LAPORAN TUGAS BESAR

IF-604 PENGOLAHAN CITRA

DETEKSI PLAT NOMOR KENDARAAN MENGGUNAKAN ALGORITMA CANNY EDGE DETECTION DENGAN GAUSSIAN BLUR FILTER

INSTITUT TEKNOLOGI HARAPAN BANGSA

Veritas vos liberabit

Disusun oleh:

1120007 - Jonathan Farrell Lewi

1121024 - Jonathan Hadiwijaya Ang

1121028 - David Kharis Elio M

1121029 - Stefanus Titan

1121030 - Juan Vincent Nugrahaputra

PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2023

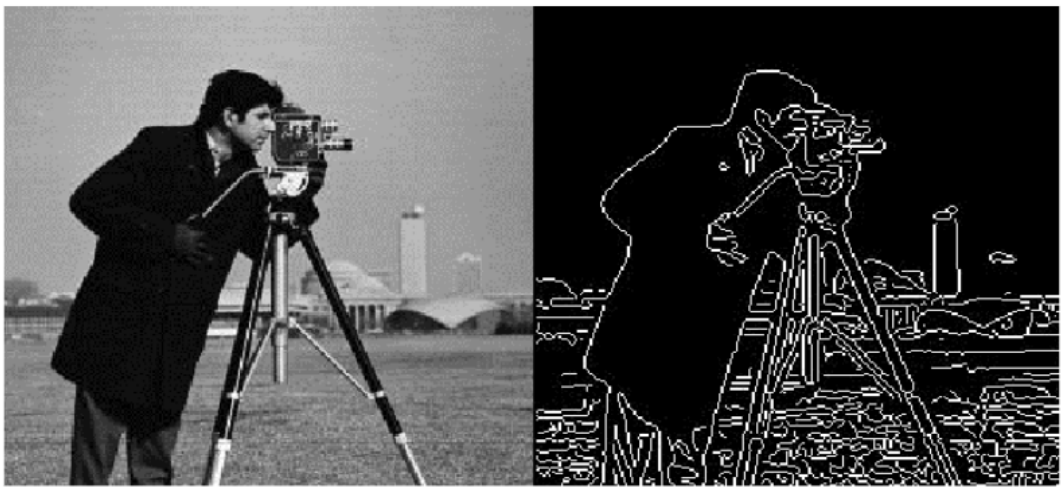
A. Abstrak

Deteksi plat nomor kendaraan adalah salah satu aplikasi dari pengolahan citra yang digunakan untuk mengidentifikasi plat nomor suatu kendaraan dalam citra. Ada beberapa bidang yang bisa menggunakan teknik ini, seperti penegakan hukum, pariwisata, marketing, dan lain lain. Banyak metode yang diusulkan untuk mengimplementasikan teknik tersebut. Penelitian ini mengusulkan metode deteksi tepi untuk memungkinkan Sistem Pengenalan Plat melalui situasi praktis, seperti berbagai kondisi lingkungan atau meteorologi. Alat pemrosesan gambar digunakan untuk memproses area plat nomor kendaraan, mengubah ukurannya, dan mengubahnya menjadi grayscale sebelum memfilter gambar untuk menghilangkan objek-objek kecil.

B. Metode

1. *Canny Edge Detection Algorithm*

Canny Edge Detection Algorithm adalah salah satu algoritma untuk mendeteksi tepi yang paling optimal. Kriteria yang terdapat dalam Canny Edge Detection Algorithm adalah: *error rate* yang lebih rendah dibandingkan dengan algoritma pendeteksi tepi lainnya, selisih jarak tepi sebenarnya dengan tepi yang dideteksi sekecil mungkin, dan hanya ada satu respon untuk satu tepi[3]. Berdasarkan kriteria tersebut cara kerja dari *Canny Edge Detection Algorithm* adalah dengan langkah pertama yaitu *smoothing* citra untuk menghilangkan *noise* lalu mencari gradasi citra untuk *highlight* area yang memiliki derivatif spasial yang tinggi, algoritma kemudian jalan di sepanjang ini wilayah ini dan *suppress* piksel apapun yang tidak maksimum. *Array* gradien sekarang dikurangi lebih lanjut oleh histeresis. Histeresis digunakan untuk jalan di semua piksel yang tersisa yang belum *suppressed*. Histeresis menggunakan dua ambang batas dan jika besarnya di bawah ambang batas pertama, maka akan disetel ke nol (tidak membuat tepi). Jika besarnya di atas ambang batas tinggi, maka akan dijadikan tepi. Dan jika besarnya berada di antara 2 ambang batas, maka ditetapkan ke nol kecuali ada jalur dari piksel ini ke piksel dengan gradien di atas T2.



