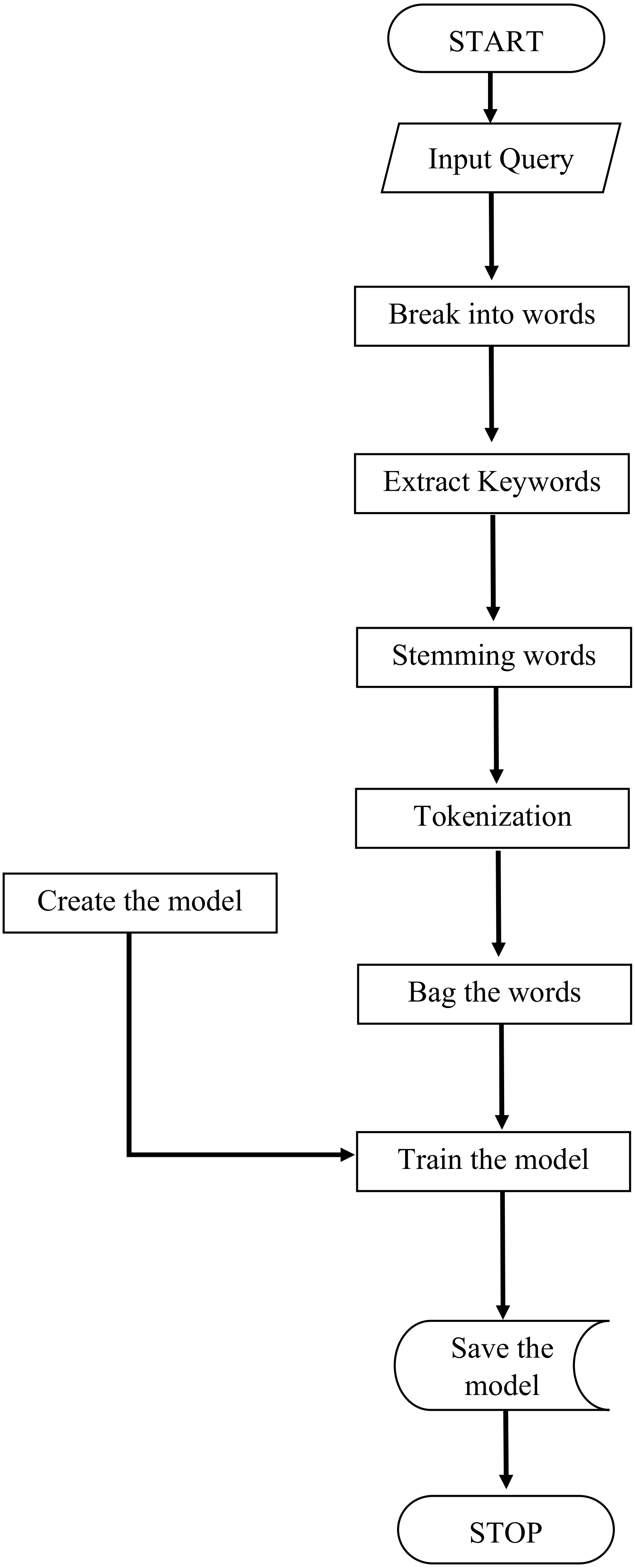


1

FIGURE : Pre-processing Assistance

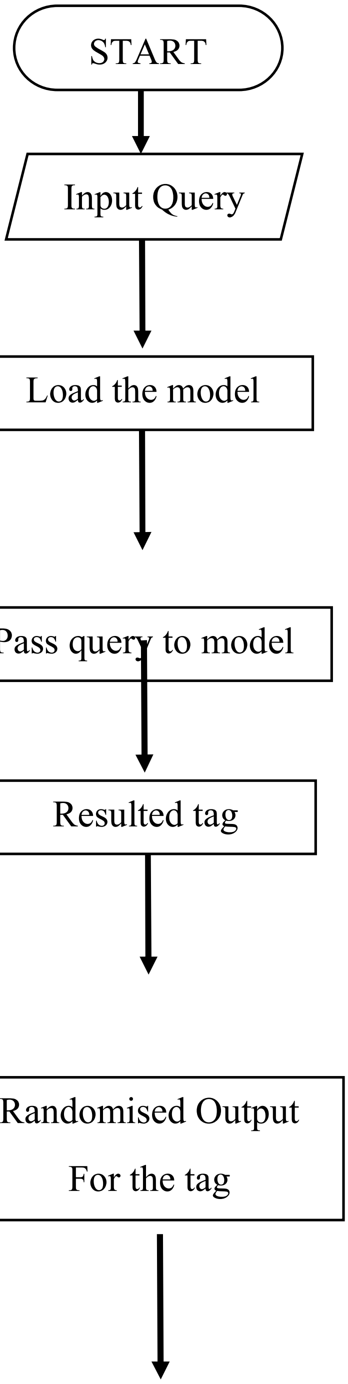
