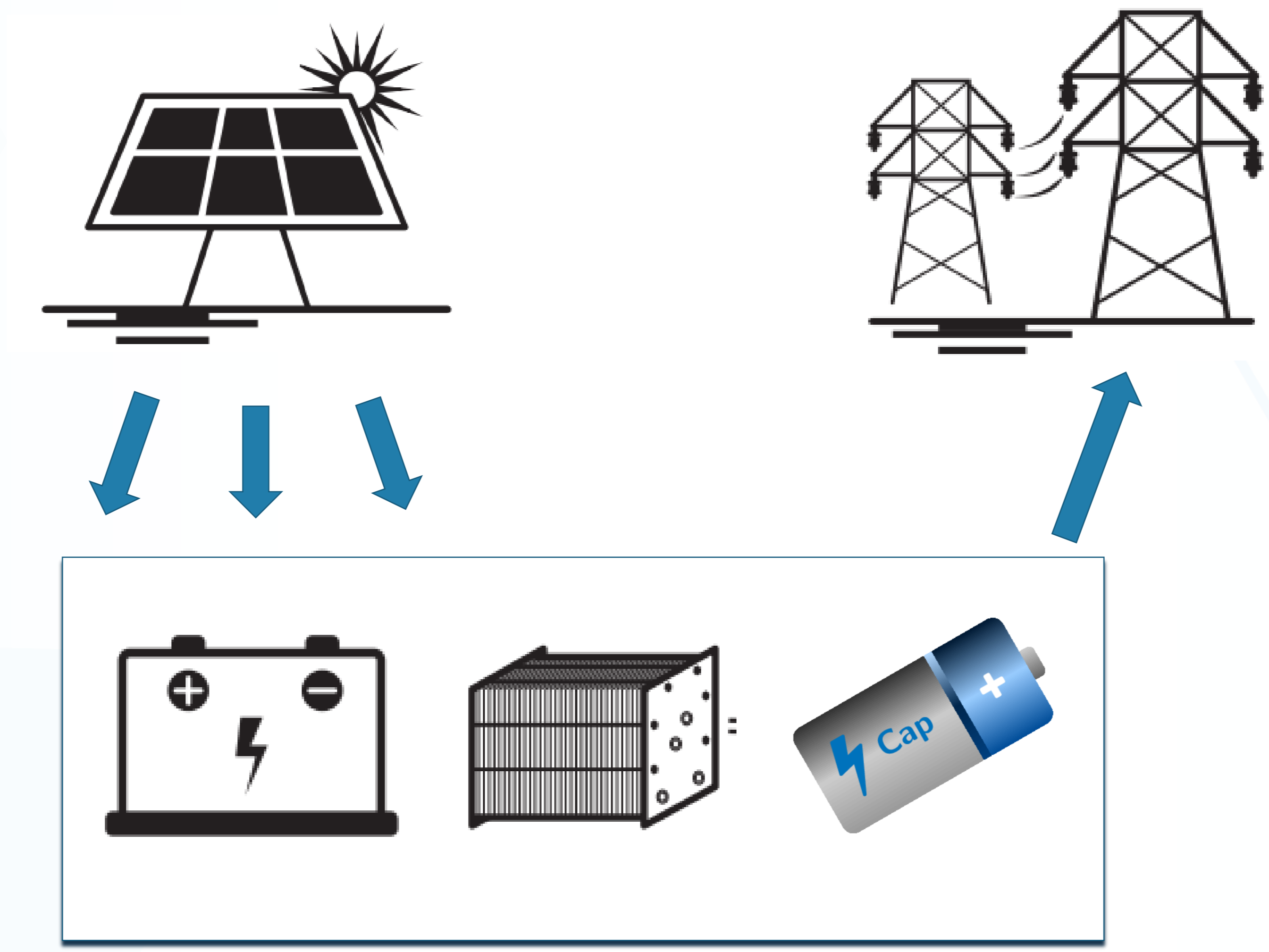


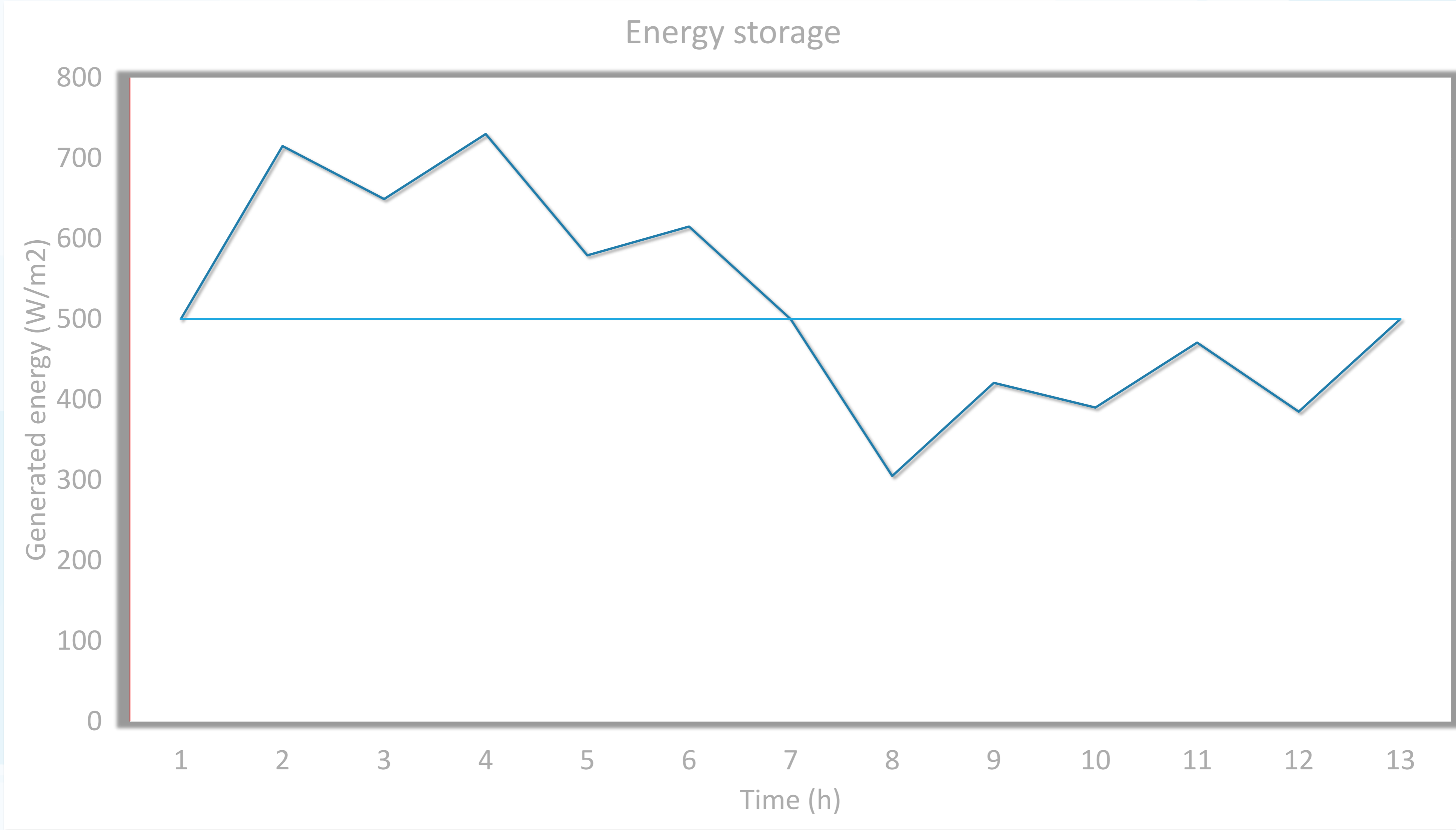
Controlling solar electric system using Java