Citra sebelum dan sesudah proses Canny Edge Detection

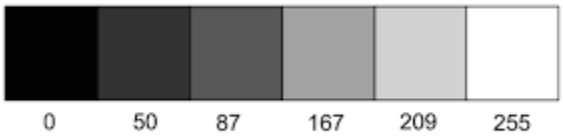
2. *Gaussian Blur*

Image smoothing adalah teknik yang menggunakan operasi konvolusi antara citra dan filter kernel. Teknik ini mempertajam gambar, memungkinkan kita melihat detail objek dengan mengurangi noise gambar. Salah satu teknik untuk menghaluskan gambar adalah pengaburan Gaussian, yang mengambil rata-rata nilai bobot dari setiap piksel tetangga.

Citra sebelum dan sesudah proses Gaussian Blur

3. *Grayscale Color*

Citra grayscale adalah citra yang hanya memiliki satu tingkat keabuan. Warna abu-abu dari citra grayscale adalah warna kanal R (merah), G (hijau), dan B (biru), yang memiliki intensitas yang sama. Warna pewarnaan ini berkisar dari hitam (intensitas rendah) hingga putih (intensitas tinggi) dan karenanya memiliki variasi yang luas.



Contoh grayscale pixel value

C. Implementasi

1. Input Gambar

```
In [2]: img = cv2.imread("1_Shaun-Elliot-from-Hexham-with-the-two-cars-he-won-in-online-draws2.png")

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis("off")
plt.title("Original Image")
plt.show()
```

Kami menggunakan library openCV yaitu cv2.imread untuk melakukan read input gambar di komputer kita. Karena cv2 itu langsung mengubah image menjadi format

BGR maka kita langsung mengubah gambarnya ke RGB lagi dengan menggunakan fungsi cv2.COLOR_BGR2RGB.

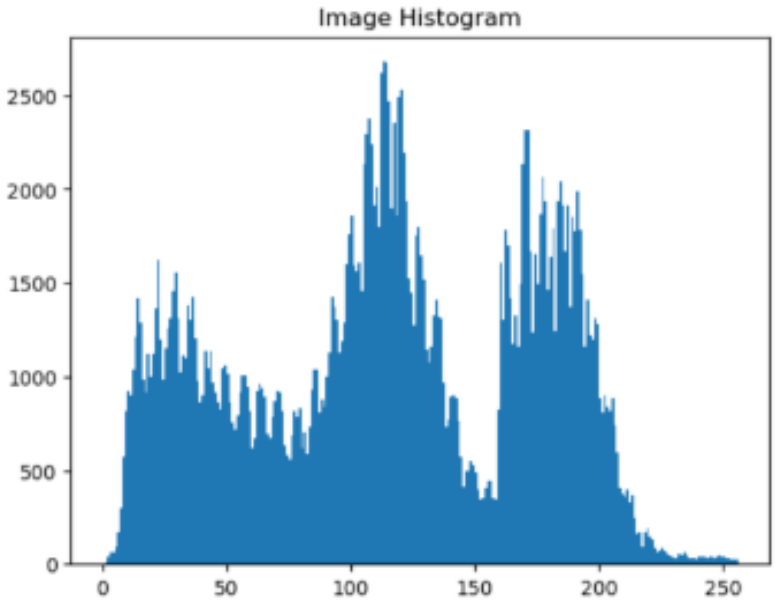
2. Image Preprocessing

In [3]:
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gaus = cv2.GaussianBlur(gray, (5, 5), 0)

