Pixel Intensity Transformations
Project Documentation

Elizabeth Chilcoat, Kelle Clark, Michael Heath

October 2, 2020

1 Usage and System Dependencies

This project is developed with the intent to investigate enhancing an image through intensity transformations that only consider the value at each pixel. The transformations require at least one image to be loaded, and the default format for these images is grayscale. However, the user may optionally choose to load and transform color images. An alert is given to the user when loading the first image about the default grayscale and choice to alternatively view the image using color. The unary transformations included are: Negative, Log, Bitplane, Gamma, Arithmetic, Complement, Piecewise Linear and Binarization/Threshold. A user can also elect to select a second image, image 2, which will be read in the same format as image 1. If two images are selected, the user may perform binary transformations from a collection of Binary Set or Bitwise Logic Operations. The Binary Set operations include Union, Intersection and Difference. The Logic operations include And, Or and Xor.

1.1 Dependencies

The user should check to see if the following libraries are installed or do a batch install with

```
pip install -r requirements.txt
```

The libraries and versions used by the team include:

1.2 Usage

To run in the command line:

```
>python pixel_ops.py [params]
```

where optional parameters are:

```
>python pixel_ops.py -h
usage: pixel_ops.py [-h] [--  ]

Pixel Operations v1.0

optional arguments:
  -h, --help  show this help message and exit
  --          A Graphical User Interface with no arguments.
```

When run, the program will launch a GUI and instruct the user to load the first image, image 1. Many of the pixel based transformations are intended to be used with Grayscale images; however, as mentioned in the introduction, the user is given an option for the image to be read in the default Grayscale or in Color [see Figure 1 and Figure 2].

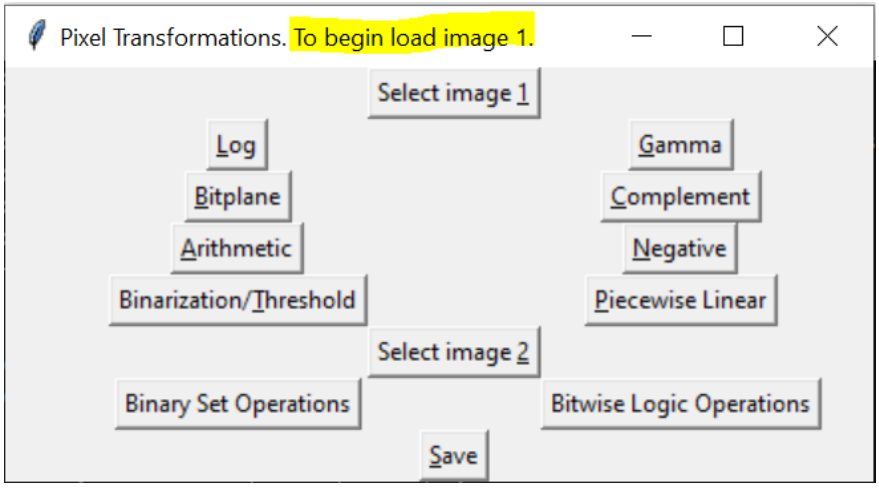


Figure 1: "Initial Graphical User Interface"

Note that if the user selects any of the operations before image 1 is selected, then a warning is given [Figure 3. Once the user selects image 1, the image is displayed on the left of the interface. [see Figure 4]