Aleksander Radovan, HUIAK, King ICT, TVZ, VVG, RIT Croatia, Algebra
Branko Mihaljević, HUIAK, RIT Croatia

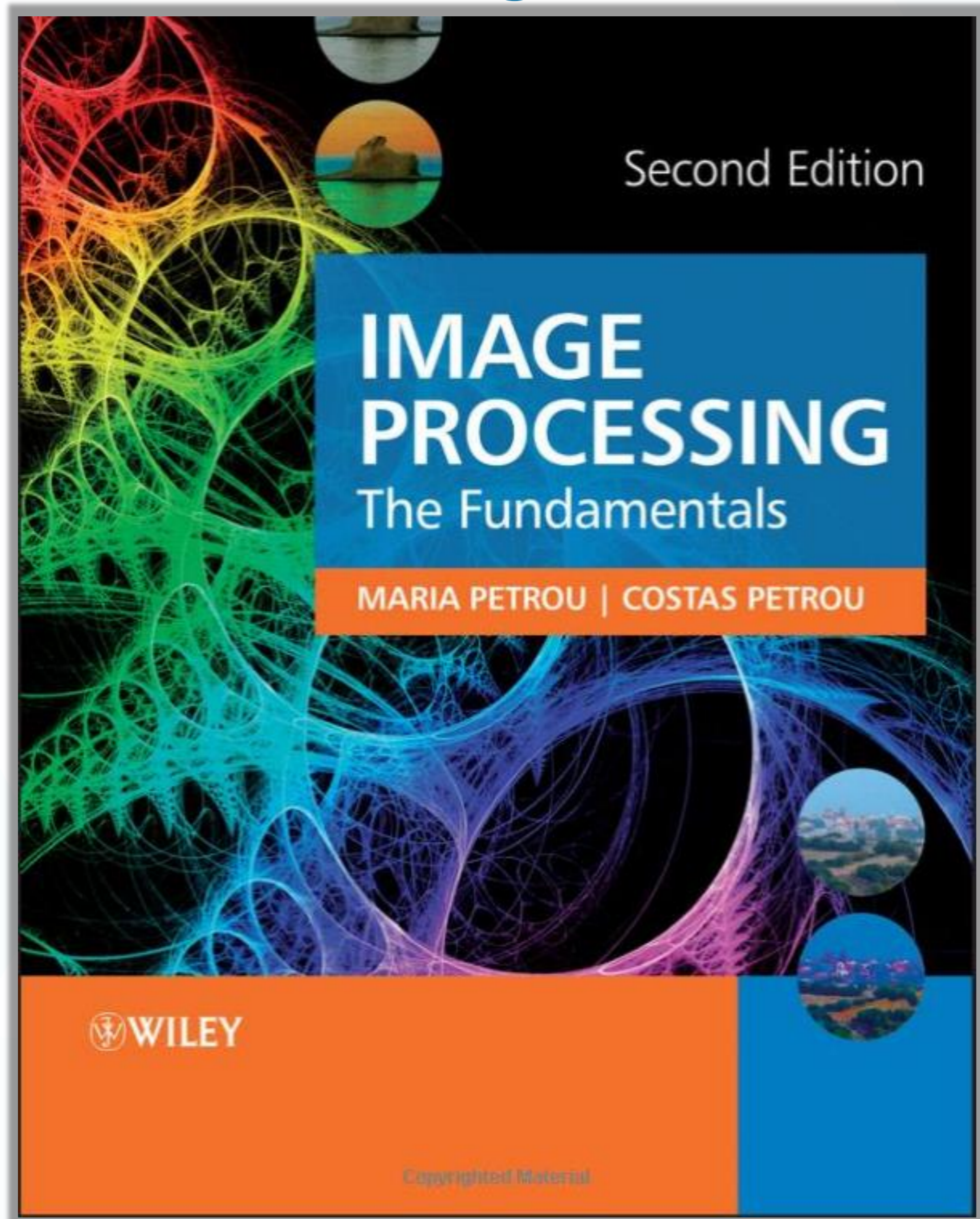
Basic concept



Problem



Starting point



The idea