plt.imshow(cv2.cvtColor(gaus, cv2.COLOR_BGR2RGB))
plt.axis("off")
plt.title("Preprocessed Image Gaussian Blur")
plt.show()
```

Disini kita mengubah citra ke mode grayscale lalu menggunakan smoothing berupa gaussian blur.

In [4]:
```
plt.hist(gray.ravel(),256,[0,256])
plt.title('Image Histogram')
plt.show()
```

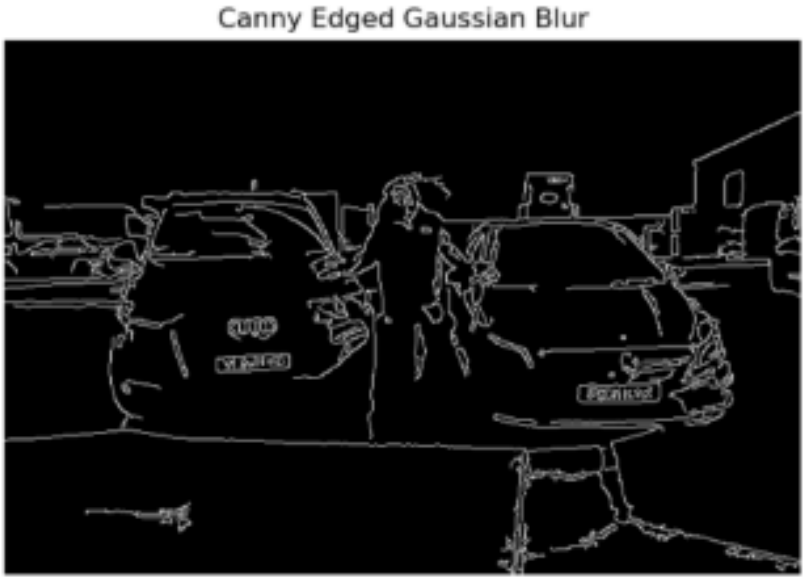


Ini adalah value dari grayscale yang kita bisa lihat dalam bentuk histogram.

3. Canny Edge Gaussian Blur

```
In [5]: canny_gaus = cv2.Canny(gaus, 30, 200)

        plt.imshow(cv2.cvtColor(canny_gaus, cv2.COLOR_BGR2RGB))
        plt.axis("off")
        plt.title("Canny Edged Gaussian Blur")
        plt.show()
```



Pada langkah ini kita menerapkan algoritma *canny edge detection* menggunakan library openCV dengan fungsi .Canny(), parameter pertama merupakan input citra yang akan kita gunakan, parameter kedua adalah minVal untuk hysteresis threshold, dan parameter ketiga adalah maxVal untuk hysteresis threshold.

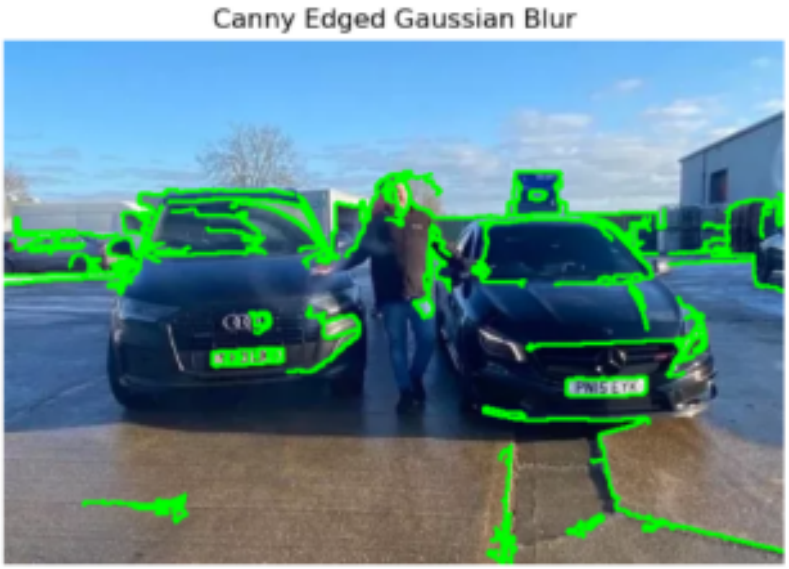
In [6]:
```
img1 = img.copy()
cnt_canny_gaus, new = cv2.findContours(canny_gaus.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(img1, cnt_canny_gaus, -1, (0, 255, 0), 2)

