SinarFe7
Seminar Nasional Fortei Regional 7

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

Robot Penerima Tamu dengan Sistem Pengenalan Wajah dan Suara

[1] Aaron Marselino Lukman, [2] Petrus Santoso
Program Studi Teknik Elektro, Universitas Kristen Petra, Surabaya
[1]m23416010@john.petra.ac.id, [2] petrus@petra.ac.id

Abstract - This Project will be focused on integration of face recognition and voice recognition to make a reception robot. This reception robot system uses Nvidia Jetson Nano as the main embedded device. There are 2 sensors used, the camera for image input and microphone for sound input. As for the actuator there are 2 devices used, the LCD monitor for the output image and the speaker for the sound output. The entire program on this system is written in Python with Ubuntu as the operating system.Based on the test results, the system successfully performed face recognition and voice recognition. The highest accuracy for the face recognition was 96%. The execution time of the face recognition program was very fast with an average time of 0.0544 second. The amount of face data that is recognized does not affect the execution time of the face recognition capability. With modified dictionary and language model data of voice recognition, the accuracy was 84.65%. This success rate is higher than using the default data which is only 26.32%.

Keywords — Receptionist Robot, face recognition, voice recognition, Jetson Nano

***Abstrak* - Proyek ini bertujuan untuk mengintegrasikan antara pengenalan wajah dan pengenalan suara untuk membuat robot penerima tamu. Sistem robot penerima tamu ini menggunakan Nvidia Jetson Nano sebagai perangkat *embedded* utama. Sensor yang digunakan ada 2, yaitu kamera sebagai masukan gambar dan mikropon sebagai masukan suara. Sedangkan untuk aktuator terdapat 2, yaitu lcd monitor sebagai keluaran gambar dan speaker sebagai keluaran suara. Keseluruhan program pada sistem ini dibuat dengan bahasa Python pada *operating system* Ubuntu. Berdasarkan hasil pengujian, sistem telah berhasil melakukan pengenalan wajah dan pengenalan suara. Untuk pengenalan wajah akurasi tertinggi didapatkan angka 96%. Waktu eksekusi dari program pengenalan wajah sangat cepat dengan rata-rata waktu 0,0544 detik. Jumlah data wajah yang dikenali tidak memengaruhi waktu dari eksekusi program pengenalan wajah. Dengan memodifikasi data pada pengenalan suara, didapatkan akurasi 84,65%. Akurasi ini jauh lebih tinggi daripada menggunakan data *default* yang hanya mendapatkan akurasi 26,32%.**

Kata Kunci — Robot penerima tamu, pengenalan wajah, pengenalan suara, Jetson Nano

I. PENDAHULUAN

Robot pada umumnya digunakan untuk menyelesaikan kegiatan yang tidak mampu atau tidak ingin dilakukan oleh manusia. Hal ini akan membuat robot memiliki banyak kelebihan yang tidak dibandingkan manusia yaitu menghasilkan luaran dengan hasil yang sama dan akurat secara berulang-ulang, tidak memiliki rasa letih dan lelah yang bisa menguntungkan bagi pengusaha dan perusahaan [1].

Salah satu kegunaan robot dalam berinteraksi dengan manusia adalah pada pusat perbelanjaan. Tempat ini merupakan suatu tempat yang memiliki banyak pendatang dan digunakan oleh masyarakat untuk melakukan rekreasi dan melepas rasa penat setelah melakukan aktifitas [2]. Maka dari itu dengan banyaknya pengunjung yang datang, maka dibutuhkan peningkatan pelayanan. Salah satu contoh dari meningkatkan pelayanan adalah menyambut pengunjung dengan baik dan membantu pengunjung dalam memberikan penunjuk arah atau informasi yang dibutuhkan melalui robot. Robot inilah yang digunakan untuk menggantikan peran manusia karena lebih efektif dan efisien.

Di Indonesia sendiri masih sedikit robot yang berguna untuk menerima tamu, baik dari memberikan salam, menunjukkan arah, maupun memberikan informasi. Padahal menurut penjelasan di paragraf sebelumnya, robot penerima tamu ini sangat penting untuk meningkatkan pelayanan. Salah satu contoh robot penerima tamu yang sudah dibuat sebelumnya ada pada penelitian dengan judul "*An Affective Guide Robot in Shopping Mall*" [2]. Robot tersebut memiliki beberapa fungsi dari pemandu arah, menjalin hubungan yang baik, dan menampilkan iklan. Pada fungsi pemandu arah pengunjung pusat perbelanjaan akan ditampilkan sebuah map lokasi yang diminta oleh pengunjung melalui pengenalan suara. Pada fungsi menjalin hubungan yang baik, robot bisa berkomunikasi layaknya manusia dan bisa bergerak untuk berinteraksi dengan pengunjung. Untuk fungsi iklan, nantinya akan ditampilkan toko beserta produk-produk yang dijual.