PREDICTING THE MOMENT WHEN THE CLOUD COVERS THE SUN

01

DETECT THE
CLOUD EDGES

02

DETERMINE THE
SPEED AND
MOVE DIRECTION

03

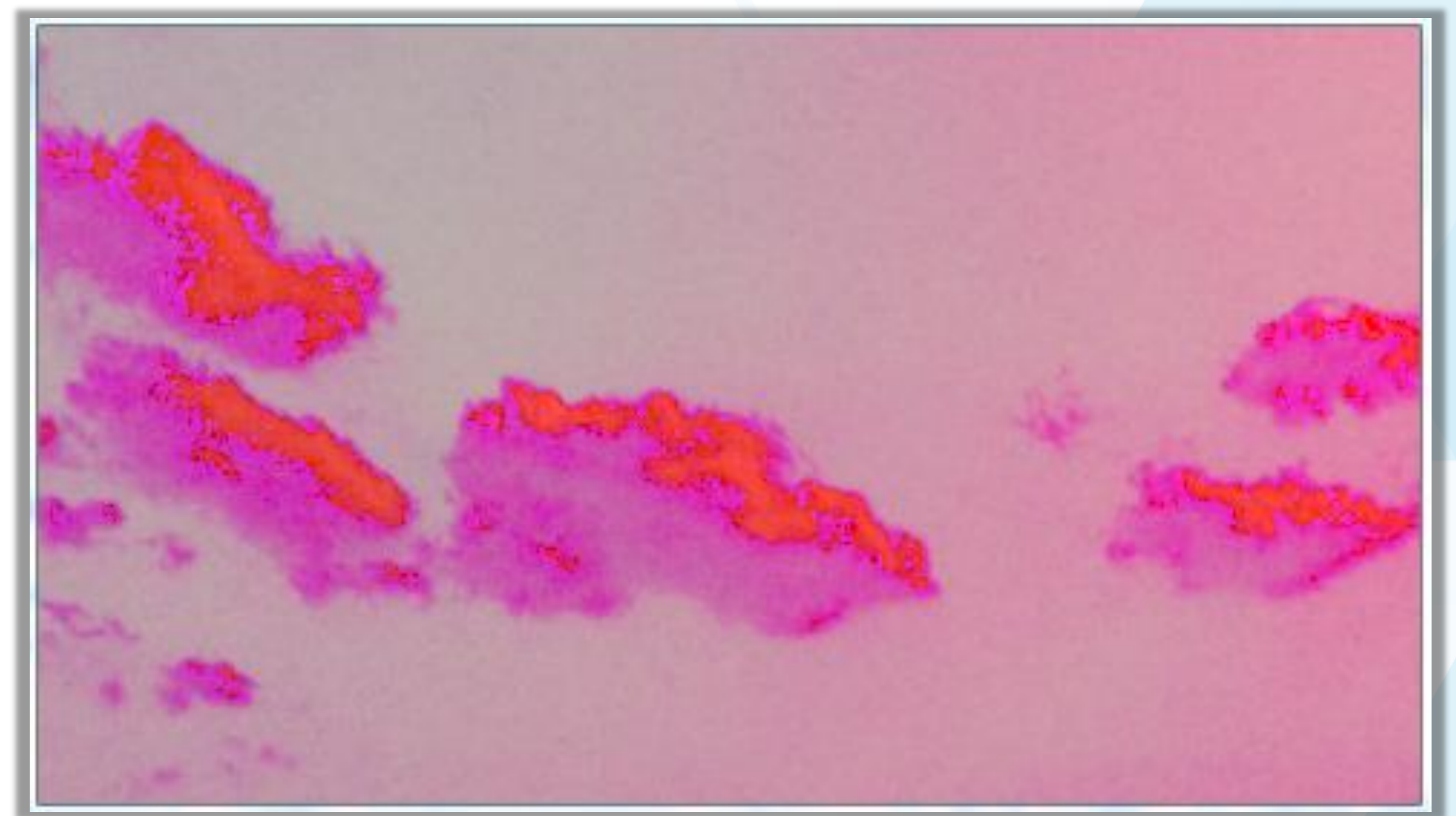
PREDICT THE
MOMENT OF
COVERING THE
SUN

04

PREDICT THE
LENGTH OF THE
SHADOW

