IoT - Parking Lot Detection Based on Image Processing

I M. E. Listartha, K. F. Apriyana, I G. W. Pramartha, I G. K. K. Putra,
I W. S. Nirawana, S. Rusditya, G. Indrawan, K. Y. E. Aryanto
Program Studi Magister Ilmu Komputer,
Universitas Pendidikan Ganesha
Jalan Udayana No 11 Singaraja, Bali
Email: listartha@gmail.com, ferry.13jack@gmail.com, dewahyu.pramartha@gmail.com,
igedekarang@gmail.com, sugiantanirawana22@gmail.com, seftian.rusditya17@gmail.com,
gindrawan@undiksha.ac.id, yota.ernanda@undiksha.ac.id

Abstrak

Penelitian ini bertujuan untuk membangun sebuah sistem IoT untuk deteksi kondisi lot parkir kendaraan yang dimana semua hasil deteksi ditampilkan dalam sebuah aplikasi android. Hal ini bertujuan untuk memudahkan mendapatkan status sebuah area parkir dengan memanfaatkan perangkat mobile android. Proses deteksinya sendiri menggunakan sumber video dari kamera yang diproses dengan memanfaatkan teknik Gaussian Blur dengan nilai radius 5x5 dan Canny Edge Detection untuk segmentasi citra. Penentuan ukuran pixel untuk perkiraan ukuran kendaraan di dapatkan dari beberapa sample yang digunakan dan di dapatkan nilai yang sesuai untuk nilai deteksi adalah 200pixel. Nilai ukuran objek dibandingkan dengan hasil deteksi untuk mendapatkan informasi mengenai status lot parkir yang ada dan hasil tersebut disimpan dalam database. Semua sistem tersebut dijalankan pada perangkat Raspberry Pi yang digunakan juga sebagai server dari perangkat deteksi parkir. Untuk aplikasi client, perangkat yang digunakan berbasis android yang bekerja dengan melakukan query data ke server Raspberry dan menampilkan data terbaru dalam sebuah table. Dari hasil pengujian didapatkan akurasi proses deteksi hingga data yang di tampilkan adalah 90,9%.

Kata Kunci — parkir cerdas, gaussian blur, canny edge detection, internet of thing, raspberry pi, android

Abstract

This study aims to build an IoT system for the detection of vehicle lot lot conditions in which all detection results are displayed in an android app. It aims to make it easier to get the status of a parking area by utilizing android mobile devices. The detection process itself uses video sources from a camera processed by utilizing the Gaussian Blur technique with a radius value of 5x5 and Canny Edge Detection for image segmentation. The pixel size determination for the approximate vehicle size is obtained from the samples used and the corresponding value for detection value is 200pixels. The value of the object size is compared to the detection result to obtain information about the existing lottery status and the results are stored in the database. All such systems run on Raspberry Pi devices that are used as well as servers from parking detection devices. For client applications, android based devices work by querying data to the Raspberry server and displaying the latest data in a table. From the test results obtained accuracy detection process until the data in the show is 90.9%.

Keywords— smart parking, gaussian blur, canny edge detection, internet of thing, raspberry pi, android

1. Pendahuluan

Era modern telah menunjukkan perkembangan setiap tahun. Banyak teknologi-teknologi canggih yang telah dibuat oleh pengembang. Salah satu perkembangan teknologi dalam bidang transportasi yang dapat kita temukan adalah sistem pelayanan parkir. Dewasa ini perparkiran dalam suatu gedung sudah mulai menggunakan sistem komputerisasi dalam pengoperasiannya, tetapi pengguna parkir masih saja terkendala atau kesulitan dalam mencari tempat parkir yang kosong dengan mengelilingi area parkir sehingga kurang efisien dan membutuhkan waktu yang lama. Jika proses pelayanan tersebut dapat digantikan dengan menggunakan sistem yang lebih modern (otomatisasi sistem) akan sangat menguntungkan, baik itu bagi perusahaan yang bersangkutan terlebih lagi bagi pengguna parkir itu sendiri.

