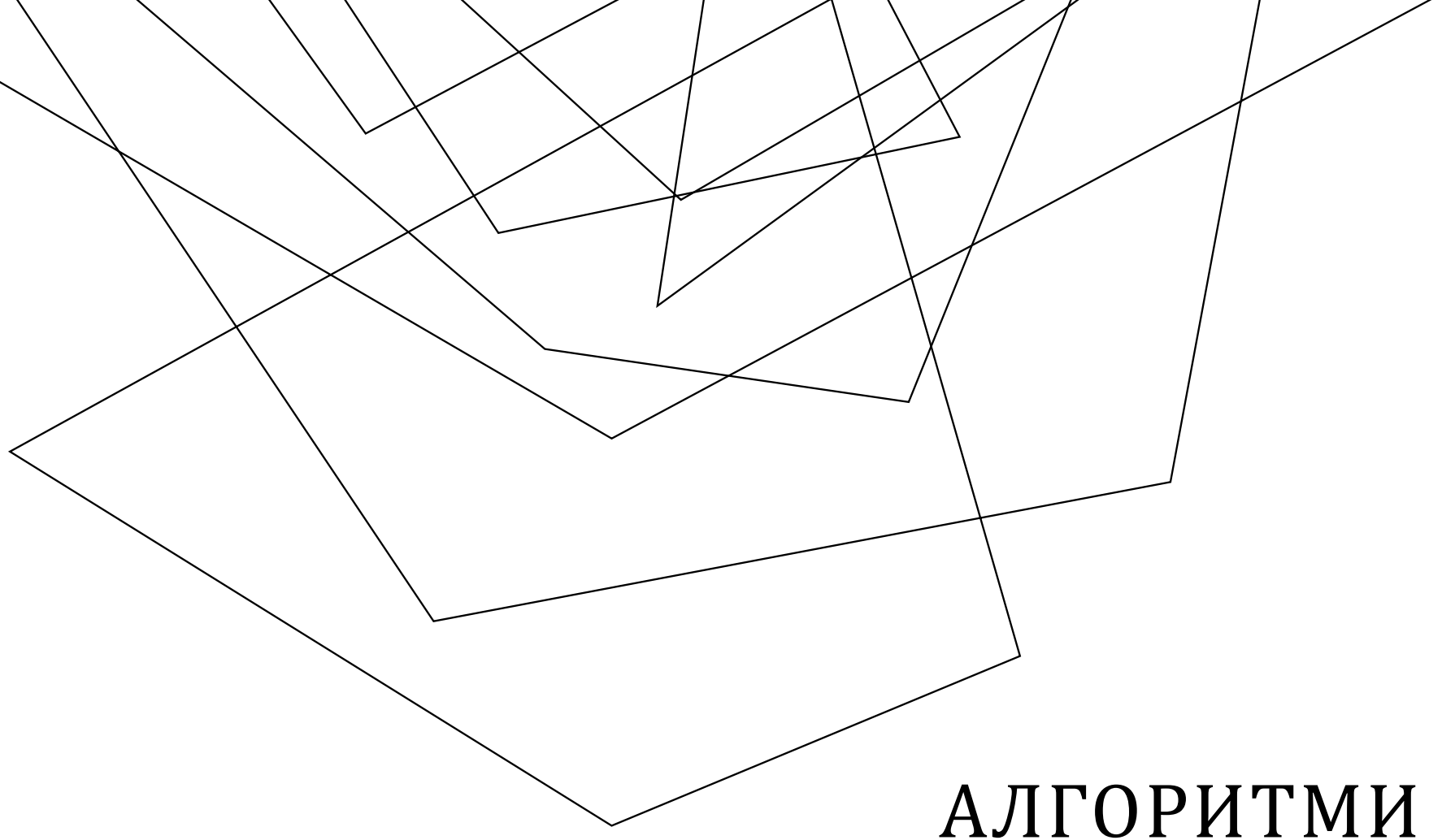


АЛГОРИТМИ ЗА ДЕТЕКЦИЈА НА РАБОВИ

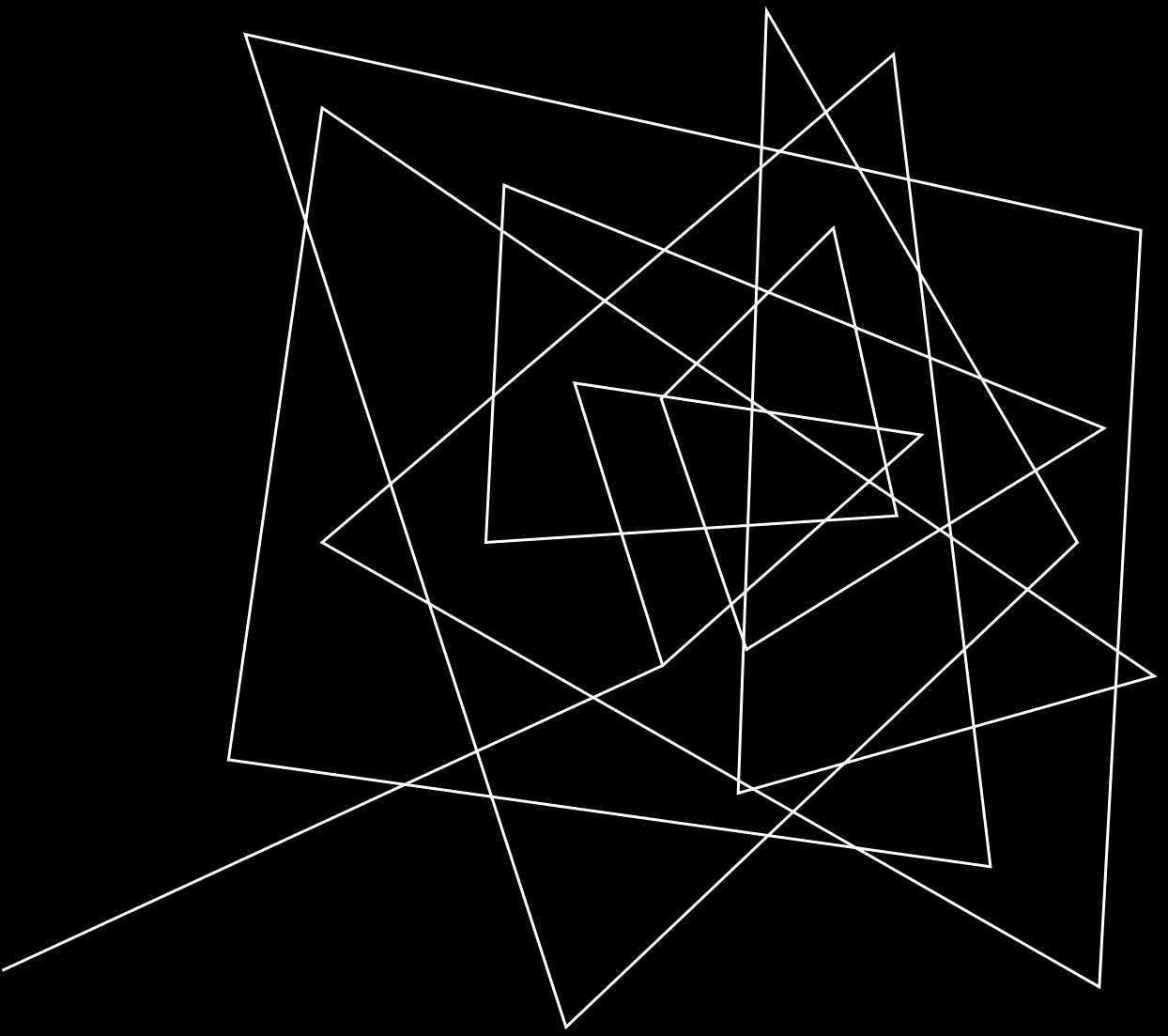
Бојан Ристов

СОДРЖИНА

ВОВЕД

Областа на детекцијата на рабови како дел од дигиталното процесирање на слики го револуционизира начинот на кој ги анализираме и интерпретираме визуелните информации, овозможувајќи ни да извлечеме вредни сознанија од сликите и нивна анализа. Способноста за прецизно откривање на рабовите е од суштинско значење за задачи како што се сегментација на слики, препознавање објекти и екстракција на карактеристики.