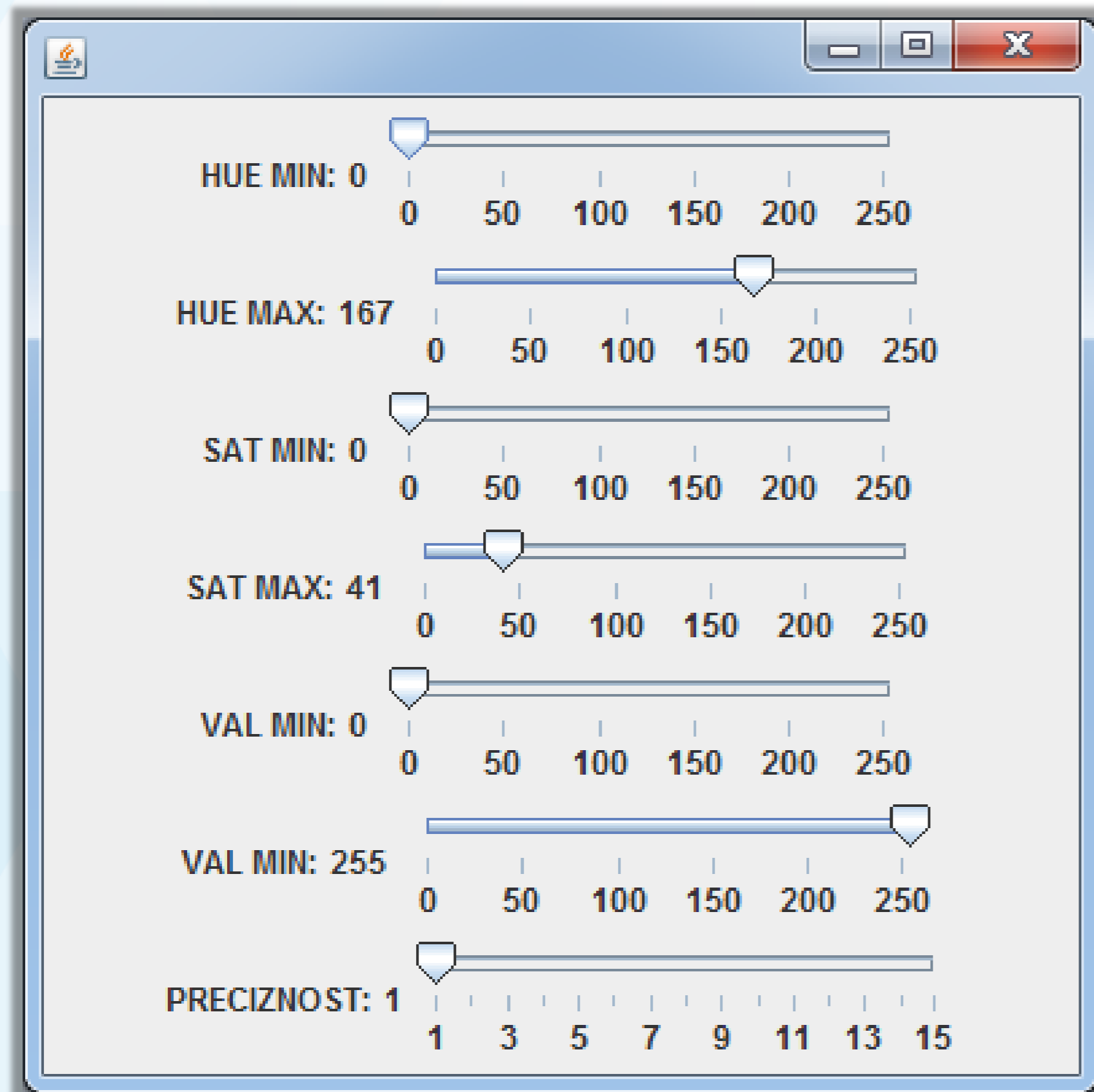
PHASE: DETECTING THE CLOUD EDGES

STEP 1: CONVERTING RGB -> HSV COLOR MODEL



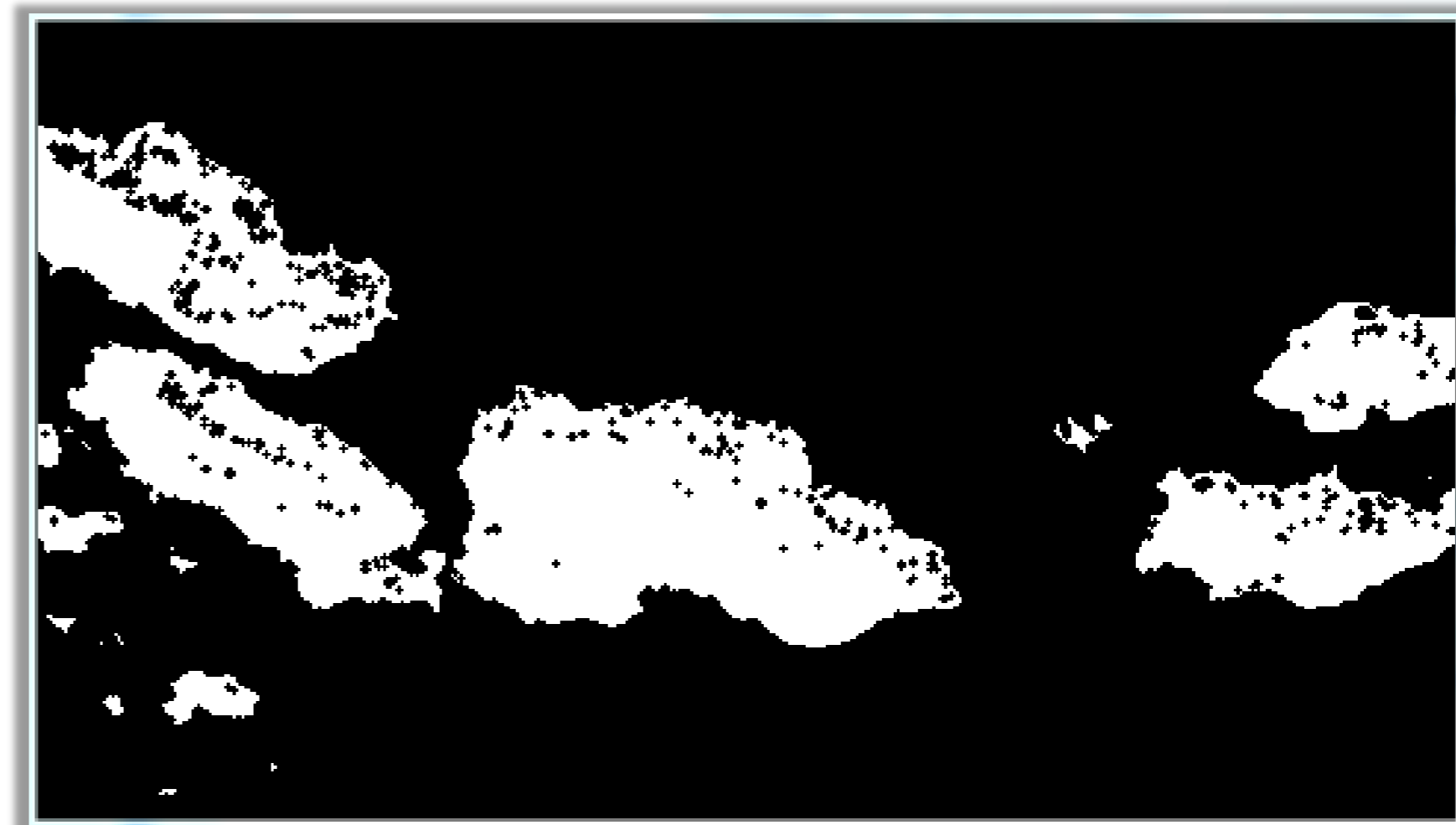
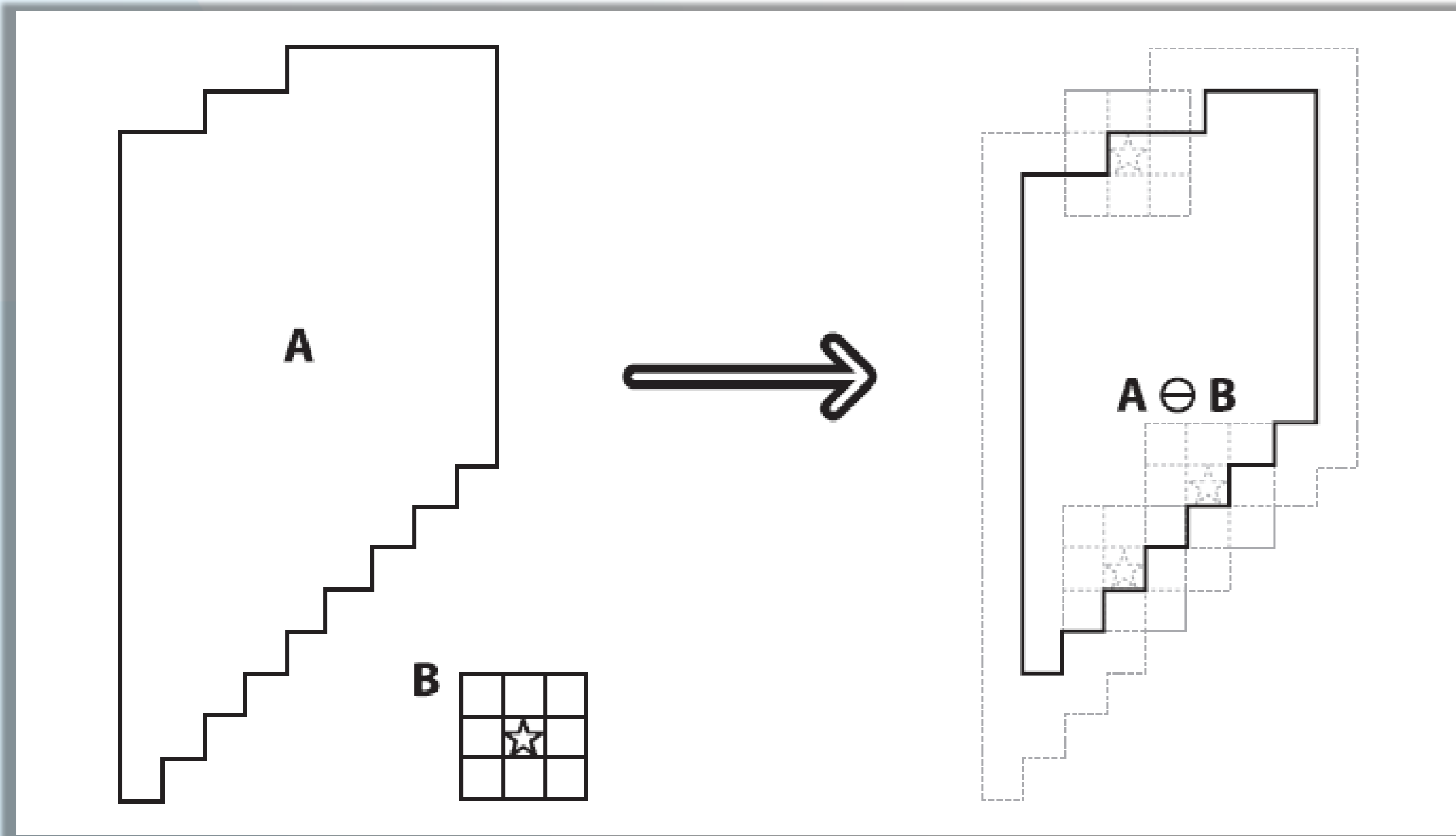
PHASE: DETECTING THE CLOUD EDGES