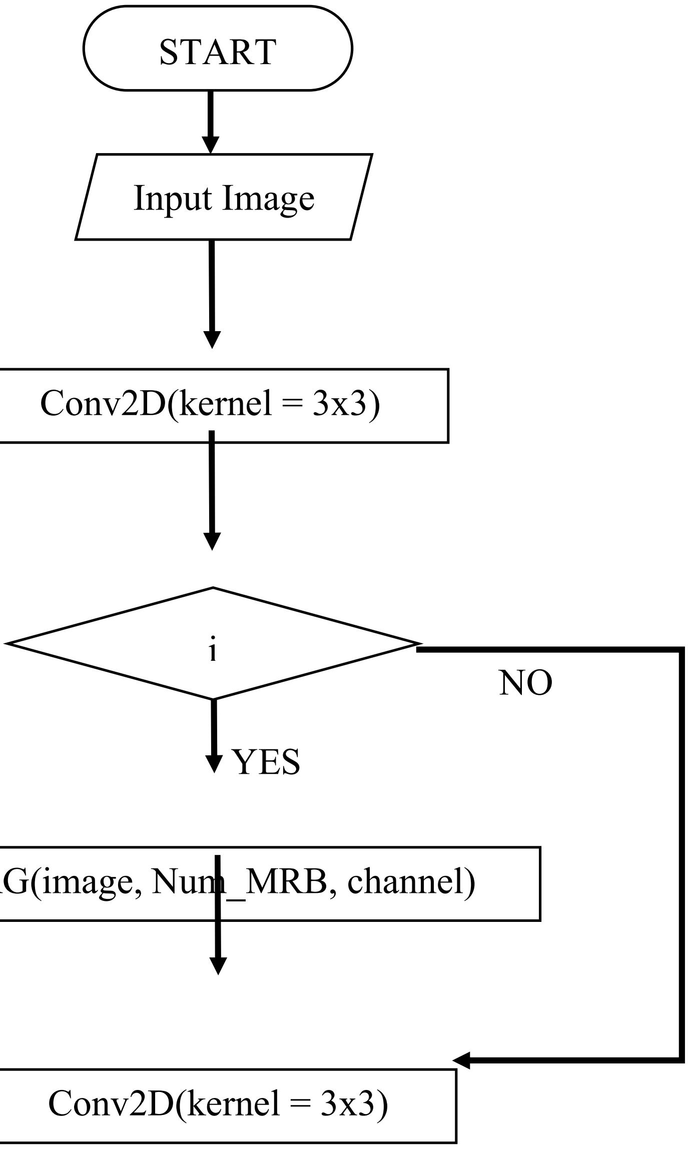

START
Input Query
Load the model
Pass query to model
Resulted tag
Randomised Output
For the tag

