Jurnal Algoritma
E-ISSN: 2302-7339 | P-ISSN: 1412-3622
INSTITUT TEKNOLOGI GARUT

https://jurnal.itg.ac.id/index.php/algoritma
DOI: 10.33364/algoritma/v.1-1.xxxx

Deteksi jumlah jari menggunakan Mediapipe dan Opencv dengan perhitungan akurasi

Rivan Febrian[1], Ihsan Hafizh Kurniawan[2]
[1,2]Institut Teknologi Garut, Indonesia

[1]2206111@itg.ac.id
[2]2206130@itg.ac.id

1.PENDAHULUAN

1.1 Latar belakang

Pendeteksian jari merupakan salah satu bidang dalam pemrosesan citra dan visi komputer yang banyak diterapkan dalam berbagai aplikasi, seperti pengenalan gestur, interaksi manusia-komputer, serta autentikasi biometrik. Salah satu teknologi yang banyak digunakan untuk tugas ini adalah MediaPipe, sebuah pustaka open-source yang dikembangkan oleh Google. MediaPipe menyediakan solusi berbasis machine learning untuk pelacakan tangan secara real-time dengan akurasi tinggi dan efisiensi yang optimal, bahkan pada perangkat dengan sumber daya terbatas. Dengan menggunakan model deep learning yang telah dioptimalkan, MediaPipe Hand Tracking mampu mendeteksi dan melacak posisi jari serta titik-titik kunci (landmarks) tangan dalam citra atau video. Teknologi ini mempermudah pengembangan aplikasi berbasis gestur tanpa memerlukan perangkat keras khusus, sehingga semakin banyak digunakan dalam berbagai inovasi di bidang augmented reality (AR), gaming, dan sistem kontrol berbasis gerakan.

Pengolahan citra digital yaitu bidang ilmu yang mempelajari mengenai bagaimana suatu citra itu dibentuk, diolah dan dianalisis sehingga dapat menghasilkan sebuah informasi yang bisa dipahami oleh manusia.Dapat pula di jelasakan bahwa pengolahan citra digital adalah manipulasi dan interprestasi digital dari citra dengan bantuan komputer. Citra sendiri merupakan fungsi dari intensitas cahaya yang direpresentasikan ke dalam bidang 2 dimensi.(Adolph, 2016)

Pengolahan citra digital adalah sebuah teknologi visual yang dipakai untuk mengamati dan menganalisis sebuah objek tanpa berhubungan secara langsung dengan objek yang diamati itu. Teknologi ini bisa dipakai untuk mengevaluasi mutu suatu produk tanpa merusak produk itu sendiri.

MediaPipe adalah Kerangka Kerja untuk membangun alur kerja pembelajaran mesin untuk memproses data deret waktu seperti video, audio, dll. Kerangka Kerja lintas platform ini berfungsi pada Desktop/Server, Android, iOS, dan perangkat tertanam seperti Raspberry Pi dan Jetson Nano(Daniel Tanugraha et al., 2022).

Deteksi tepi Canny adalah teknik yang digunakan dalam pemrosesan gambar untuk menemukan tepi pada gambar. Dinamai sesuai dengan penciptanya, John Canny, yang mengembangkan teknik ini pada tahun 1986. Algoritma deteksi tepi Canny digunakan untuk menemukan tepi pada gambar dengan mencari maxima lokal pada turunan pertama gambar(Supriyatin, 2020)

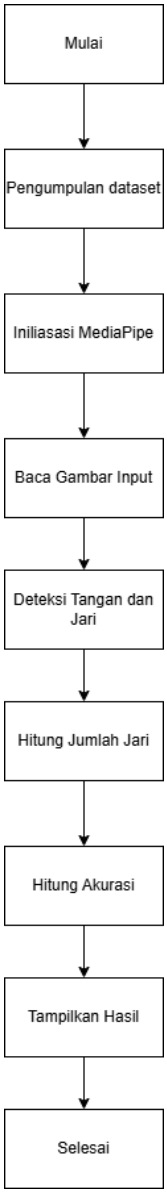
1.2 Rumusan Masalah

1. Bagaimana cara mengintegrasikan library Mediapipe dan algoritma deteksi tepi Canny untuk mendeteksi angka yang ditunjukkan oleh jari secara akurat?
2. Bagaimana sistem ini dapat diimplementasikan secara real-time dengan mempertimbangkan berbagai kondisi lingkungan?