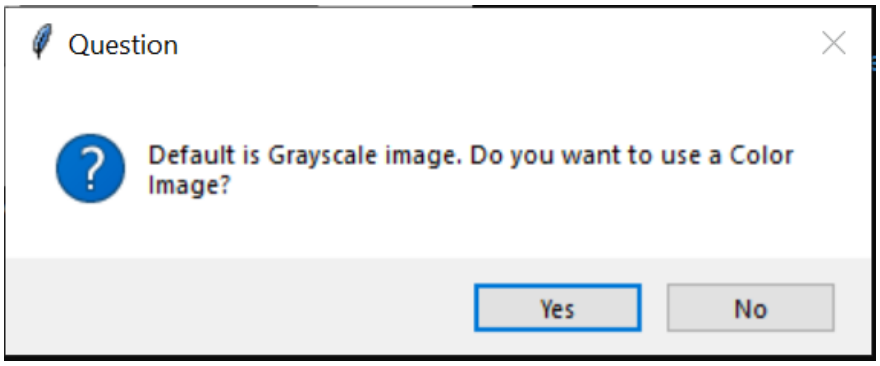


Figure 2: "When image 1 is selected, the user is given a choice of default Grayscale or Color."

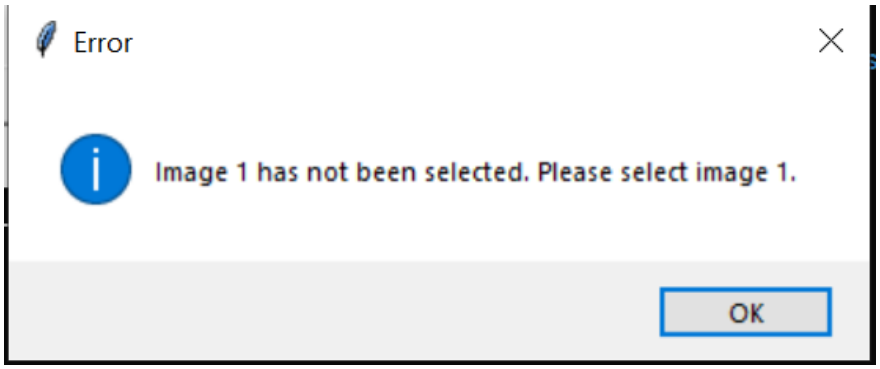


Figure 3: "An alert is given if user selects an operation before selecting an image 1."

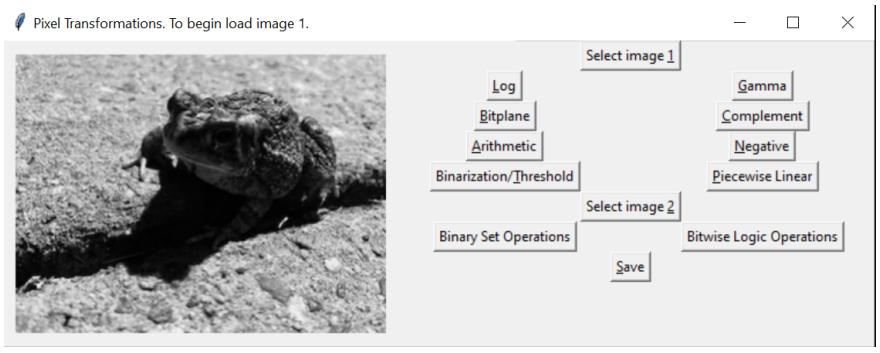


Figure 4: "When image 1 is selected, the image is displayed on the left panel of the interface."

The result of using any of the operations is placed on the right of the interface. In Figure 5, the Log transformation has been applied to image 1 resulting in the darker pixel values increasing to reveal details in the crevice of the pavement.

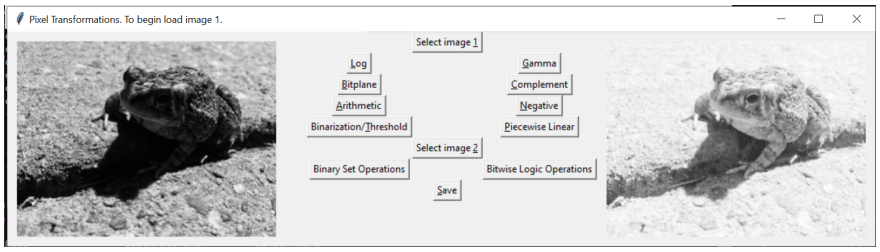


Figure 5: "When a transformation has been chosen, the resulting image is displayed in the right panel of the interface."

