IF6083Vision
Assignment - 1

Oleh:

Made Arbi Parameswara (23522002)

PROGRAM STUDI MAGISTER INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

1. *Theory Questions*

1.1 *Show that it you use the line equation ρ = xcosθ + ysinθ, each image point (x,y) results in a sinusoid in (ρ, θ) Hough space. Relate the amplitude and phase of the sinusoid to the point (x,y).*

Dalam transformasi *Hough*, garis direpresentasikan secara unik dalam koordinat polar sebagai

$\rho = x\cos\theta + y\cos\theta$. Jika persamaan ini di-*plot* ke titik tertentu (x, y) dengan memvariasikan θ dengan rentang $0 - 2\pi$ maka ρ akan membentuk kurva sinus dengan ruang yang memiliki *term* sebagai (ρ, θ) . Tinggi dari sinusoid (amplitude), terkait dengan seberapa jauh titik tersebut dari titik asal: Semakin jauh titik (x, y) dari titik asal maka kurva akan memiliki amplitudo yang lebih tinggi. Sebaliknya, semakin dekat titik dengan titik asal, amplitudo akan semakin kecil. Fase dari kurva sinus ini berhubungan langsung dengan θ. Kemiringan garis akan sangat tergantung pada fasenya.

1.2 *Why do we parametrize the line in terms (ρ, θ)instead of the slope and intercept (m, c)? Express the slope and intercept in term of (ρ, θ).*

Parameterisasi garis dengan *term* (ρ, θ) diripada menggunakan kemiringan dan *intercept* (m, c) adalah jika kita menggunakan *slope-intercept*, ruang solusi bisa sangat luas, sebesar seluruh keanggoataan bilangan real, yaitu $-\infty$ sampai ∞. Sedangkan jika menggunakan (ρ, θ) ruang soulsi berada pada rentang terbatas yaitu 0 sampai 2π.

Persamaan garis berdasarkan *term* (ρ, θ),

$$x\cos\theta + y\sin\theta = \rho \tag{1}$$

Subtitusikan persamaan $y = mx + c$ ke persamaan (1), sehingga

$$x\ \cos\theta + (mx + c)\ \sin\theta = \rho \tag{2}$$

Lakukan ekspansi:

$$x\ \cos\theta + mx\ \sin\theta + c\ sin\theta = \rho \tag{3}$$

Lakukan pengelompokan berdasarkan *variable*:

$$x\ (\cos\theta + m\sin\theta) + c\ sin\theta = \rho \tag{4}$$

Lakukan komparasi dengan persamaan $y = mx + c$ maka koefisien dari x adalah m dan koefisien dari c adalah sebagai berikut:

$$m = -\frac{\cos\theta}{\sin\theta} = -\cot\theta \tag{5}$$

$$c = \frac{\rho}{\sin\theta} \tag{6}$$

1.3 ***Assuming that the image points (x, y) are in an image of width W and height H, that is, x in [1, W], y in [1, H]. What is the maximum absolute value of ρ, and what is the range for θ?***

Nilai maksimum dari ρ terjadi ketika sebuah garis melewati sudut dari sebuah gambar, sudut terjaih dari gambar dengan asal $(0,0)$ adalah jarak diagonal pada (W, H) atau dapat diekspresikan dengan:

$$\rho_{max} = \sqrt{W^2 + H^2} \tag{7}$$

Jika θ menyatakan sudut antara sumbu x dan garis dari titik asal ke suatu titik pada garis yang direpresentasikan dalam ruang *Hough*. Maka θ akan memiliki nilai pada rentang 0 sampai π sehingga *range* dari θ adalah:

$$0 \leq \theta \leq \pi$$

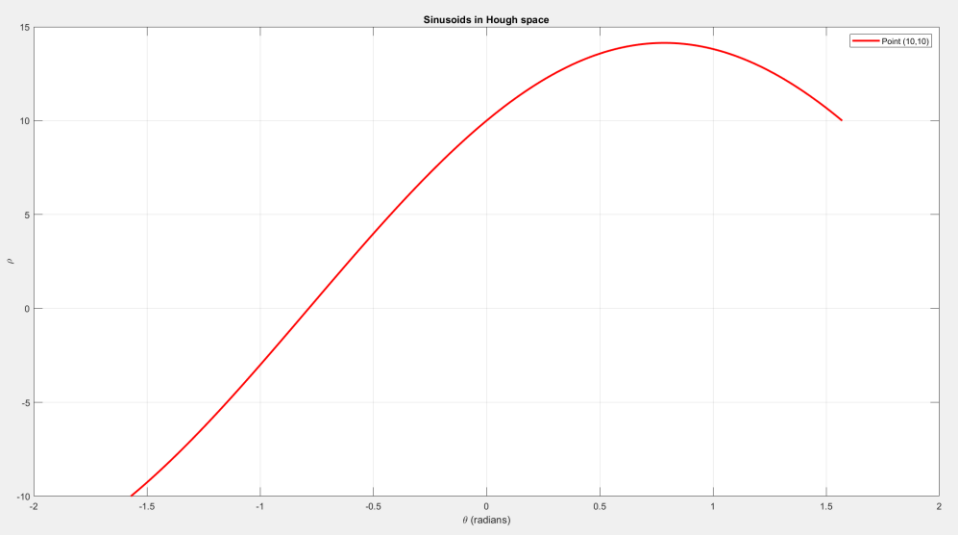
Atau dalam derajat:

$$0° \leq \theta \leq 180°$$

1.4 ***For point (10,10) and points (20,20) and (30,30) in the image, plot the corresponding sinusoid waves in Hough space, and visualize how their intersection point defines the line. What is (m, c) for this line? Please use Matlab to plot the curves and report the result in your write-up.***