Фокусот на овој семинарски труд е да навлезе во областа на алгоритмите за детекција на рабови, со посебен акцент на нивната имплементација, анализа и евалуација. Мојата примарна цел е да истражам и имплементирам неколку широко користени алгоритми за откривање рабови, испитувајќи ги нивните основни принципи, силни страни и ограничувања.



ПОЗНАТИ АЛГОРИТМИ ЗА ДЕТЕКЦИЈА НА РАБОВИ

SOBEL OPERATOR

Sobel Operator е широко користен алгоритам за откривање рабови базиран на градиент кој го проценува градиентот на сликата за да ги идентификува рабовите. Ги анализира промените на интензитетот во хоризонталните и вертикалните насоки на сликата за да ја одреди големината и насоката на градиентите.

Во алгоритмот Sobel Operator, формулацијата вклучува употреба на конволуциони кернели за да се процени градиентот на сликата во хоризонтална и вертикална насока. Овие кернели се дизајнирани да ја доловат брзината на промена на интензитетот на пикселите и да обезбедат мерка за големината на градиентот на секој пиксел.

X – Direction Kernel

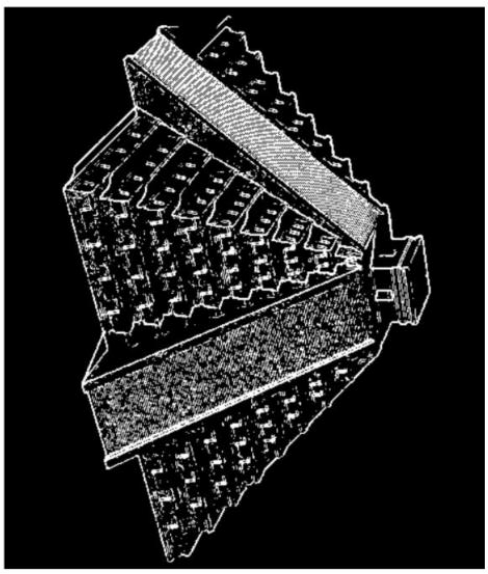
-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