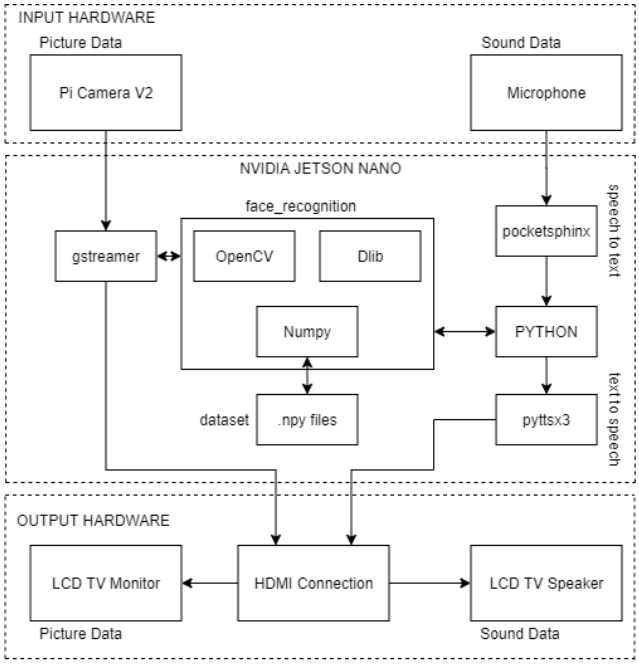
Dari latar belakang masalah dan referensi penelitian sebelumnya, penulis akan membuat robot penerima tamu untuk meningkatkan pelayanan di tempat umum, baik di pusat perbelanjaan, sekolah-sekolah, ataupun tempat-tempat yang memiliki banyak pengunjung. Pada proyek ini penulis akan mengurangi dan menambahkan beberapa fitur agar sesuai dengan kebutuhan dan kemampuan dari penulis. Pertama-tama untuk pengenalan suara tetap ada akan tetapi dibuat lebih sederhana. Kedua robot tidak memiliki interaksi fisik karena robot tidak bisa bergerak. Ketiga untuk menjalin hubungan yang lebih baik maka ditambahkan pengenalan wajah untuk menggantikan *tag* rfid. Hal ini dilakukan agar robot bisa berkenalan dengan orang baru ataupun menyapa orang yang sudah pernah datang sebelumnya. Keempat, fungsi dari robot penerima tamu ini ada 3, yaitu memberi salam, menampilkan informasi, dan menunjukkan arah

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

II. PERENCANAAN DAN IMPLEMENTASI

Proyek ini berfokus untuk mengintegrasikan program pengenalan wajah dan program pengenalan suara. Robot ini memiliki 3 fungsi utama yaitu pemberian salam, menampilkan informasi, dan menunjukkan arah dari suatu lokasi tertentu. Pengenalan wajah merupakan salah satu fitur yang digunakan untuk robot bisa melakukan pemberian salam. Sedangkan untuk pengenalan suara, digunakan untuk menggantikan *input* dari suatu perangkat *embedded* yang pada umumnya menggunakan *keyboard* ataupun *mouse.*

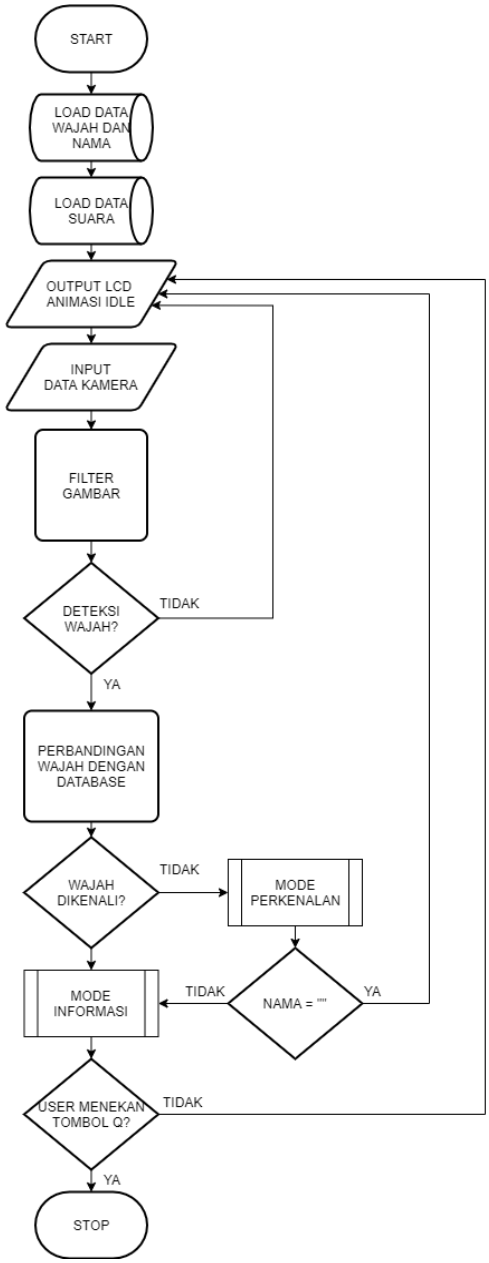
Sistem yang dibuat ini merupakan gabungan dari beberapa *input* dan *output* yang di program menggunakan bahasa Python pada sebuah perangkat *embedded* bernama Jetson Nano. Sistem ini memanfaatkan beberapa *library* yang sudah ada untuk melakukan pengenalan wajah dan suara. Pada sistem 2 data yang akan dilakukan proses yaitu data gambar dan data suara. Untuk meilihat sistem secara keseluruhan bisa dilihat pada Gambar 1.



