

SciComp with Py

Edge Detection as Derivative of Luminosity Part 2

Vladimir Kulyukin
Department of Computer Science
Utah State University



Outline

- Review
- Edge Detection with OpenCV Canny Algorithm
- Visual Comparison of Gradient and Canny Algorithms and the Effects of No-Blurring and Blurring



Review



Gradients

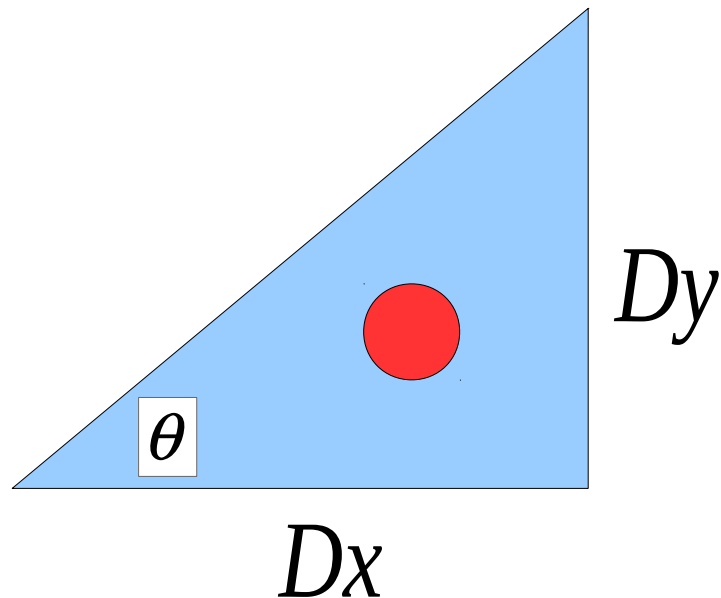
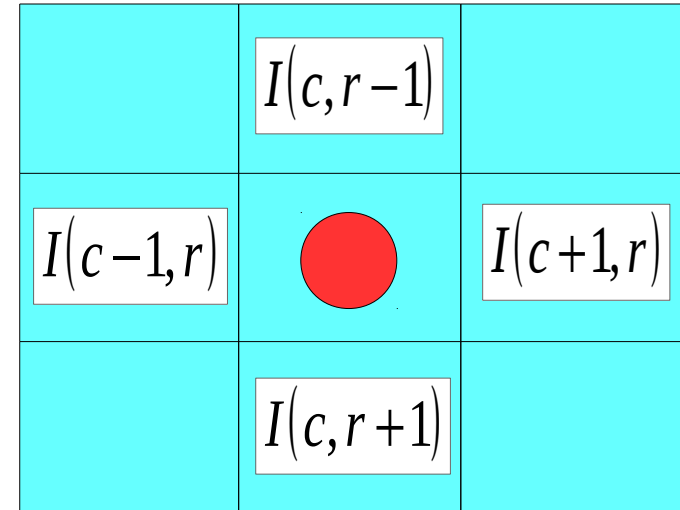
- Gradients are changes in image intensity/color
- Gradients, if viewed as vectors, have directions and magnitudes
- Gradients can be computed for each pixels or for image regions



Vertical & Horizontal Changes: Dy & Dx

$$Dy = I(c, r - 1) - I(c, r + 1)$$

$$Dx = I(c + 1, r) - I(c - 1, r)$$



$\|G\| = \sqrt{Dy^2 + Dx^2}$ is the gradient's magnitude at $I(c, r)$

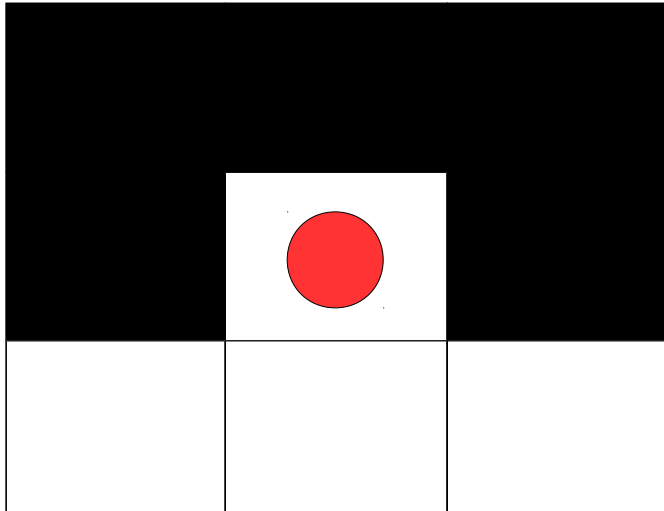
$\theta = \tan^{-1}\left(\frac{Dy}{Dx}\right)$ is the gradient's orientation

What if $Dx = 0$? In this case, we can set Dx to some small default value, e.g., 1.