Kendala tersebut bisa terjadi pada saat harus memutari parkir minimal satu kali untuk mendapat lokasi yang pas. Kemudian parkir biasanya hanya satu arah, saat terlewat susah untuk mundur kembali. Berdasarkan permasalahan tersebut, maka perlu di buat aplikasi *smart parking* berbasis IoT yang bisa memberitahukan lot parkir kosong kepada pengguna, sehingga mereka bisa memarkir kendaraan tanpa harus mencari secara manual kembali. Jika proses pelayanan bisa diganti dengan menggunakan sistem yang lebih modern (sistem otomasi) makan akan sangat menguntungkan, baik itu untuk perusahaan yang bersangkutan maupun pengguna parkir itu sendiri [1].

2. Tinjauan Pustaka

Dalam penulisan tesis ini penulis mengumpulkan segala informasi dari referensi, literatur yang sesuai dengan topik dan menggunakan media internet sebagai bahan referensi tambahan.

2.1. *Python*

Dalam penelitian ini, pembangunan aplikasi pemetaan menggunakan bahasa pemrograman *Python*. Bahasa ini dipilih karena mudah dipelajari, memiliki struktur data tingkat tinggi dan pendekatan pemrograman berbasis objek (*Object Orriented Programming*) yang sederhana namun efektif[2].

Ada yang membedakan antara *Python* dengan bahasa pemrograman lain seperti C/C++ dan *Java* yang memerlukan kompilasi terlebih dahulu untuk bisa dijalankan. *Python* merupakan *interpreted language* atau *scripting language* artinya kita tinggal menuliskan program kemudian langsung bisa menjalankannya tanpa harus mengompilasinya terlebih dahulu. Sebagai interpreter *Python* memiliki keuntungan misalnya kita tidak perlu secara eksplisit mendeklarasikan jenis variabel apakah *String* atau integer, layaknya bahasa pemrograman *compiler* seperti C/C++ dan Java. Karena *Python* sebagai bahasa pemrograman *interpreter* ia tidak memiliki file *binary* seperti halnya pada C/C++ dimana kita bisa merahasiakan *source code*, dan cukup mendistribusikan file *binary* untuk bisa dijalankan. Di *Python*, *source code* adalah program itu sendiri yang bisa langsung dijalankan

2.2. *OpenCV*

Diambil dari *OpenCV*.org[3]. *OpenCV (Open Source Computer vision Library)* adalah software pustaka *open source* untuk *computer vision* dan *machine learning*. *OpenCV* dibangun untuk menyediakan infrastruktur umum untuk aplikasi *computer vision* dan untuk mempercepat penggunaan persepsi mesin ke dalam produk komersial.

Gambar 1. Fondasi *OpenCV*

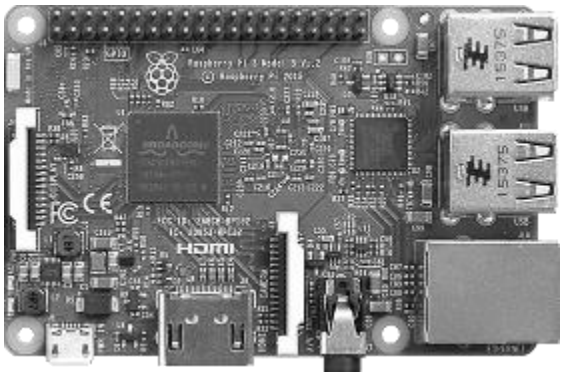
Penelitian ini menggunakan versi 3.2 yang dapat diintegrasikan pada bahasa *Python* dengan mudah, dimana instalasi hanya dilakukan dengan memindahkan file cv2.pyd dari paket instalasi *OpenCV* ke folder instalasi *Python* di *site-packages*. Paket file ini secara default hanya disediakan untuk *Python* versi 2.7 dan mendukung untuk arsitektur x86, ARM maupun x64.