ИМПЛЕМЕНТАЦИЈА НА SOBEL OPERATOR АЛГОРИТМОТ

```python
import cv2
import numpy as np

# Load the images
image_filenames = ['IMG1.jpg', 'IMG2.jpg', 'IMG3.jpg', 'IMG4.jpg', 'IMG5.jpg']
for filename in image_filenames:
    # Read the image
    img = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)

    # Apply Sobel operator
    sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)  # Sobel operator in X direction
    sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)  # Sobel operator in Y direction

    # Calculate gradient magnitude and direction
    gradient_magnitude = np.sqrt(sobelx ** 2 + sobely ** 2)
    gradient_direction = np.arctan2(sobely, sobelx)

    # Normalize gradient magnitude to 0-255 range
    gradient_magnitude = cv2.normalize(gradient_magnitude, None, 0, 255, cv2.NORM_MINMAX, dtype=cv2.CV_8U)

    # Threshold the gradient magnitude to obtain the binary edge map
    threshold_value = 50  # Adjust this threshold value as needed
    _, edge_map = cv2.threshold(gradient_magnitude, threshold_value, 255, cv2.THRESH_BINARY)

    # Display the resulting edge map
    cv2.imshow('Edge Map', edge_map)

    # Wait for a key press and then close the windows
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Sobel Operator	Оригинални фотографии
	
	
	

PREWITT OPERATOR

Операторот Prewitt е уште еден популарен алгоритам за детекција на рабови што се користи во обработката на слики. Тој е сличен на операторот Sobel и исто така работи со пресметување на градиентите на сликата. Операторот Prewitt, како и операторот Sobel, применува неколку конволуциони кернели на сликата за да го пресмета градиентот во хоризонтална и вертикална насока.

Операторот Prewitt, исто како и Sobel, ги детектира рабовите со истакнување на региони со значителни промени во интензитетот. Сепак, коефициентите на кернелот на операторот Sobel и Prewitt се разликуваат, што резултира со малку различни пресметки на градиент.

-1	0	+1
-1	0	+1
-1	0	+1

G_x

+1	+1	+1
0	0	0
-1	-1	-1

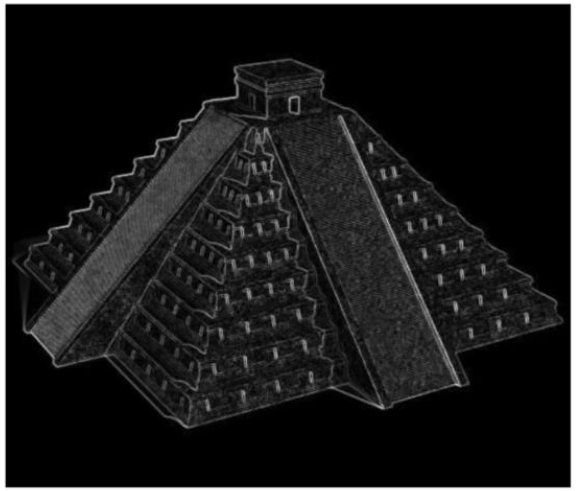
G_y

ИМПЛЕМЕНТАЦИЈА НА PREWITT OPERATOR АЛГОРИТМОТ

```
import numpy as np
from PIL import Image
import cv2

image_files = ["IMG1.jpg", "IMG2.jpg", "IMG4.jpg"]

for image_file in image_files:

    img = np.array(Image.open(image_file)).astype(np.uint8)
    gray_img = np.round(0.299 * img[:, :, 0] +
                        0.587 * img[:, :, 1] +
                        0.114 * img[:, :, 2]).astype(np.uint8)
    # Prewitt Operator
    h, w = gray_img.shape

    horizontal = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])  # s2
    vertical = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])  # s1
    newgradientImage = np.zeros((h - 2, w - 2))

