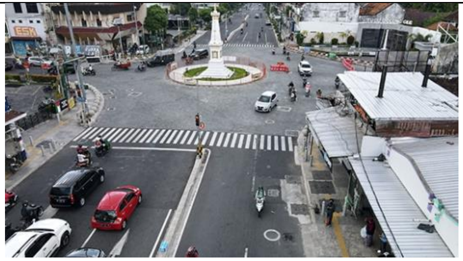
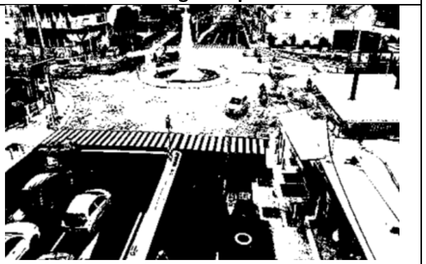
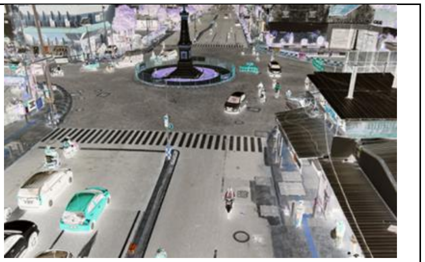
Form Evaluasi Project Tahap Akhir (UAS)

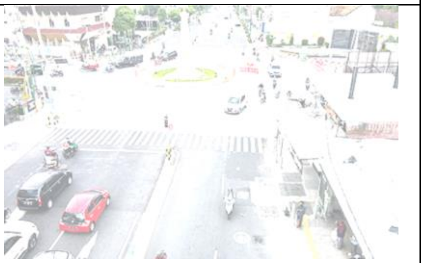
NAMA : Cecep Wahyu Cahyana
NIM : 2110410285
KELAS : E

FITUR DASAR :

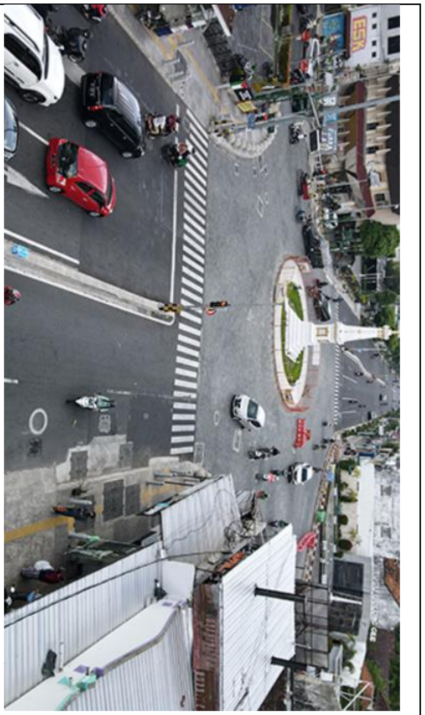
No	Fitur	Ada/Tidak	Code Fungsi / Algoritma Utama	Image Input	Image Output

1	Image Thresholding	Ada	def ImgThresHolding(img_input,coldepth):
			    #solusi 1
			    #img_output=ImageOps.invert(img_input)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			    pixels = img_output.load()
			    thresholdpic =img_input.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            if thresholdpic[i,j] < (127,127,127):
			                pixels[i,j] = (0, 0, 0)
			            elif thresholdpic[i,j] >= (127, 127, 127):
			                pixels[i,j] = (255, 255, 255)
			
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output
					

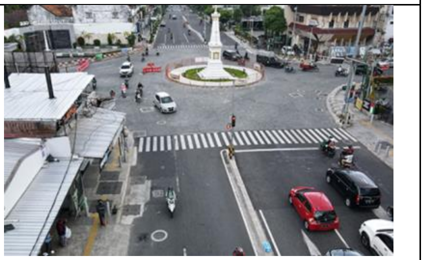
2	Image Negative	Ada	def ImgNegative(img_input,coldepth):
			    #solusi 1
			    #img_output=ImageOps.invert(img_input)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			    pixels = img_output.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            r, g, b = img_input.getpixel((i, j))
			            pixels[i,j] = (255-r, 255-g, 255-b)
			