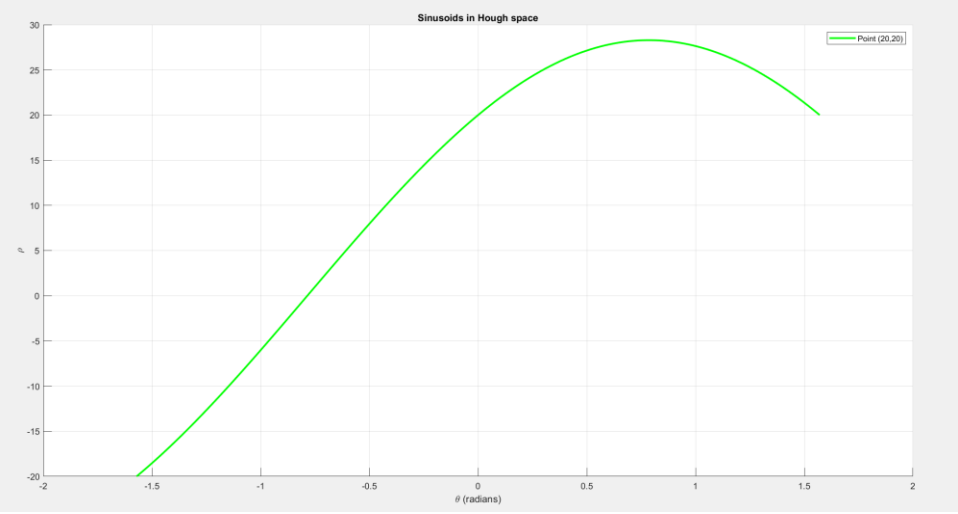
Visualisasi *sinusoid waves* pada ruang Hough untuk beberapa *point*:

- Point (10,10):



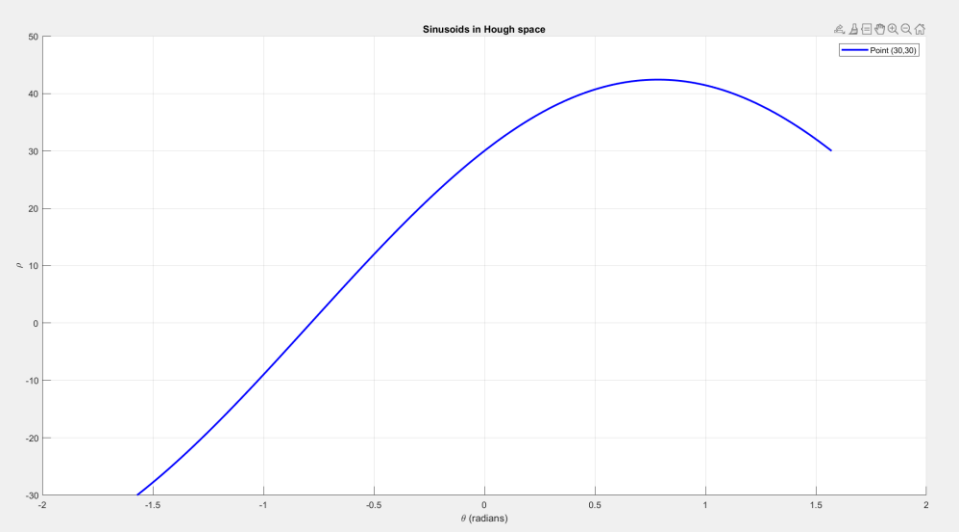
Gambar 1. *Plot* dari point (10,10) pada ruang Hough

- Point (20,20):



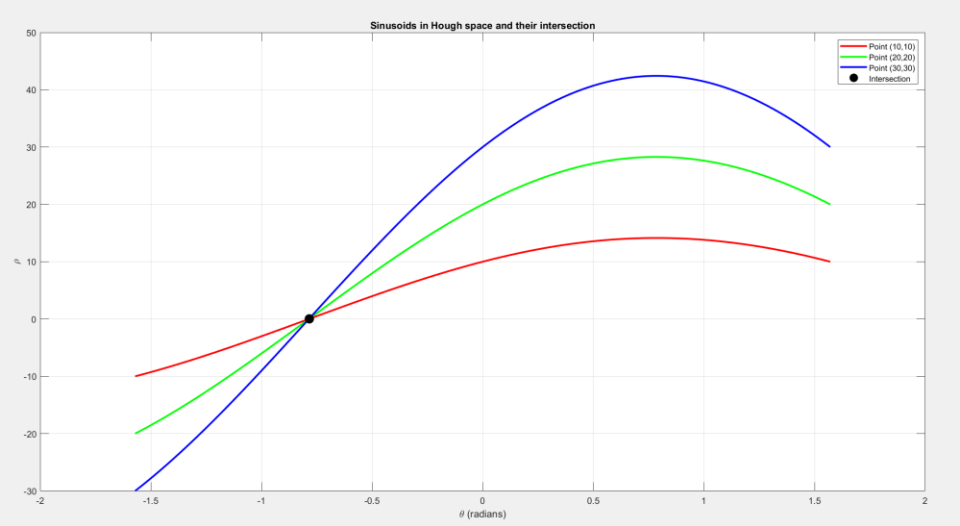
Gambar 2 *Plot* dari point (20,20) pada ruang Hough