1.3 Tujuan

1. Mengembangkan sistem real-time untuk mendeteksi angka pada jari menggunakan Mediapipe dan algoritma deteksi tepi Canny.
2. Mengevaluasi akurasi dan keandalan sistem dalam berbagai kondisi lingkungan.

2. METODE PENELITIAN

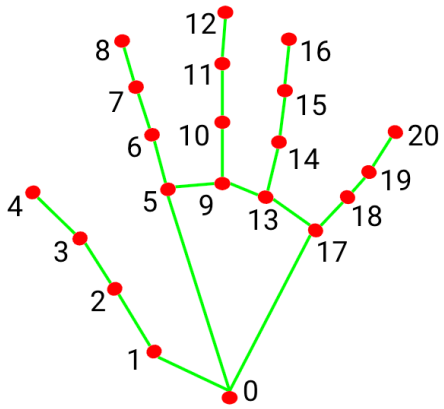


2.1 MediaPipe

Pengenalan gerakan tangan dengan MediaPipe adalah suatu teknik untuk mengubah input video dengan mempertahankan informasi penting tentang gerakan tangan, sementara gaya atau tekstur visual dari video tersebut diubah (Beno et al., 2022). Informasi penting tersebut terkait dengan pengenalan gestur tangan, baik oleh manusia maupun mesin, dan dapat dimanfaatkan dalam berbagai aplikasi seperti interaksi manusia dan komputer.

a. pengumpulan data

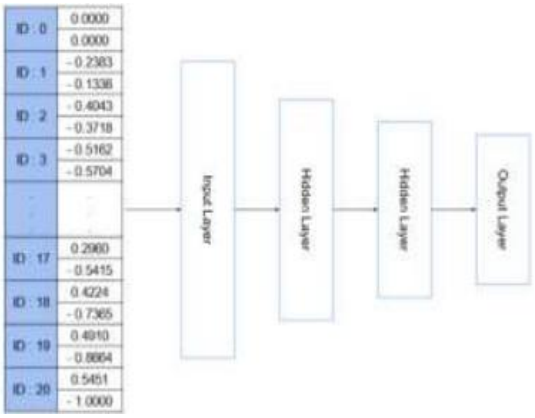
Pemilihan dataset merupakan langkah awal yang penting dalam penelitian ini. Dataset yang digunakan merupakan hasil rekaman gesture tangan dengan menggunakan MediaPipe. MediaPipe menyediakan model untuk mendeteksi landmark tangan, dimana model ini mendeteksi posisi tangan dengan 21 keypoint. Keypoint inilah yang disimpan sebagai dataset untuk melatih model agar bisa mendeteksi gesture tangan apa yang sedang dibuat. Ilustrasi keypoint dalam mediapipe ditunjukan pada Gambar



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Proses pengumpulan data melibatkan beberapa variasi gesture. Setiap gesture disimpan keypoint-nya, masing-masing gestur direkam untuk tangan kanan dan tangan kiri. Gestur yang dipilih ditunjukan pada Gambar

b. struktur model



Dataset yang digunakan untuk training dan menguji model merupakan data keypoint dari gestur yang didapatkan dari MediaPipe yang disimpan dalam csv saat pengumpulan data. Data train yang digunakan sebanyak 75% dan data test sebanyak 25% dari dataset yang ada. Proses pelatihan model dimulai dengan membangun arsitektur jaringan saraf menggunakan TensorFlow Keras. Model dibuat dalam bentuk sekuensial