FIGURE : Using Pretrained



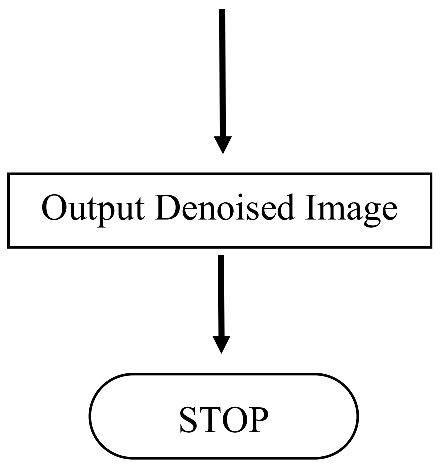
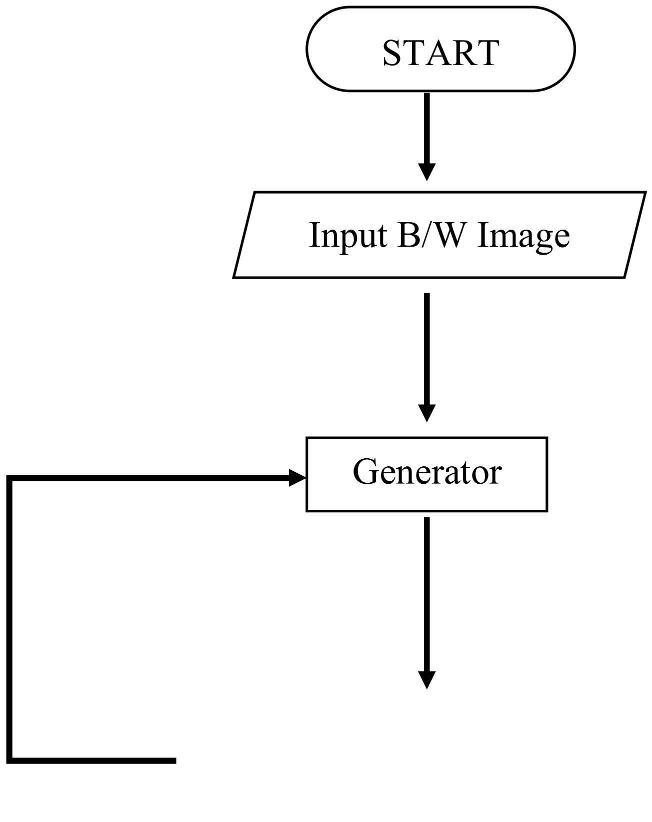