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output
					

3	Image Brightness	Ada	def BrightnessUp(img_input, coldepth):
			    #solusi 1
			    #img_output=ImageOps.invert(img_input)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			    pixels = img_output.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            r, g, b = img_input.getpixel((i, j))
			            pixels[i,j] = (150+r, 150+g, 150+b)
			            if(r<0 and g<0 and b<0):
			                r=0
			                g=0
			                b=0
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
					
					Figure 1. Brigtness Up

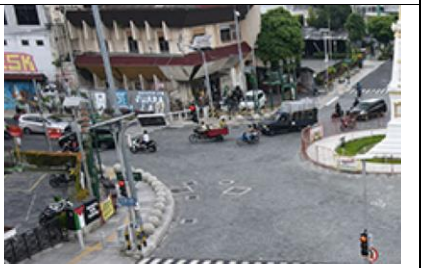
			return img_output
			def BrightnessDown(img_input,coldepth):
			    #solusi 1
			    #img_output=ImageOps.invert(img_input)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			    pixels = img_output.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            r, g, b = img_input.getpixel((i, j))
			            pixels[i,j] = (r-100, g-100, b-100)
			            if(r<0 and g<0 and b<0):
			                r=0
			                g=0
			                b=0
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output
					
					Figure 2. Brigtness Down

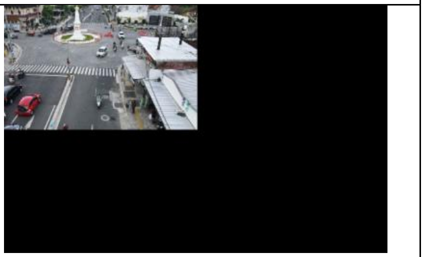
4	Image Rotation	Ada	def ImgRotate_90(img_input,coldepth,deg,direction):
			    #solusi 1
			    #img_output=img_input.rotate(deg)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[1],img_input.size[0]))
			    pixels = img_output.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            if direction=="C":
			                r, g, b = img_input.getpixel((j,img_output.size[0]-i-1))
			            else:
			                r, g, b = img_input.getpixel((img_input.size[1]-j-1,i))
			            pixels[i,j] = (r, g, b)
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output
			
			def ImgRotate_180(img_input,coldepth,deg,direction):
			    #solusi 1
			    #img_output=img_input.rotate(deg)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			    pixels = img_output.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            if direction=="C":
			                r, g, b = img_input.getpixel((i,img_output.size[1]-j-1))
			            else:
			                r, g, b = img_input.getpixel((img_input.size[0]-j-1,i))
			            pixels[i,j] = (r, g, b)
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
					

			img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output
			--- 	
			
			def ImgRotate_270(img_input,coldepth,deg,direction):
			    #solusi 1
			    #img_output=img_input.rotate(deg)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[1],img_input.size[0]))
			    pixels = img_output.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            if direction=="C":
			                r, g, b = img_input.getpixel((img_output.size[1]-j-1,i))
			            else:
			                r, g, b = img_input.getpixel((img_input.size[1]-j-1,i))
			            pixels[i,j] = (r, g, b)
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output		

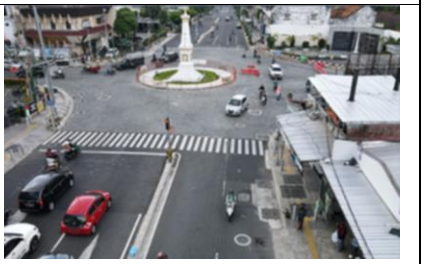
5	Image Flipping	Ada	def ImgFlip(img_input,coldepth,deg,direction):
			    #solusi 1
			    #img_output=img_input.rotate(deg)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			    pixels = img_output.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            if direction=="C":
			                r, g, b = img_input.getpixel((img_output.size[0]-i-1,j))
					

			`            else:`
			`                r, g, b = img_input.getpixel((img_input.size[0]-j-1,i))`
			`            pixels[i,j] = (r, g, b)`
			
			`    if coldepth==1:`
			`        img_output = img_output.convert("1")`
			`    elif coldepth==8:`
			`        img_output = img_output.convert("L")`
			`    else:`
			`        img_output = img_output.convert("RGB")`
			