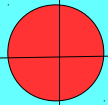


Example

Image



Pixel Values

0	0	0
0		0
255	255	255

$$dy = 0 - 255 = -255; dx = 0 - 0 \approx 1$$

$$\|G\| = \sqrt{(-255)^2 + 1^2} = 255.00196078 \approx 255$$

$$\theta = \left(\tan^{-1} \left(\frac{-255}{1} \right) \right) \frac{180}{\pi} = -89.775311^\circ \approx -90^\circ$$

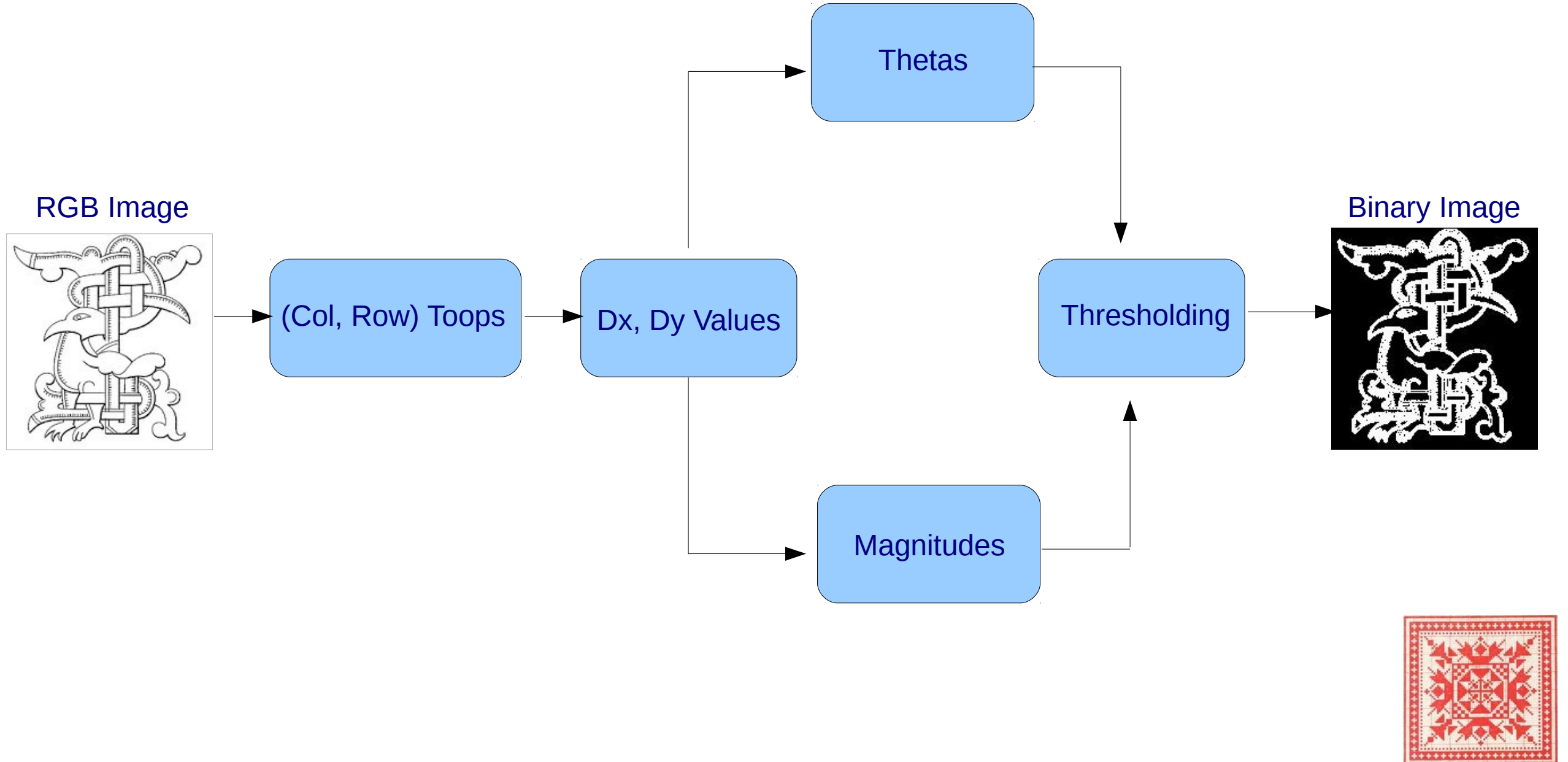


Edge Detection with Gradients

- Grayscale RGB pixels with relative luminosity
- Compute ***Dy*** and ***Dx*** at each pixel
- Compute gradient's magnitude and orientation for each pixel
- Implement an edge detection pipeline