Gambar 1. Blok diagram sistem keseluruhan

A. *Alur Program Sistem*

Pada alur program terdapat 3 buah *flowchart*. *Flowchart* pertama merupakan sistem secara keseluruhan dengan adanya 2 *subroutine* atau *function*. 2 *flowchart* lainnya merupakan penjelasan dari 2 *subroutine* tersebut yang akan ditampilkan lebih spesifik. *Flowchart* pertama ada pada Gambar 2.



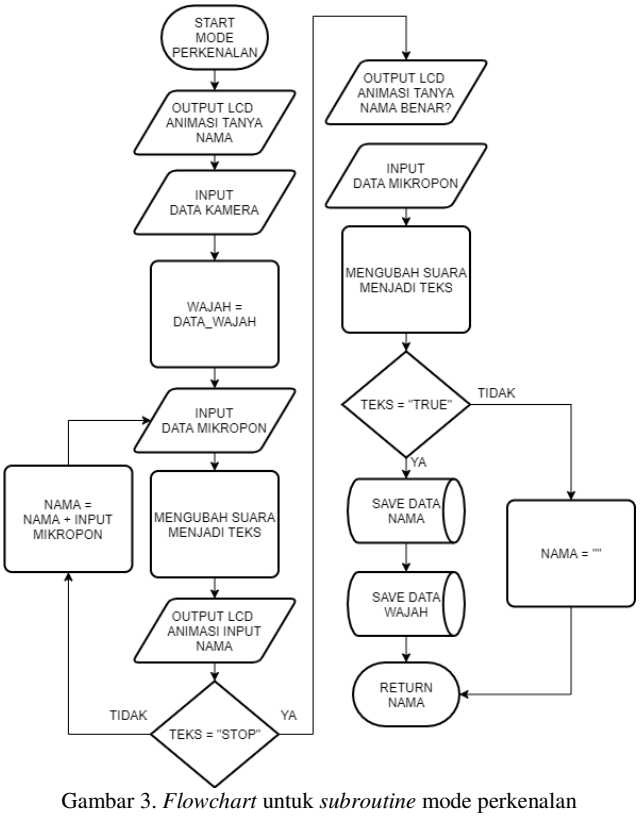
Gambar 2. *Flowchart* sistem secara keseluruhan

Pada tahap awal program akan melakukan *load* semua data wajah dan suara yang sudah tersimpan sebelumnya dalam sebuah *file*. Untuk melakukan interaksi dan mengetahui bahwa robot sedang dalam keadaan *idle* akan ditampilkan sebuah animasi wajah robot. Hal ini akan terpanggil secara terus-menerus jika wajah tidak terdeteksi. Saat wajah terdeteksi maka program akan melakukan perbandingan wajah yang terdeteksi tadi dengan data wajah yang sebelumnya di *load*

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

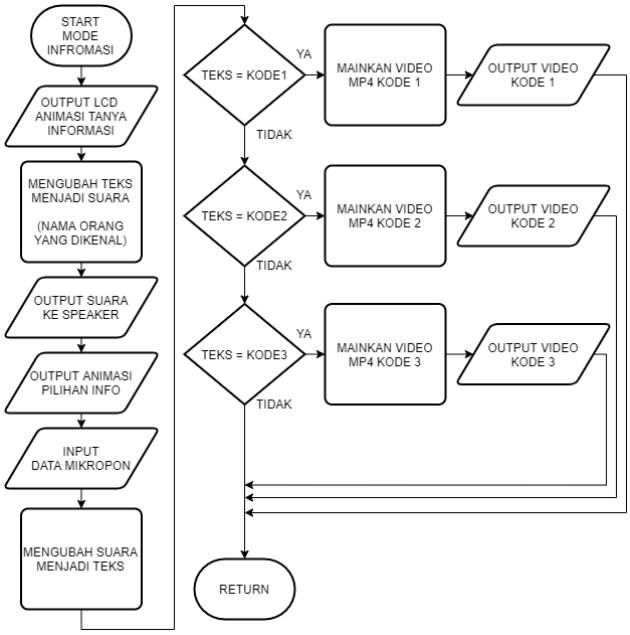
oleh program. Jika wajah yang terdeteksi tadi ada pada data wajah maka bisa diartikan bahwa wajah telah dikenali.