			`    return img_output`		

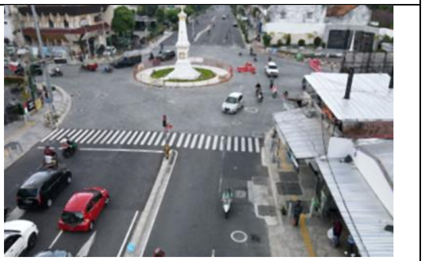
6	Image Zooming	Ada	`def Img_Zoom(img_input,coldepth):`
			`    #solusi 1`
			`    #img_output=ImageOps.invert(img_input)`
			
			`    #solusi 2`
			`    if coldepth!=24:`
			`        img_input = img_input.convert('RGB')`
			
			`    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))`
			`    pixels = img_output.load()`
			`    for i in range(img_output.size[0]):`
			`        for j in range(img_output.size[1]):`
			`            r, g, b = img_input.getpixel((i*0.5, j*0.5))`
			`            pixels[i,j] = (r, g, b)`
			
			`    if coldepth==1:`
			`        img_output = img_output.convert("1")`
			`    elif coldepth==8:`
			`        img_output = img_output.convert("L")`
			`    else:`
			`        img_output = img_output.convert("RGB")`
			
			`    return img_output`
					

7	Image Shrinking	Ada	`def ImgZoomout(img_input,coldepth):`
			`    #solusi 1`
			`    #img_output=ImageOps.invert(img_input)`
			
			`    #solusi 2`
			`    if coldepth!=24:`
			`        img_input = img_input.convert('RGB')`
			
			`    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))`
			`    pixels = img_output.load()`
			`    for i in range(img_output.size[0]):`
			`        for j in range(img_output.size[1]):`
					

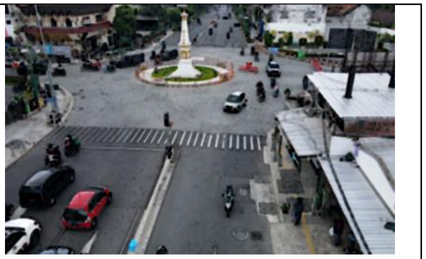
			············r, g, b = img_input.getpixel((i, j))
			            pixels[i*0.5,j*0.5] = (r, g, b)
			
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output		

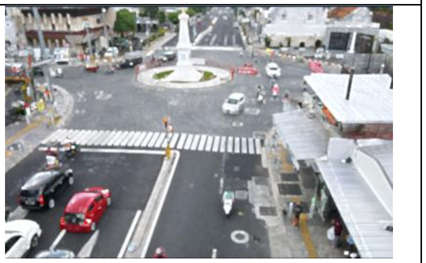
8	Mean Filtering	Ada	def ImgMean(img_input, coldepth):
			
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    kernel = 3
			    task1=[]
			    task2=[]
			    task3=[]
			    index = kernel // 2
			    img_output = Image.new('RGB', (img_input.size[0], img_input.size[1]))
			    pixels = img_output.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            for z in range(kernel):
			                if i + z - index < 0 or i + z - index > img_input.size[0] - 1:
			                    for c in range (kernel):
			                        task1.append(0)
			                        task2.append(0)
			                        task3.append(0)
			                else:
			                    if j + z - index < 0 or j + index > img_input.size[1] - 1:
			                        task1.append(0)
			                        task2.append(0)
			                        task3.append(0)
			                    else:
			                        for k in range (kernel):
			                            r,g,b = img_input.getpixel((i+z-index,j+k-index))
			                            task1.append(r)
			                            task2.append(g)
			                            task3.append(b)
			
			            # menghitung temp
			            pixels[i,j] = (round((sum(task1))/len(task1)),
			                        round((sum(task2))/len(task2)),
			                        round((sum(task3))/len(task3)))
					

			task1=[]
			            task2=[]
			            task3=[]
			
			    if coldepth == 1:
			        img_output = img_output.convert("1")
			    elif coldepth == 8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output		