    for i in range(1, h - 1):
        for j in range(1, w - 1):
            horizontalGrad = (horizontal[0, 0] * gray_img[i - 1, j - 1]) + \
                             (horizontal[0, 1] * gray_img[i - 1, j]) + \
                             (horizontal[0, 2] * gray_img[i - 1, j + 1]) + \
                             (horizontal[1, 0] * gray_img[i, j - 1]) + \
                             (horizontal[1, 1] * gray_img[i, j]) + \
                             (horizontal[1, 2] * gray_img[i, j + 1]) + \
                             (horizontal[2, 0] * gray_img[i + 1, j - 1]) + \
                             (horizontal[2, 1] * gray_img[i + 1, j]) + \
                             (horizontal[2, 2] * gray_img[i + 1, j + 1])
            verticalGrad = (vertical[0, 0] * gray_img[i - 1, j - 1]) + \
                           (vertical[0, 1] * gray_img[i - 1, j]) + \
                           (vertical[0, 2] * gray_img[i - 1, j + 1]) + \
                           (vertical[1, 0] * gray_img[i, j - 1]) + \
                           (vertical[1, 1] * gray_img[i, j]) + \
                           (vertical[1, 2] * gray_img[i, j + 1]) + \
                           (vertical[2, 0] * gray_img[i + 1, j - 1]) + \
                           (vertical[2, 1] * gray_img[i + 1, j]) + \
                           (vertical[2, 2] * gray_img[i + 1, j + 1])
            # Edge Magnitude
            mag = np.sqrt(pow(horizontalGrad, 2.0) + pow(verticalGrad, 2.0))
            newgradientImage[i - 1, j - 1] = mag
    # Convert the image to uint8 and scale the values to 0-255
    newgradientImage = (255 * (newgradientImage /
np.max(newgradientImage))).astype(np.uint8)
    cv2.imshow("Prewitt Edges", newgradientImage)
    cv2.waitKey(0)
cv2.destroyAllWindows()
```

Оригинални + фотографии

Prewitt + Operator



ROBERTS CROSS OPERATOR

Roberts Cross Operator е алгоритам за откривање рабови кој ги детектира рабовите на сликата со пресметување на големината на градиентот користејќи пар од 2x2 конволуциони кернели. Овие кернели се применуваат посебно на сликата за приближување на градиентот во хоризонтална и вертикална насока. За да се пресметаат рабовите со помош на операторот Roberts Cross, сликата се спојува со овие две кернели одделно. Големината на градиентот потоа се пресметува како квадратен корен од збирот на квадратните градиенти во хоризонталната и вертикалната насока.

Овој алгоритам може да биде чувствителен и може да произведе потенки рабови во споредба со другите алгоритми. Тој е лесен за имплементација и бара минимални пресметковни ресурси.

$$|G| = \sqrt{Gx^2 + Gy^2} \quad \left(|G| = |Gx| + |Gy|\right)$$

+1	0
0	-1

Gx

0	+1
-1	0

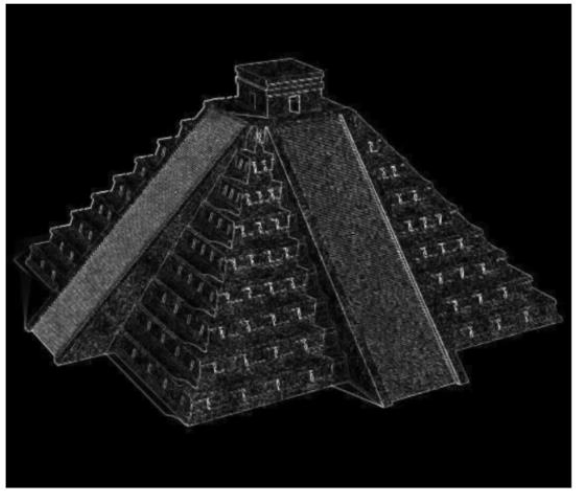
Gy

ИМПЛЕМЕНТАЦИЈА НА ROBERTS CROSS OPERATOR АЛГОРИТМОТ

```python
import cv2
import numpy as np
from scipy import ndimage
import os

# Path to the directory containing the images
image_dir = os.path.dirname(os.path.abspath(__file__))

# List of image filenames
image_files = ["IMG1.jpg", "IMG2.jpg", "IMG3.jpg", "IMG4.jpg", "IMG5.jpg"]

# Loop through the image files
for image_file in image_files:
    # Read the image
    image_path = os.path.join(image_dir, image_file)
    img = cv2.imread(image_path, 0).astype('float64')
    img /= 255.0

    # Apply the Roberts Cross Operator
    roberts_cross_v = np.array([[1, 0], [0, -1]])
    roberts_cross_h = np.array([[0, 1], [-1, 0]])
    vertical = ndimage.convolve(img, roberts_cross_v)
    horizontal = ndimage.convolve(img, roberts_cross_h)
    edged_img = np.sqrt(np.square(horizontal) + np.square(vertical))
    edged_img *= 255

    # Display the image

    cv2.imshow("Roberts Cross Edges", edged_img.astype(np.uint8))
    cv2.waitKey(0)