2.3. *Raspberry Pi*

Raspberry Pi adalah modul *microcomputer* yg juga mempunyai *input output digital port* seperti pada board *microcontroller*[4]. Diantara kelebihan Rasberry Pi dibanding *board microcontroller* yang lain yaitu mempunyai *Port*/koneksi untuk *display* berupa TV atau Monitor PC serta koneksi USB untuk *Keyboard* serta *Mouse*. *Raspberry Pi* dibuat di inggris oleh *Raspberry Pi Foundation* Pada awalnya *Raspberry Pi* ditunjukan untuk modul pembelajaran ilmu komputer disekolah.

Raspberry Pi sejak dirilis pada tahun 2012 telah memiliki lima model, empat diantara dapat digunakan oleh orang umum namun satu untuk tujuan pengembangan. Sebagai sebuah IoT *(Internet of Things*), dengan cepat dan mudah Raspsberry Pi dapat *start-up* dan berselancar di internet, atau untuk menulis program seperti layaknya sebuah kit pengembangan SDK (*Software Development Kit*) untuk alat

tersebut, untuk mengontrol perangkat eksternal atau menjalankan robot dan seperti tujuan awal dari proyek *Rapsberry Pi* ini, yaitu untuk membangkitkan daya tarik bagi anak-anak sekolahan agar aktif menggeluti dunia teknologi informatika.



Gambar 2. *Raspberry Pi*

Untuk menggunakan *Raspberry Pi* diperlukan *operating system* (contoh OS: *windows, Linux, mac, Unix* dst) yg dijalankan dari *SD card* pada *board Raspberry* tidak seperti pada *board microcontroller* AVR yang selama ini kita pakai tanpa OS[5]. *Operating system* yang banyak dipakai antara lain *Linux* distro *Raspbian*. OS yang bisa di jalankan di *Raspberry* antara lain: *Arch Linux ARM, Debian GNU/Linux, Gentoo, Fedora, FreeBSD, NetBSD, Plan 9, Inferno, Raspbian OS, RISC OS* dan *Slackware Linux*.

2.4. *Gaussian Blur*

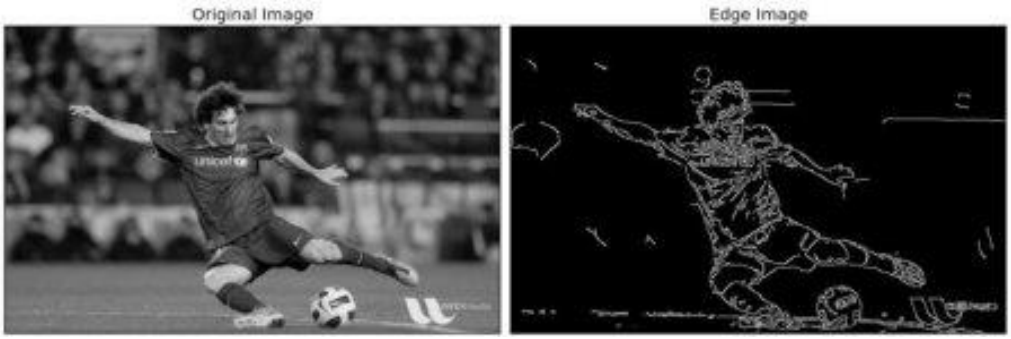
Gaussian Blur merupakan konvolusi citra dengan fungsi *gaussian*. Istilah ini dibuat karena *gaussian* memiliki beberapa sifat yang cukup unik. Pertama karena hasil transformasi *fourier* fungsi *gaussian* ternyata menghasilkan *gaussian* juga[6]. Kedua yang menarik dari fungsi *gaussian* adalah untuk implementasi fungsi *gaussian* 2D. Fungsi *gaussian* 2D secara matematis memiliki sifat dapat dipisahkan (*separable*). Sifat inilah yang sangat membantu dalam mengimplementasi proses pengkaburan (*blurring*).