9	Median Filtering	Ada	def Imgmedian(img_input,coldepth):
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]), "white")
			    pixels = img_output.load()
			    mask = [(0,0)] * 9
			    for i in range(1, img_output.size[0]-1):
			        for j in range(1, img_output.size[1] - 1):
			            mask[0] = img_input.getpixel((i-1,j-1))
			            mask[1] = img_input.getpixel((i-1,j))
			            mask[2] = img_input.getpixel((i-1,j+1))
			            mask[3] = img_input.getpixel((i,j-1))
			            mask[4] = img_input.getpixel((i,j))
			            mask[5] = img_input.getpixel((i,j+1))
			            mask[6] = img_input.getpixel((i+1,j-1))
			            mask[7] = img_input.getpixel((i+1,j))
			            mask[8] = img_input.getpixel((i+1,j+1))
			            mask.sort()
			            pixels[i,j] = (mask[4])
			
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			    return img_output
					

10	Edge Detection Pilihan 1	Tidak			
11	Edge Detection Pilihan 2	Tidak			
12	Gaussian Filtering	Tidak			

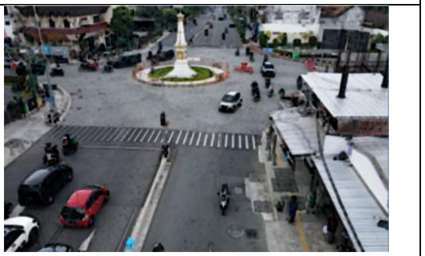
13	Erosi	Ada	def Imgerosi(img_input,coldepth,direction):
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			    pixels = img_output.load()
			    for i in range(1, img_output.size[0]-1):
			        for j in range(1,img_output.size[1]-1):
			            masukan = [img_input.getpixel((i-1, j-1)),
			                    img_input.getpixel((i-1, j)),
			                    img_input.getpixel((i-1, j+1)),
			                    img_input.getpixel((i, j-1)),
			                    img_input.getpixel((i, j)),
			                    img_input.getpixel((i, j+1)),
			                    img_input.getpixel((i+1, j-1)),
			                    img_input.getpixel((i+1, j)),
			                    img_input.getpixel((i+1, j+1))]
			            erosi_min = min (masukan)
			            erosi_max = max (masukan)
			            if direction == "min":
			                pixels[i,j] = erosi_min
			            else:
			                pixels[i,j] = erosi_max
			
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output
					

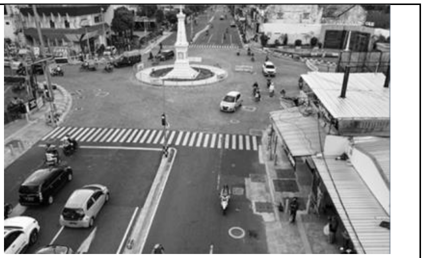
14	Dilasi	Ada	def Imgdilasi(img_input,coldepth,direction):
			    #solusi
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			    pixels = img_output.load()
			    for i in range(1,img_input.size[0]-1):
			        for j in range(1, img_output.size[1]-1):
			            masukan= [img_input.getpixel((i-1, j-1)),
			                    img_input.getpixel((i-1, j)),
			                    img_input.getpixel((i-1, j+1)),
			                    img_input.getpixel((i, j-1)),
			                    img_input.getpixel((i, j)),
					

			img_input.getpixel((i, j+1)),
			                img_input.getpixel((i+1, j-1)),
			                img_input.getpixel((i+1, j)),
			                img_input.getpixel((i+1, j+1))]
			            dilasi_min = min (masukan)
			            dilasi_max = max (masukan)
			            if direction == "max":
			                pixels[i,j] = dilasi_max
			            else:
			                pixels[i,j] = dilasi_min
			
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output		

15	Opening	Ada	def ImgOpening(img_input,coldepth,direction):
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]),"white")
			    pixel = img_input.load()
			    pixels = img_output.load()
			    for i in range(1,img_input.size[0]-1):
			        for j in range(1, img_output.size[1]-1):
			            masukan= [img_input.getpixel((i-1, j-1)),
			                img_input.getpixel((i-1, j)),
			                img_input.getpixel((i-1, j+1)),
			                img_input.getpixel((i, j-1)),
			                img_input.getpixel((i, j)),
			                img_input.getpixel((i, j+1)),
			                img_input.getpixel((i+1, j-1)),
			                img_input.getpixel((i+1, j)),
			                img_input.getpixel((i+1, j+1))]
			            opening_min = min (masukan)
			            opening_max = max (masukan)
			            if direction == "open":
			                pixels[i,j] = opening_min and opening_max
			            else:
			                pixels[i,j] = pixel[i,j]
			
			    if coldepth==1:		

			img_output = img_output.convert("1")
			elif coldepth==8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output		