dengan beberapa lapisan untuk memproses input data. Lapisan pertama menerima vektor dengan dimensi 21×2, yang merupakan data koordinat x dan y dari 21 keypointtangan. Untuk mencegah overfitting, dua lapisan dropout diterapkan dengan probabilitas masing-masing 0,2 dan 0,4. Model juga memiliki dua lapisan dense (fully connected) dengan masing-masing 20 dan 10 neuron, menggunakan fungsi aktivasi ReLU untuk menangkap pola non-linear. Lapisan keluaran menggunakan fungsi aktivasi softmax dengan jumlah neuron yang sesuai dengan jumlah kelas yaitu 4, untuk mengklasifikasikan gestur tangan berdasarkan probabilitas. Model dikompilasi menggunakan Adam optimizer, yang merupakan algoritma pembaruan bobot yang efisien, serta fungsi loss sparse categorical crossentropy, karena data label diberikan dalam format integer. Metrik accuracy dipilih untuk memantau performa model selama pelatihan. Proses pelatihan dilakukan hingga maksimal 1000 epoch, dengan dua callback yang digunakan dalam proses ini adalah ModelCheckpoint untuk menyimpan bobot model terbaik secara otomatis setelah setiap epoch, dan EarlyStopping dengan toleransi 20 epoch untuk menghentikan pelatihan jika performa validasi tidak meningkat, sehingga menghindari overfitting. Dengan konfigurasi ini, model dapat dilatih secara optimal untuk mengenali gestur tangan berdasarkan dataset yang tersedia.

2.2 Deteksi Canny

Operator Canny, yang dikemukakan oleh John Canny pada tahun 1986, terkenal sebagai operator deteksi tepi yang optimal. Algoritma ini memberikan tingkat kesalahan yang rendah, melokalisasi titik-titik tepi (jarak piksel-piksel tepi yang ditemukan deteksi dan tepi yang sesungguhnya sangat pendek), dan hanya memberikan satu tanggapan untuk satu tepi. berdasarkan (Tsani & Harliana, 2019), terdapat enam langkah yang dilakukan untuk mengimplementasikan deteksi tepi Canny. Keenam langkah tersebut dijabarkan berikut ini :
Pertama, dilakukan penipisan terhadap citra dengan tujuan untuk menghilangkan derau. Hal ini dapat dilakukan dengan menggunakan filter Gaussian dengan cadar sederhana. Cadar yang digunakan berukuran jauh lebih kecil daripada ukuran citra. Contoh ditunjukkan pada Gambar

$1/115$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Setelah penghalusan gambar terhadap derau dilakukan, dilakukan proses untuk mendapatkan kekuatan tepi (edge strength). Hal ini dilakukan dengan menggunakan operator Gaussian. Selanjutnya, gradien citra dapat dihitung melalui rumus :

$$|G| = |G_x| + |G_y|$$

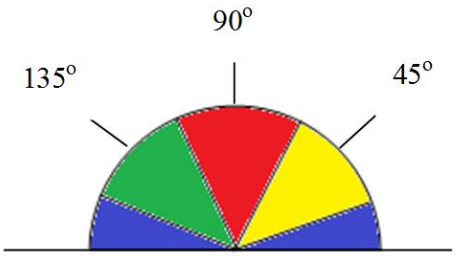
Langkah ketiga berupa penghitungan arah tepi. Rumus yang digunakan untuk keperluan ini :

$$theta = tan^{-1}(G_y, G_x)$$

Setelah arah tepi diperoleh, perlu menghubungkan antara arah tepi dengan sebuah arah yang dapat dilacak dari citra. Sebagai contoh, terdapat susunan piksel berukuran 5 x 5 seperti terlihat pada Gambar 2. Dengan melihat piksel “a” tampak bahwa a hanya memiliki 4 arah berupa 0 derajat, 45 derajat, 90 derajat, dan 135 derajat.

X	X	X	X	X
X	X	X	X	X
X	X	**a**	X	X
X	X	X	X	X
X	X	X	X	X

Selanjutnya, arah tepi yang diperoleh akan dimasukkan ke dalam salah satu kategori dari keempat arah tadi berdasarkan area yang tertera pada Gambar 3. Berikut adalah aturan konversi yang berlaku :