# Close all windows
cv2.destroyAllWindows()
```

Оригинални фотографии

Roberts Cross Operator



CANNY OPERATOR

Алгоритмот Canny Edge Detection е популарна и широко користена техника за детекција на рабови во компјутерската визија и обработката на слики. Тој е познат по неговата ефикасност во откривањето на рабовите со ниски стапки на грешки и прецизна локализација.

Операторот Canny обезбедува одлични резултати во откривањето на рабовите и е отпорен на грешки. Широко се користи во различни задачи за компјутерска визија, како што се откривање објекти, сегментација на слики и екстракција на карактеристики. Сепак, неговата повеќестепена природа го прави пресметковно поскап во споредба со другите алгоритми за откривање рабови.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

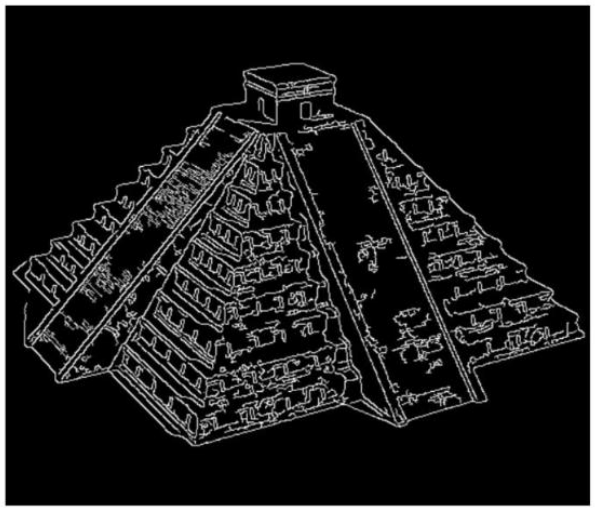
Gy

ИМПЛЕМЕНТАЦИЈА НА CANNY OPERATOR АЛГОРИТМОТ

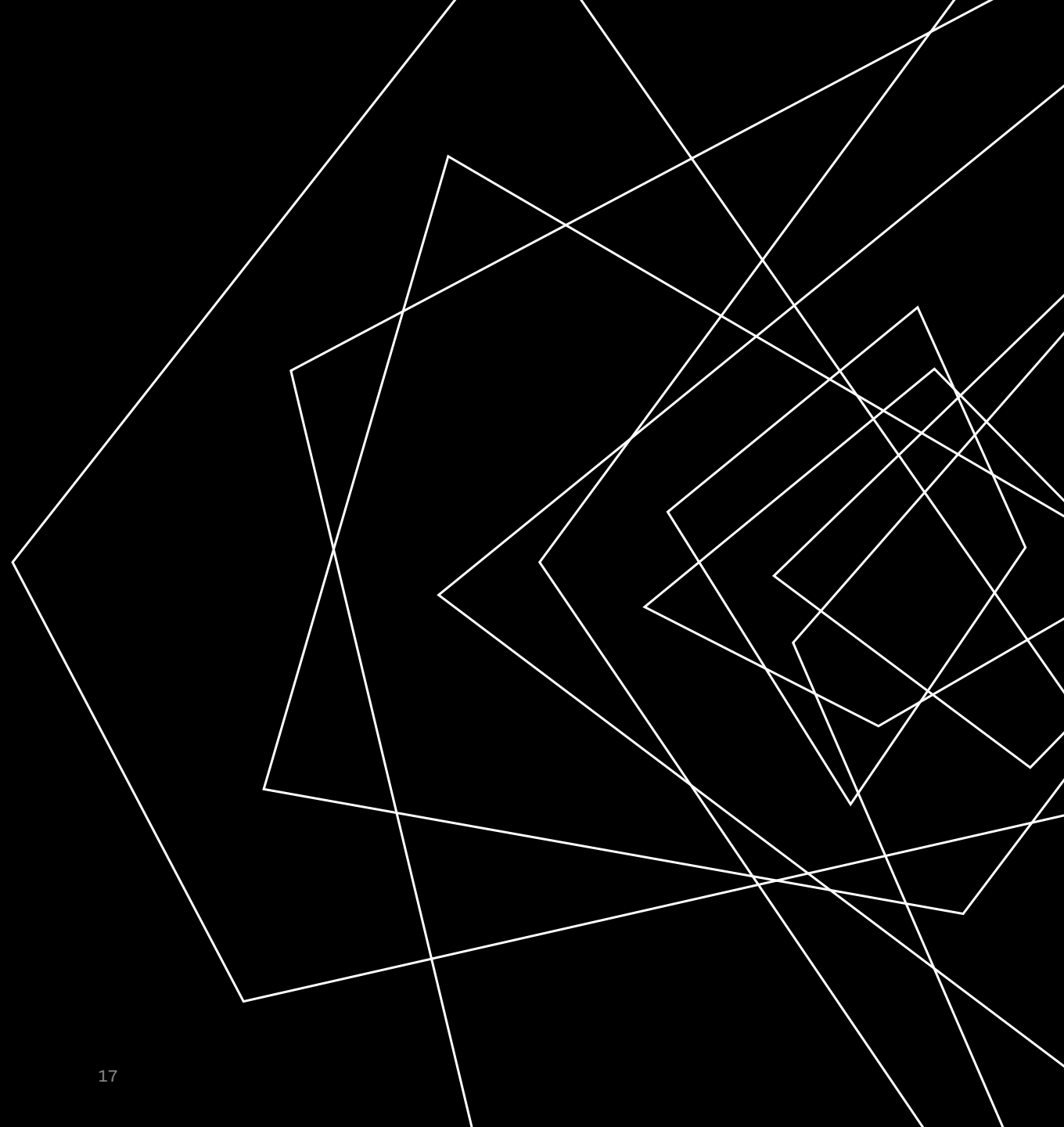
```python
import cv2
import os