STEP 2: SETTING THE THRESHOLD



PHASE: DETECTING THE CLOUD EDGES

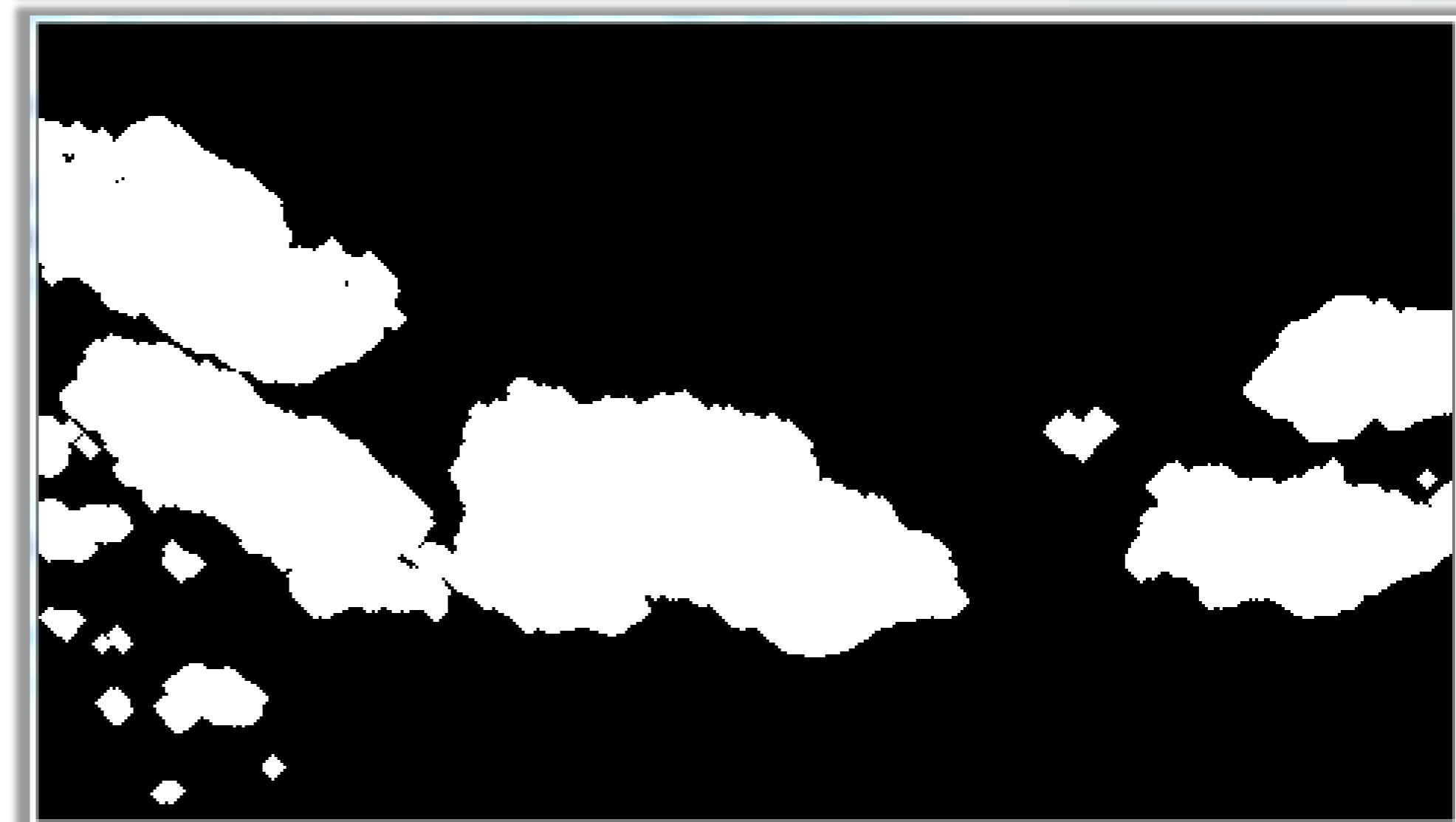
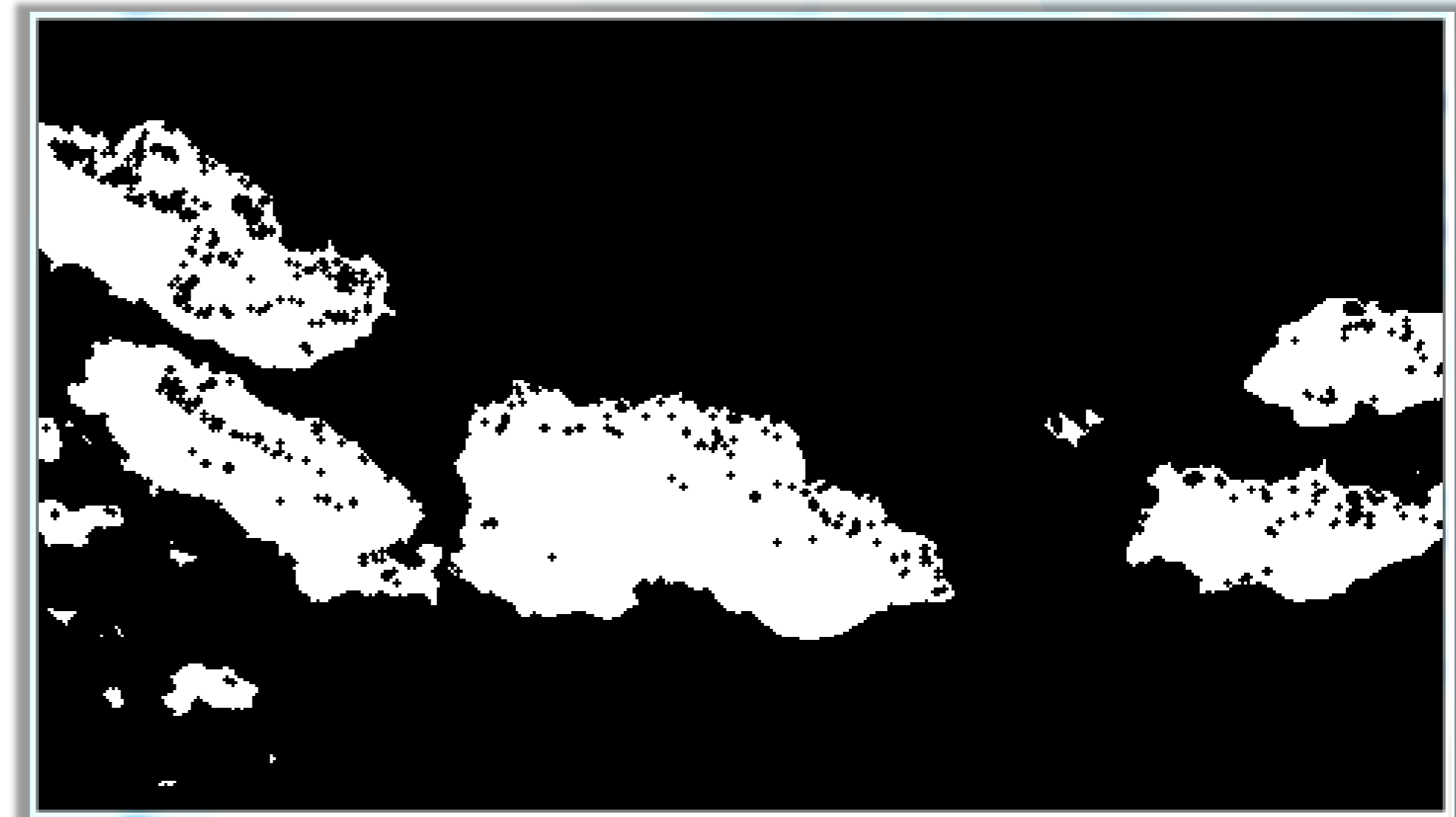
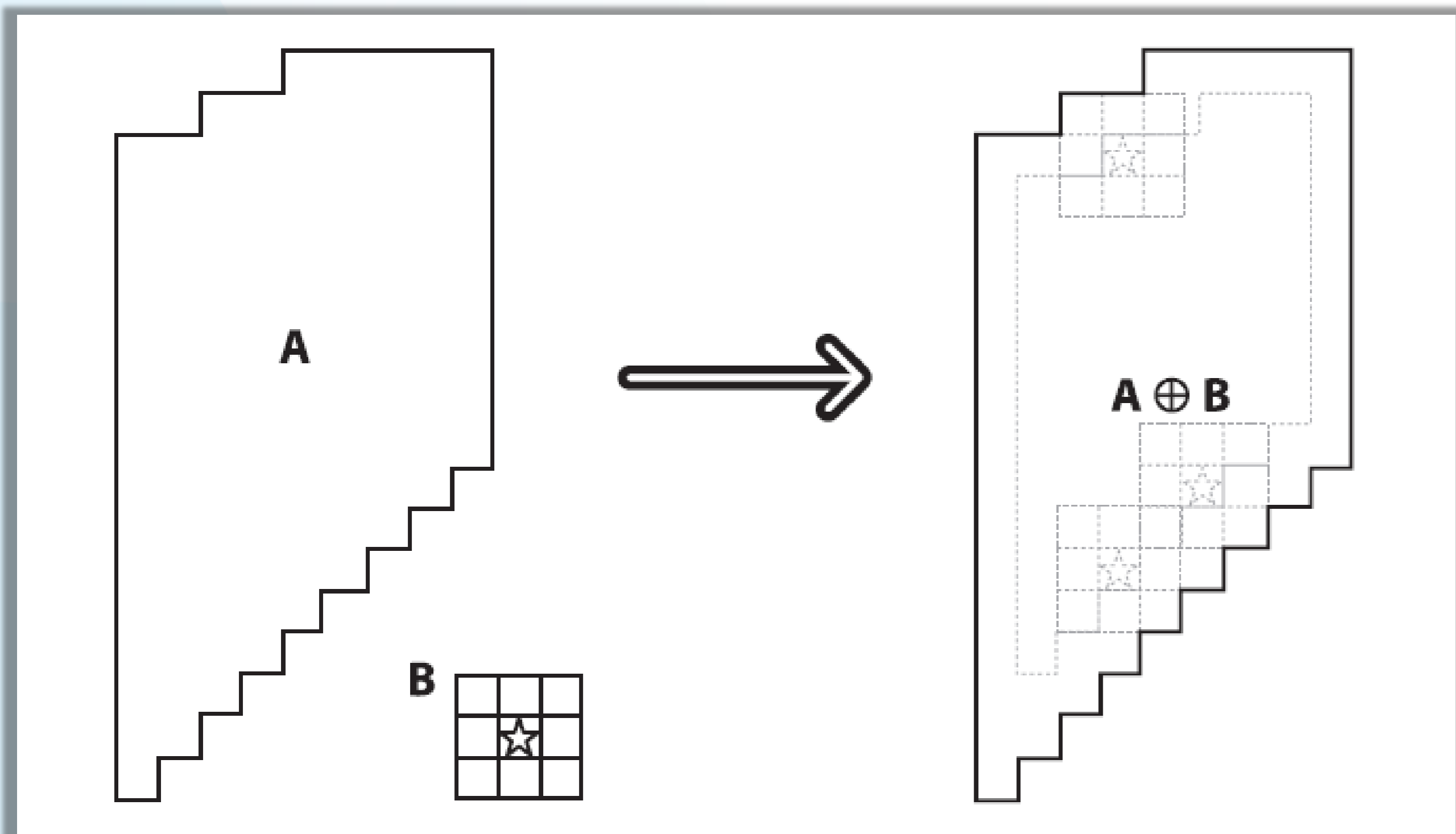
STEP 3: APPLYING EROSION