- Point (30,30):



Gambar 3 *Plot* dari point (30,30) pada ruang Hough

- Persimpangan dari ketiga point:



Gambar 4 Titik persimpangan dari *point* (10,10), (20,20), dan (30,30) ruang *Hough*

m pada persamaan garus ini diekspresikan oleh persamaan (5) dan dan c pada persamaan garis ini diekspresikan oleh persamaan (6). Sehingga jika dikalkulasikan pada titik intersection nilai dari m dan c dapat dikomputasikan dengan menggunakan algoritma berikut pada MATLAB:

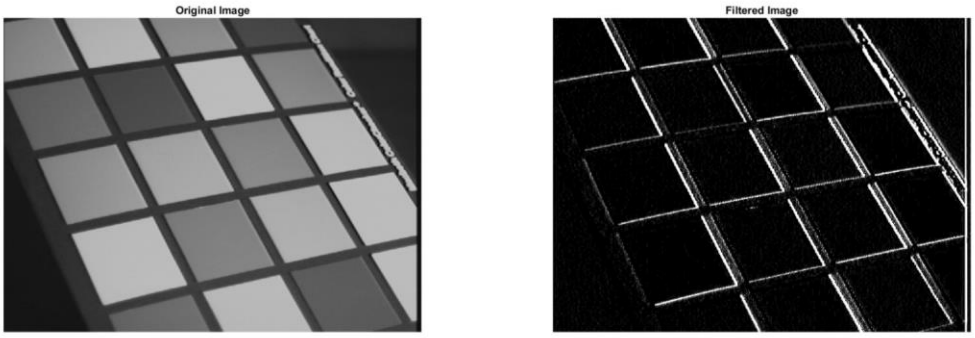
```
% Definsikan point-point
points = [10, 10; 20, 20; 30, 30];

% Definisikan range dari theta
theta = linspace(-pi/2, pi/2, 400);

% Kalkulasikan nilai rho untuk setiap titik each point
rho1 = points(1,1)*cos(theta) + points(1,2)*sin(theta);
rho2 = points(2,1)*cos(theta) + points(2,2)*sin(theta);
rho3 = points(3,1)*cos(theta) + points(3,2)*sin(theta);

% Cari titik perpotongan dengan mencari varian minimum nilai rho
[~, idx] = min(var([rho1; rho2; rho3]));

% Define the points
points = [10, 10; 20, 20; 30, 30];

% Gunakan nilai rho dan theta pada titik persimpangan
rho_intersection = rho1(idx);
theta_intersection = theta(idx);

% Kalkulasikan m dan c
m = -cos(theta_intersection) / sin(theta_intersection);
c = rho_intersection / sin(theta_intersection);

fprintf('The slope m is: %f\n', m);
fprintf('The y-intercept c is: %f\n', c);
```

```
Slope m adalah: 1.003945
y-intercept c adalah: -0.039446
```

2. Implementation

2.1 Convolution

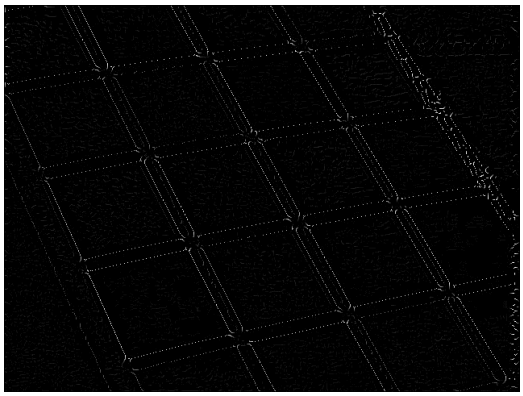
Input yang gunakan adalah img01.jpg dengan parameter h berupa prewitt filter pada gradient sumbu x dengan nilai $\begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix}$. Berikut merupakan hasil dari konvolusi saya dibandingkan dengan original image menggunakan fungsi myImageFilter dengan scharr filter:



Gambar 5 Perbandingan Original Image dan Filtered Image dengan Konvolusi Menggunakan Prewitt Filter pada fungsi myImageFilter

2.2 Edge Detection

Berikut merupakan hasil *edge detection* menggunakan sobel filter dan sigma = 2:

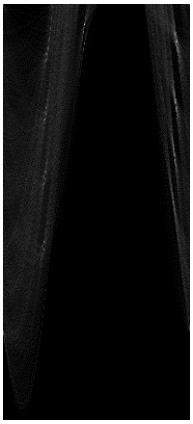


Gambar 6 *Output Edge Detection* Dengan Menggunakan Fungsi myEdgeFilter

2.3 Hough Transform

Berikut merupakan hasil hough *transform* dengan menggunakan:

- threshold = 0.03
- rhoRes = 2
- thetaRes = pi/90

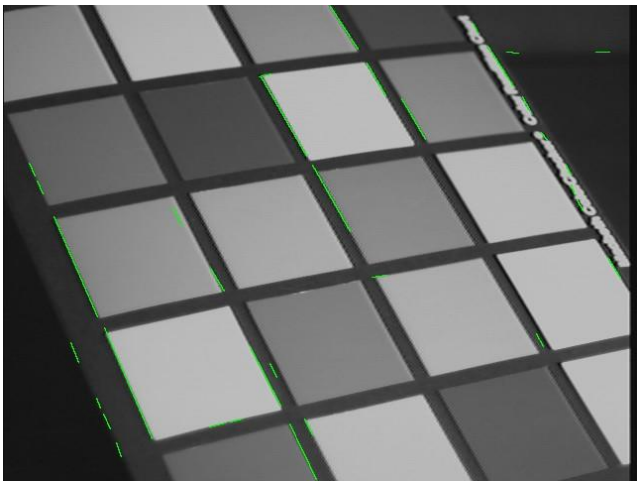


Gambar 7 *Output Hough Transform* dengan Menggunakan Fungsi myHoughTransform

2.4 Finding Lines & 2.5 Fitting Line Segments for Visualization

Berikut merupakan hasil *finding lines* dan *fitting lines for visualization* dengan *parameters*:

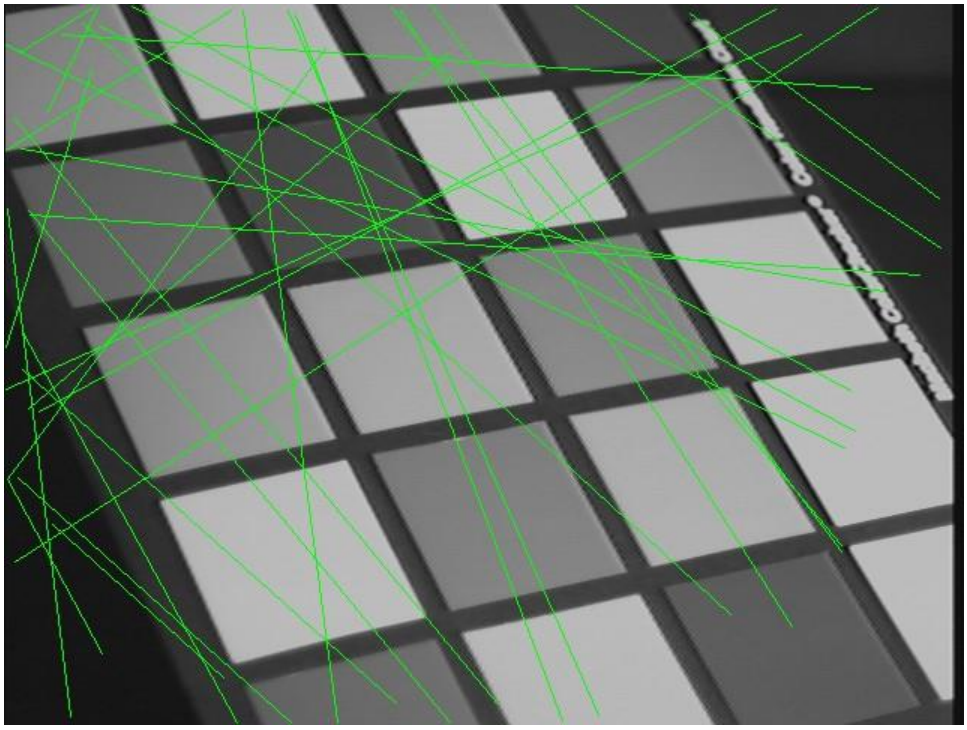
- sigma: 2
- threshold: 0.03
- rhoRes: 2
- thetaRes: pi/90
- nLines : 50



Gambar 8 *Output houghScript* yang Mengkombinasikan Penggunaan Fungsi myEdgeFilter, myHoughTransform, dan myHoughLines

2.5.x Implement Houglines Yourself

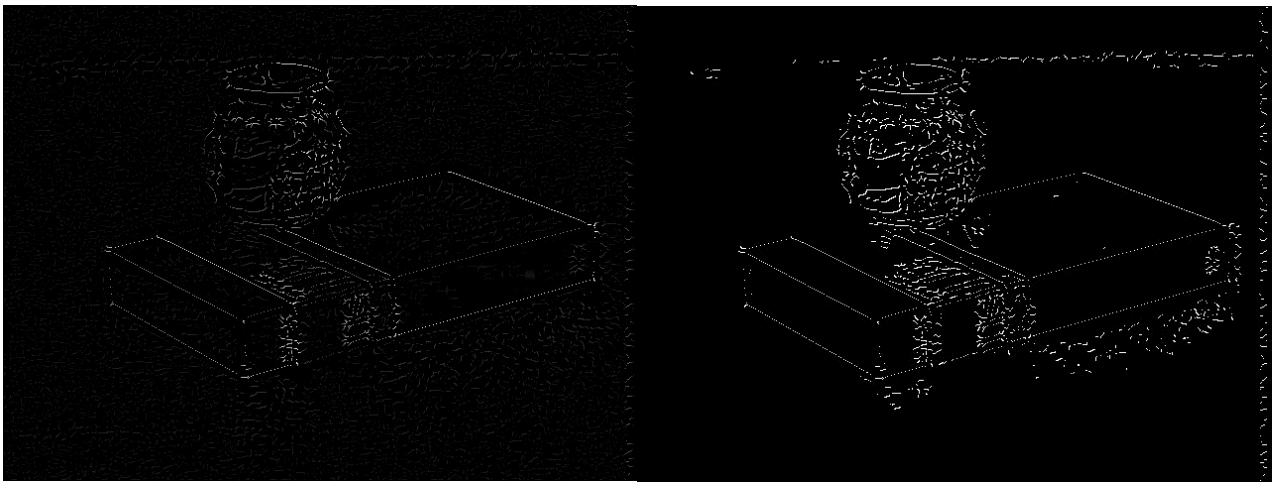
Hasil implementasi houghlines dengan fungsi myHoughLineSegments jauh dari kata berhasil. Perbedaan parameter default yang digunakan matlab dengan fungsi saya mempengaruhi bagaimana hasil dari houghlines. Dengan tuning parameter lebih lanjut, akan ada kemungkinan dapat menyamai fungsi dari houghlines default MATLAB.