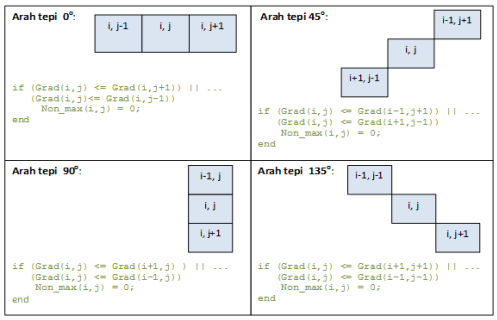


- Semua arah tepi yang berkisar antara 0 dan 22,5 serta 157,5 dan 180 derajat (warna biru) diubah menjadi 0 derajat.
- Semua arah tepi yang berkisar antara 22,5 dan 67,5 derajat (warna kuning) diubah menjadi 45 derajat.
- Semua arah tepi yang berkisar antara 67,5 dan 112,5 derajat (warna merah) diubah menjadi 90 derajat.
- Semua arah tepi yang berkisar antara 112,5 dan 157,5 derajat (warna hijau) diubah menjadi 135 derajat.

Setelah arah tepi diperoleh, penghilangan non-maksimum dilaksanakan dilaksanakan. Penghilangan non-maksimum dilakukan di sepanjang tepi pada arah tepi dan menghilangkan piksel-piksel (piksel diatur menjadi 0) yang tidak dianggap sebagai tepi. Dengan cara seperti itu, diperoleh tepi yang tipis.

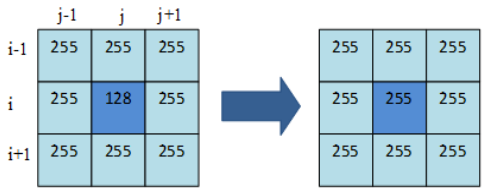
Langkah keenam berupa proses yang disebut *hysteresis*. Proses ini menghilangkan garis-garis yang seperti terputus-putus pada tepi objek. Caranya adalah dengan menggunakan dua ambang T1 dan T2. Lalu, semua piksel citra yang bernilai lebih besar daripada T1 dianggap sebagai piksel tepi. Selanjutnya, semua piksel yang terhubung dengan piksel tersebut dan memiliki nilai lebih besar dari T2 juga dianggap sebagai piksel tepi.

Bagian penting yang perlu dijelaskan adalah penghilangan non-maksimum dan peng-ambangan histeresis. Penghilangan non-maksimum dilakukan dengan mula-mula menyalin isi larik Grad (yang berisi besaran gradien) ke Non_max. Selanjutnya, penghilangan non-maksimum dilaksanakan dengan memperhatikan dua titik tetangga yang terletak pada arah tepi (yang tersimpan dalam Theta). Misalnya, arah tepi adalah 0. Apabila titik yang menjadi perhatian mempunyai koordinat (r, c), dua titik tetangga berupa (r, c-1) dan (r, c+1). Apabila gradien titik perhatian lebih besar daripada gradien kedua tetangga, nilainya akan dipertahankan. Sebaliknya, jika nilai titik perhatian lebih kecil daripada nilai salah satu atau kedua gradien tetangga, nilainya akan diabaikan (diubah menjadi nol).



Peng-ambangan histeresis dilakukan dengan melibatkan dua ambang T1 (ambang bawah) dan ambang T2 (ambang atas). Nilai yang kurang dari T1 akan diubah menjadi hitam (nilai 0) dan nilai yang lebih dari T2 diubah menjadi putih (nilai 255). Lalu, bagaimana nilai yang lebih dari atau sama dengan T1 tetapi kurang dari T2? Oleh karena itu, untuk sementara nilai pada posisi seperti itu diberi nilai 128, yang menyatakan nilai abu-abu atau belum jelas, akan dijadikan 0 atau 255 (Riana et al., 2023).

Selanjutnya, dilakukan pengujian untuk mendapatkan kondisi seperti tercantum pada Gambar 5. Apabila kondisi seperti itu terpenuhi, angka 128 diubah menjadi 255. Proses pengujian seperti itu dilakukan sampai tidak ada lagi perubahan dari nilai 128 menjadi 255. Tahap selanjutnya, semua piksel yang bernilai 128 yang tersisa diubah menjadi nol.