Gambar 3. Efek *Gaussian Blur*

Dalam operasi perata-rataan tetangga (konvolusi) digunakan jendela atau area yang menyatakan hubungan ketetanggaan yang dilibatkan dalam operasi. Semakin besar ukuran jendela tersebut maka gambar akan semakin kabur dan juga waktu yang diperlukan untuk melakukan operasi tersebut. Dengan adanya operasi yang bersifat dapat dipisahkan, proses konvolusi bisa dilakukan dua kali namun hanya menggunakan jendela berdimensi satu. proses konvolusi pertama kali dilakukan untuk tiap baris dan dilanjutkan dengan konvolusi untuk tiap kolom sebagai data berdimensi satu juga. Akibat partisi ini operasi konvolusi bisa mengeksploitasi paralelisme baik menggunakan *thread* ataupun prosesor grafik.

2.5. *Canny Edge Detection*

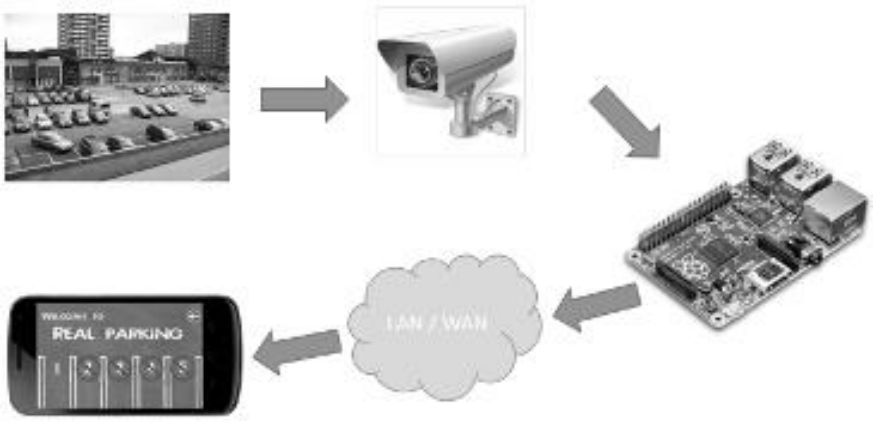
Algoritma *Canny Edge Detection* merupakan salah satu teknik *edge detection* yang cukup populer penggunaannya dalam pengolahan citra[7]. Salah satu alasannya adalah ketebalan *edge* yang bernilai satu *pixel* yang dimaksudkan untuk melokalisasi posisi *edge* pada citra secara seakurat mungkin. Algoritma *Canny Edge Detection* secara umum beroperasi sebagai penghalusan untuk mengurangi dampak *noise* terhadap pendeteksian *edge*, menghitung potensi *gradien* citra, *non-maximal supression* dari *gradien* citra untuk melokalisasi *edge* secara presisi, *hysteresis thresholding* untuk melakukan klasifikasi akhir.



Gambar 4. Hasil deteksi tepi.

3. Metode Penelitian

Struktur *smart parking* berbasis IoT ini dibuat dalam bentuk permodelan yang terdiri dari tempat parkir yang terintegrasi dengan kamera. Kamera dipilih sebagai input data dikarenakan umumnya ada disetiap parkir umum/apartemen dan pemasangnya lebih mudah dalam instalasinya dibandingkan pemasangan sensor khusus disetiap lot parkir. Jangkauan kamera/cctv bisa mencakup area yang luas, namun dalam penelitian ini akan menggunakan sebuah *webcam* untuk mendapat *input* video dari kendaraan yang parkir pada lot parkir yang tersedia. Berikutnya dalam penelitian ini menggunakan komponen *Raspberry Pi* berguna untuk menganalisis data video untuk didapatkan lot yang kosong dan yang tidak. Perangkat raspiberry pi ini terhubung pada jaringan WAN /LAN untuk menerapkan fungsi-fungsi yang di miliki oleh *OpenCV* dalam menganalisis data video. Aplikasi *mobile* akan menampilkan seluruh informasi mengenai lot parkir yang kosong ataupun yang tidak. Berikut gambar topology *smart parking* berbasis IoT.