FIGURE : MIRNet Neural Network for Image Denoising



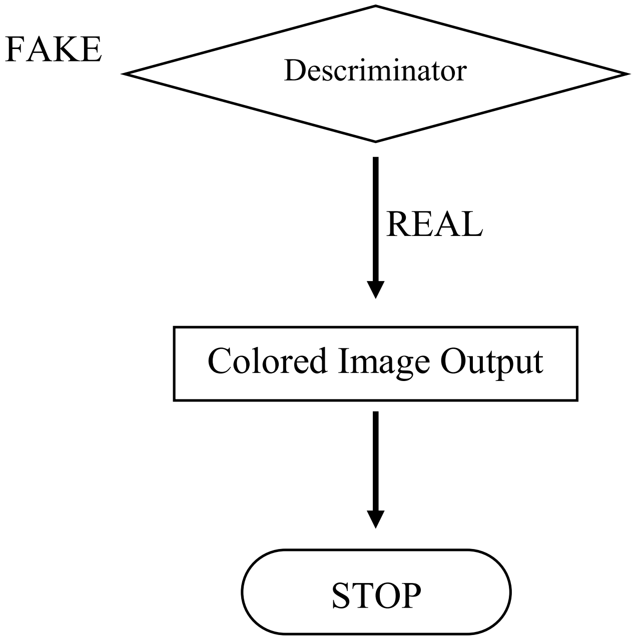