Edge Detection Pipeline



Edge Detection with OpenCV



Canny Edge Detection

- OpenCV has a number of edge detection algorithms; one of them is Canny
- Canny edge detection is an edge detection algorithm developed by John Canny in the 1980's
- Canny edge detection: 1) noise reduction through blur; 2) use a mask (aperture) to compute gradients; 3) non-maximum suppression; 4) thresholding



Non-Maximum Pixel Suppression

- Canny edge detection takes two grayscale value parameters: *minVal* and *maxVal*
- If the gradient at a pixel is above *maxVal*: this pixel is called *strong* pixel
- If the gradient at a pixel is below *minVal*, this pixel is called a non-edge pixel
- If the gradient at a pixel is a between *minVal* and *maxVal*, then it is retained if it is connected (is in close proximity) to a strong pixel
- If a strong pixel is a local maximum but its gradient is not in the direction of the edge, it is removed
- Images are typically blurred before edge detection



Detecting Edges w/o Blurring in OpenCV

```
def detectEdgesWithoutBlur(input_path, output_path):  
    image = cv2.imread(input_path)  
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  
    image_edges = cv2.Canny(gray_image, 100, 200, apertureSize=3, L2gradient=True)  
    cv2.imshow('Input', image)  
    cv2.imshow('Gray Image', gray_image)  
    cv2.imshow('Edges', image_edges)  
    cv2.waitKey(0)  
    cv2.imwrite(args['output_path'], image_edges)  
    del gray_image  
    del image_edges
```

Py source in `cv_detect_edges.py`



Detecting Edges w/ Blurring in OpenCV

```
def detectEdgesWithBlur(input_path, output_path, gauss_blur_mask):  
    image = cv2.imread(input_path)  
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  
    blurred_image = cv2.GaussianBlur(gray_image, gauss_blur_mask, 0)  
    image_edges = cv2.Canny(blurred_image, 100, 200, apertureSize=3, L2gradient=True)  
    ## rest of code to show, save, and delete images is not shown
```