When image 2 is also selected it is displayed immediately to the left of the operations along with image 1. The result of the binary operation is displayed on the right after the binary operation has been selected. In Figure 6, two color images are selected and the Union operation is performed.

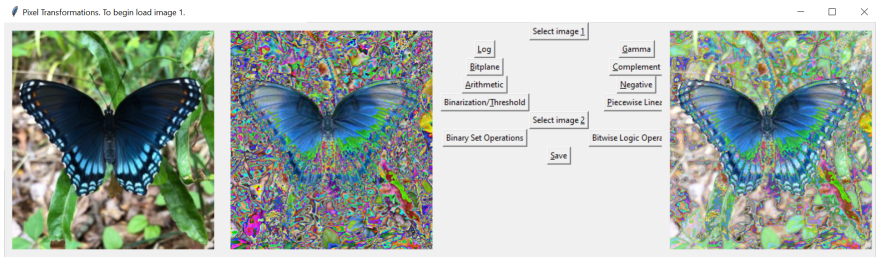


Figure 6: 'The input for binary operations, image 1 and image 2 are shown left to right with the resulting image on the far right.'"

A user is able to save any resulting image by using the save button in the bottom right corner of the operations pane. A dialog box opens to prompt the user allowing the user to chose the name and directory to save the image to. The team used a collection of images from and placed in Input then saved the resulting images using the save feature of the GUI to save the transformed images to Output. The team used the image browser project one to view these images.

To quit the program, the user enters in q from the keyboard.

2 User Requirements Log

The design, implementation and testing of the system are based on satisfying the following stated user requirements.

2.1 Project Requirements:

1. The system should have a Graphical User Interface
2. The system should allow a user to select a method to be applied
3. The system should display all methods available to the user
4. The system should implement the method to create the negative image
5. The system should implement versions to produce bit plane images
6. The system should implement the method to produce the log transformation of an image
7. The system should implement the method to produce the power law gamma transformation of an image
8. The system should implement the method to create a linear transformation of an image
9. The system should implement the method to produce a piece-wise linear transformation of an image
10. The system should implement arithmetic operations on an images
11. The system should implement image set operations on an image
12. The system should implement binarization thresholding on an image
13. The system should perform logic operations on a binary image
14. The system should allow a user to input the path to the image to be used as input
15. The system should prompt the user for the necessary parameters of the method chosen
16. The system should allow a user to chose to save a transformed image
17. The system should prompt the user for the name to save a file as
18. The system should terminate when the user presses the “q” on the keyboard
19. The system should run with the command pixel-ops
20. The system should display help to run the program with the command pixel-ops -h

3 Software Model, Development Methodology

The team configured and added to the programs written to satisfy the first two assignments for this project that allowed the team to meet the stated requirements. The use of project one, image browser, allowed the team to view the saved images through traversing all images in folders grouped according to the transformations saved in an Output file.

4 Testing

4.1 Test Objective: What if the user enters option -h?

Test run:

```
>python pixel_ops.py -h
```

Test result:

```
usage: pixel_ops.py [-h] [--  ]

Pixel Operations v1.0

optional arguments:
  -h, --help  show this help message and exit
  --          A Graphical User Interface with no arguments.
```

4.2 Test Objective: What if the user does not select an image 1 before trying to use an operator?

Test run: Each operation was selected in turn before selecting an image.

Test result: A dialogue box was displayed prompting the user to select image 1.

4.3 Test Objective: What if the user tries to use a binary operator before selecting a second image?

Test run: Each of the binary operation buttons was selected in turn before a second image was selected.

Test result: A dialogue box was displayed prompting the user to select image 2.