Gambar 9 Implementasi Myhoughlinessegment Menggunakan Ec Berdasarkan Modifikasi Houghscript

3. **Experiments**

- Dalam eksprimen ini menggunakan img02.jpg dengan parameter tertentu, sigma: 2, threshold: 0.03, rhoRes: 2, thetaRes: pi/90, nLines: 50

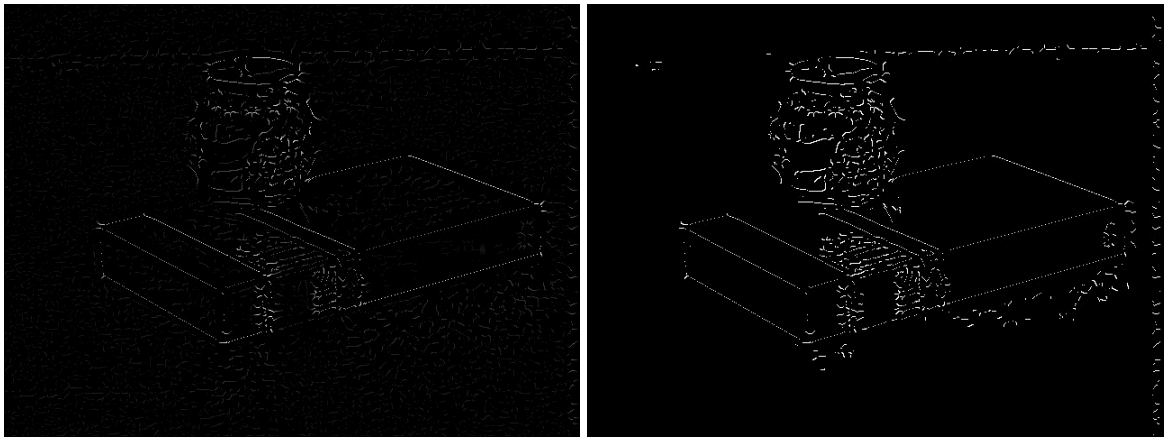


Gambar 10 *Edge Detection* (Kiri) dan A*pplying Threshold Result* (Kanan)

Gambar 11 Hasil *Hough Transform* (Kiri) dan Hasil *Find Lines* (Kanan)

Hasil dari hough lines kurang baik dikarenakan, noise pada gambar masih terdeteksi dan beberapa lines yang penting masih belum terdeteksi dengan baik.

- Mengubah sigma menjadi 2.5, eksperimen ini menggunakan parameter tertentu, sigma: 2.5, threshold: 0.5, rhoRes: 2, thetaRes: pi/90, nLines: 50

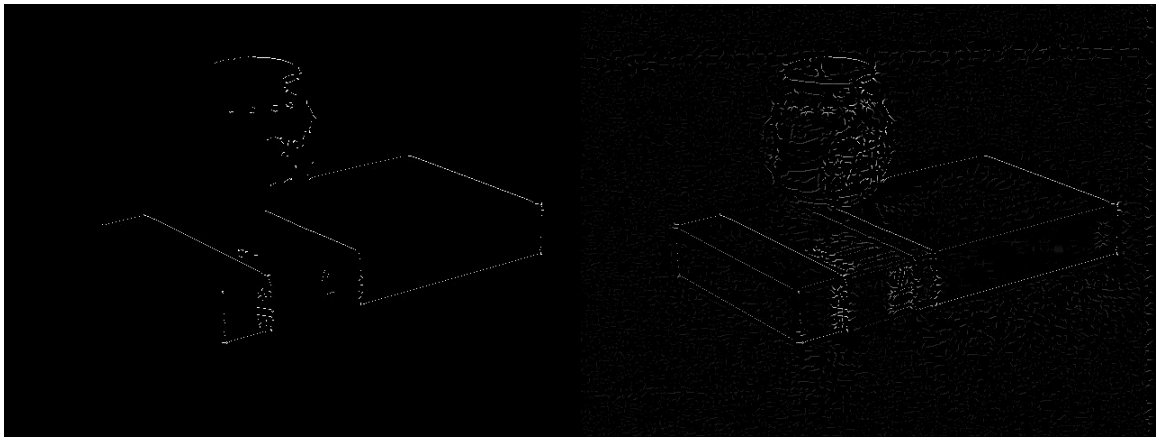


Gambar 12 *Edge Detection* (Kiri) dan A*pplying Threshold Result* (Kanan)

Gambar 13 Hasil *Hough Transform* (Kiri) dan Hasil *Find Lines* (Kanan)

Dengan mengaplikasikan sigma yang lebih tinggi, noise dari lines berkurang dikarenakan effect smoothing yang dilakukan oleh sigma, sehingga noise yang dideteksi berkurang cukup drastis.