Pada proses percabangan wajah dikenali maka terdapat 2 jalan, yaitu masuk ke mode perkenalan yang nantinya wajah tersebut akan disimpan dalam *file* data wajah, atau mode informasi yang berguna agar pengunjung bisa bertanya informasi yang dibutuhkan. Di bawah mode perkenalan terdapat percabangan khusus yang melihat nilai *return* dari mode perkenalan. Percabangan ini dibuat untuk membedakan perkenalan yang berhasil dan tidak berhasil. Jika berhasil maka nilai *return* tidak kosong dan akan masuk ke mode informasi. Sedangkan untuk nilai *return* yang menghasilkan nilai kosong akan langsung kembali ke animasi *idle*. Pada akhir program akan disediakan sintaksis jika tombol "Q" pada *keyboard* ditekan maka program akan memberhentikan sistem secara keseluruhan.



Gambar 3. *Flowchart* untuk *subroutine* mode perkenalan

Pada Gambar 3 bisa dilihat alur dari *subroutine* mode perkenalan. Pada awalnya akan ditampilkan animasi berupa robot yang sedang menanyakan nama dan meminta pengguna untuk berada pada posisi yang tepat untuk menangkap gambar wajah yang dibutuhkan. Setelah gambar wajah ditangkap program akan melakukan konversi data wajah tersebut dalam bentuk *array* yang akan disimpan pada suatu variabel sementara. Setelah itu, program akan mendengarkan huruf atau abjad dalam Bahasa Inggris yang sudah ditentukan untuk mengeja nama dari pengguna tersebut sampai pengguna berkata "stop". Program akan berlanjut untuk memastikan apakah nama sudah benar atau tidak. Jika pengguna mengatakan "True" nama dan data wajah tadi akan disimpan, jika tidak maka variabel nama akan dikosongkan dan kembali ke program awal tanpa menyimpan data apapun.



Gambar 4. *Flowchart* untuk *subroutine* mode informasi

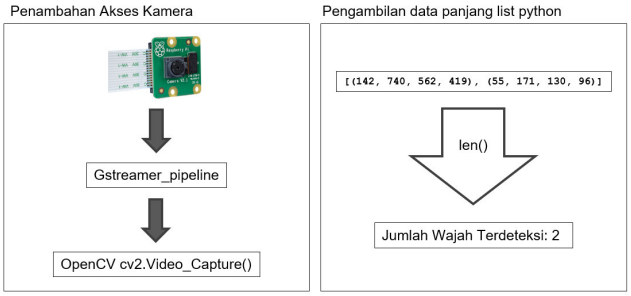
Pada Gambar 4 bisa dilihat alur dari *subroutine* dari mode informasi. Saat masuk ke dalam mode ini awalnya saat wajah dikenali program akan mengeluarkan *string* berupa nama orang yang dikenali tersebut. Untuk melakukan sapaan ini program akan menampilkan animasi bertanya dengan model orang berbicara dan menambahkan suara yang dilakukan oleh *text to speech* oleh sebuah *library*. Setelah itu pengguna akan diminta untuk mengatakan kode yang akan dibutuhkan informasinya. Jika kode tersebut ada pada pilihan maka kode yang terpilih tadi akan menampilkan video dari kode tersebut. Pada *flowchart* ini hanya diwakilkan 3 kode saja untuk menghemat tempat. Untuk program yang sebenarnya akan ada 9 percabangan yang masing-masing mewakili video infromasi yang ada.

B. Program Deteksi Wajah

Pemrograman deteksi wajah pada proyek ini menggabungkan *library* face_recognition milik Adam Geitegey [3] dengan CSI-Camera milik Kangalow [4]. *Library* face_recognition digunakan untuk melakukan deteksi wajah, dengan menghasilkan nilai *array* lokasi dari setiap wajah yang ditemukan. Sedangkan CSI-Camera digunakan untuk

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

mengakses kamera pada Jetson Nano agar bisa digunakan dalam pemrograman Python. Karena untuk deteksi wajah hanya diperlukan jumlah dari wajah yang ditemukan, maka *array* yang didapatkan tadi hanya diambil panjangnya saja menggunakan perintah len(). Penambahan program untuk deteksi wajah ini bisa dilihat pada Gambar 5

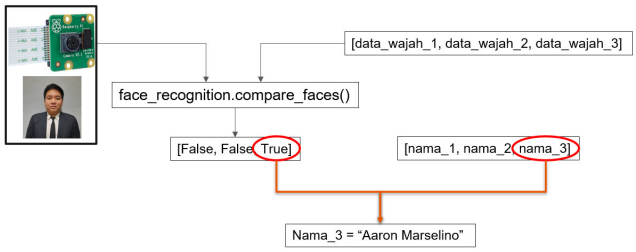


Gambar 5. Penambahan program deteksi wajah

C. *Program Pengenalan Wajah*

Program untuk melakukan pengenalan wajah menggunakan *library* yang sama dengan sistem deteksi wajah, yaitu face_recognition. Pada program ini untuk melakukan pengenalan wajah perintah yang digunakan adalah face_recognition.compare_faces(). Pada sintaksis ini terdapat nilai toleransi. Nilai ini bisa diubah-ubah sesuai kebutuhan semakin kecil angkanya maka semakin ketat pengenalan wajah yang dilakukan. Jika tidak diisi maka secara *default* nilai toleransi adalah 0,6.

