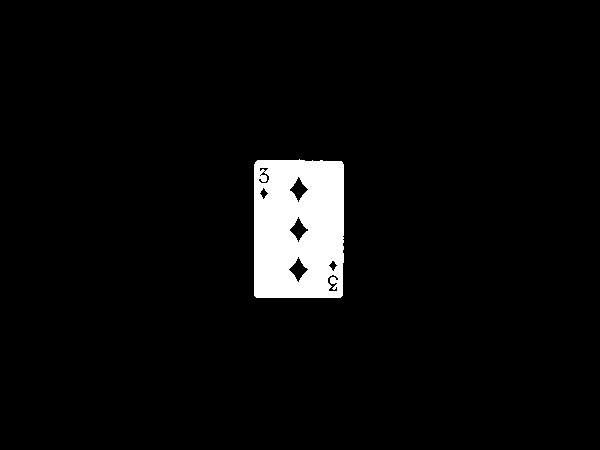
## Part 1: Object to be detected: Playing Cards

Our object of choice is playing cards from a standard 52 card deck. The main goal will be to detect the cards themselves and therefore being able to count how many cards are in a given image. One of the possible refinements would be detecting the suit and rank of each card. We made this choice of object because playing cards appeared to be easy enough to detect in images with very little distractions and proportionally harder the more noise and distractions you add. Furthermore the challenge of detecting rank and suit seemed appropriate as an refinement option.

|  |  |
| --- | --- |
| **Criterium** | **Possible Values** |
| Minimum/Maximum size | All images are of size < 600x600, object size varies and doesn’t need to be defined |
| Lighting variations | Indoors, Artificial Light, No direct sunlight |
| Rotation variations | All possible rotations, front facing up |
| Occlusion | No occlusion, partial occlusion |
| Other | Only background noise, cards may be viewed from an angle, no white backgrounds |

## Part 2: Pipeline

-Phase 1

In the first phase of our pipeline we get rid of background noise and output a binary image where the white pixels indicate the white parts of a card. This is achieved by first converting the input image to grayscale and applying our custom threshold to it, which looks for the first pixel value that at least 10% of all pixels share, going from 255 to 0. After this value has been found we threshold at the pixel value 30 % below it. This has proven to give good results for playing cards, since we can expect to have large areas of the image be white. After thresholding the image we apply opening to a compare Image and geodesic dilation of the threshold image with the compare image to get rid of any background noise that might still be left. A simple example of the application of Phase 1 can be seen in Image 1 and Image 2.

1 Image 1: Input

Image 2: Output Phase 1

-Phase 2

-Phase 3

## Part 3: Parameters

An important part of every program are the defined parameters by which the measures are taken, since they directly influence the outcome of the pipeline and are crucial for getting the right result out of a multitude of input images.

The first 2 important parameters are used in our custom automatic threshold. First we look for the first pixel value that is shared by at least **10%** of all pixels, to determine the predominant white value, afterwards we threshold on that value times **0.7** to add a bit of leeway. This choice of values was empirically determined and appeared to produce great results, when looking for playing cards in our images. A known problem that results out of this method and choice of parameters is the restriction that we cannot use images with white backgrounds, since we assume that the white of the cards is the “whitest” color in the given image.

Another is important parameter is the sampling accuracy used for calculating the Hough graph, which would in our case be **600**. This means that we draw **600** lines through every edge pixel in order to determine the lines going through the image. A sample size this big allows us to inspect almost every possible rotation. Accompanying the accuracy is the threshold used to determine the strongest value; it has to be chosen to fit our accuracy, since drawing more lines means more possible matches. For this we chose the value **100** as we found that it produced the most desirable results.

## Part 4: Example Outputs and Discussion

Lelelel