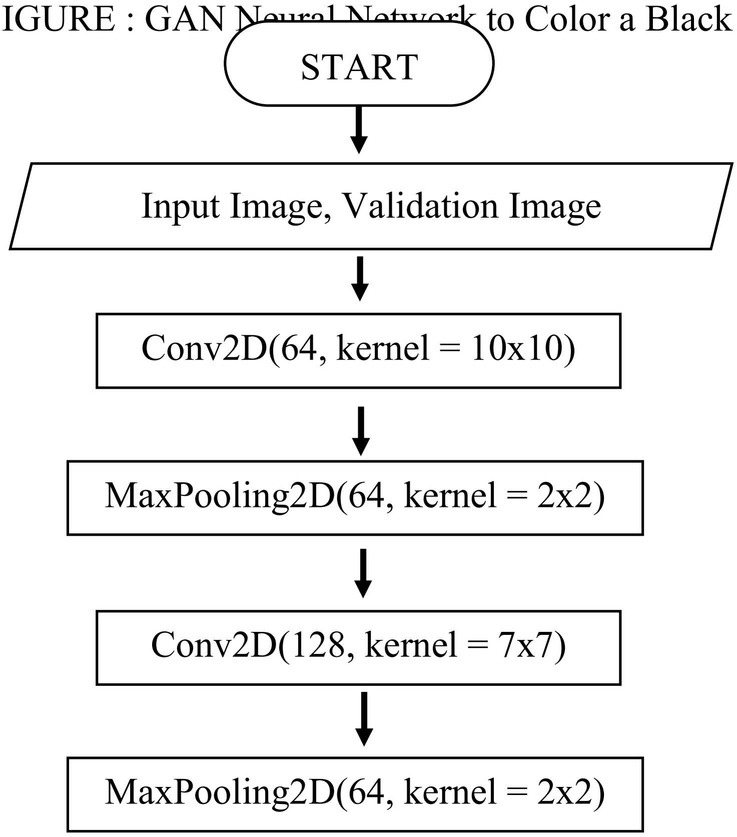
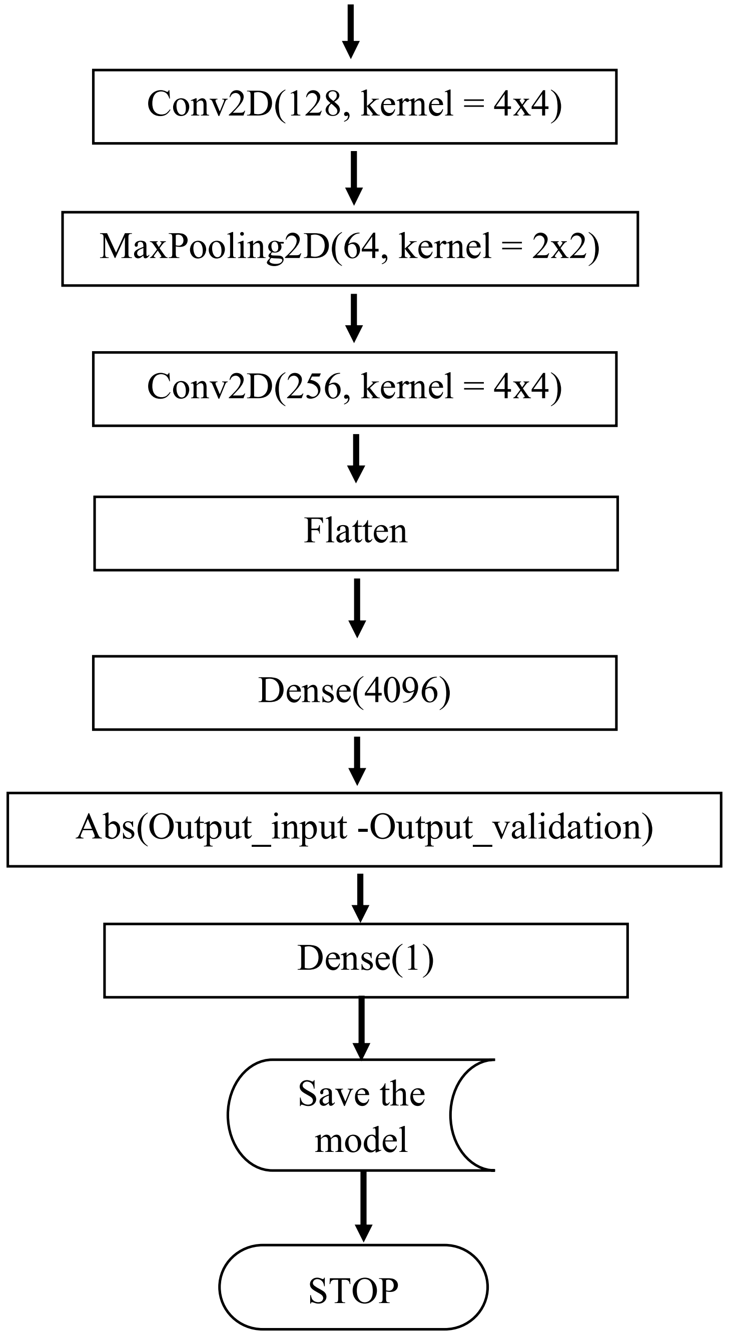
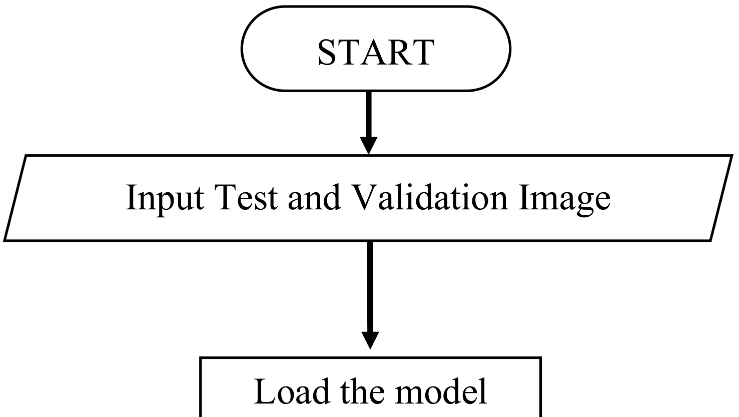