plt.imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB))
plt.axis("off")
plt.title("Canny Edged Gaussian Blur")
plt.show()
```

Pada langkah ini kita mencari kontur / kurva yang membentuk suatu objek yang memiliki intensitas yang sama menggunakan fungsi findContours(), parameter pertama adalah citra yang digunakan dalam kasus ini citra yang sudah dilakukan operasi *canny edge detection*, parameter kedua adalah mode pengambilan kontur dalam kasus ini RETR_LIST yaitu mode yang sederhana semua kontur memiliki hirarki / level yang sama, sedangkan parameter ketiga adalah metode aproksimasi kontur yaitu CHAIN_APPROX_SIMPLE berfungsi untuk *compress* jumlah kontur. Lalu untuk menampilkan hasilnya adalah dengan fungsi drawContours(), parameter pertama yaitu target citra, parameter kedua adalah kontur yang didapatkan, parameter ketiga adalah indeks kontur untuk ditampilkan (-1 untuk semua kontur), lalu parameter keempat adalah warna, dan terakhir adalah ketebalan.

In [7]:
```
img1_sorted = img.copy()
cnt_canny_gaus_sorted = sorted(cnt_canny_gaus, key = cv2.contourArea, reverse = True)[:30]
cv2.drawContours(img1_sorted, cnt_canny_gaus_sorted, -1, (0, 255, 0), 2)

plt.imshow(cv2.cvtColor(img1_sorted, cv2.COLOR_BGR2RGB))
plt.axis("off")
plt.title("Canny Edged Gaussian Blur")
plt.show()
```



Langkah terakhir adalah untuk sorting kontur yang ada berdasarkan area kontur yang diperoleh dengan objek contourArea, sorting descending oleh karena parameter ketiga yaitu reverse yang disetel menjadi true kemudian sliced sehingga hanya 30 elemen pertama yang tersisa, elemen lainnya dibuang.

4. Result

In [8]:
```
img1_loop_contour = img.copy()

screenCntx = []

for c in cnt_canny_gaus_sorted:
    contour_perimeter = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.018 * contour_perimeter, True)

    if len(approx) == 4:
        screenCntx.append(approx)


cv2.drawContours(img1_loop_contour, screenCntx, -1, (0, 255, 0), 3)

plt.imshow(cv2.cvtColor(img1_loop_contour, cv2.COLOR_BGR2RGB))
plt.axis("off")
plt.title("Canny Edged Gaussian Blur")
plt.show()
```

Di step ini dilakukan looping untuk mencari contour yang bergaris 4 atau berbentuk persegi, jika ada dua plat nomor yang terdeteksi maka akan terdeteksi dua duanya seperti gambar diatas.

D. Analisis Pengujian

1. Satu plat nomor Gaussian kernel 3x3

2. Satu plat nomor Gaussian kernel 5x5

3. Satu plat nomor Gaussian kernel 7x7

4. Dua plat nomor Gaussian kernel 3x3

5. Dua plat nomor Gaussian kernel 5x5

6. Dua plat nomor Gaussian kernel 7x7

Berdasarkan analisis pengujian yang telah dilakukan, Gaussian kernel yang lebih kecil dari 5x5 memiliki terlalu banyak noise sehingga plat tidak terdeteksi dengan optimal, dan Gaussian kernel yang lebih besar dari 5x5 menjadi terlalu blur sehingga plat juga tidak terdeteksi dengan optimal. Maka ukuran kernel terbaik adalah 5x5 karena tidak terlalu banyak noise dan juga tidak banyak blur.

E. Referensi

[1] Mousa, A. (2012). Canny edge-detection based vehicle plate recognition. International Journal of signal processing, Image processing and pattern recognition, 5(3), 1-8.

[2] Maini, Raman, and Himanshu Aggarwal. "Study and comparison of various image edge detection techniques." International journal of image processing (IJIP) 3.1 (2009): 1-11.

[3] J. F. Canny. “A computational approach to edge detection”. IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, no. 6, pp. 679-697, 1986