- Mengubah threshold menjadi 0.5, eksperimen ini menggunakan parameter tertentu, sigma: 2, threshold: 0.5, rhoRes: 2, thetaRes: pi/90, nLines: 50

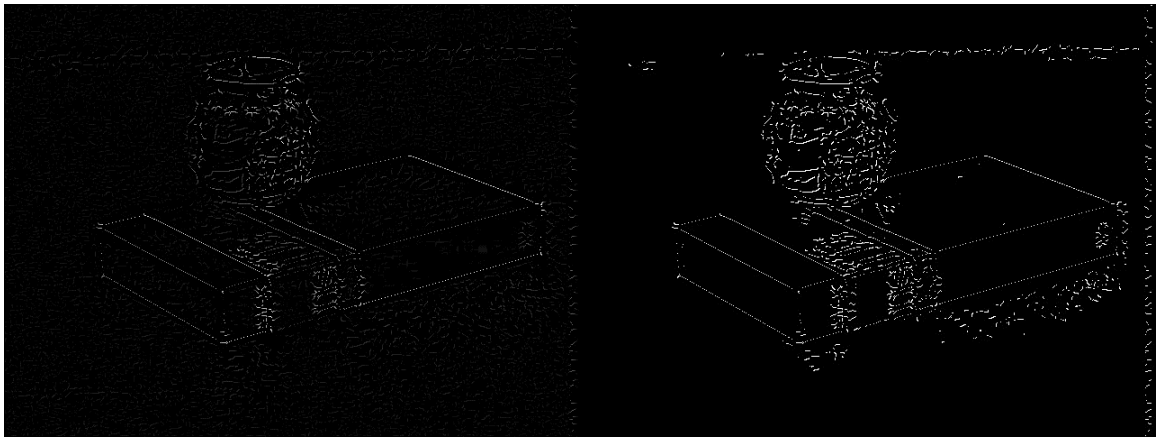


Gambar 14 *Edge Detection* (Kiri) dan A*pplying Threshold Result* (Kanan)

Gambar 15 Hasil *Hough Transform* (Kiri) dan Hasil *Find Lines* (Kanan)

Dengan mengaplikasikan threshold yang lebih tinggi, noise dari lines berkurang, namun belum semua side utama yang terdeteksi lines pada gambar.

- Mengubah *rho resolution* menjadi 0.75, eksperimen ini menggunakan parameter tertentu, sigma: 2, threshold: 0.5, rhoRes: 0.75, thetaRes: pi/90, nLines: 50

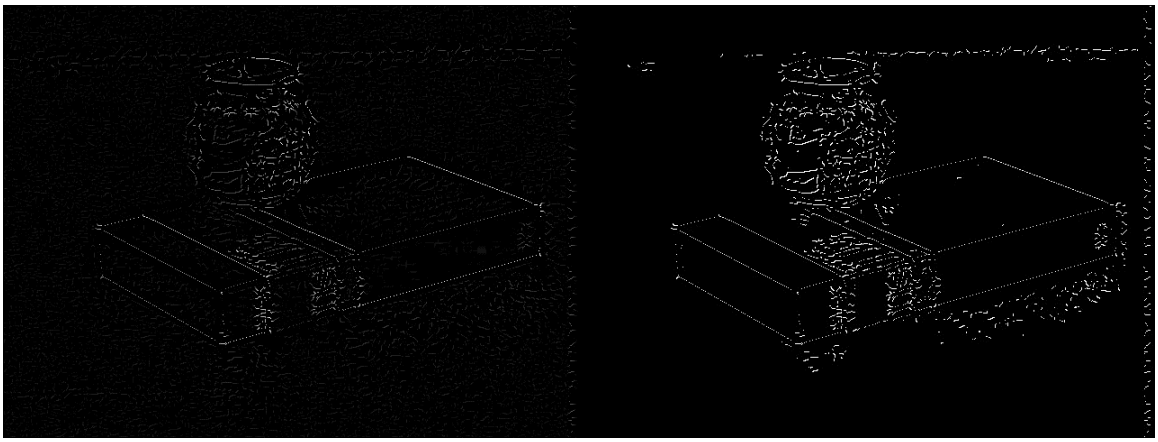


Gambar 16 *Edge Detection* (Kiri) dan A*pplying Threshold Result* (Kanan)

Gambar 17 Hasil *Hough Transform* (Kiri) dan Hasil *Find Lines* (Kanan)

Dengan mengaplikasikan rho resolution yang lebih tinggi, lines menjadi lebih clear namun semakin terputus-putus.

