JTERA (Jurnal Teknologi Rekayasa), Vol. 6, No. 2, Desember 2021, Hal. 329-340
Terakreditasi "**Peringkat 3**" oleh Kemenristek/BRIN, Nomor SK: 85/M/KPT/2020
DOI: 10.31544/jtera.v6.i2.2021.329-340
p-ISSN 2548-737X
e-ISSN 2548-8678

Perancangan Aplikasi Pengumpulan Data Statistik Penguasaan Bola Berbasis *Background Subtraction*

Muhammad Yusuf Fadhlan[#], Hertog Nugroho, Luthfiana Adam Maldini
Jurusan Teknik Elektro, Politeknik Negeri Bandung
Jl. Gegerkalong Hilir, Desa Ciwaruga, Kec. Parongpong, Kab. Bandung Barat, Jawa Barat 40559, Indonesia
[#]muhammad.yusuf.ttel410@polban.ac.id

Abstrak

Pada pertandingan sepakbola, penguasaan bola menjadi suatu penilaian dan bahan evaluasi yang menunjukan peluang untuk mencetak. Perhitungan penguasaan bola dilakukan dengan menghitung waktu dari total waktu pertandingan secara manual. Meskipun hanya garis besar penilaian dari pertandingan sepakbola, masih banyak sekali kesalahan dalam menulis statistik pertandingan karena kesalahan manusia, khususnya di Indonesia. Untuk mangatasi masalah tersebut, pada penelitian ini dibuat sebuah aplikasi pengumpulan data statistik penguasaan bola menggunakan *background subtraction*. Beberapa *frame* pada rekaman pertandingan sepakbola diambil untuk dibuat *background* referensi lalu dilakukan proses substraksi dengan *frame* pada saat pertandingan berlangsung. Hasil substraksi tersebut kemudian akan membentuk *rectangular border* yang sisi-sisi nya mengindikasikan objek terluar yang terdeteksi. Posisi *rectangular* tersebut akan dibedakan menjadi lima kondisi yang menentukan statistik penguasaan bola. Hasil pendeteksian tersebut dibandingkan dengan penilaian manusia untuk menilai akurasi pendeteksian. Hasil dari penelitian ini adalah prototipe berupa aplikasi sebagai penerapan metode *background subtraction*. Berdasarkan hasil penelitian yang telah dilakukan diperoleh akurasi pendeteksian sebesar 91%. Sebagian besar kesalahan terjadi karena objek diam terlalu lama yang membuat objek tersebut menjadi *background* dan tidak terdeteksi oleh sistem.

Kata kunci: aplikasi, *background subtraction, machine vision*, penguasaan bola, statistik pertandingan

Abstract

In football matches, possession of the ball becomes an assessment and evaluation material that shows opportunities to score. Calculation of ball possession is done by calculating the time of the total game time manually. Even though it is only an outline of the assessment of football matches, there are still many errors in writing match statistics due to human error, especially in Indonesia. To overcome this problem, in this study an application was made to collect ball possession statistical data using background subtraction. Several frames on the recording of a football match are taken to make a reference background and then a subtraction process is carried out with the frame during the match. The result of the subtraction will then form a rectangular border whose sides indicate the outermost object detected. The rectangular position will be divided into five conditions that determine ball possession statistics. The detection results are compared with human judgments to assess the detection accuracy. The result of this research is a prototype in the form of an application as the application of the background subtraction method. Based on the results of the research that has been done, the detection accuracy is 91%. Most of the errors occur because the object is idle for too long which makes the object in the background and is not detected by the system.

Keywords: *applications, background subtraction, ball possession, machine vision, match statistics*

I. PENDAHULUAN

Sepakbola merupakan salah satu cabang olahraga yang banyak diminati oleh dunia, termasuk Indonesia. Sepakbola aalah permainan yang dimainkan oleh dua regu yang masing-masing regu terdiri dari sebelas orang pemain, yang lazim disebut kesebelasan [1]. Dalam sepakbola, penguasaan bola atau *ball possession* adalah salah satu poin penting untuk memenangkan sebuah pertandingan sepakbola yang menunjukkan peluang untuk mencetak gol. Perhitungan statistik *ball possession* dilakukan dengan menghitung waktu dari total waktu pertandingan secara manual.

Diterima: 9 Agustus 2021; Direvisi: 15 November 2021; Disetujui: 14 Desember 2021

Meskipun hanya garis besar statistik pertandingan sepakbola, di Indonesia khususnya masih banyak sekali kesalahan dalam menulis statistik pertandingan atau seringnya terjadi *human error*, sebagai contoh bagaimana menentukan *ball posession* yang masih sering salah [2].

Pada era digitalisasi seperti ini banyak aktivitas dalam kehidupan sehari-hari yang dilakukan secara daring. Peran telekomunikasi dan *Internet of Things* (IoT) dalam perkembangan teknologi di masa digitalisasi ini sangat penting sekali karena penyebaran informasi juga kini telah berkembang dari komunikasi dengan menggunakan surat, komunikasi dengan kabel seperti telepon, *fiber optic*, dan komunikasi nirkabel seperti komunikasi *cellular* untuk mengirimkan SMS atau telepon, sampai menggunakan internet untuk berkomunikasi.