Ada beberapa tahap untuk melakukan pengenalan wajah ini. Tahap pertama adalah deteksi wajah. Jika wajah tidak terdeteksi maka sintaksis compare_faces() tidak akan terpanggil. Untuk tahap kedua yaitu melakukan komparasi. Program ini akan menghasilkan letak dari dataset yang bernilai "True" yang mengartikan ada wajah yang terdeteksi. Tahap ketiga merupakan tahap untuk mengambil data nama dari wajah yang dikenali. Setelah letak nilai "True" ditemukan nama dengan urutan yang sama akan menjadi hasil dari program ini. Hal ini dikarenakan data nama dan data wajah memiliki urutan yang sama. Alur program secara sederhana bisa dilihat pada Gambar 6

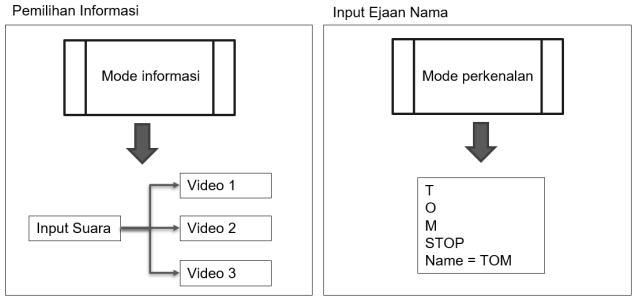


Gambar 6. Alur program sederhana dari pengenalan wajah

D. *Program Pengenalan Suara (Speech to text)*

Program pengenalan suara ini digunakan untuk 2 hal pada sistem yaitu untuk mendengarkan pengenalan nama dan untuk melakukan pemilihan informasi. Program ini menggunakan *library* yang sudah ada yaitu pocketpshinx [5] . *Library* ini dipilih karena kemampuannya untuk melakukan pengubahan dari suara menjadi teks secara *offline*.

Kegunaan dari *speech to text* ini ada 2 yaitu untuk menampilkan informasi dan mengeja nama dari pengguna yang belum dikenali. Kedua kegunaan ini akan dimasukkan ke dalam *function* yaitu mode informasi dan mode perkenalan. Contoh alur sederhana yang menggunakan *speech to text* sebagai inputnya bisa dilihat pada Gambar 7.



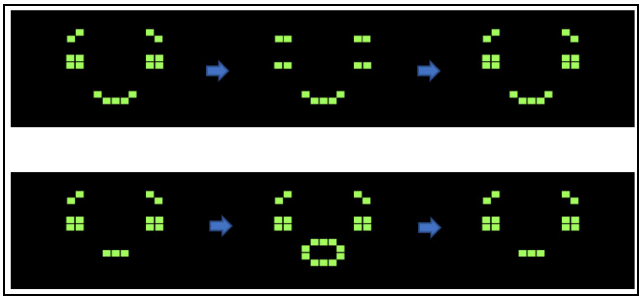
Gambar 7. Alur sederhana 2 kegunaan *speech to text*

E. *Program Menampilkan Animasi dan Informasi*

Pada sistem tampilan animasi dan informasi ini terdapat 2 buah program yang digunakan, yaitu menggunakan *video player* dan OpenCV. Untuk video player digunakan gstreamer melalui program python sedangkan OpenCV digunakan untuk menghasilkan animasi. Masalah yang terjadi sehingga dibutuhkannya 2 macam program adalah pada OpenCV tidak bisa menampilkan video dengan suara. Sehingga OpenCV ini akan digunakan untuk animasi yang tidak perlu sinkronisasi antara suara dengan gambar seperti animasi *idle* dan animasi tanya. Sedangkan untuk menampilkan informasi seperti arah dan petunjuk yang diperlukan untuk menampilkan mp4 dengan suara, maka dibutuhkan video player seperti gstreamer.

Untuk menampilkan animasi terdapat 2 *library* penting yang dibutuhkan yaitu OpenCV dan time. OpenCV digunakan untuk mengambil gambar dari *folder* dan menampilkannya *Library* time disini digunakan untuk menentukan lama waktu gambar tersebut ditampilkan. Lama waktu ini akan membuat gambar berganti-ganti dengan selang waktu yang berbeda-beda dan membuat gambar seolah-olah bergerak. Contoh animasi bisa dilihat pada Gambar 8.