3. HASIL DAN PEMBAHASAN

Pada penelitian ini, dilakukan pengembangan sistem deteksi jumlah jari menggunakan MediaPipe dan OpenCV serta perhitungan akurasi hasil deteksi. Metodologi yang digunakan mencakup beberapa tahapan utama yang dilakukan secara sistematis, yaitu inisialisasi pustaka dan model deteksi tangan, membaca gambar input, deteksi tangan dan jumlah jari yang terangkat, penerapan algoritma deteksi tepi menggunakan Canny, perhitungan akurasi, serta penampilan hasil deteksi dalam bentuk visual dan numerik.

1. Inisialisasi Library dan Model Deteksi Tangan

Tahapan pertama adalah inisialisasi pustaka dan model deteksi tangan. Pada tahap ini, pustaka yang diperlukan seperti OpenCV digunakan untuk membaca dan memproses gambar, MediaPipe digunakan untuk mendeteksi tangan beserta landmark jari, serta pustaka google.colab.patches digunakan untuk menampilkan gambar dalam lingkungan Google Colab. Selanjutnya, model MediaPipe Hands diinisialisasi untuk mendeteksi tangan dan mengidentifikasi 21 titik landmark tangan dalam gambar yang akan diproses.

2. Membaca Gambar

Setelah model diinisialisasi, langkah berikutnya adalah membaca gambar input yang berisi tangan menggunakan OpenCV. Gambar dibaca dari path file yang telah ditentukan dan ditampilkan sebelum diproses lebih lanjut. Tahap ini penting untuk memastikan bahwa gambar dapat diakses dan digunakan sebagai input untuk sistem.

3. Deteksi Tangan Dan Jari

Tahapan selanjutnya adalah deteksi tangan dan jumlah jari yang terangkat. Gambar yang telah dibaca dikonversi ke format RGB karena model MediaPipe memproses gambar dalam format ini. Setelah itu, model MediaPipe digunakan untuk mendeteksi tangan dalam gambar dan mengidentifikasi landmark jari. Jika tangan terdeteksi, sistem akan menggambar landmark tangan pada gambar untuk memvisualisasikan hasil deteksi. Untuk menentukan jumlah jari yang terangkat, sistem mengevaluasi posisi ujung jari (tip) dan dasar jari (base). Jika posisi ujung jari lebih tinggi dari dasar jari dalam sumbu vertikal, maka jari tersebut dianggap terangkat. Proses ini dilakukan untuk empat jari utama, yaitu telunjuk, jari tengah, jari manis, dan kelingking.

4. Deteksi Tepi (Edge Detection) menggunakan Canny

Setelah jumlah jari terangkat terdeteksi, sistem akan melakukan deteksi tepi menggunakan algoritma Canny Edge Detection. Proses ini dimulai dengan mengonversi gambar ke dalam format grayscale, karena deteksi tepi lebih optimal pada gambar hitam putih. Selanjutnya, algoritma Canny diterapkan dengan menggunakan threshold atas dan bawah tertentu untuk mengekstrak kontur dan tepi tangan. Hasil dari proses ini akan menunjukkan garis-garis tepi tangan dan jari yang terdeteksi.

5. Perhitungan Akurasi Deteksi Jari

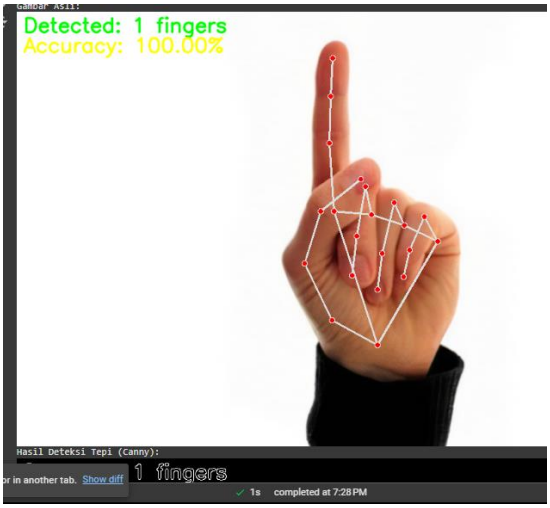
Tahapan berikutnya adalah perhitungan akurasi deteksi jari. Akurasi dihitung dengan membandingkan jumlah jari yang terdeteksi dengan jumlah jari yang sebenarnya dalam gambar. Perhitungan akurasi menggunakan rumus:

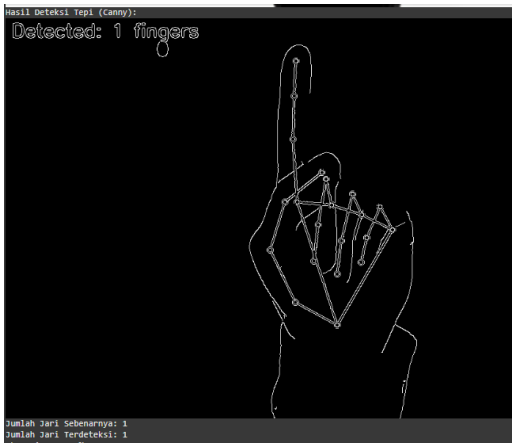
$$\text{Akurasi} = (1 - |\text{Aktual} - \text{Prediksi}|) \times 100\%$$

Jika jumlah jari sebenarnya adalah 0 dan sistem juga mendeteksi 0 jari, maka akurasi 100%. Namun, jika jumlah jari yang terdeteksi tidak sesuai dengan jumlah jari sebenarnya, maka akurasi akan berkurang. Perhitungan ini bertujuan untuk mengevaluasi sejauh mana sistem dapat mengenali jumlah jari dengan benar.

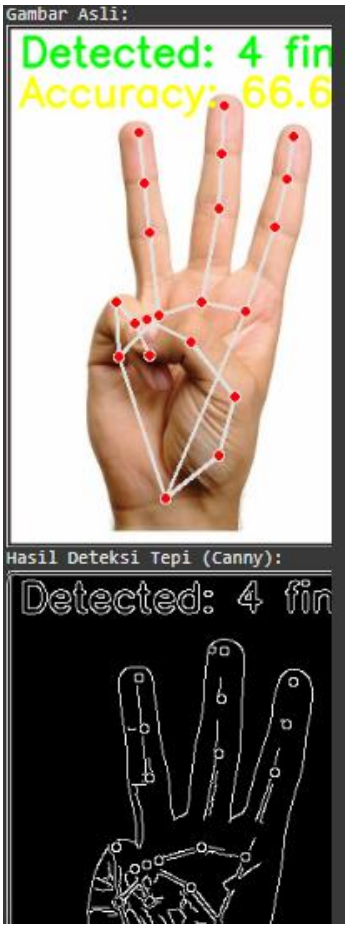
6. Menampilkan Hasil Deteksi dan Akurasi

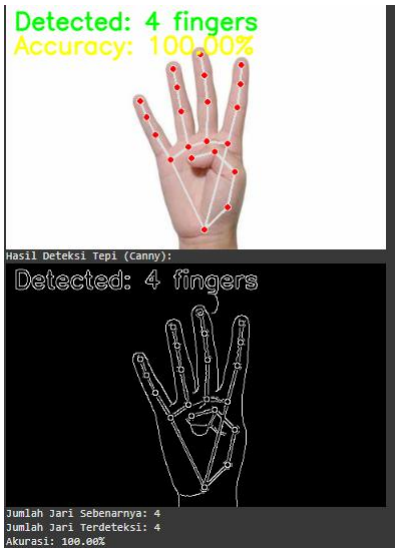
Tahapan terakhir adalah menampilkan hasil deteksi dan akurasi. Hasil deteksi jari akan ditampilkan dalam bentuk visual, di mana sistem akan menambahkan teks pada gambar yang menunjukkan jumlah jari yang terdeteksi serta tingkat akurasinya. Selain itu, gambar hasil deteksi tepi juga akan ditampilkan sebagai referensi tambahan. Informasi numerik mengenai jumlah jari yang sebenarnya, jumlah jari yang terdeteksi, serta akurasi deteksi juga dicetak di layar untuk dianalisis lebih lanjut.

