Unleash Your Genius: Decode the Hidden Image and Outsmart the Kryptonians!

Samarth Barve

***Abstract*— In this paper, we introduce a novel approach to tackle the enigmatic task of decoding a concealed image through a complex interplay with an input image. This task demands the creation of an efficient algorithm capable of unveiling the hidden image by scrutinizing the feedback image it generates. Our proposed solution is instantiated within the Robot Operating System (ROS) framework, leveraging an 8-bit, 3-channel input image. Our algorithm adheres to a predefined set of principles derived from the correlations between the pixel values of the input and hidden images, facilitating the generation of the corresponding output image. Through iterative application of these principles, we successfully unveil the concealed image within the given framework. This paper delves into the intricacies of the algorithm's conception, its implementation within ROS, and its efficacy in meeting the challenge posed.**

I. INTRODUCTION

The sudden appearance of extraterrestrial beings from the technologically advanced planet of Krypton poses a formidable task for humanity: unraveling a covert image hidden within a system governed by intricate relationships with an input image. This paper endeavors to meet this challenge by presenting a sophisticated algorithmic solution implemented using the Robot Operating System (ROS).

II. PROBLEM STATEMENT

The challenge posed by the Kryptonian mystery tasks humanity with unveiling a concealed image within a system governed by intricate relationships with an input image. Operating on 8-bit, 3-channel images, where each pixel encapsulates values for Red (R), Blue (B), and Green (G), the problem can be succinctly stated: given an input image and predefined relationships between the pixel values of the hidden image and the input image, the goal is to devise an efficient algorithm capable of producing an output image that exposes the hidden content while adhering to specified rules for each channel (R, G, B).

To elucidate the interconnections between the pixel values of the input and hidden images, the following guidelines are established:

1) Red Channel (R)
   - When the Red value of the input image (R input) is less than the Red value of the hidden image (R hidden), the corresponding Green value in the output image (G output) is set to 0.
   - If R input equals R hidden, G output is set to 127.
   - If R input surpasses R hidden, G output is set to 255.
2) Green Channel (G)
   -
   - If the Green value of the input image (G input) is lower than the Green value of the hidden image (G hidden), the corresponding Blue value in the output image (B output) is set to 0.
   - Should G input match G hidden, B output is designated as 127.
   - If G input exceeds G hidden, B output is configured as 255.
3) Blue Channel (B)
   - In instances where the Blue value of the input image (B input) is inferior to the Blue value of the hidden image (B hidden), the corresponding Red value in the output image (R output) is assigned as 0.
   - When B input equals B hidden, R output is assigned a value of 127.
   - If B input surpasses B hidden, R output is designated as 255.

The algorithm's task is to iteratively apply these rules to each pixel in the input image, thereby generating the corresponding output image that reveals the concealed content within the provided system.

By unraveling this remarkable challenge and effectively implementing the algorithm, we showcase humanity's intellectual prowess and adeptness in analyzing complex relationships within images. The destiny of the world hangs on our capacity to unveil the hidden image, unless Superman intervenes to save the day.

III. INITIAL ATTEMPTS

In our initial efforts to unveil the concealed image within the given system, we devised a straightforward algorithm involving iterative adjustments of pixel values guided by predefined rules. However, this approach proved to be highly inefficient and time-consuming.

Our initial algorithm began by initializing the input image as an 8-bit, 3-channel image with all pixel values set to 127. Subsequently, we subjected this image to a comparison function that assessed the pixel values of the input and hidden images based on the specified relationships. Depending on the feedback received for each pixel, we incrementally modified the pixel value by 1 in an attempt to converge on the desired image.

While this initial approach was conceptually simple, it quickly became impractical for larger images due to the maximum of 128 iterations required. The computational

overhead per iteration was significant, leading to prolonged execution times. It became evident that a more efficient and optimized strategy was imperative to meet the challenge's demands.

```
# Initialize input image
input_image = np.full((height, width,
channels), 127, dtype=np.uint8)

# Iterate through pixels and adjust values
based on rules
for i in range(max_iterations):
    # Compare input and hidden images based
    on rules


for row in range(height):
 for col in range(width):
  if input_image[row, col, 0] <
  hidden_image[row, col, 0]:

    input_image[row, col, 1] = 0
     elif input_image[row, col, 0] ==
            hidden_image[row, col, 0]:

          input_image[row, col, 1] = 127
       elif input_image[row, col, 0] >
       hidden_image[row, col, 0]:

        input_image[row, col, 1] = 255
# Apply similar logic for
other channels (G and B)
```

In the subsequent sections of this paper, we present our refined algorithm, crafted based on a more streamlined approach. We delineate the enhanced methodology, delve into the implementation intricacies within the ROS framework, and furnish experimental results showcasing the efficacy of our optimized solution in efficiently unraveling the hidden image within the confines of the Kryptonian challenge.