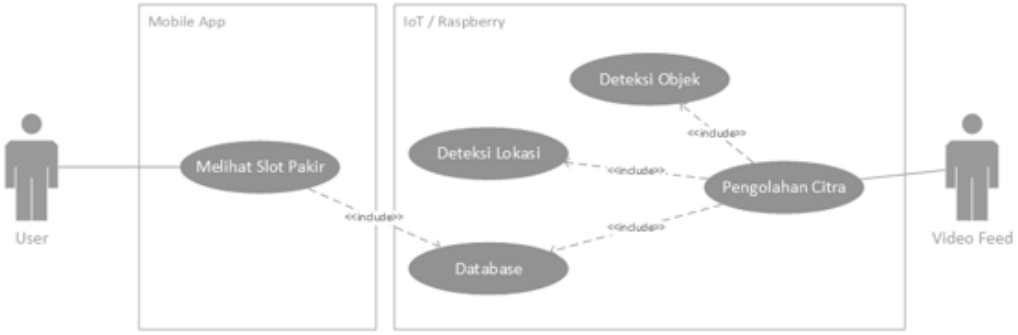


Gambar 5. Topologi Sistem

Semua informasi mengenai lot yang kosong akan dianalisis dengan algoritma tertentu. Semua informasi tersebut akan secara *realtime* di *update* ke aplikasi *mobile client*.

3.1. *Use case* Diagram

Dalam membuat aplikasi *mobile smart parking* berbasis IoT ini menggunakan *use case* diagram yang dapat dilihat pada gambar 6.

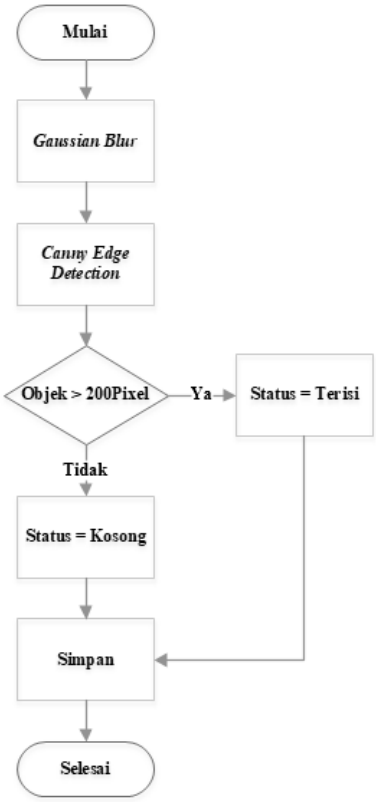


Gambar 6. *Use case* Diagram

Sistem di bangun menjadi dua bagian dimana, terdapat sistem pada *Raspberry Pi* dan perangkat *android*. Sistem pada *Raspberry* memiliki proses yang lebih kompleks dimana sistem ini menangani proses pengambilan data dari sumber video, mengolahnya menjadi informasi, kemudian menyimpannya pada *database mysql* sehingga dapat di *query* menggunakan bahasa *php* untuk membuat tampilan web. Tampilan ini akan di *request* oleh aplikasi pada *android* menggunakan *protocol http* yang secara tampilan akan diperbaharui setiap 3 detik.

3.2. Alur Program

Proses deteksi dilakukan pada setiap gambar yang didapatkan dari data video, deteksi dilakukan pada setiap *frame* yang ada sehingga hasil deteksi didapatkan langsung dari setiap *frame.* Alur proses deteksi ditampilkan pada gambar 7



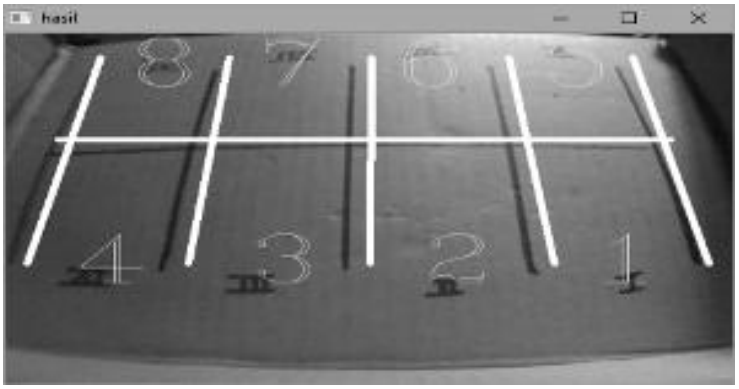
Gambar 7. Alur Deteksi pada Setiap *Frame.*