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

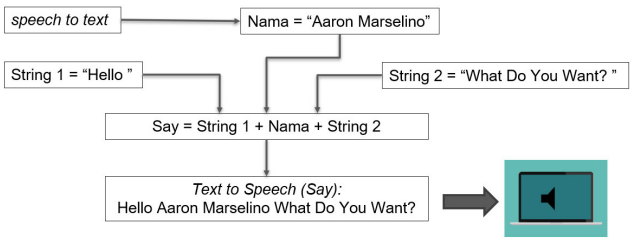


Gambar 8. Contoh animasi yang ditampilkan

Untuk *video player* digunakan program yang diambil dari github dengan *link*: https://github.com/hadware/gstreamer-python-player [6]. Pada folder github ini diambil sebuah *example* python dengan nama player-minimal.py. akan tetapi terdapat problem pada program ini sehingga diperlukan program tambahan untuk mengetahui kapan video selesai dan mematikannya. Maka dari itu ditambahakn program untuk menunggu EOS (*End of State*) dan melakukan set_state ke kondisi NULL. Program tambahan ini didapatkan dari kode Python pada *link*: https://github.com/gkralik/python-gst-tutorial/blob/master/basic-tutorial-1.py milik Gregor Kralik [7].

F. Program Mengubah Teks Menjadi Suara

Pada sistem ini untuk mengubah suara menjadi teks digunakan *library* yang dinamakan pyttsx3 [8]. *Library* ini akan secara otomatis mengubah teks dalam *string* menjadi suara yang dikirimkan ke *speaker* yang terdapat pada televisi melalui kabel HDMI. Kegunaan dari sistem ini adalah nantinya robot akan memanggil nama dari wajah yang dikenali. Karena nama dari orang tersebut tidak selalu sama dan bisa berubah-ubah maka dibutuhkan system *text to speech*. Hal yang perlu diperhatikan pada program ini adalah *speak rate*. Semakin kecil maka suara yang dihasilkan semakin lambat Gambar alur program dari *text to speech* ada pada Gambar 9.



Gambar 9. Alur program *text to speech*

G. Modifikasi Penyimpanan Data Wajah

Data wajah disimpan dalam bentuk numpy *file*. Maka dari itu, dibutuhkan *library* Numpy untuk menggunakannya. Untuk mengambil dan menyimpan data digunakan sintaksis np.load() dan np.save(). Numpy dipilih karena data yang digunakan untuk *library* face_recognition merupakan data *array*. Data untuk pengenalan wajah ini terdapat 2 jenis yaitu data *array* dari setiap wajah berupa 128 angka (known_face_encodings) dan data *array* nama dari setiap orang yang sudah dikenali (known_face_names). Untuk menghasilkan 128 angka *array* atau *embedding* ini bisa dilakukan dengan memanggil sintaksis face_recognition.face_encodings(). Perbedaan dari kedua dataset ini bisa dilihat pada Tabel 1

Tabel 1. Perbedaan dataset wajah dan nama

Keterangan	**Known_face_names**	**Known_face_encodings**
Kegunaan	Menyimpan nama dari setiap orang	Menyimpan dataset wajah yang telah dilakukan *encoding*
Bentuk	String List atau String Array	Multiple Array
Metode Penambahan	numpy.append()	numpy.vstack()
Contoh	["Aaron", "Biden", "Obama"]	[[128 array data_1] [128 array data_2] [128 array data_3]]

H. Modifikasi Penyimpanan Data Suara

Pada sistem pengenalan suara ini untuk meningkatkan akurasi maka dilakukan perubahan pada *file* LM dan *file* DIC dengan cara membuat kamus sendiri dengan bantuan program bernama lmtool. Cara untuk meningkatakan akurasi dari *speech to text* ini didapat dari web ghatage.com tentang *Make Pocketshinx recognize new words* [9]. Metode ini merupakan cara untuk membatasi banyaknya kata yang digunakan dalam *library* pocketsphinx. Sehingga dengan batasan ini diharapkan akurasi dari pengenalan suara ini bisa lebih baik. Data suara baru akan memiliki data-data meliputi "0-9", "A-Z", "STOP", "TRUE", dan "FALSE".