- Mengubah theta resolution menjadi pi/180, eksperimen ini menggunakan parameter tertentu, sigma: 2, threshold: 0.5, rhoRes: 2, thetaRes: pi/180, nLines: 50

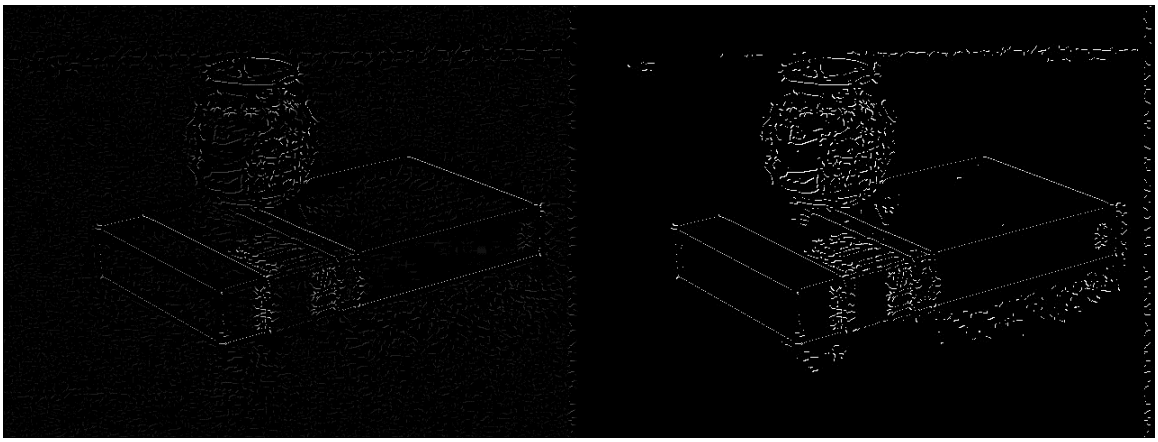


Gambar 18 *Edge Detection* (Kiri) dan A*pplying Threshold Result* (Kanan)

Gambar 19 Hasil *Hough Transform* (Kiri) dan Hasil *Find Lines* (Kanan)

Dengan mengaplikasikan theta resolution yang lebih tinggi, ruang deteksi hough lines mnejadi lebih tinggi, dan garis terputus-putus sedikit berkurang.

- Mengubah nLines menjadi 150, eksperimen ini menggunakan parameter tertentu, sigma: 2, threshold: 0.5, rhoRes: 2, thetaRes: pi/90, nLines: 150

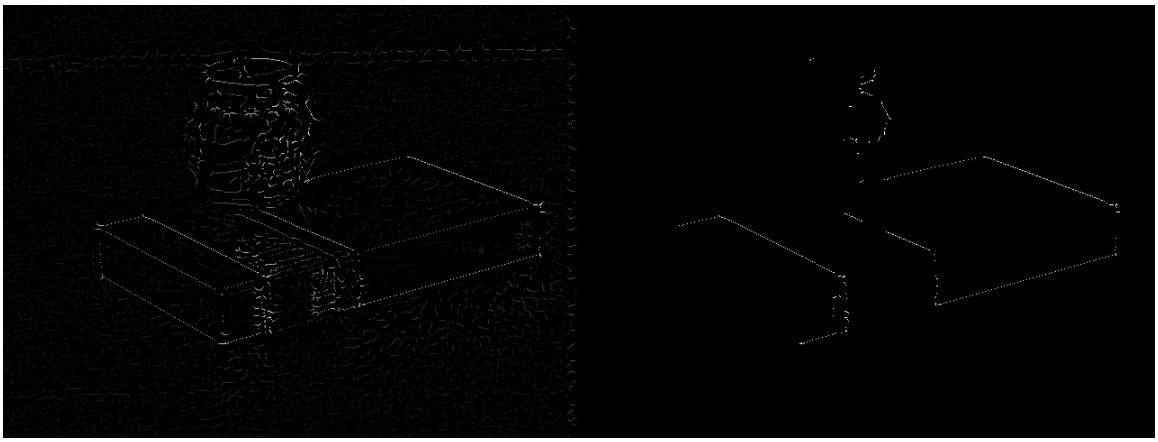


Gambar 20 *Edge Detection* (Kiri) dan A*pplying Threshold Result* (Kanan)

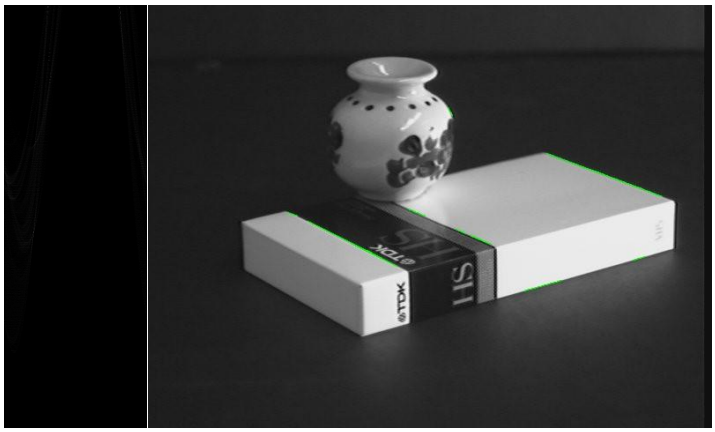
Gambar 21 Hasil *Hough Transform* (Kiri) dan Hasil *Find Lines* (Kanan)

Dengan mengaplikasikan nLines lebih tingg, jumlah garis yang dihasilkan oleh houghlines meningkat.