4. Hasil dan Pembahasan

Penerapan aplikasi yang dibangun sepenuhnya menggunakan bahasa pemrograman *Python* dan diintegrasikan dengan modul dari *OpenCV*, semua komponen ini di *install* dan dijalankan pada *Raspbery* Pi dengan sistem operasi *Raspbian*.

4.1. Pengambilan Gambar

Proses ini menggunakan modul dari *OpenCV* dalam mengambil gambar dari sumber video[8]. Kamera diletakan secara baik untuk mengambil area parkir yang diinginkan, pengambilan diposisikan supaya tidak terganggu saat ada objek lain yang menghalangi dan semua lot parkir tetap terlihat biarpun berisi kendaraan. Permodelan area parkir dilakukan seperti pada gambar 8.

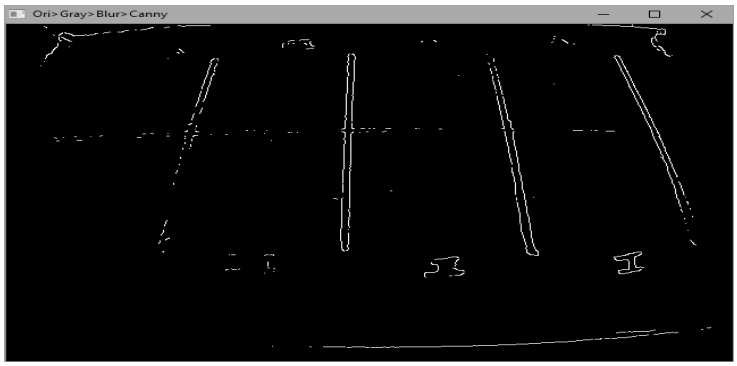


Gambar 8. Pemetaan Pengambilan Gambar

Penyesuaian area yang di pantau disesuaikan dengan ukuran asli setiap lot di area parkir, garis putih vertikal dan horizontal pada video mempresentasikan area yang di monitor.

4.2. Mencari Nilai Segmentasi

Video sumber yang masih berwarna harus di konversi menjadi abu-abu sesuai dengan syarat proses selanjutnya[9]. Gambar abu-abu ini dihaluskan dengan menghilangkan gangguan bintik yang muncul dengan menggunakan *Gaussian Blur*[10]. Nilai yang digunakan adalah dengan radius 5x5 sesuai dengan nilai contoh yang diberikan pada manual *OpenCV* yang selanjutnya dilakukan proses deteksi tepi. Hasil deteksi tepi ditampilkan pada gambar 9.



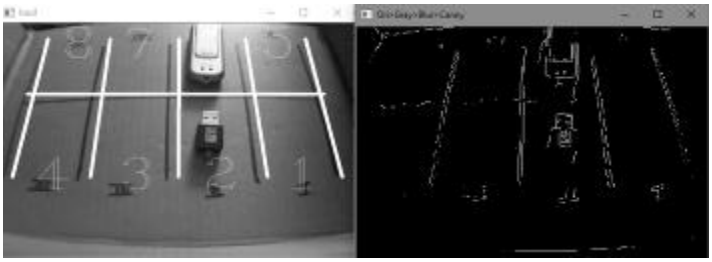
Gambar 9. Segmentasi Gambar

Dengan melakukan proses deteksi tepi pada video didapatkan akurasi yang lebih baik dalam proses deteksi objek karena pada video antara objek dan latar belakang sudah terpisah[7].