Source: Learning OpenCV, O'Reilly, p. 117

PHASE: DETECTING THE CLOUD EDGES

STEP 4: APPLYING DILATATION



Source: Learning OpenCV, O'Reilly, p. 116

PHASE: DETECTING THE CLOUD EDGES

STEP 5: CONTOUR TRACKING

THE ALGORITHM

For every contour on the image

Detect the starting pixel on the image

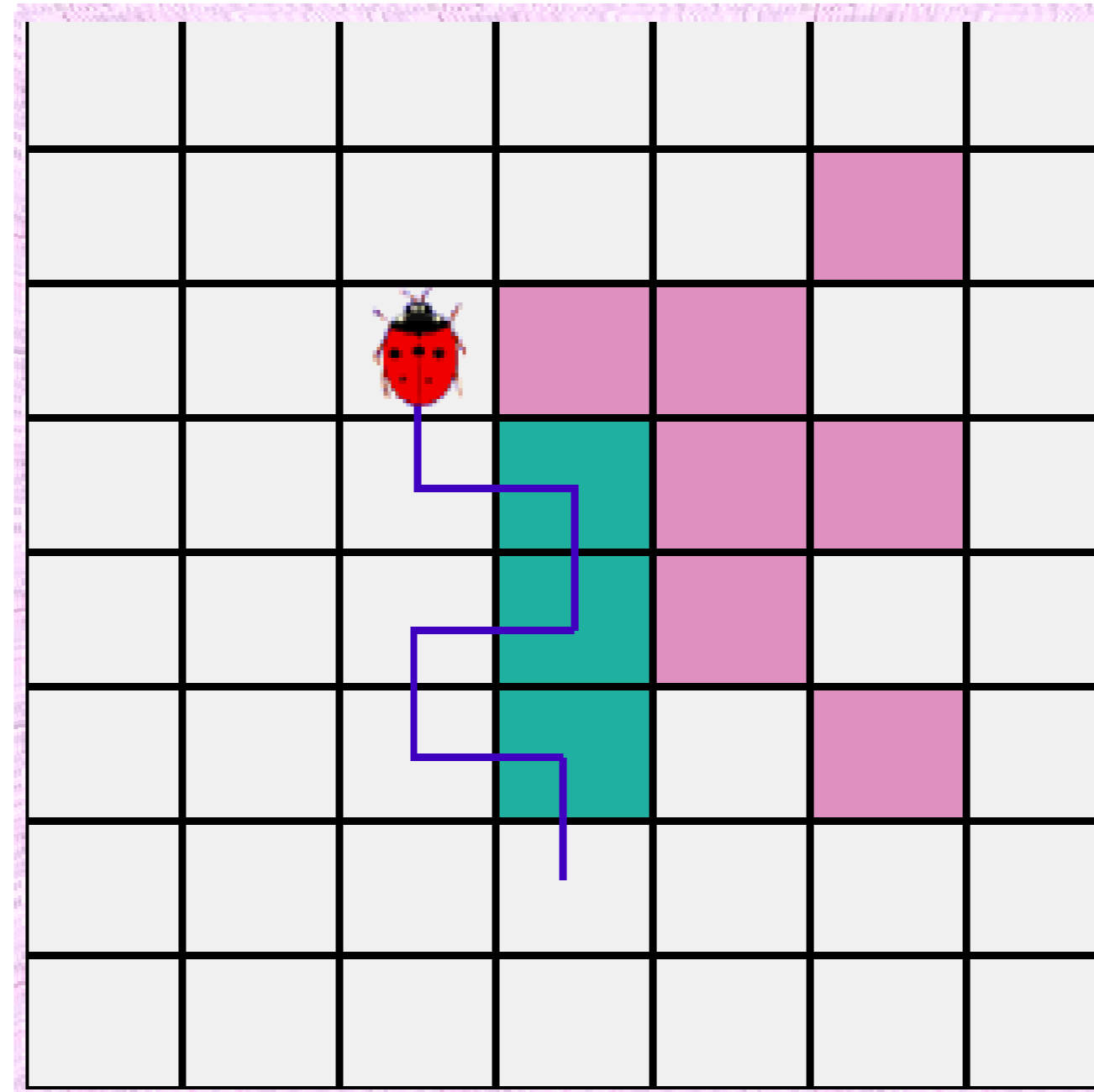
Repeat until the return to the starting pixel

If the current position is the white pixel

Move one pixel to the left

If the current position is the black pixel

Move one pixel to the right



Source: www.imageprocessingplace.com



PROOF OF CONCEPT VIDEO

The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer:** Shows a project structure with folders like 'Canny', 'Histogram', 'Hough', 'KDI', 'src', 'JRE S', 'Oper', 'Refer', 'lib', 'MarvinE', 'MarvinF', 'MarvinP', 'MarvinS', 'OpenCV', 'Prewitt', 'Segment', 'Sobel-1', 'Sobel-2', and 'SURF'.
- TrackingClouds.java Control Panel:** A dialog box with sliders for:
 - HUE MIN: 0 (range 0-250)
 - HUE MAX: 100 (range 0-250)
 - SAT MIN: 0 (range 0-250)
 - SAT MAX: 170 (range 0-250)
 - VAL MIN: 98 (range 0-250)
 - VAL MIN: 213 (range 0-250)
 - PRECIZNOST: 2 (range 1-15)
- Web Cam Live:** A window showing a live video feed of a person's face with red contour lines overlaid on it.
- Code Editor:** Shows the source code for 'TrackingClouds.java' with line numbers 172-187.
- Outline:** Shows a class hierarchy with methods like 'trackingClouds()' and 'tSun(IollImage)'.
- Taskbar:** Shows the Windows taskbar with icons for Chrome, File Explorer, and the application, along with the system tray showing 'HR' and the date '8.8.2013' at '23:23'.