IV. Final Approach

To address the inefficiencies encountered in the initial attempts, we devised a more optimized solution centered around the binary search algorithm. This refined approach markedly diminished the maximum number of iterations necessary to unveil the concealed image within the given system. The following code snippet elucidates the final approach:

```
# Initialization
final_r = 127 *
np.ones((img_size[0], img_size[1]))

final_g = 127 *
np.ones((img_size[0], img_size[1]))

final_b = 127 *
np.ones((img_size[0], img_size[1]))
change = 128

# ROS initialization
rospy.init_node('player_node')
rate = rospy.Rate(10)
pub1 = rospy.Publisher
('guess', Image, queue_size=10)

rospy.Subscriber('result', Image, guessCallback)

# Iterative process
while not rospy.is_shutdown():
    guess = cv2.merge([final_b,final_g,final_r])
    msg = bridge.cv2_to_imgmsg(guess)
    pub1.publish(msg)
    rate.sleep()
    if imgMsg_res is not None:
        if (np.sum(imgMsg_res!=127) == 0):
            rospy.signal_shutdown("Task Complete")
            exit()
        final_r = final_r *
        (imgMsg_res[:,:,1] == 127)

        final_r = (final_r - change) *
        (imgMsg_res[:,:,1] == 255)

        final_r = (final_r + change) *
        (imgMsg_res[:,:,1] == 0)

        final_g = final_g *
        (imgMsg_res[:,:,0] == 127)

        final_g = (final_g - change) *
        (imgMsg_res[:,:,0] == 255)

        final_g = (final_g + change) *
        (imgMsg_res[:,:,0] == 0)

        final_b = final_b *
        (imgMsg_res[:,:,2] == 127)

        final_b = (final_b - change) *
        (imgMsg_res[:,:,2] == 255)

        final_b = (final_b + change) *
        (imgMsg_res[:,:,2] == 0)

        imgMsg_res = None
        change = change / 2
```

In this final approach, we adopted a binary search algorithm to efficiently update the pixel values of the output image channels (red, green, and blue) based on the comparison

results obtained from the hidden image (imgMsg_res). Here's an overview of the key steps:

1) **Initialization**: We initialize the final red (final_r), final green (final_g), and final blue (final_b) channels as 8-bit images with all pixel values set to 127. The variable "change" is set to 128, representing the maximum change value in each iteration.
   Iterative Process: The algorithm iterates until a solution is found or the ROS shutdown signal is received. Within each iteration, the following steps are performed:
   -
   - **Generate Guess**: We merge the current red, green, and blue channels into a single guess image.
   - **Publish Guess**: The guess image is published as a ROS image message for processing.
   - **Comparison and Update**: If the comparison result (imgMsg_res) is available, we update the pixel values of the output image channels based on the comparison results. The rules provided in the problem statement are selectively applied to update the pixel values.
   - **Halving the Change**: After updating the pixel values, the "change" value is halved to reduce the range of possible values for the next iteration, facilitating faster convergence towards the hidden image.
2) **Task Completion**: If the sum of elements in imgMsg_res that are not equal to 127 is zero, it indicates that all pixel values have been successfully matched to reveal the hidden image. In this case, the algorithm sends a shutdown signal to ROS and exits.

By implementing the binary search algorithm and the optimized update process described above, the final approach notably reduces the maximum number of iterations required to unveil the hidden image, efficiently converging towards the solution.

V. Results and Observation

Compare your results with all the available algorithms which you may have used to tackle the PS. If possible, present your and their results in tabular / graphical format.

Explain the trend of results in details. Mention the drawbacks (if any) of your algo compared to other algo and the reason of picking up the approach over the other if you have implemented any algo over the other.

CONCLUSION

In conclusion, our devised solution effectively tackles the formidable Kryptonian challenge of deciphering a concealed image within a complex system. Through the implementation of an algorithm utilizing ROS and adhering to the prescribed rules, we proficiently generate the requisite output image, leveraging the intricate interplay between the pixel values of the input and hidden images. Our solution not only showcases the intellectual prowess of humanity but also enriches our comprehension of image analysis and algorithmic design in the realm of intricate image relationships.