4.4 Test Objective: What if the user enters invalid parameters for a transformation?

Tests run and results:

- Gamma selected with -1 entered for exponent, an alert is given. The user can try again or can cancel.
- Gamma selected with gamma correctly entered, the value of the constant multiplier was entered as -1. The response was an alert was given. The user can try again or can cancel.
- Arithmetic selected with value of c negative for addition, subtraction, multiplication and division. An alert is given for the minimum value of c. The user is prompted to try again or can cancel.
- Arithmetic selected. Choice of operator is division and a zero value for constant. An alert is given. The user is prompted to try again or can cancel.
- Binarization selected. Choices out of range provided, i.e. less than 0 and greater than 255. An alert is given. Also only integer values are accepted. The user is prompted to try again or can cancel.

4.5 Test Objective: What occurs at the edge cases if a pixel value is 0 or 255 and the Log transformation is selected? How does the program handle log(0), or if performing the transformation log(1 + pixel), log(255+ 1)?

Test run: The pure-black image and the pure-white image from the solid color image corpus folder are both used with the Log transformation. With the pure-black image, all 0 pixel values, selected as image 1 the resulting log image is visibly indistinguishable from the pure-black image. When the pure white image is used with the log transformation, the log image is slightly off white.

4.6 Test Objective: What does the program do if the Piecewise Linearization is selected with (r1,s1) and (r2, s2) and r1 = r2 which would result in division by 0 in the algorithm?

Test run: An image 1 and the log transformation operation is selected. The point (r1, s1) is entered. An alert is displayed if r2 is entered that is not at least r1. The program forces a user to enter in the left most point first.

4.7 Test Objective: Does the program allow a user to save an image?

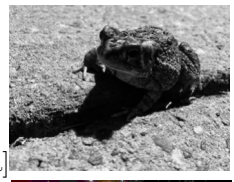
Test run: The image corpus was collected and the transformations were run on these images.

Test Results: The results of these images were saved in the Output photos folder in the repository. The name of the saved image reflects the original image from the used:

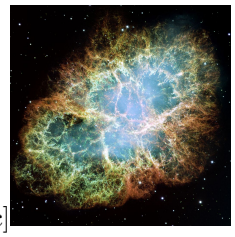
imageUsed_transformationName_parameter_parameter.png

4.8 Test Objective: Does the program execute all methods?

Tests run: An image corpus was created including solid color images, grayscale images and color images. The team chose parameter values that increased and decreased the impact of the transformations visually. For instance the Log transformation was used on a darker image like frog.png to show that the new image is lighted to reveal more detail. The Log transformation is also applied to the Crab Nebula image to reveal stars that are faint in the original color image.

[a] 

[b] 

[c] 

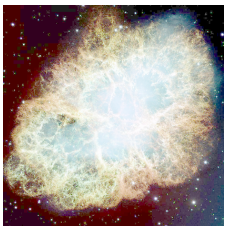
[d] 

Figure 7: (a) frog (b) frog log (c) crab nebula (d) crab nebula log

The grey color image was combined with the grayscale flowers image to investigate the union, intersection, difference as well as and, or and xor as shown in Figure 8. There seems to be no difference in the image using xor and or with flowers and grey.

To better see these Logical operators, the team used color images, yellow and dragonfly, to investigate the union, intersection, difference as well as and, or and xor. These results as shown in Figure 9. There seems to be no difference in the image using xor and or with flowers and grey. This was particularly useful is seeing a difference between the or and the xor. Note: the dragonfly.bmp image would not render here.