Py source in `cv_detect_edges.py`



Grayscale & Blurring



Picture is grayscale and blurred



Detected Edges



Visual Comparison of Edge Detection Results

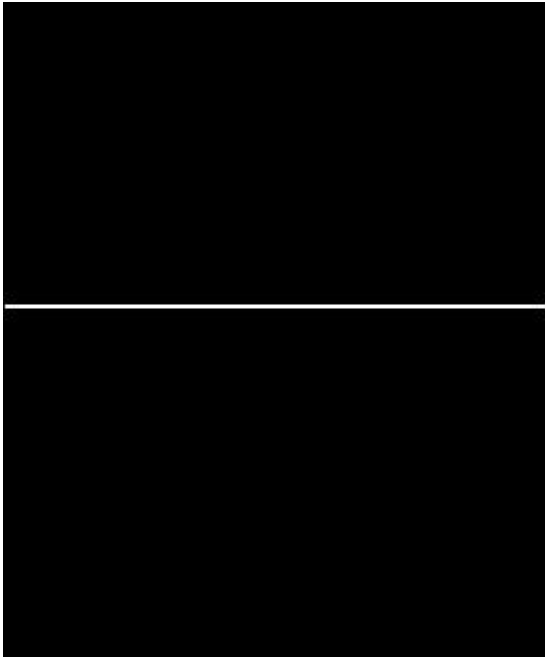


GD vs. OpenCV

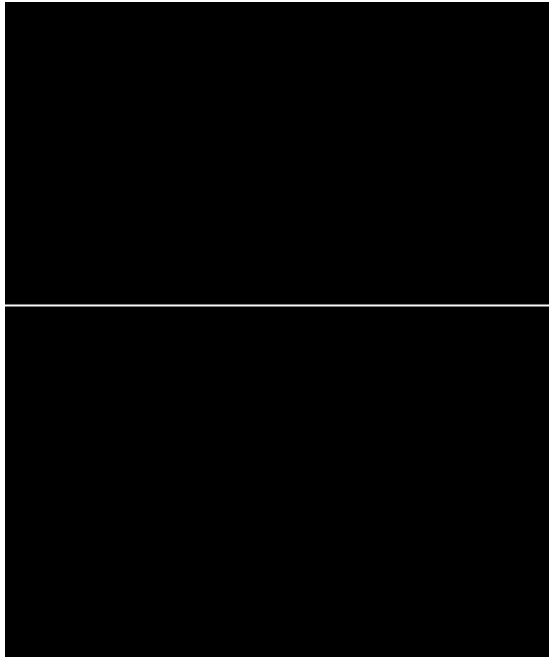
Original



GD



Canny

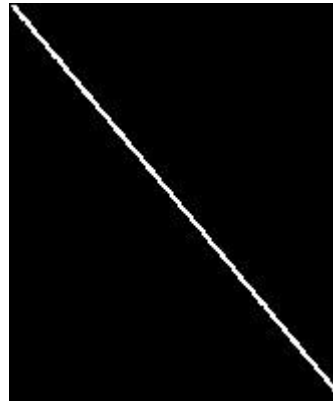


GD vs. OpenCV

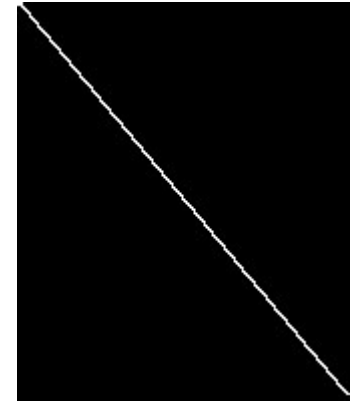
Original



GD

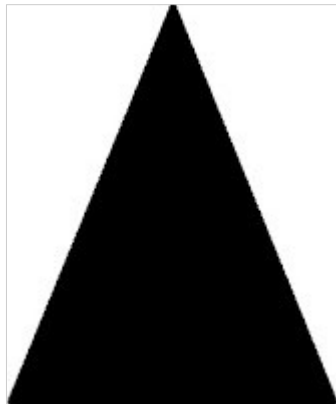


Canny

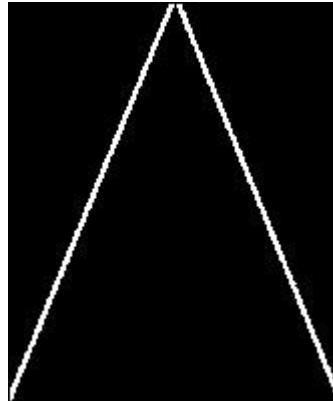


GD vs. OpenCV

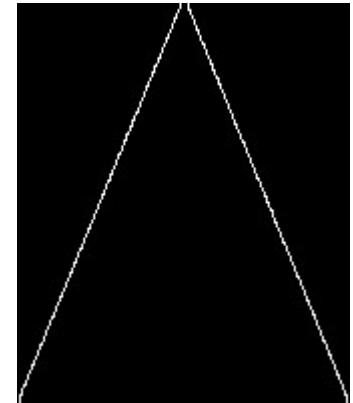
Original



GD



Canny



GD vs. OpenCV

Original



GD



Canny



GD vs. OpenCV

Original



GD



Canny

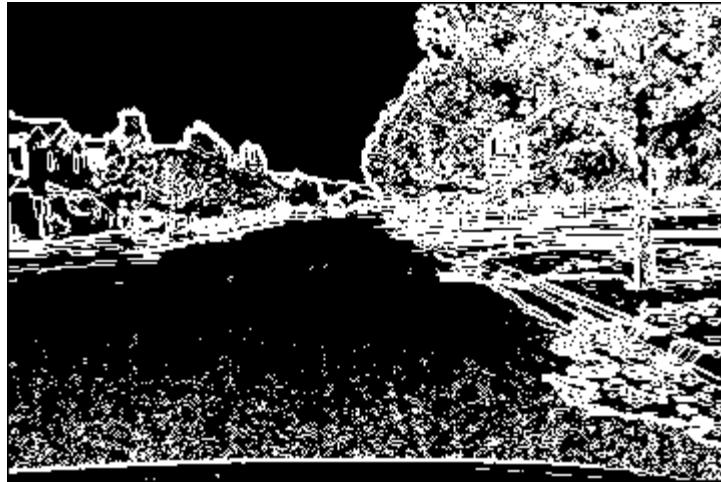


GD vs. OpenCV

Original



GD

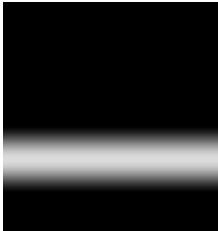


Canny



GD vs. OpenCV

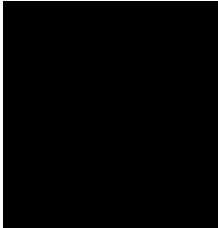
Original



GD

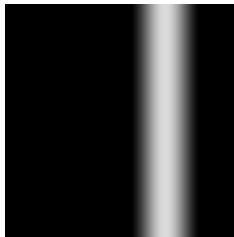


Canny



GD vs. OpenCV

Original



GD



Canny



Some Conclusions

- On simple images, the simple gradient algorithm detects edges as well as the OpenCV Canny algorithm
- On more complex images, the simple gradient algorithm detects edges more crudely than the OpenCV Canny algorithm
- Blurring does make a difference if you want fewer edges detected
- Canny may have trouble with wide blurry edges



References

- https://en.wikipedia.org/wiki/Edge_detection
- http://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html