Berdasarkan beberapa penelitian atau sistem yang telah ada khususnya untuk mendeteksi gerak dapat digunakan dengan beberapa cara yaitu dengan menggunakan metode *Colour Image Processing* [3], *background subtraction* [4], [5], *template matching* [6], menggunakan sensor [7], dan dengan menggunakan metode *Bayes* [8]. Metode *colour image processing* mendeteksi dengan cara mencari nilai *red-green-blue* (RGB) dan dalam mendeteksi pemain dalam pertandingan sepakbola dapat dilakukan dengan cara menyesuaikan RGB dari *background* atau lapangan sepakbola tersebut sehingga pemain yang RGB nya beda dari *background* tersebut dapat terdeteksi tetapi akan sulit terdeteksi bila seragam pemain RBG-nya sama dengan *background*. Metode *template matching* adalah proses suatu citra masukan yang mengandung *template* tertentu dibandingkan dengan *template* pada basis data [9]. Penerapan pada pendeteksian pemain dalam pertandingan sepakbola dapat dilakukan dengan memasukan *template* dari setiap pemain kedalam basis data. Namun, hal tersebut membutuhkan *template* yang banyak dan tentunya akan membutuhkan ruang penyimpanan dan komputasi yang besar. Penggunaan sensor seperti *passive infrared* (PIR) dapat mendeteksi gerakan dan bahkan pancaran suhu tubuh dari objek manusia yang berada dalam ruangan [7]. Tetapi, penerapan menggunakan sensor untuk mendeteksi pemain pada pertandingan sepakbola akan sulit dilakukan karena ukuran lapangan sepak bola yang luas dan jangkauan sensor untuk dapat mendeteksi yang terbatas. Dalam pendeteksian menggunakan metode *Bayes*, dimana metode ini akan bekerja sesuai input yang diberikan berupa gambar yang dijadikan data *training* untuk kamera sehingga gambar atau data baru yang diterima kamera akan terdeteksi jika tidak sesuai dengan *input* [8]. Metode ini dapat diterapkan dalam pendeteksian pemain dalam pertandingan sepakbola. Namun, objek tidak akan terdeteksi jika objek diam atau tidak bergerak. Metode *background subtraction* dapat mendeteksi substraksi pada *background* dengan mengubah citra biner dan menentukan tingkat kepekaan perubahan piksel pada *background*. Metode ini dapat diterapkan dalam pendeteksian pemain dalam pertandingan sepakbola dengan menjadikan beberapa *frame* sebelum *current frame* sebagai *background* sehingga objek yang berada di atas lapang dapat terdeteksi.

Berdasarkan hal tersebut di atas, didapatkan sebuah gagasan untuk membuat sistem yang dapat mendeteksi penguasaan bola yang terjadi dalam rekaman pertandingan sepakbola dengan memanfaatkan metode *background subtraction* yang hasilnya akan disajikan di dalam *website* dalam bentuk grafik. Gagasan ini bertujuan untuk mendeteksi *event* yang terjadi di dalam rekaman pertandingan sepakbola, serta bermanfaat untuk mengurangi kesalahan penilaian penguasaan bola atau *ball possession* dalam pertandingan sepakbola yang terjadi karena *human error* dan untuk memudahkan dalam menilai penguasaan bola tersebut.

II. METODE PENELITIAN

Perancangan

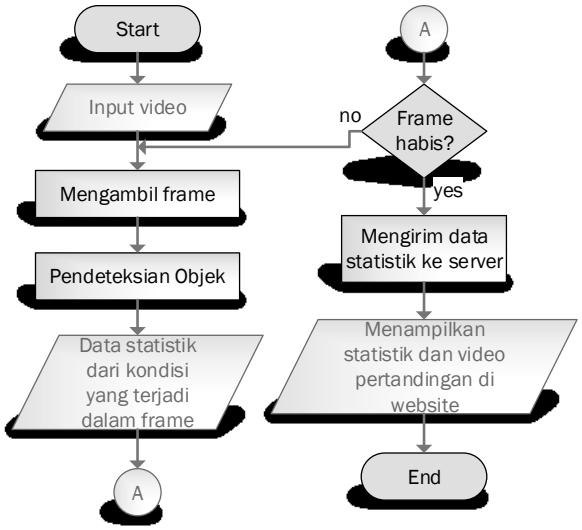
Perancangan sistem dari aplikasi yang dibuat adalah sebagai berikut:

1. Video pertandingan sepakbola dimasukan ke aplikasi *website* sebagai *input*. Dari video tersebut pada program diambil beberapa *frame* untuk dibuat sebagai *background.*
2. *Frame* selanjutnya pada video akan dikurangi (*subtrack*) dengan *background* yang hasilnya akan berupa *frame* dengan warna hitam dan putih. Warna hitam berarti intensitas warna di *frame* dengan *background* sama atau hasil *subtrack* bernilai mendekati 0, 0, 0 (hitam). Sedangkan warna putih berarti intensitas warna di *frame* dengan *background* berbeda atau hasilnya tidak mendekati 0, 0, 0.
3. Selanjutnya akan dibuat *rectangle* yang sisi-sisi nya berdasarkan objek terluar yang terdeteksi (pemain) kecuali kiper dan hakim garis. Wilayah pendeteksian dibuat dinamis agar kiper dan hakim garis tidak ikut terdeteksi.
4. Berdasarkan ukuran dan posisi *rectangle* tersebut ditentukan 5 buah kondisi. Kondisi-kondisi tersebut adalah:

a) Serangan intens di tim A
b) Serangan normal di tim A
c) Kondisi normal (*netral*)
d) Serangan normal di tim B
e) Serangan intens di tim B

5. Dari kondisi tersebut akan dihitung lamanya kondisi tersebut dan pada waktu kapan kondisi tersebut terjadi yang kemudian akan dibuat statistik penguasaan bola dengan membandingkannya dengan total waktu pertandingan tersebut.
6. Proses akan kembali ke poin 1 sebelum video pertandingan sepakbola selesai.
7. Statistik tersebut akan disimpan di *server* dan diolah menjadi grafik yang ditampilkan di *website*.

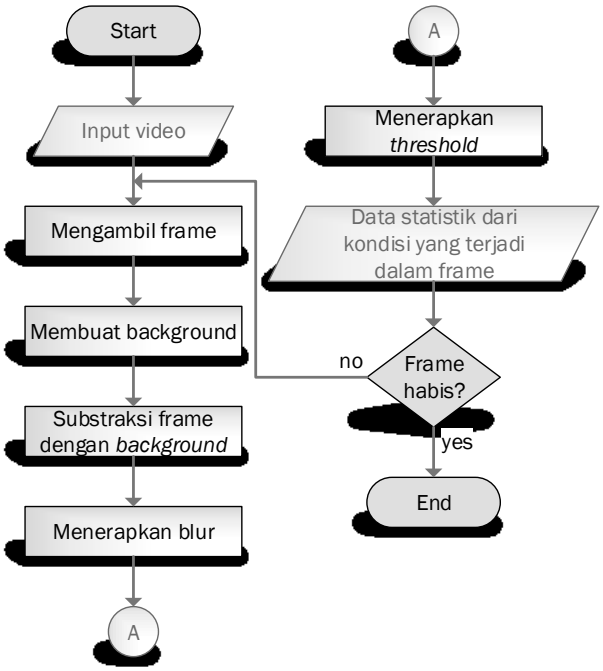
Gambar 1 merupakan diagram alir sistem keseluruhan dari penelitian ini yang memperlihatkan urutan bekerjanya sistem untuk mendapatkan statistik dalam pertandingan yang diawali oleh input data berupa video yang diolah setiap *frame*-nya. Pada gambar tersebut, data yang diolah dari *frame* ialah data terstruktur berupa intensitas warna atau RGB yang berdimensi sesuai dengan dimensi piksel dari *frame* tersebut. Dari data tersebut diterapkan metode *background subtraction* untuk pendeteksian objek berupa pemain di lapangan kecuali kiper dan hakim garis. Koordinat dari objek yang terdeteksi tersebut diambil untuk menentukan *event* yang terjadi pada frame tersebut. Dari data *event* yang terjadi tersebut akan diolah dan dikirim ke *server* yang kemudia akan dijadikan grafik yang ditampilkan dalam *website*.



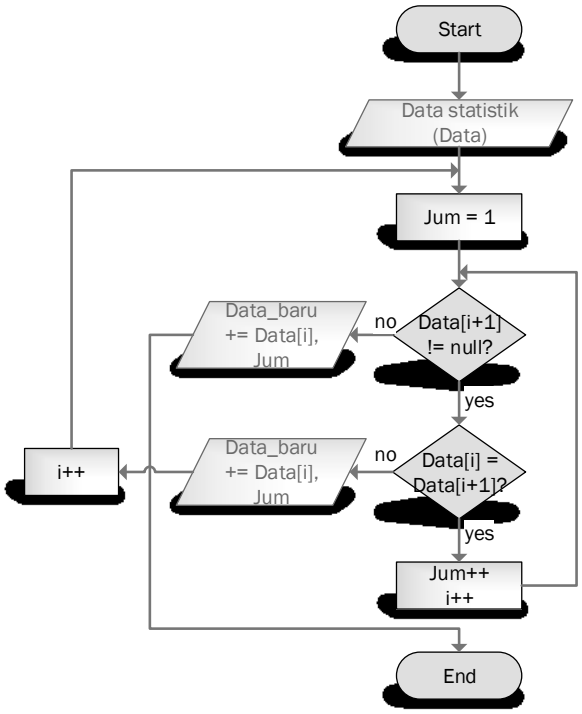
Gambar 1. Diagram alir sistem keseluruhan