16	Closing	Ada	def ImgClosing(img_input,coldepth,direction):
			#solusi 2
			if coldepth!=24:
			img_input = img_input.convert('RGB')
			
			img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]),"white")
			pixel = img_input.load()
			pixels = img_output.load()
			for i in range(1,img_input.size[0]-1):
			for j in range(1, img_output.size[1]-1):
			masukan= [img_input.getpixel((i-1, j-1)),
			img_input.getpixel((i-1, j)),
			img_input.getpixel((i-1, j+1)),
			img_input.getpixel((i, j-1)),
			img_input.getpixel((i, j)),
			img_input.getpixel((i, j+1)),
			img_input.getpixel((i+1, j-1)),
			img_input.getpixel((i+1, j)),
			img_input.getpixel((i+1, j+1))]
			closing_min = min (masukan)
			closing_max = max (masukan)
			closing= closing_max and closing_min
			if direction == "close":
			pixels[i,j] = closing
			else:
			pixels[i,j] = pixel[i,j]
			
			if coldepth==1:
			img_output = img_output.convert("1")
			elif coldepth==8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output
					

17	RGB to Grayscale	Ada	def ImgGreyscale(img_input,coldepth):
			    #solusi 1
			    #img_output=ImageOps.invert(img_input)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			
			    img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			    pixels = img_output.load()
			    threshold = img_input.load()
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            r, g, b = img_input.getpixel((i, j))
			            pixels[i,j] = (r, r, r)
			
			    if coldepth==1:
			        img_output = img_output.convert("1")
			    elif coldepth==8:
			        img_output = img_output.convert("L")
			    else:
			        img_output = img_output.convert("RGB")
			
			    return img_output
					

18	RGB to HSV/HLS Conversion	Tidak			

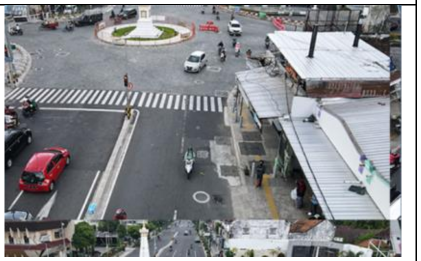
FITUR TAMBAHAN :

No	Fitur	Ada/Tidak	Code Algoritma	Image Input	Image Output

1	Image Blending	Ada	def ImgBlending(img_input,coldepth,img2,x,y,value):
			    #solusi 1
			    #img_output=ImageOps.invert(img_input)
			
			    #solusi 2
			    if coldepth!=24:
			        img_input = img_input.convert('RGB')
			        img2 = img2.open("C:/Users/M.S.I/Downloads/Undiksha Material/praktikum1/Foto/Foto2.jpg")
			
			        img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			        pixels = img_output.load()
			
			        lebar=img2.size[0]
			        tinggi=img2.size[1]		

			value2=10-value
			
			    for i in range(img_output.size[0]):
			        for j in range(img_output.size[1]):
			            r, g, b = img_input.getpixel((i,j))
			
			            if i>=lebar+x or j>=tinggi+y or i<x or j<y:
			                r2=r
			                g2=g
			                b2=b
			            else:
			                r2, g2, b2 = img2.getpixel((i-x, j-y))
			            pixels[i,j] (int((r*value2+r2*value)/10), int((g*value2+g2*value)/10), int((b*value2+b2*value)/10))
			    img_output = img_output.convert("RGB")
			