TEST VIDEO



add media or start a new project to enable

PHASE: DETERMINING THE SPEED AND DIRECTIONS

STEP 1: DETERMINING THE CENTROID OF THE CLOUD



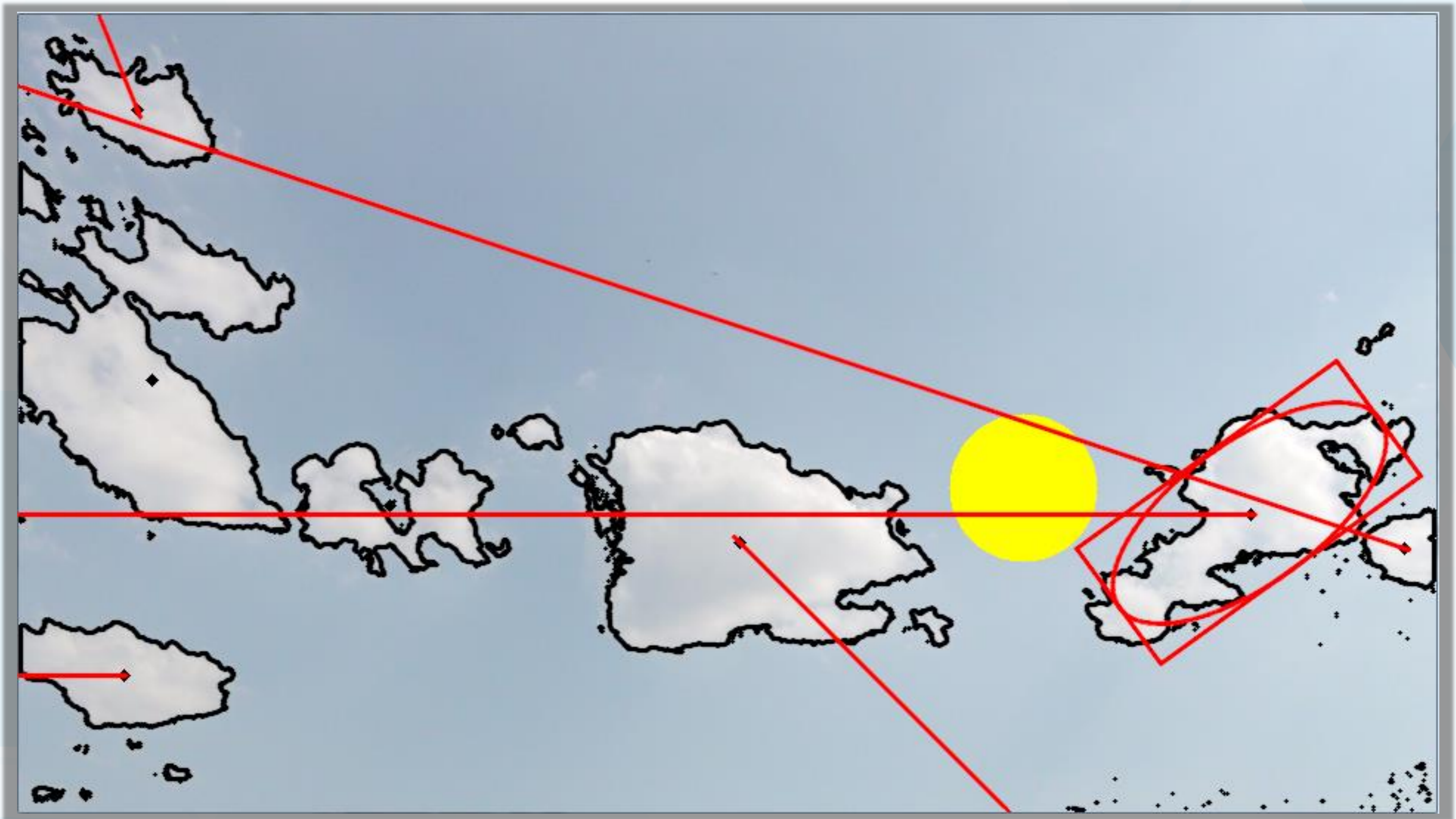
PHASE: DETECTING THE CLOUD EDGES

STEP 2: DETERMINING THE SPEED OF CLOUD MOVEMENT



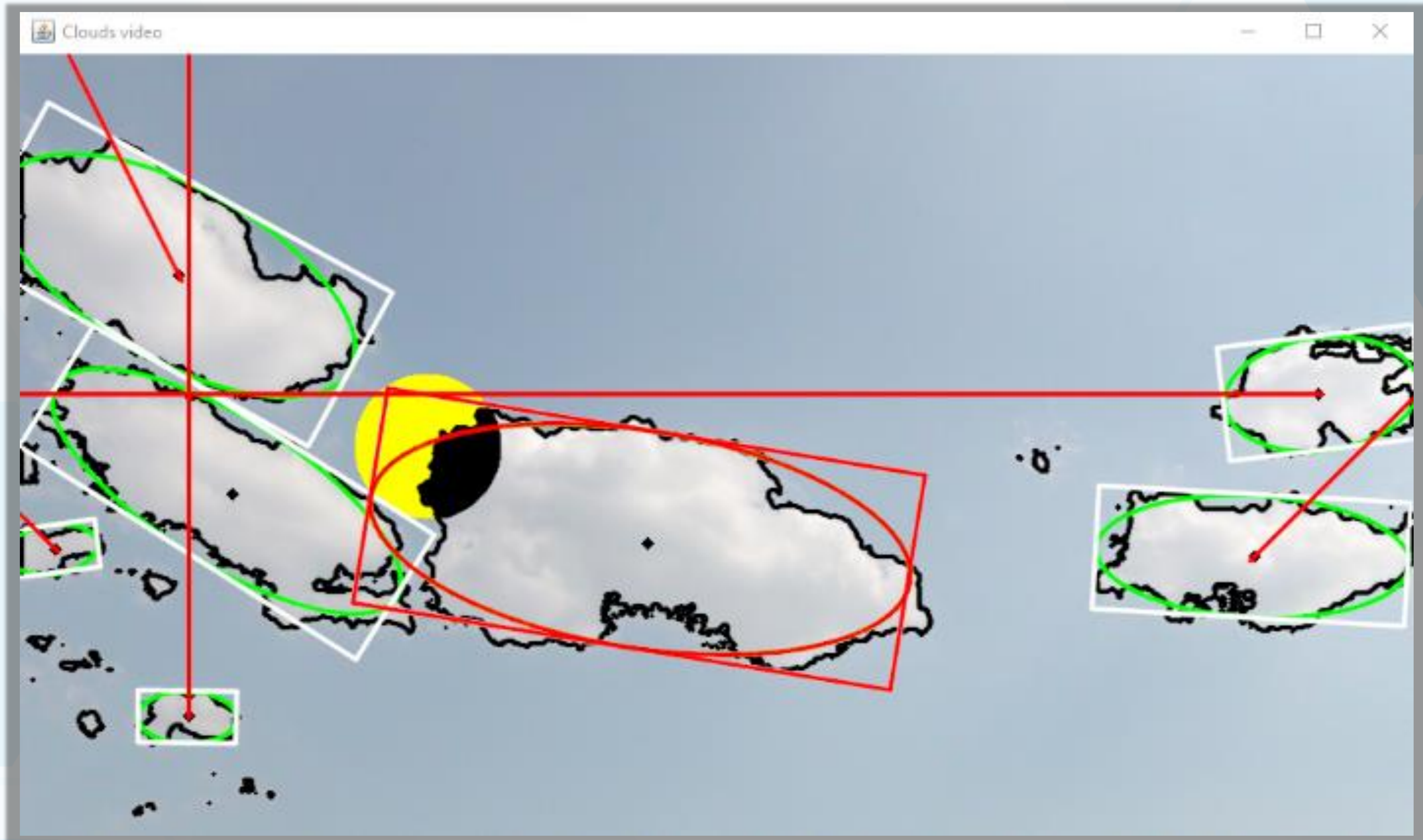
PHASE: PREDICTING THE MOMENT OF SUN COVERING

STEP 1: DETECT THE NEAREST CLOUD TO THE SUN

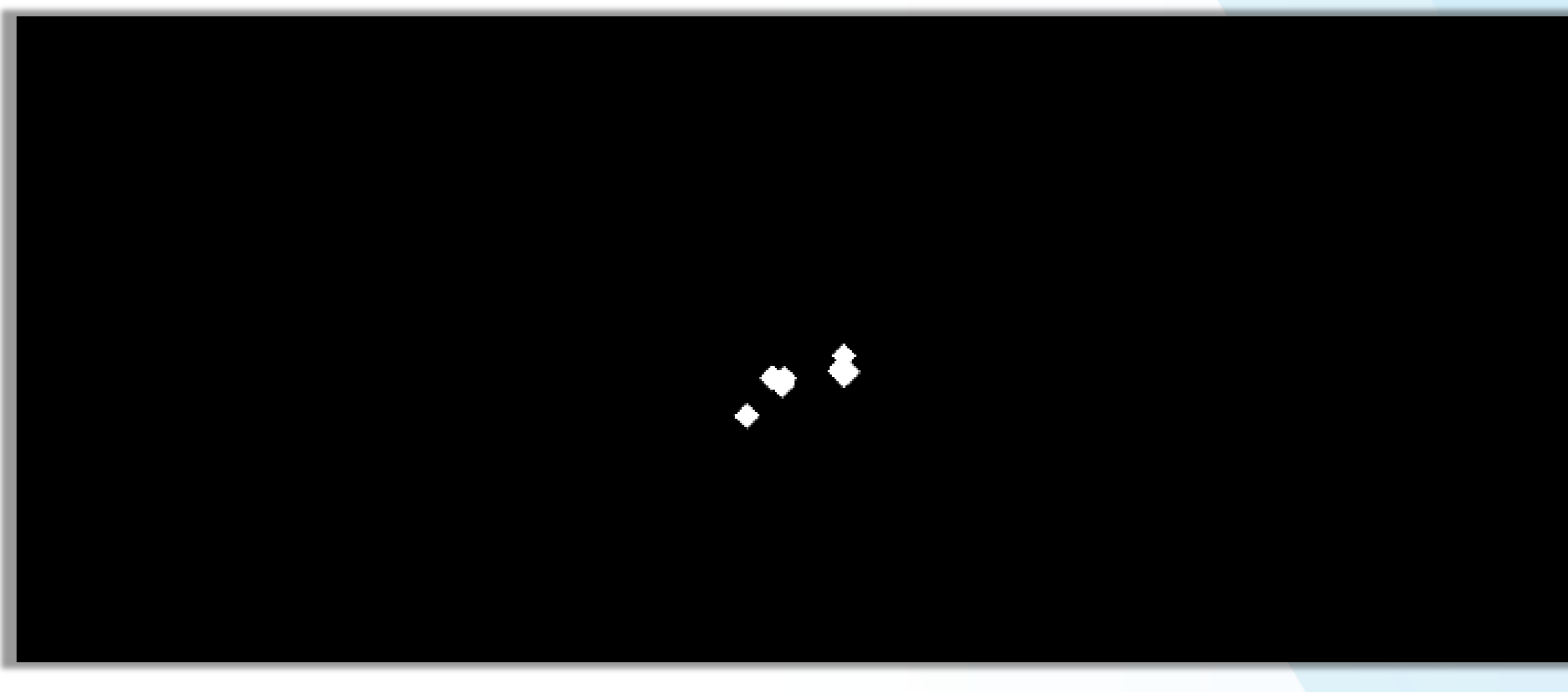
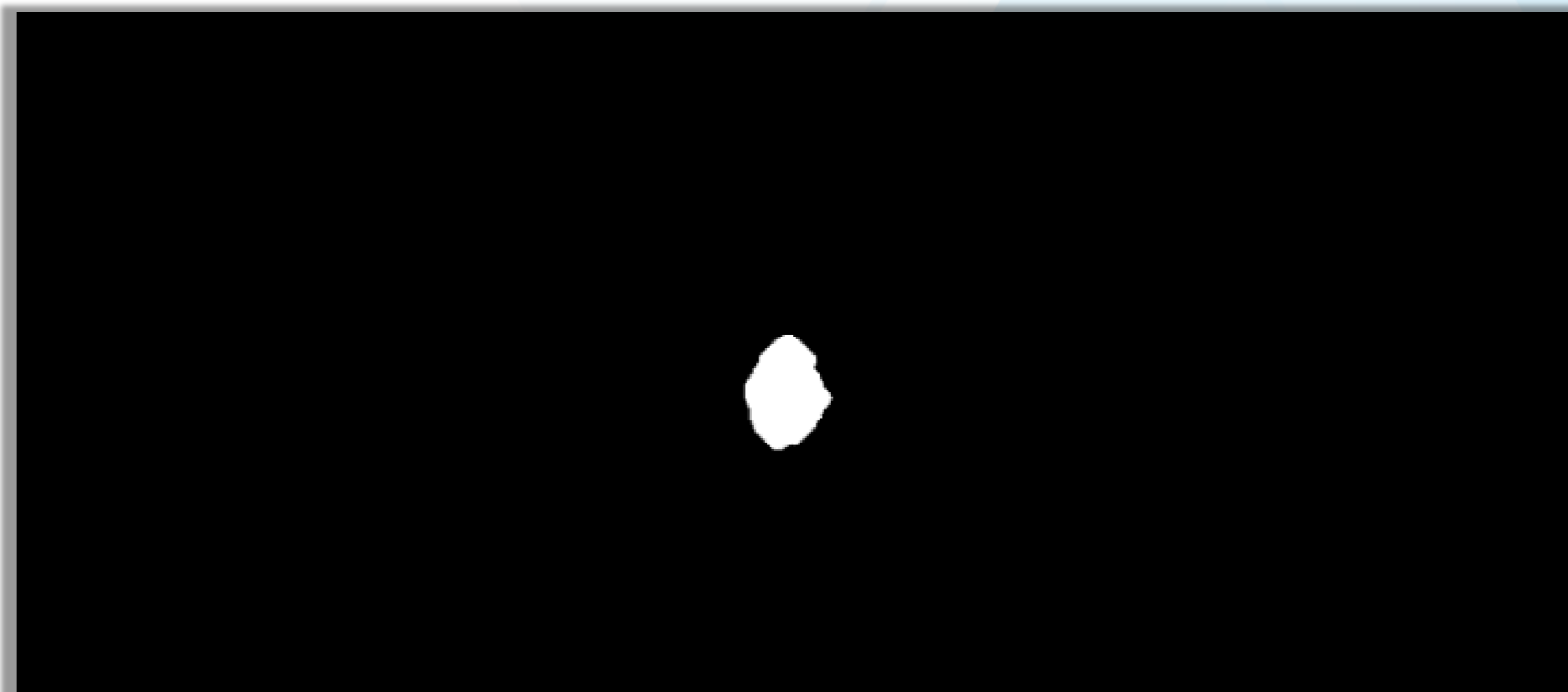


PHASE: PREDICTING THE MOMENT OF SUN COVERING

STEP 2: DETECT THE MOMENT WHEN THE SHADOW STARTS



PHASE: PREDICTING THE LENGTH OF THE SHADOW



PHASE: DETECTING THE CLOUD EDGES

ADDING THE AI TO DETERMINE THE THRESHOLD LEVELS

The screenshot displays a Java IDE with several windows:

- Control Panel:** Features sliders for Hue (MIN: 0, MAX: 240), Sat (MIN: 0, MAX: 18), and Val (MIN: 0, MAX: 247). It also includes a 'STRUCTURE ELEMENT' slider and a list of files.
- Clouds video:** A central window showing a video frame of a cloudy sky with a black circle and red arrows indicating detected cloud edges.
- Histogram:** A window showing a histogram of the video frame with 'Count' on the y-axis (0 to 45,000) and 'Value' on the x-axis (0 to 250). The histogram is color-coded by channel (Red, Green, Blue).