Urutan proses atau algoritma dalam melakukan deteksi objek dapat dilihat pada Gambar 2. Diawali dengan melakukan *input* data berupa video yang akan diolah setiap *frame*-nya satu per satu. Gabungan dari beberapa *frame* akan diolah menjadi *background* karena metode *background subtraction* yang diterapkan membutuhkan *background* yang kemudian akan dikurangi dengan *frame* selanjutnya untuk mendapatkan *foreground mask* yang berwarna hitam dan putih. *Foreground mask* tersebut dibuat blur dan diberi *threshold* untuk mengurangi *noise* yang terjadi karena objek kecil yang tidak diinginkan ikut terdeteksi. Setelah itu dapat ditentukan *event* yang terjadi dalam *frame* dengan membuat *rectangle* dari objek terluar yang terdeteksi (selain kiper dan hakim garis) dilihat dari koordinat *rectangle* tersebut. *Event* tersebut dijadikan sebuah data yang dikumpulkan dari *frame* awal sampai *frame* akhir.

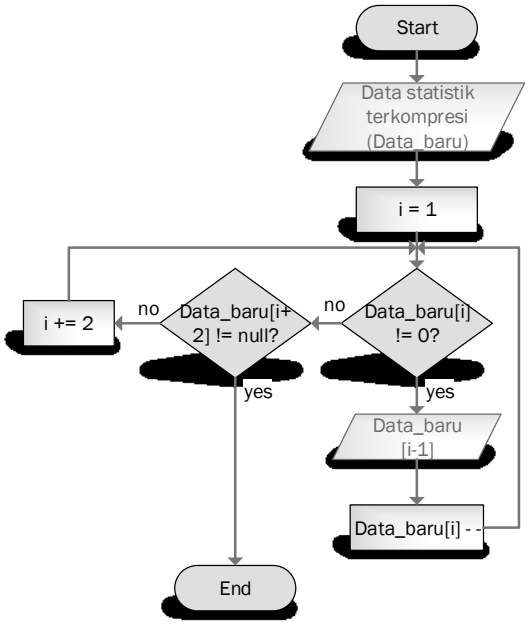
Algoritma dari kompresi data yang dilakukan dalam penelitian ini dapat dilihat pada Gambar 3. Kompersi yang diterapkan adalah kompresi *Run Length Encoding* (RLE), dimana data sejenis yang berurutan akan diubah menjadi 2 karakter (atau lebih) yaitu karakter itu sendiri dan jumlah karakter tersebut dalam sebuah deret. Sebagai contoh, data "AAAABBBAAA" akan diubah menjadi "A4B3A3". Data tersebut di-*sweep* satu per satu dari indeks terkecil hingga indeks terbesar.



Gambar 2. Diagram Alir Pendeteksian Objek

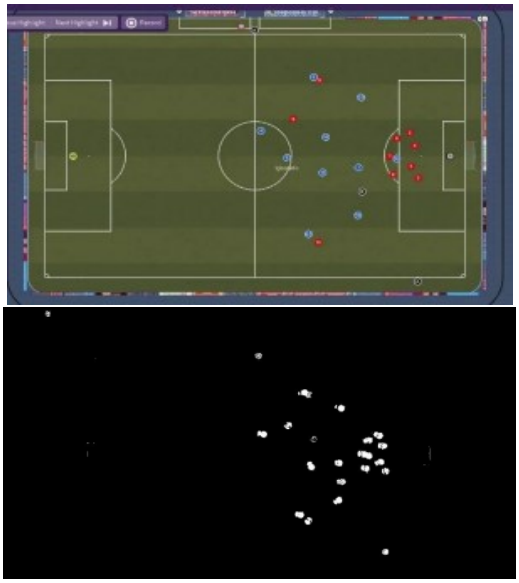


Gambar 3. Diagram alir kompresi data



Gambar 4. Diagram Alir Dekompresi Data

Diagram alir atau algoritma dekompresi data yang digunakan dalam penelitian ini dapat dilihat pada Gambar 4. Diawali dengan *input* data yang telah dikompresi berupa list dengan tipe data *char*. Indeks genap menunjukan sebuah karakter dan indeks ganjil menunjukan jumlah dari karakter tersebut dalam sebuah deret. Proses dekompresi ini dilakukan dengan mengubah 2 indeks menjadi satu deret. Sebagai contoh, data [“A”, “2”, “B”, “10”, “A”, “3”] akan diubah menjadi “AABBBBBBB BBBAAA”. Data tersebut di-*sweep* secara berurutan dari indeks terkecil hingga indeks terbesar.



Gambar 5. *Background* lapangan dengan objek pemain yang bergerak (atas) dan hasil dari perbandingan objek bergerak dan *background* (bawah)

B. Metode Background Substraction

Background subtraction yang juga dikenal sebagai *foreground detection* adalah salah satu teknik pada bidang pengolahan citra dan *computer vision* yang bertujuan untuk mendeteksi/mengambil *foreground* dari *background* untuk diproses lebih lanjut. *Background* sangat penting dalam proses pelacakan objek, karena untuk mendeteksi adanya gerakan perlu dipisahkan antara objek yang bergerak dan objek yang diam [4].