			  return img_output		

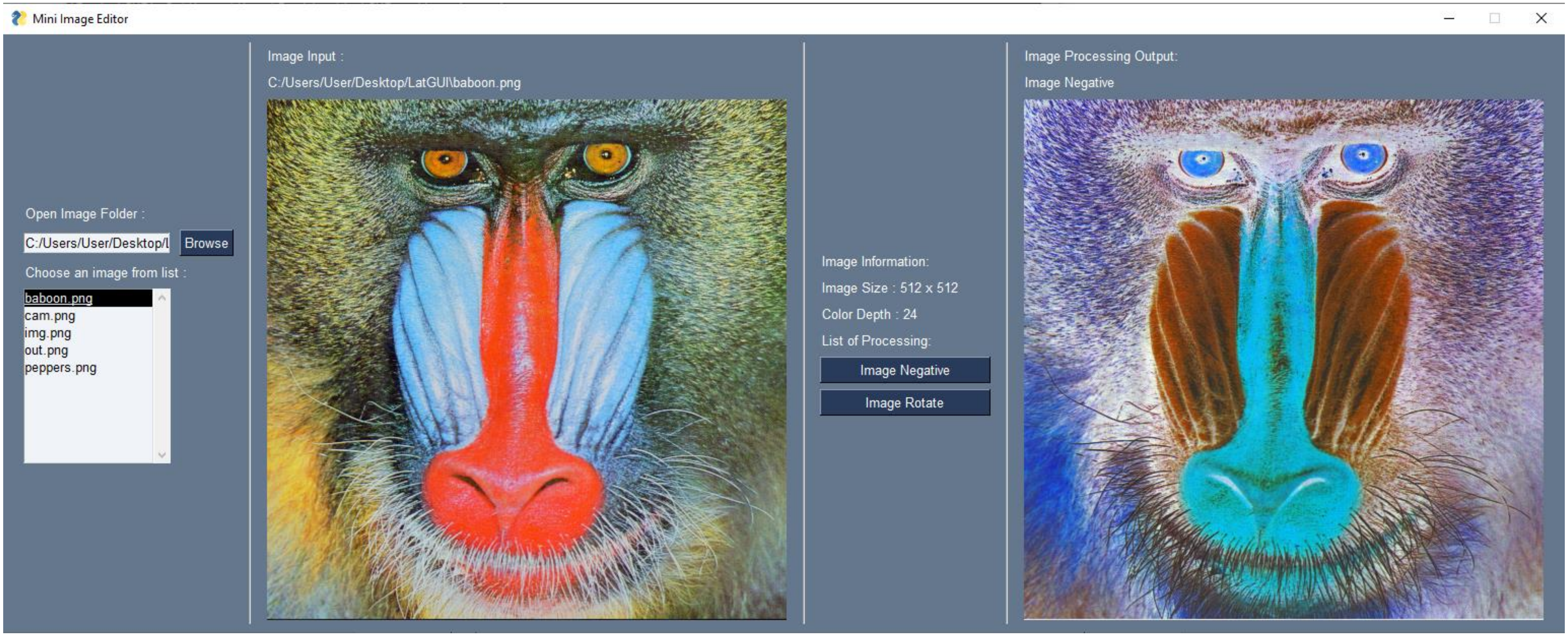
2	Image Logarithmic	Tidak			

3	Image Translation	Ada	def ImgTranslation(img_input,coldepth):
			  #solusi 1
			  #img_output=ImageOps.invert(img_input)
			
			  #solusi 2
			  if coldepth!=25:
			    img_input = img_input.convert('RGB')
			
			  img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			  pixels = img_output.load()
			  for i in range(img_output.size[0]):
			    for j in range(img_output.size[1]):
			      r, g, b = img_input.getpixel((i, j))
			      pixels[i-50,j-50] = (r, g, b)
			
			  if coldepth==1:
			    img_output = img_output.convert("1")
			  elif coldepth==8:
			    img_output = img_output.convert("L")
			  else:
			    img_output = img_output.convert("RGB")
			
			  return img_output
					

4	Edge Detection Pilihan 3	Tidak			
5	Edge Detection Pilihan 4	Tidak			
6	Edge Detection Pilihan 5	Tidak			

7	Top Hat	Tidak			

PRINT SCREEN ANTARMUKA UTAMA :



Open Image Folder :

C:/Users/User/Desktop/l

Choose an image from list :

baboon.png
cam.png
img.png
out.png
peppers.png

Image Input :

C:/Users/User/Desktop/LatGUI\baboon.png

Image Information:

Image Size : 512 x 512

Color Depth : 24

List of Processing:

Image Processing Output:

Image Negative