FIGURE : GAN Neural Network to Color a Black and


START
Input Image, Validation Image
Conv2D(64, kernel = 10x10)
MaxPooling2D(64, kernel = 2x2)
Conv2D(128, kernel = 7x7)
MaxPooling2D(64, kernel = 2x2)


Conv2D(128, kernel = 4x4)
MaxPooling2D(64, kernel = 2x2)
Conv2D(256, kernel = 4x4)
Flatten
Dense(4096)
Abs(Output_input -Output_validation)
Dense(1)
Save the model
STOP


START
Input Test and Validation Image
Load the model

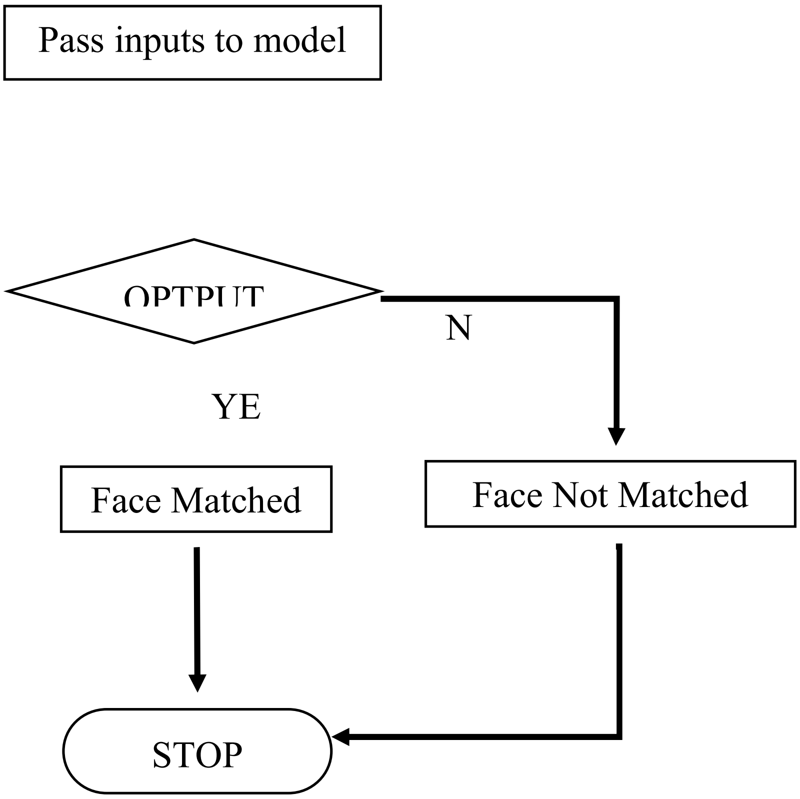


FIGURE : Siamese Neural Network for One Shot Image

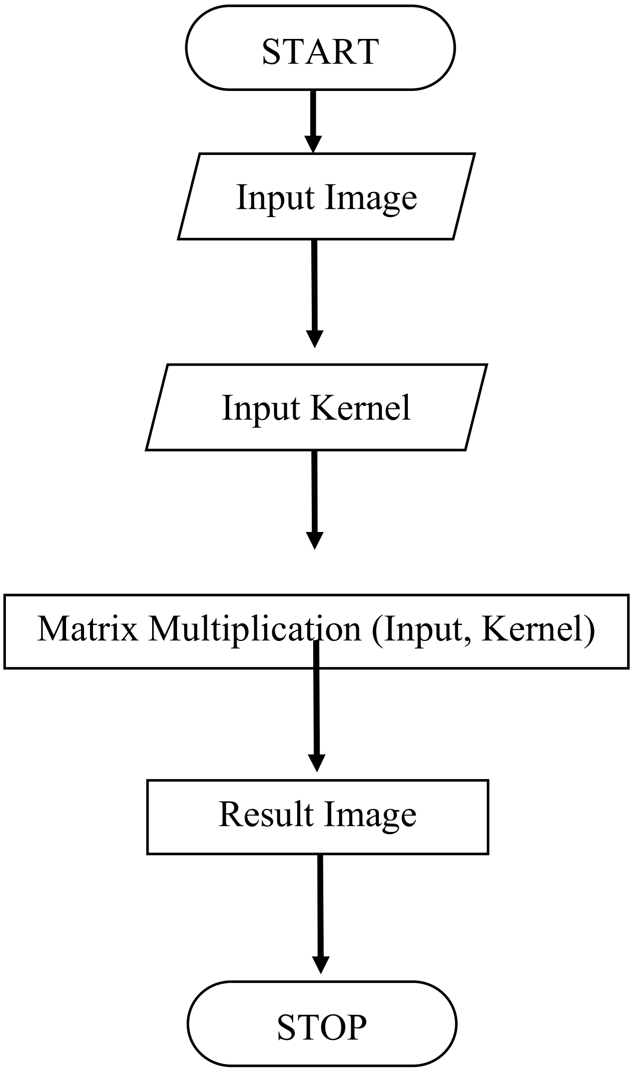


FIGURE : Manual Operation on Image Data to Denoise, Smoothing

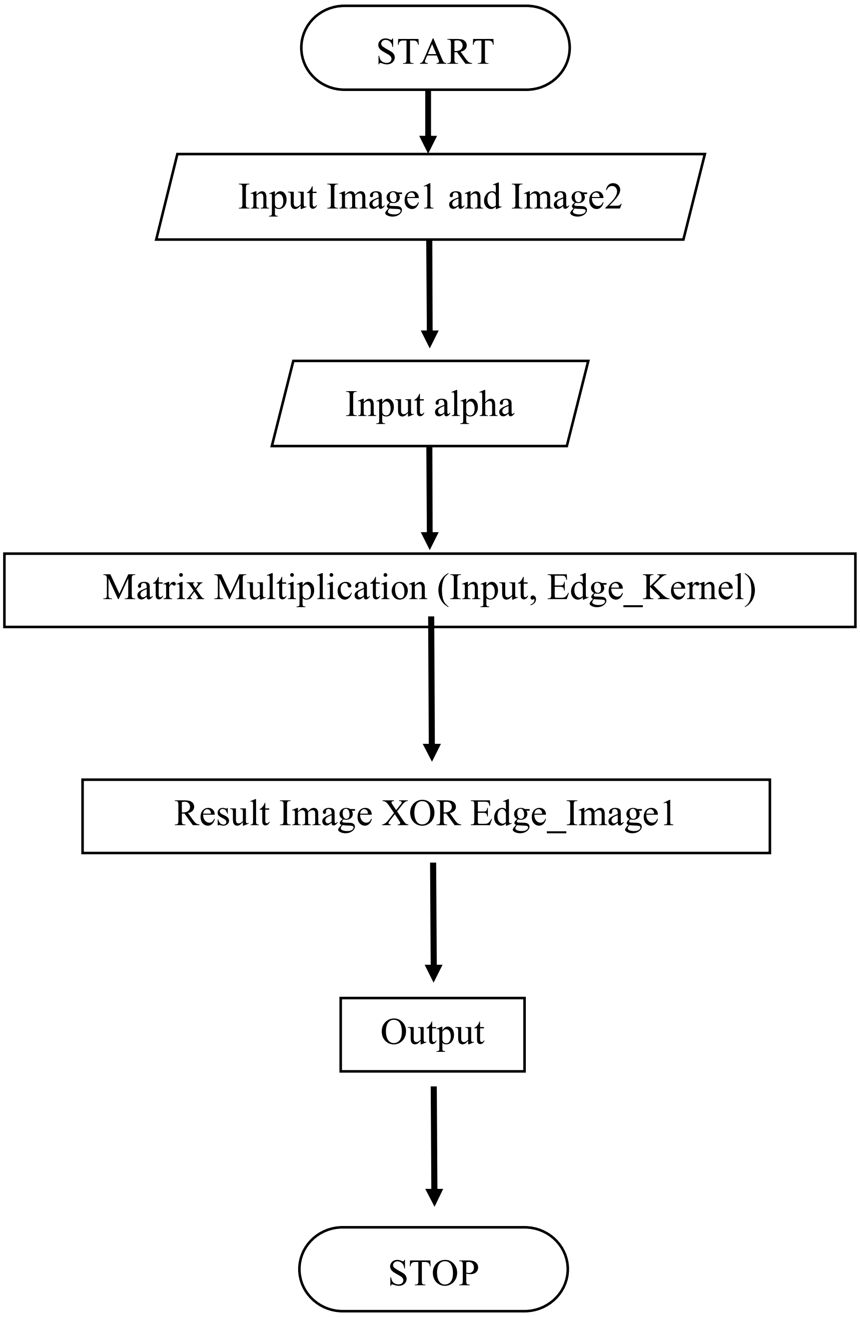

START
Input Image1 and Image2
Input alpha
Matrix Multiplication (Input, Edge_Kernel)
Result Image XOR Edge_Image1
Output
STOP