def apply_canny_edge_detection(image):
    # Conversion to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Applying Gaussian blur
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Applying Canny edge detection
    edges = cv2.Canny(blurred, 50, 150)

    return edges

# Get the file paths of the images
image_folder = '.'  # Current folder
image_files = ['IMG1.jpg', 'IMG2.jpg', 'IMG3.jpg', 'IMG4.jpg', 'IMG5.jpg']

# Process each image
for image_file in image_files:
    image_path = os.path.join(image_folder, image_file)

    # Read the image
    image = cv2.imread(image_path)

    # Apply Canny edge detection
    edges = apply_canny_edge_detection(image)

    # Display the edges image
    cv2.imshow('Edges', edges)

    # Wait for key press and close the windows
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Оригинални + фотографии

Canny + Operator



ДЕМО АПЛИКАЦИЈА




Избери некој од наведените алгоритми!
Canny алгоритам
Sobel алгоритам
Prewitt алгоритам
Roberts Cross алгоритам
Примени алгоритам
Зачувај


Избери некој од наведените алгоритми!
Canny алгоритам
Sobel алгоритам
Prewitt алгоритам
Roberts Cross алгоритам
Примени алгоритам

АНАЛИЗА НА АЛГОРИТМИТЕ И ЗАКЛУЧОК

Изборот на алгоритам за откривање рабови зависи од специфичните барања на апликацијата.Забележително е дека алгоритмот за откривање на рабови Canny резултира со поголема прецизност во откривањето на рабовите и аглите, но не е најдобар за наоѓање повторливи агли. Во однос на ефикасноста, операторите Prewitt, Roberts и Sobel се брзи во споредба со другите за откривање на рабовите.

Алгоритмот Canny обезбедува висока точност и робусност, но бара повеќе пресметковни ресурси. Операторите Sobel и Prewitt нудат едноставност и разумни перформанси, додека операторот Roberts Cross е лесна опција погодна за поедноставни задачи за откривање рабови. Разбирањето на силните и слабите страни на овие алгоритми помага во изборот на најсоодветниот врз основа на потребите на апликацијата.

ВИ БЛАГОДАРАМ НА ВНИМАНИЕТО

Изработил: Бојан Ристов

Ментор: Проф. Д-р Ивица Димитровски