- Code Editor:** Displays Java code for processing the video frames, including methods for calculating maximum and average values.

```
121 histogramFrame.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
122 slidersFrame.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
123
124 listaRubnihPravaca.add(new Pravac(cvPoint(0,0), cvPoint(959,0)));
125 listaRubnihPravaca.add(new Pravac(cvPoint(0,539), cvPoint(959,539)));
126 listaRubnihPravaca.add(new Pravac(cvPoint(0,0), cvPoint(0,1079)));
127 listaRubnihPravaca.add(new Pravac(cvPoint(959,0), cvPoint(959,539)));
128
129
130 private double maxValue(double array[]){
131     double max = Arrays.stream(array).max().getAsDouble();
132     return max;
133 }
134
135 private double averageValue(double array[]){
136     double sum = Arrays.stream(array).sum();
```

Output console shows:

```
Vrijeme: 24 sekundi
POVRšina SUNCA: 2906.0
SUNCE JE PREKRIVENO 97.72506481010502%
NOVE VRIJEDNOSTI:
H MIN = 0
H MAX = 240
S MIN = 0
S MAX = 18
V MIN = 0
V MAX = 247
```

SUMMARY

The image is taken from the video every 12 seconds

OpenCV (JavaCV) does a good job

A DLL file is needed to run it on Windows

Instead of simulating the future cloud movement, it's better to move simulate the Sun movement

Using Java in combination with other components the solar electric system can be optimized

Thank you!



<https://www.linkedin.com/in/aleksander-radovan-57a29a35/>



@alex_hujak