- Mengaplikasikan seluruh parameter *tuning*, eksperimen ini menggunakan parameter tertentu, sigma: 2.5, threshold: 0.5, rhoRes: 0.75, thetaRes: pi/180, nLines: 150.



Gambar 22 *Edge Detection* (Kiri) dan A*pplying Threshold Result* (Kanan)



Gambar 23 Hasil *Hough Transform* (Kiri) dan Hasil *Find Lines* (Kanan)

Dengan mengaplikasikan parameter tuning, algoritma yang mendeteksi houghlines berjalan cukup baik dengan beberapa sides utama berhasil dideteksi oleh houghlines, dengan sedikit noise yang terdeteksi. Garis putus-putus juga tidak banyak terdeteksi. Namun secara keseluruhan hasil masih bisa ditingkatkan lagi dengan melakukan proses bluring. Dikarenakan penggunaan parameter fungsi hough transform yang terbatas maka hasil dari deteksi houghlines hanya berjalan cukup baik, untuk mampu menghasilkan pendeteksi lines yang lebih baik ada baiknya menggunakan lebih banyak parameter. Bukan hanya itu terdeteksinya banyak noise pada gambar sebelum parameter tuning mengindikasikan bahwa perlunya tahap pre-processing gambar yang lebih lanjut sehingga hasil dari deteksi hough lines diharapkan lebih baik.

4. Try your own images

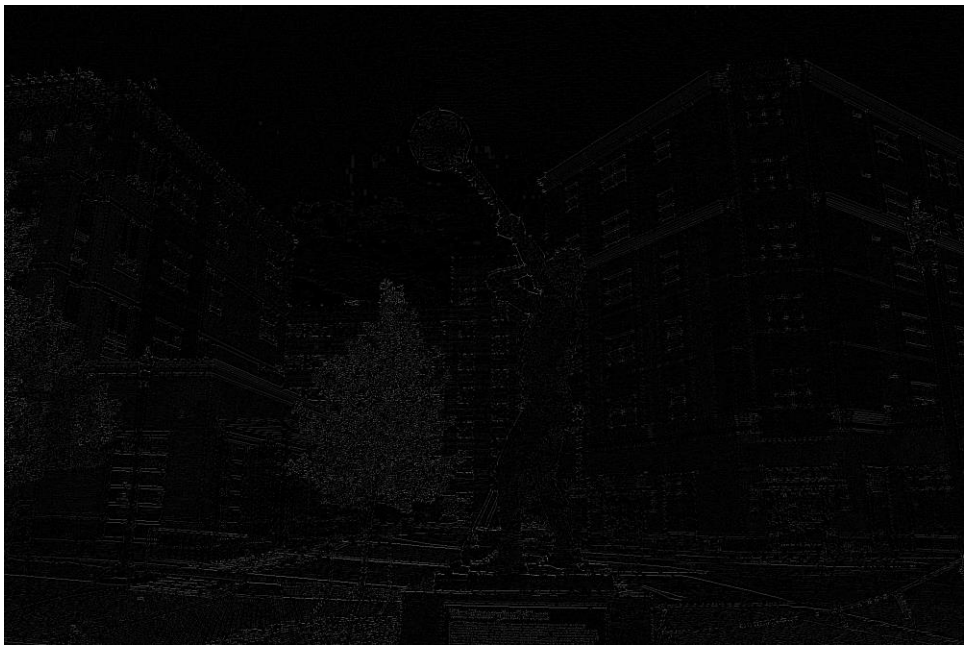
Hasil dibawah merupakan salah satu hasil dari lima gambar yang saya dapatkan dari internet untuk dilakukan beberapa langkah find lines menggunakan hough transform. Hasil menggunakan parameter tuning pada bagian 3 (Eksperimen)

Gambar 24 *Edge Detection* (Kiri) dan A*pplying Threshold Result* (Kanan)

Gambar 25 Hasil *Hough Transform* (Kiri) dan Hasil *Find Lines* (Kanan)

Hasil menggunakan parameter sebelum parameter *tuning* dengan parameter sebagai berikut:

- sigma: 2
- threshold: 0.03
- rhoRes: 2
- thetaRes: pi/90
- nLines: 50



Gambar 26 *Edge Detection*

Gambar 27 A*pplying Threshold Result*

Gambar 28 Hasil *Hough Transform* (Kiri) dan Hasil *Find Lines* (Kanan)

Berdasarkan gambar 25 dan 26, parameter tuning pada bagian eksperimen hanya berjalan baik pada gambar tertentu, yakni gambar img.02. Ketika diaplikasikan dengan gambar lainnya ternyata hasil find lines menggunakan houghlines hamper tidak terdeksi. Gambar 26, terlihat bahwa *edge detection* mendeteksi terlalu banyak noise. Meskipun telah menggunakan teknik *non-maximum suppression* pada proses edge detection, banyak noise yang masih teridentifikasi sebagai edge. Hal yang sama ditujukan pada gambar 27, dimana noise dari threshold terlihat sangat banyak. Sementara itu, Gambar 28 pada hasil *find lines* menunjukkan banyak garis yang luput dari deteksi oleh houghlines. Hanya beberapa garis utama yang berhasil dideteksi, sementara mayoritasnya tidak. Walaupun telah dilakukan transformasi dengan *hough transform*, hasil integrasinya kurang optimal dan banyak ketidaksesuaian yang terjadi.