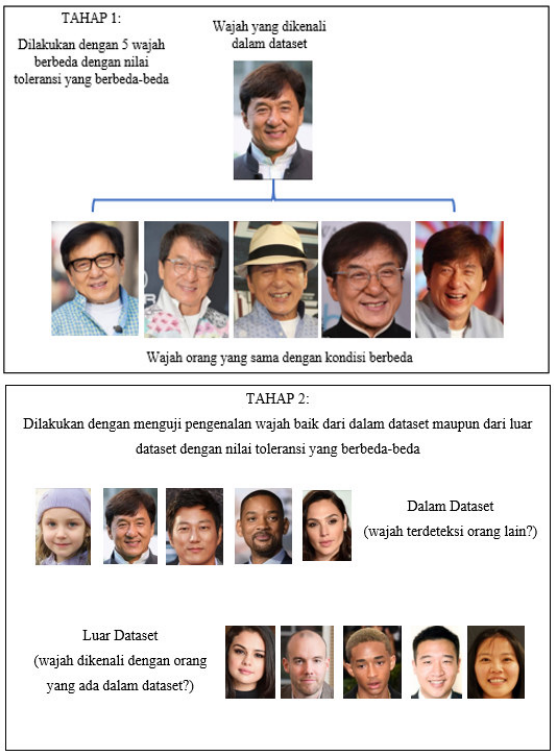
III. PENGUJIAN DAN HASIL

Pada bab ini akan dilakukan beberapa pengujian untuk mengetahui efektifitas dan efisiensi dari *library* ataupun sistem yang digunakan untuk robot penerima tamu ini.

A. Pengujian Pengenalan Wajah Berdasarkan Nilai Toleransi

Pengujian ini dilakukan untuk menentukan nilai toleransi pada program pengenalan wajah untuk mendapatkan akurasi yang paling tinggi. Pengujian ini dilakukan dalam 2 tahap. Untuk tahap pertama dilakukan pengenalan wajah sama dengan kondisi berbeda sebanyak 5 wajah. Sedangkan untuk tahap kedua dilakukan percobaan sebanyak 10 wajah dengan pembagian 5 wajah dalam *database* dan 5 wajah dari luar *database*. Nantinya kedua tahap ini akan dilakukan berulang kali dengan nilai toleransi yang berbeda-beda. Ilustrasi pengujian ini bisa dilihat pada Gambar 10.

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551



Gambar 10. Ilustrasi pengujian program pengenalan wajah

Tabel 2. Hasil akurasi total dari tahap 1 dan tahap 2

Nilai Akurasi Total			Toleransi Tahap 1				
			0.4	0.45	0.5	0.55	0.6
			68%	92%	100%	100%	100%
Toleransi Tahap 2	0.4	100%	84%				
	0.45	100%		96%			
	0.5	70%			85%		
	0.55	60%				80%	
	0.6	30%					65%

Dari Tabel 2 bisa diketahui bahwa akurasi paling tinggi adalah menggunakan nilai toleransi 0.45. hal ini dibuktikan dari hasil rata-rata akurasi antara tahap 1 dan tahap 2 yang menghasilkan nilai akurasi pengenalan wajah 96%. Sehingga dari sini untuk nilai dari variabel *tolerance* pada face_recognition.compare_faces() akan menggunakan 0.45 untuk mendapatkan hasil yang terbaik.

B. Pengujian Pengenalan Wajah Terhadap Data Wajah yang Dikenali

Pengujian ini dilakukan untuk mengetahui berapa waktu yang dibutuhkan untuk melakukan pengenalan wajah dengan jumlah data yang berbeda-beda. Pengujian dilakukan dengan menggunakan 8 dataset berbeda jumlah yang sudah disimpan sebelumnya. Waktu yang dicatat pada pengujian ini adalah waktu untuk mengambil dataset (load) dan waktu eksekusi program pengenalan sebanyak 5 kali percobaan dari orang yang sama.

Tabel 3. Data pengujian waktu pengenalan wajah

Jumlah Dataset	Waktu Eksekusi (detik)						
	Load	Waktu Pengenalan Wajah (detik)					Rata-Rata
		1	2	3	4	5	
3	0.034	**2.99**	0.05	0.04	0.05	0.05	0.048
5	0.039	**2.94**	0.06	0.05	0.05	0.07	0.059
10	0.038	**2.89**	0.04	0.07	0.06	0.04	0.057
15	0.040	**3.04**	0.06	0.04	0.06	0.04	0.053
20	0.035	**2.95**	0.07	0.05	0.04	0.04	0.053
30	0.034	**2.95**	0.37	0.04	0.03	0.04	0.040
40	0.033	**3.00**	0.08	0.05	0.04	0.04	0.054
50	0.043	**3.11**	0.09	0.07	0.05	0.04	0.068
Rata-Rata							0.05442

Dari Tabel 3 diketahui bahwa tidak ada perbedaan waktu yang signifikan antar jumlah dataset. Hanya saja terlihat bahwa pengenalan wajah pertama kali sejak program dijalankan akan memakan waktu lebih lama daripada pengenalan wajah berikutnya. Hal ini dikarenakan program membutuhkan waktu untuk melakukan *load* pertama kali dan ini membutuhkan waktu yang cukup lama. Karena waktu pengenalan wajah memiliki waktu yang sangat berbeda maka waktu pertama ini akan dianggap sebagai *outlier*. Sehingga waktu rata-rata untuk melakukan pengenalan wajah adalah 0.05442 detik

C. Pengujian Pengenalan Suara

Pengujian pengenalan suara ini dilakukan untuk mengetahui berapa akurasi yang bisa didapatkan dengan menggunakan *library* pocketsphinx yang sudah dimodifikasi pada *dictionary* dan *language model*. Dilakukan juga pengujian menggunakan *dictionary* dan *language model default* saat menginstall pocketsphinx. Sebagai pembanding dilakukan juga pengujian dengan bantuan *google translate* dimana nantinya *voice* pada *google translate* juga akan menjadi suara yang masuk pada mikropon. Hal ini dibuat dengan tujuan untuk mengetahui apakah pelafalan penulis dengan pelafalan *google translate* terdapat perbedaan yang signifikan. Pengujian dilakukan dengan menggunakan *reserve word* yang sebelumnya sudah disimpan pada dataset pengenalan suara.