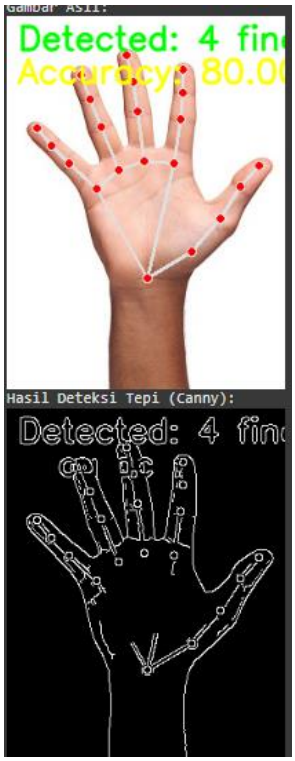

Detected: 1 fingers
Accuracy: 100.00%
Hasil Deteksi Tepi (Canny):
1 fingers


Hasil Deteksi Tepi (Canny):
Detected: 1 fingers
Jumlah Jari Sebenarnya: 1
Jumlah Jari Terdeteksi: 1


Gambar Asli:
Hasil Deteksi Tepi (Canny):


Gambar Asli:
Hasil Deteksi Tepi (Canny):


Detected: 4 fingers
Accuracy: 100.00%
Hasil Deteksi Tepi (Canny):
Detected: 4 fingers
Jumlah Jari Sebenarnya: 4
Jumlah Jari Terdeteksi: 4
Akurasi: 100.00%


Hasil Deteksi Tepi (Canny):

No	Deteksi	Aktual	Akurasi (%)
1	1	1	100
2	2	2	100
3	4	3	67
4	4	4	100
5	4	5	80
Rata Rata			89,4

4. KESIMPULAN

Penelitian ini mengembangkan sistem deteksi jumlah jari menggunakan MediaPipe dan OpenCV dengan akurasi rata-rata 89,4%. Sistem ini efektif mendeteksi jari yang terangkat pada gambar statis, meskipun terdapat batasan pada pencahayaan buruk, latar belakang kompleks, atau posisi tangan yang tidak jelas. Deteksi tepi dengan algoritma Canny memberikan visual tambahan untuk analisis kontur tangan. Untuk pengembangan selanjutnya, disarankan untuk menggunakan teknik pra-pemrosesan gambar lebih lanjut dan model deep learning untuk meningkatkan akurasi serta menguji sistem pada berbagai kondisi pencahayaan dan latar belakang. Implementasi real-time juga direkomendasikan untuk aplikasi seperti pengenalan gestur.

REFERENSI

Adolph, R. (2016). *済無No Title No Title No Title*. 1–23.

Beno, J., Silen, A. ., & Yanti, M. (2022). No 主観的健康感を中心とした在宅高齢者における 健康関連指標に関する共分散構造分析Title. *Braz Dent J.*, *33*(1), 1–12.

Daniel Tanugraha, F., Pratikno, H., Musayanah, M., & Indah Kusumawati, W. (2022). Pengenalan Gerakan Olahraga Berbasis (Long Short- Term Memory) Menggunakan Mediapipe. *Journal of Advances in Information and Industrial Technology*, *4*(1), 37–45. https://doi.org/10.52435/jaiit.v4i1.182

Riana, D., Uki Eka Saputri, D., & Hadianti, S. (2023). Klasifikasi Alexnet dan Deteksi Tepi Canny untuk Identifikasi Citra Repomedunm. *Jurnal Informasi Dan Teknologi*, *5*(1), 191–198. https://doi.org/10.37034/jidt.v5i1.295

Supriyatin, W. (2020). Perbandingan Metode Sobel, Prewitt, Robert dan Canny pada Deteksi Tepi Objek Bergerak. *ILKOM Jurnal Ilmiah*, *12*(2), 112–120. https://doi.org/10.33096/ilkom.v12i2.541.112-120

Tsani, N. B., & Harliana, H. (2019). Implementasi Deteksi Tepi Canny Dengan Transformasi Powerlaw Dalam Mendeteksi Stadium Kanker Serviks. *Jurnal Ilmiah Intech : Information Technology Journal of UMUS*, *1*(01), 22–33. https://doi.org/10.46772/intech.v1i01.35