Dari definisi di atas maka dapat disimpulkan bahwa *background subtraction* merupakan metode yang umumnya digunakan untuk mendeteksi objek yang bergerak pada video dari kamera yang diam atau statis (*stationary camera*). Proses deteksi objek bergerak (*motion detection*) dengan metode *background subtraction* didasarkan pada perbandingan antara *background* referensi dengan *frame*. Atau secara numerik merupakan proses pengurangan nilai setiap *pixel* dari *frame* dengan *background.* Gambar 5 menunjukkan ilustrasi *background substraction* pada *frame* pertandingan sepak bola.

C. Metode Analisis

Data yang diuji dalam pengujian ini berupa video pertandingan sepakbola dari *videogame* FIFA Manager 2019 yang diproses menggunakan metode *background subtraction* dengan parameter sebagai berikut:

1. Ketepatan (*Accuracy*)

Ketepatan sistem dalam melakukan pendeteksian *event* diuji dengan cara membandingkan penilaian sistem terhadap *event* yang terjadi pada *frame* dengan penilaian manusia (*responden*) yang dikumpulkan melalui kuisioner yang disebarkan. Jika penilaian sistem dengan penilaian *responden* sama akan bernilai 1, sedangkan jika penilaian sistem dengan penilaian *responden* berbeda akan bernilai 0. Ketepatan atau *accuracy* ini dapat dihitung menggunakan persamaan 1 berikut.

$$Accuracy = \frac{\sum_{i=1}^{t} Ri}{t\ x\ n}\ X\ 100\% \quad (1)$$

Keterangan:

R = jumlah penilaian responden dengan sistem yang sesuai

n = jumlah responden

t = jumlah frame yang diuji

2. Waktu Komputasi

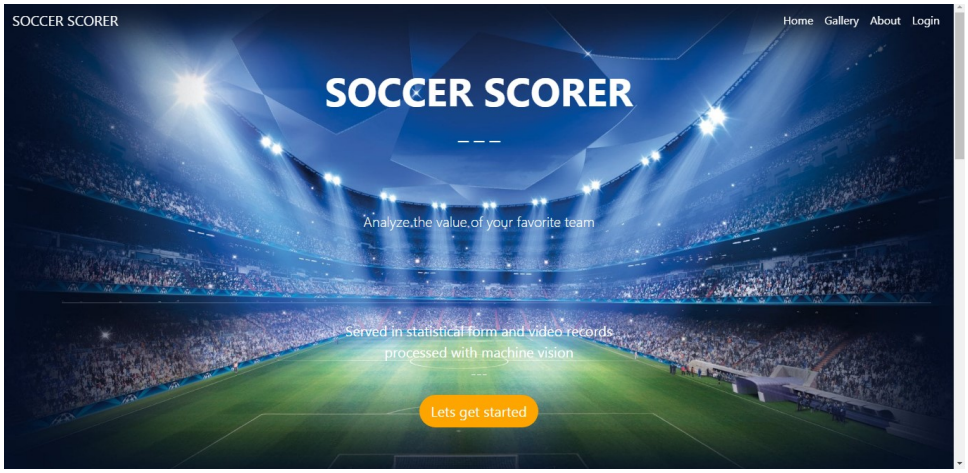
Waktu komputasi diuji dengan cara menjalankan sistem dengan spesifikasi yang berbeda yaitu dengan menggunakan laptop dan menggunakan Google Colab dari data uji yang sama. Data uji tersebut berupa video dengan durasi 10 menit, 25 menit, dan 35 menit 32 detik. Dan dari waktu komputasi tersebut dapat dihitung *frame per second* (fps) komputasi tersebut. *Frame per second* tersebut dapat dihitung dengan persamaan 2 berikut.