4.3. Deteksi Objek

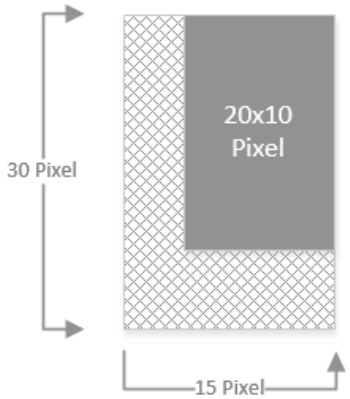
Sesuai dengan kondisi asli pada area yang dimonitor, area lot dibagi menjadi 8 area monitor dimana jika ada objek pada area ini akan dihitung jumlah pixel yang dimiliki dan jika melebihi nilai yang

ditentukan maka akan dianggap area lot itu sudah terisi. Sebelumnya objek yang akan digunakan sebagai kendaraan dihitung jumlah pixel dan dicari nilai terkecilnya. Dari nilai tersebut diambil nilai 200pixel sebagai nilai terkecil untuk proses deteksi.



Gambar 10. Deteksi Objek.

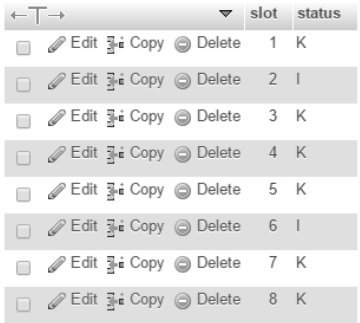
Hasil deteksi kemudian disimpan dalam database *mysql* yang dipasang pada *Raspberry Pi* secara langsung pada setiap gambar dari video yang diproses. Ukuran perbandingan ukuran kendaraan dan area parkir digambarkan dalam gambar 11.



Gambar 11. Perbandingan Luas Lot Parkir dan Objek 200 Pixel

4.4. Tabel Database

Pembangunan tabel untuk menyimpan data ini hanya terdiri dari kolom lot dan status. Kolom lot mempresentasikan area lot yang dimonitor pada parkir dan status adalah untuk menyatakan keadaan parkir apakah terisi atau tidak. Gambar 12 menampilkan contoh data pada tabel tersebut, pengambilan data ini dilakukan dengan memanfaatkan aplikasi *phpmyadmin*.



	slot	status
Edit Copy Delete	1	K
Edit Copy Delete	2	I
Edit Copy Delete	3	K
Edit Copy Delete	4	K
Edit Copy Delete	5	K
Edit Copy Delete	6	I
Edit Copy Delete	7	K
Edit Copy Delete	8	K

Gambar 12. Struktur Tabel

Pada kolom status, K digunakan untuk mempresentasikan Kosong yang artinya lot parkir tidak berisi kendaraan dan I artinya lot sedang terisi kendaraan.

4.5. Aplikasi Web pada *Android*

Pada perangkat *android* dibangun aplikasi berbasis web yang melakukan *query* kedalam database yang dibangun. Nilai tabel ditampilakan pada halaman web dan mempilkan status area parkir secara langsung yang terbarui setiap 3 detik.

Saat ada perubahan pada data, sistem akan memberikan notifikasi untuk memberikan status terakhir mengenai kondisi area parkir. Dalam aplikasi *android* yang di bangun, aplikasi melakukan koneksi pada alamat IP yang didefenisikan sebagai sumber data.

Gambar 13. Aplikasi *Android* dan Rapsberry Pil

4.6. Hasil Pengujian

Tabel 1. Hasil Pengujian Data

Simulasi	Jumlah Kendaraan	Objek Pengganggu	Jumlah Deteksi Tersisi Per-Lot	Hasil
1Car.mp4	1	0	1	Sesuai
2Cars.mp4	2	0	2	Sesuai
3Cars.mp4	3	0	3	Sesuai
4Cars.mp4	4	0	4	Sesuai
5Cars.mp4	5	0	5	Sesuai
6Cars.mp4	6	0	6	Sesuai
7Cars.mp4	7	0	7	Sesuai
8Cars.mp4	8	0	8	Sesuai
1Melintang.mp4	4	0	5	Sesuai
Batu.mp4	3	1	4	Sesuai
Krikil.mp4	2	4	2	Tidak

Dari pengujian yang dilakukan didapatkan informasi bahwa untuk kendaraan yang parkir melintang (mengambil 2 lot parkir) terdeteksi dengan baik dengan mendapatkan hasil deteksi dua lot itu terpakai. Hal ini terjadi karena setengah badan kendaraan itu mengambil lebih dari 200pixel area lot lainnya, dan kondisi ini membuat area lot itu tidak bisa digunakan oleh kendaraan lain.