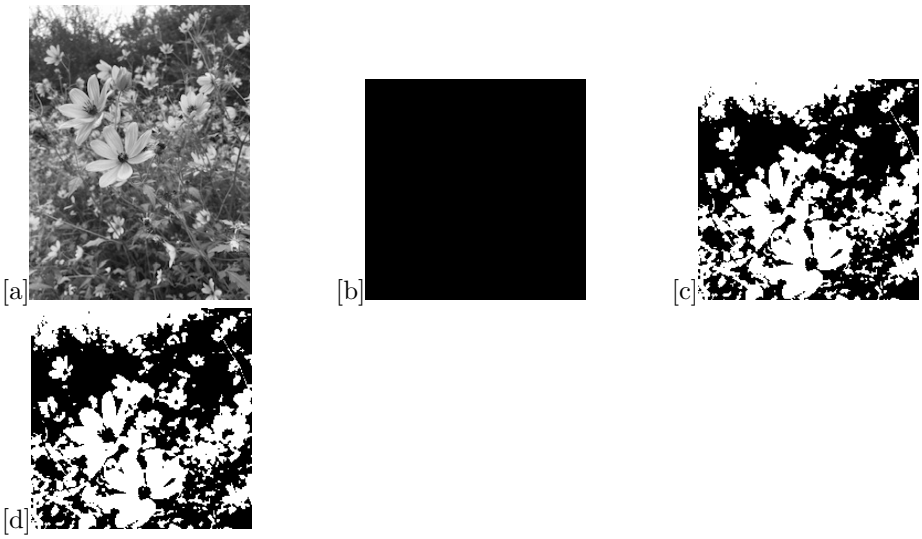


Figure 8: (a) flower (b) flower and grey (c) flower or grey (d) flower xor grey

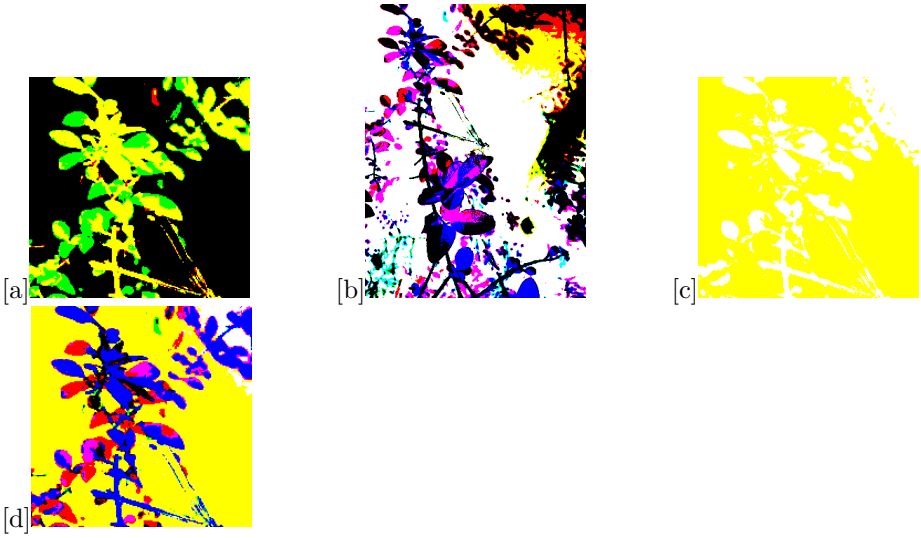


Figure 9: (a)dragonfly and yellow (b) dragonfly not (c) dragonfly or yellow (d) dragonfly xor yellow

The parameters for arithmetic ranged so that the changes could be discerned. Below is arithmetic operators subtraction and addition is used with the values 0, 25, 50 and 100 in Figure 10.