Tabel 4. Hasil pengujian pengenalan suara

Data yang Digunakan	Suara Pengguna	Suara *voice google translate*
Data *default*	39.47%	13.16%
Data modifikasi	81.58%	87.72%

Dari hasil Tabel 4 dapat dilihat bahwa terjadi peningkatan akurasi yang sangat signifikan. Terlihat dari yang pengujian awal yang hanya menghasilkan 26.32% naik menjadi 84.65%. Pada pelafalan juga didapatkan hasil yang lebih baik dalam

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

menggunakan *voice* dari *google translate* dimana penulis hanya mendapatkan 81.58% sedangkan *voice google translate* mendapatkan akurasi 87.72%. Hal ini mengartikan bahwa pelafalan akan memengaruhi program pengenalan suara yang dilakukan.

IV. KESIMPULAN

Dari seluruh perencanaan, implementasi, dan pengujian dari robot penerima tamu ini, didapatkan kesimpulan sebagai berikut:

1. Pengenalan wajah memiliki akurasi tertinggi yaitu 96% dengan menggunakan nilai *tolerance* = 0.45 pada *function* compare_faces(). Untuk jumlah dataset, tidak ada pengaruh waktu yang signifikan. Eksekusi program berlangsung cukup cepat dengan rata-rata waktu 0.0544 detik.
2. Pengenalan suara hasil modifikasi memiliki akurasi yang lebih tinggi daripada menggunakan pengenalan suara dengan data *default*. Dimana hasil modifikasi menghasilkan angka akurasi 84.65%. Pelafalan juga memiliki pengaruh dengan tingkat akurasi. Pelafalan yang benar (*voice* google translate) memiliki akurasi lebih tinggi yaitu sekitar 87.72%.

V. DAFTAR PUSTAKA

[1] M. D. Putro and J. Litouw, "Robot Pintar Penyambut Costumer pada Pusat Perbelanjaan Kota Manado," *Rekayasa Elektr.*, vol. 13, pp. 8–17, 2017, Accessed: Sep. 24, 2019. [Online]. Available: https://media.neliti.com/media/publications/128171-ID-robot-pintar-penyambut-costumer-pada-pus.pdf.

[2] T. Kanda, M. Shiomi, Z. Miyashita, H. Ishiguro, and N. Hagita, "An affective guide robot in a shopping mall," in *Proceedings of the 4th ACM/IEEE International Conference on Human-Robot Interaction, HRI'09*, 2008, pp. 173–180, doi: 10.1145/1514095.1514127.

[3] A. Geitegey, "GitHub - ageitgey/face_recognition: The world's simplest facial recognition api for Python and the command line." https://github.com/ageitgey/face_recognition (accessed Jun. 22, 2020).

[4] Kangalow, "GitHub - JetsonHacksNano/CSI-Camera: Simple example of using a CSI-Camera (like the Raspberry Pi Version 2 camera) with the NVIDIA Jetson Nano Developer Kit." https://github.com/JetsonHacksNano/CSI-Camera (accessed Jun. 22, 2020).

[5] D. Prazdnichnov, "GitHub - bambocher/pocketsphinx-python: Python interface to CMU Sphinxbase and Pocketsphinx libraries." https://github.com/bambocher/pocketsphinx-python (accessed Jun. 22, 2020).

[6] "GitHub - hadware/gstreamer-python-player: Minimalist examples of audio players in python, using the new Gstreamer 1.0 API." https://github.com/hadware/gstreamer-python-player (accessed Jun. 22, 2020).

[7] K. Gregor, "python-gst-tutorial/basic-tutorial-1.py at master · gkralik/python-gst-tutorial · GitHub." https://github.com/gkralik/python-gst-tutorial/blob/master/basic-tutorial-1.py (accessed Jun. 22, 2020).

[8] N. Bhat, "GitHub - nateshmbhat/pyttsx3: offline Text To Speech synthesis for python." https://github.com/nateshmbhat/pyttsx3 (accessed Jun. 22, 2020).

[9] A. Ghatage, "Make Pocketsphinx recognize new words · Panopticon," 2012. http://ghatage.com/tech/2012/12/13/Make-Pocketsphinx-recognize-new-words/ (accessed Jun. 23, 2020).