$$fps = \frac{jumlah\ frame}{waktu\ komputasi\ (s)} \quad (2)$$

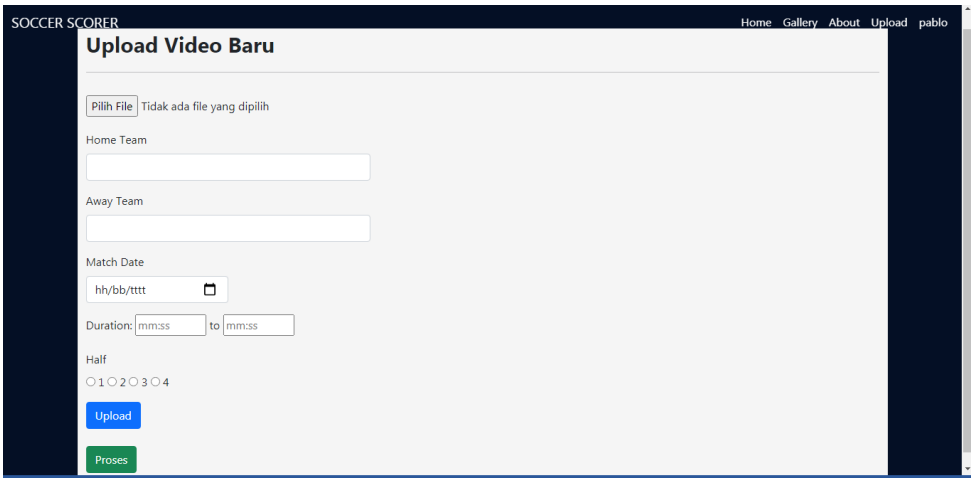
III. HASIL DAN PEMBAHASAN

A. Realisasi Sistem

Gambar 6 menunjukan tampilan depan *website*. Proses *input* yang berupa video pertandingan sepakbola dilakukan melalui *website*. Untuk dapat melakukan proses *upload* harus melakukan login terlebih dahulu pada *website*, sehingga tidak sembarang orang yang dapat melakukan *upload* pada *website* tersebut.

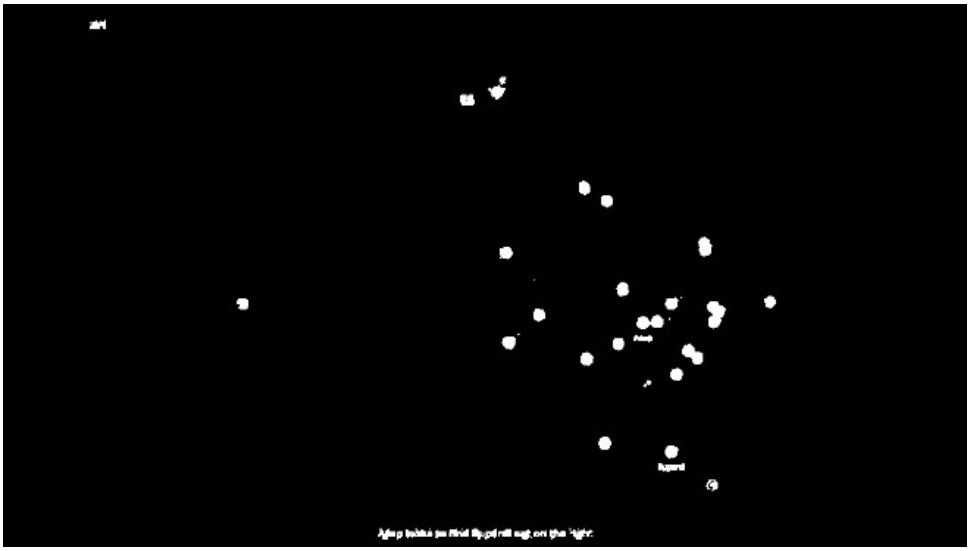


Gambar 6. Halaman depan *website*



Gambar 7. Halaman *upload website*

Gambar 8. ***Frame input***

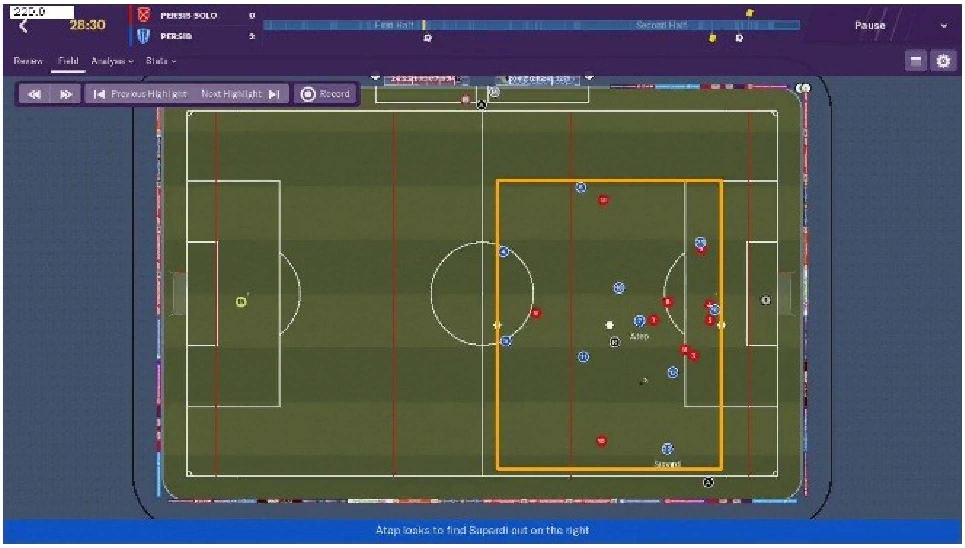


Gambar 9. ***Foreground mask***

Di halaman upload.php seperti terlihat pada Gambar 7 terdapat form untuk memilih file video yang mau di-*upload*, form nama tim *home* dan tim *away*, tanggal pertandingan, waktu di video tersebut dalam pertandingan, dan babak, karena pada penelitian ini dari satu pertandingan akan dicuplik menjadi beberapa video. Tombol upload akan mengirim data yang telah diisi di form ke tabel input_video pada database yang dikirim dengan menggunakan *method $_POST* untuk mengambil data yang dimasukan pada form. Data tersebut akan diolah sesuai dengan format yang diterapkan dalam program keseluruhan sebelum dikirim ke database. File video yang di-*upload* akan disimpan dalam satu folder yaitu folder upload. Jika tombol upload di klik, video yang baru di-*upload* akan diproses oleh program yang telah dibuat.