Untuk pengujian pada video batu, ukuran batu yang digunakan lebih dari 200pixel. Hal ini membuat lot tempat batu tersebut memiliki status terisi, sehingga mobil tidak mungkin akan bsia parkir jika ada batu besar pada lot. Hal ini disimpulkan sebagai hasil deteksi yang sesuai. Berbeda dengan video krikil, ukuran deteksinya sedikit namun karena berserakan membuatnya terdeteksi lebih dari 200pixel sehingga lot terdeksi terisi biarpun pada penerapannya, kendaraan bisa menggunakan area lot tersebut. Kesimpulan yang di dapatkan untuk pengujian diatas untuk akurasinya adalah:

$$Persentase\ Akurasi = \frac{Hasil\ Sesuai}{Total\ Sample}\ x\ 100\% = \frac{10}{11}\ x\ 100\% = 90{,}9\%$$

5. Kesimpulan dan Saran

Dari penelitian diatas didapatkan kesimpulan bahwa sistem yang dibangun mampu berjalan dengan baik dengan akurasi 90,9% yang dimana fokus utama yang menjadi nilai utamanya terletak pada proses deteksi kendaraan itu sendiri. Tidak adanya kemampuan aplikasi untuk mengenali objek yang terdapat pada lot parkir menjadi masalah yang menurunkan tingkat akurasi. Jika nantinya sistem akan dikembangkan maka diharapkan sistem dapat mengidentifikasi objek yang ada dan menentukan jenis objek tersebut dan memutuskan lebih baik untuk status dari lot parkir tersebut.

Daftar Pustaka

[1] G. Tesoriere, T. Giuffrè, R. E. Barone, M. A. Morgano, dan S. M. Siniscalchi, "Architecture for parking management in smart cities," *IET Intell. Transp. Syst.*, vol. 8, no. 5, hal. 445–452, 2014.

[2] M. Lutz, *Learning Python*, vol. 78, no. 1. 2007.

[3] A. Zelinsky, "Learning *OpenCV*???*Computer vision* with the *OpenCV* Library," *IEEE Robotics and Automation Magazine*, vol. 16, no. 3. hal. 100, 2009.

[4] *Raspberry Pi* Foundation, "*Raspberry Pi* - Teach, Learn, and Make with *Raspberry Pi*," *Www.Raspberrypi.Org*, 2012. .

[5] S. F. Barrett dan D. J. Pack, "Atmel AVR Microcontroller Primer: Programming and Interfacing," *Synth. Lect. Digit. Circuits Syst.*, vol. 2, no. 1, hal. 1–194, 2007.

[6] R. a. Hummel, B. Kimia, dan S. W. Zucker, "Deblurring *Gaussian Blur*," *Comput. Vision, Graph. Image Process.*, vol. 38, no. 1, hal. 66–80, 1987.

[7] L. Ding dan A. Goshtasby, "On the canny edge detector," *Pattern Recognit.*, vol. 34, no. 3, hal. 721–725, 2001.

[8] J. Howse, *OpenCV Computer vision with Python*. 2013.

[9] T. Kumar dan K. Verma, "A Theory Based on Conversion of RGB image to Gray image," *Int. J. Comput. Appl.*, vol. 7, no. 2, hal. 5–12, 2010.

[10] J. Flusser, T. Suk, S. Farokhi, dan C. Höschl, "Recognition of images degraded by *Gaussian Blur*," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9256, hal. 88–99.