FIGURE : To detect disturbed or displaced objects in

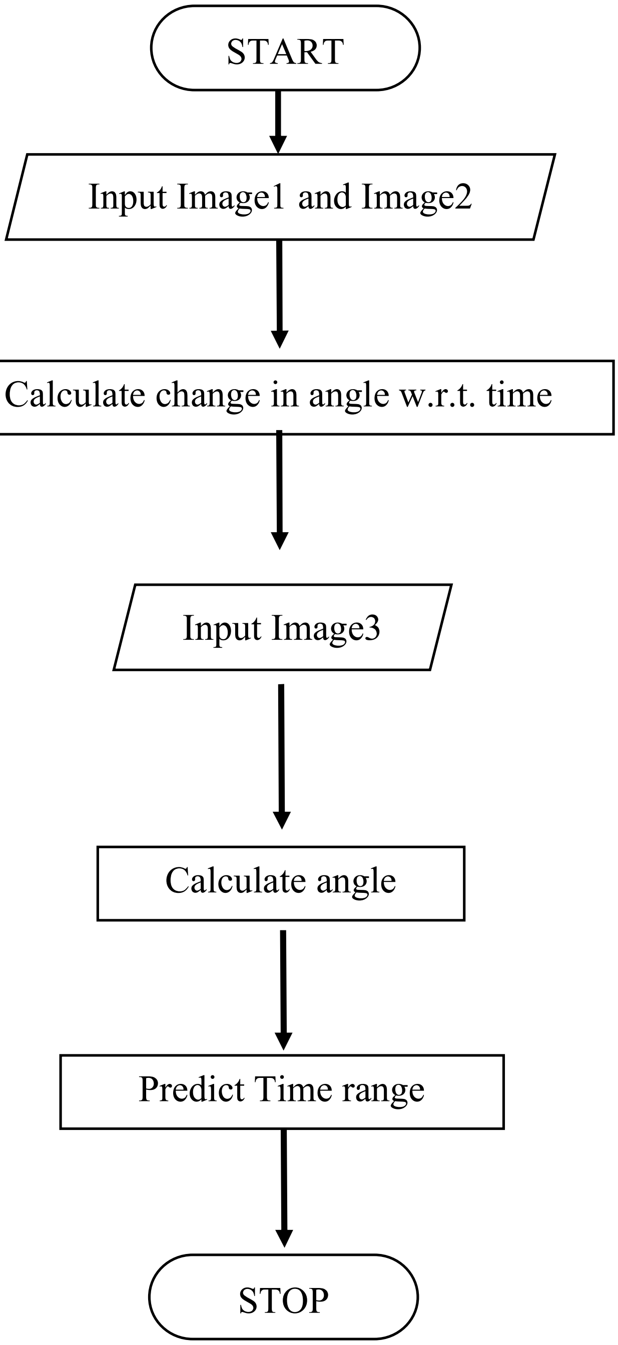


FIGURE : Shadow based time detection

ALGORITHM TO SEARCH IMAGE CONTAING FACE IN A DIRECTORY

[TO CREATE AND TRAIN NEURAL NETWORK]

STEP 1 : Start

STEP 2 : Input image

STEP 3 : Reshape image to 100 pixel by 100 pixel

STEP 4 : Add 2D convolutional layer(feature=64, kernel=(10,10), activation=RelU)

STEP 5 : Add 2D maxpooling layer(feature=64, pool_size=(2,2), padding=same)

STEP 6 : Add 2D convolutional layer(feature=128, kernel=(7,7), activation=RelU)

STEP 7 : Add 2D maxpooling layer(feature=64, pool_size=(2,2), padding=same)

STEP 8 : Add 2D convolutional layer(feature=128, kernel=(4,4), activation=RelU)

STEP 9 : Add 2D maxpooling layer(feature=64, pool_size=(2,2), padding=same)

STEP 10: Add 2D convolutional layer(feature=256, kernel=(4,4), activation=RelU)

STEP 11: Flatten the resultant matrix of STEP 10

STEP 12: Dense the flattened output to 4096 units using sigmoid activation and save

STEP 13: Pass the input image and saved result to Model function in keras

STEP 14: Add L1 distance layer to the model

STEP 15: Train the model with labelled images and output

STEP 16: Save the model

STEP 17: Stop

[TO USE THE SAVED MODEL]

STEP 1: Load the model

STEP 2: Input the test image

STEP 3: For each image in directory

STEP 4: Input validation image and test image to model

STEP 5: If model returns 1, goto the step 7

STEP 6: If no more validation image, goto the step 8

STEP 7: Display the matched image and image name

STEP 8: Stop

ALGORITHM TO UPSCALE A LOW RESOLUTION IMAGE INTO A HIGH RESOLUTION IMAGE USING PRETRAINED GAN [ESRGAN : ENHANCED SUPER RESOLUTION GENERARIVE ADVERSARIAL NETWORK]

```
STEP 1: Load the pretrained model
STEP 2: Input image
STEP 3: image = image * 1.0/255
STEP 4: data = Unsqueeze image
STEP 5: Pass data to model
STEP 6: Output High Resolution Image
STEP 7: Exit
```

[ALGORITHM TO UNSQUEEZE the image data]

```
STEP 1: data equals empty array
STEP 2: For each channel in image
STEP 3: For each row in channel
STEP 4: For each column in row
STEP 5: append value of image[channel][row][column] to data
STEP 6: Return data
STEP 7: Exit
```

ALGORITHM TO UPSCALE A IMAGE TAKEN IN LOW LIGHT INTO A BRIGHT LIGHT IMAGE USING PRETRAINED GAN [ESRGAN : ENHANCED SUPER RESOLUTION GENERARIVE ADVERSARIAL NETWORK]

```
STEP 1: Load the pretrained model
STEP 2: Input image
STEP 3: image = image * 1.0/255
STEP 4: data = Unsqueeze image
STEP 5: Create input_frames of size row=256 and column = 256 from data
STEP 6: for each input_frame in input_frames
STEP 7: pass input_frame to model
STEP 8: replace input_frame with the output of step 7
STEP 9: merge frames to create the output picture
STEP 7: Exit
```