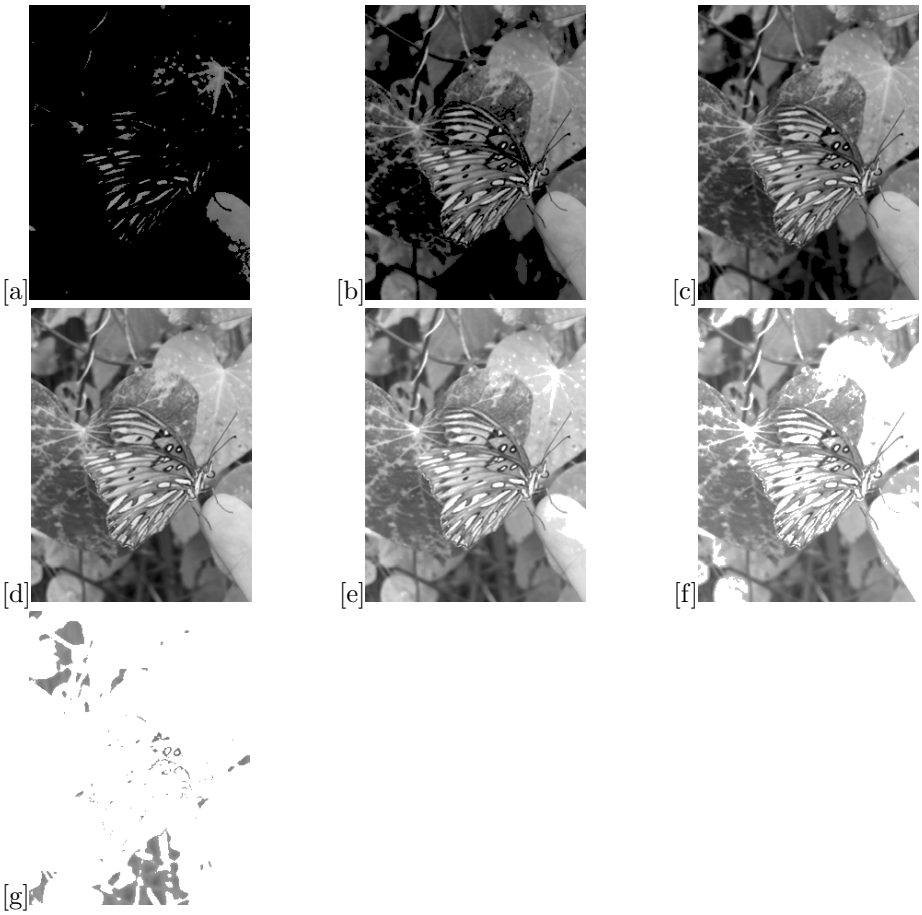


Figure 10: (a) butterfly - 100 (b) butterfly - 50 (c) butterfly - 25 (d) butterfly + 0 (b) butterfly + 25(c) butterfly + 50 (d) butterfly + 100

While not easy to display, for the white image, the not, the negative and the complement is a black image. We show the results on the frog instead in Figure 11:

Figure 11:

The team also choose pure colors to see if blue and yellow really make

green...but it turns out that blue and yellow makes a black image.

There are many output photos, and the best way to view these results is by using the team's first project image browser.py:

```
python image_browser.py Output_photos
```

The most difficult transformation turned out to be the piecewise linear transformation. Choosing values for the two points so that the image is changed enough to see and not so much so that the resulting image is all black was difficult. Figure **??** shows the teams output for piece wise linear using pairs of points.

Figure 12: (
a) moth (b) (200, 10) and (201, 100) (c) (200, 50) and (201, 100)

The multiplication and division darkened and lightened as expected. In Figure **??** the images of the butterfly and the frog are arranged from darkest to lightest with a variety of division and multiplication is used with interesting results given the wrap around operation of the pixel values.

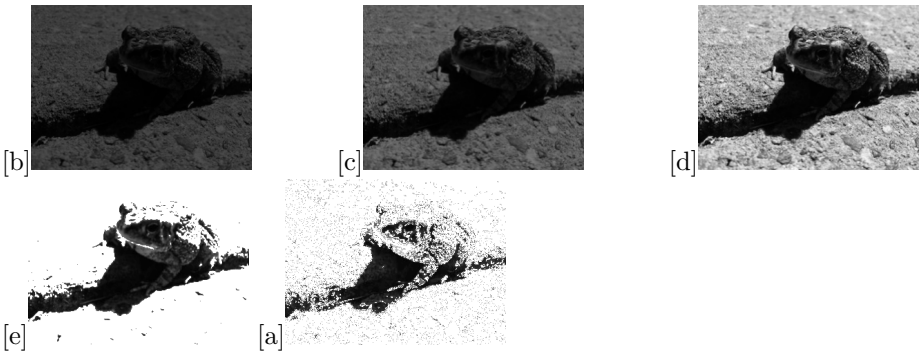


Figure 13: (a) divide by 1/3 (b) times 1/2 (c) divide by 1 (d) times 2 (e) divide by 1/4

The most spectacular images were produced using the color image of the crab nebula. Here we used operations like gamma including the exponentiation of the pixel values. The greyscale images of a snake with the gamma transformation are nice too, but just not as visually stunning as seen in Figure 14.