Frame diambil satu per-satu dari video yang diinputkan seperti terlihat pada Gambar 8 yang kemudian akan dikurangi dengan *background* untuk mendapatkan *foreground mask*. *Background* bersifat dinamis, tergantung pada *frame* asli dari video dan dibutuhkan 5 buah frame untuk membuat *background* tersebut. Oleh karena itu, *foreground mask* akan berupa *frame* putih (*blank*) jika *background* belum terbentuk. Jika *background* sudah terbentuk, *foreground mask* akan terlihat seperti pada Gambar 9.

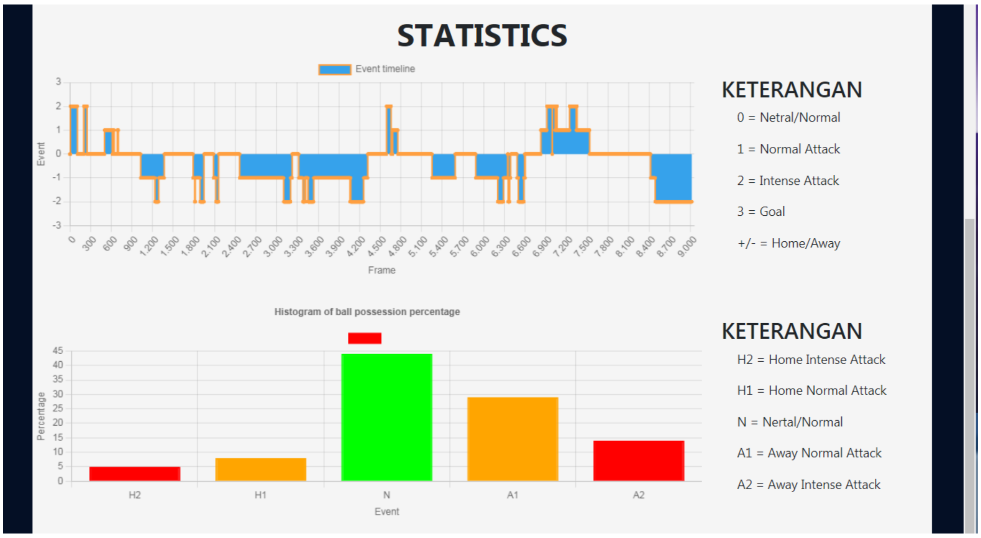
Dari *foreground mask* pada Gambar 9 tersebut, sistem membuat sebuah *bounding box* pada *frame* input yang sisi-nya berdasarkan objek terluar yang terdeteksi selain kiper dan hakim garis seperti terlihat pada Gambar 10. Dari posisi *bounding box* tersebut, sistem menentukan *event* yang terjadi pada frame tersebut. *Event* yang ditentukan oleh sistem pada setiap *frame*-nya disimpan di dalam variabel serta akan diolah dan dikompresi menjadi data statistik seperti terlihat pada Gambar 11.



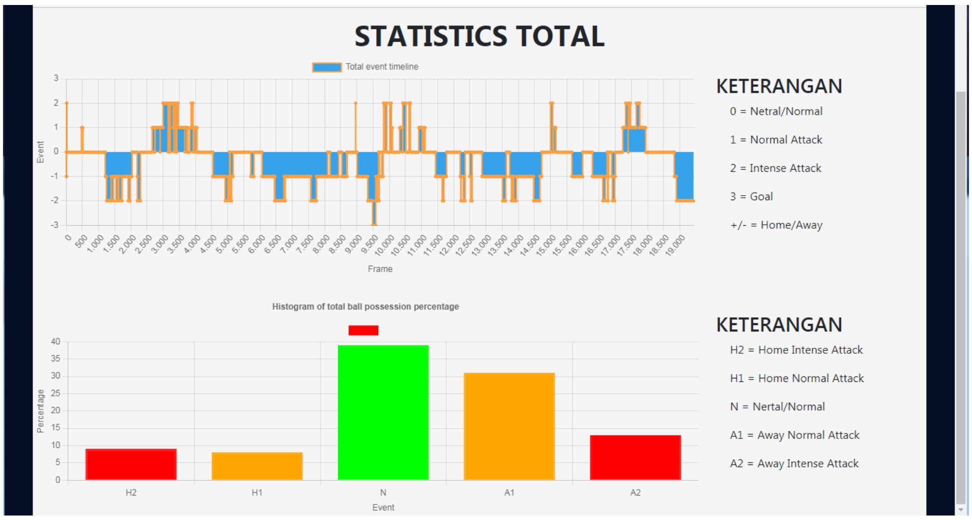
Gambar 10. *Frame output*

```
event list         :  CCCCDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
encoded event list :  C 4 D 52
```