So that our coverage is complete, here is a sample collection of the results of using bitplane and binarization. Figure 15

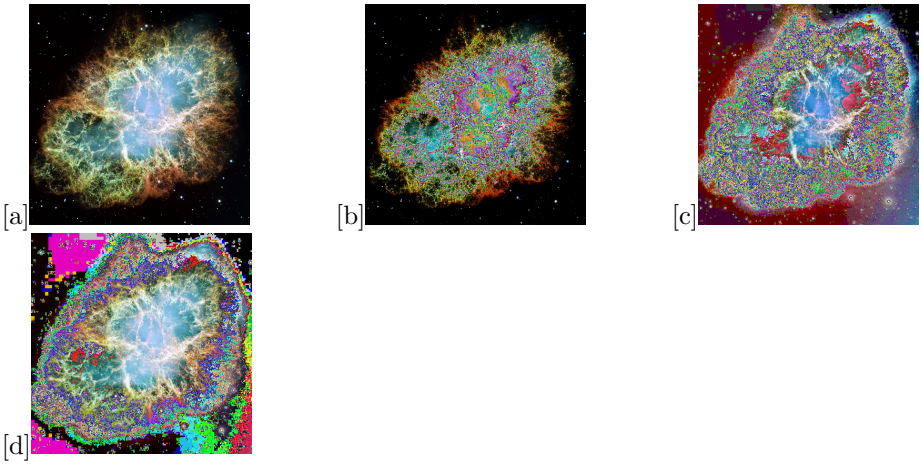


Figure 14: (a) original (b) $3pixel^2$ (c) $3pixel^{1/2}$ (d) $3pixel^{1/4}$

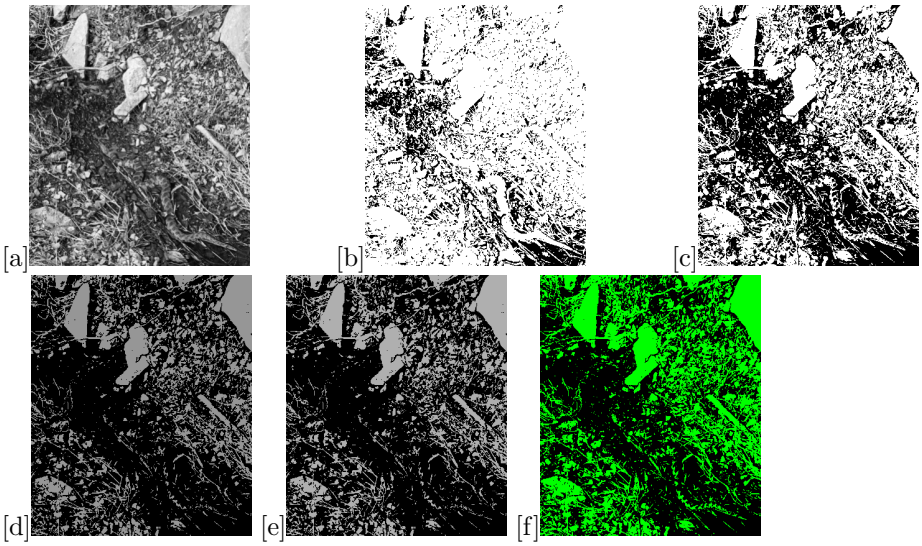


Figure 15: (a) original snake (b) - (e) binarization of snake with varying values of the threshold and new maximum value (f) bitplane green snake value 7