Gambar 11. Data statistik



Gambar 12. Grafik statistik video



Gambar 13. Grafik statistik pertandingan total

Tabel 1. Data hasil pengujian *accuracy*

NO	FRAME	PENILAIAN						TRUE	FALSE
		SISTEM	R1	R2	R3	R4	R5		
1	30	B	B	A	B	B	C	3	2
2	60	C	C	C	C	C	C	5	0
3	90	C	C	C	C	C	C	5	0
4	120	C	C	C	B	C	C	4	1
5	150	C	C	C	C	B	C	4	1
6	180	B	B	B	B	B	B	5	0
7	210	B	B	B	B	B	B	5	0
8	240	B	B	B	B	B	B	5	0
9	270	A	A	A	B	A	A	4	1
10	300	A	A	A	B	A	A	4	1
11	330	A	A	A	A	A	A	5	0
12	360	B	B	B	B	B	B	5	0
13	390	A	A	A	B	A	A	4	1
14	420	B	B	B	B	B	A	4	1
15	450	B	B	B	C	A	A	2	3
16	480	B	B	B	B	A	B	4	1
17	510	B	B	B	B	B	B	5	0
18	540	B	B	B	B	B	A	4	1
19	570	B	B	B	B	E	A	3	2
20	600	A	A	A	A	A	A	5	0
21	630	A	A	A	A	A	A	5	0
22	660	A	A	A	A	A	A	5	0
23	690	B	B	B	B	B	B	5	0
24	720	B	B	B	B	B	B	5	0
25	750	B	B	B	B	B	B	5	0
26	780	B	B	B	B	B	B	5	0
27	810	B	B	B	B	B	B	5	0
28	840	B	B	B	B	B	B	5	0
29	870	C	C	C	C	C	C	5	0
30	900	C	C	C	C	C	C	5	0
31	930	C	C	C	C	C	C	5	0
32	960	C	C	C	C	C	C	5	0
33	990	C	C	C	C	C	C	5	0
34	1020	C	C	C	C	C	C	5	0
35	1050	C	C	C	C	C	C	5	0
36	1080	C	C	C	C	C	C	5	0
37	1110	C	C	C	C	C	C	5	0
38	1140	C	C	C	C	C	C	5	0
39	1170	C	C	C	C	C	C	5	0
40	1200	C	C	C	C	C	C	5	0
41	1230	C	C	C	C	C	C	5	0
42	1260	C	C	C	D	C	C	4	1
43	1290	C	C	C	D	C	C	4	1
44	1320	C	C	C	C	C	C	5	0
45	1350	D	D	D	D	D	D	5	0
46	1380	D	D	D	D	D	D	5	0
47	1410	D	D	D	D	D	D	5	0
48	1440	D	D	D	D	D	D	5	0
49	1470	D	D	D	D	D	D	5	0
50	1500	D	D	D	D	E	D	4	1
51	1530	D	B	D	D	D	D	4	1
52	1560	D	D	D	D	D	D	5	0
53	1590	D	E	D	D	D	D	4	1
54	1620	D	E	D	D	D	D	4	1
55	1650	D	E	D	D	D	D	4	1
56	1680	D	D	D	D	D	D	5	0
57	1710	D	D	D	E	D	D	4	1
58	1740	E	E	E	E	E	E	5	0
59	1770	E	E	E	E	E	E	5	0
60	1800	D	E	D	E	D	D	3	2
TOTAL								275	25
ACCURACY								91%	

Setelah *frame* dari video input habis, data statistik dikirim ke database. Di dalam *website*, data statistik tersebut akan diolah dan didekompresi yang kemudian akan disajikan dalam bentuk grafik *event timeline* dan persentase penguasaan bola seperti terlihat pada Gambar 12 dan Gambar 13. Gambar 12 menunjukan grafik statistik selama video yang diinputkan, sedangkan Gambar 13 menunjukan akumulasi grafik statistik dari beberapa video dalam pertandingan yang sama.

B. Pengujian

1. Ketepatan (*Accuracy*)

Data hasil pengujian sample video input dengan durasi 1 menit yang diambil 1 frame per detiknya serta dinilai oleh sistem dan *responden* dapat dilihat pada Tabel 1 dengan keterangan:

Frame = urutan frame dalam video.
Sistem = penilaian *event* yang terjadi oleh sistem (A, B, C, D atau E).
Rn = penilaian *event* yang terjadi responden n (A, B, C, D atau E).
True = jumlah penilaian sistem yang sesuai dengan penilaian responden.
False = jumlah penilaian sistem yang tidak sesuai dengan penilaian responden.

Berdasarkan pada rumus 1 maka dari data hasil pengujian pada tabel 1 dapat dihitung ketepatan atau *accuracy* keseluruhan dari sistem yang telah dibuat. *Accuracy* sistem pada setiap event juga dapat dihitung menggunakan rumus 1 yang dapat dilihat pada Tabel 2.

Berdasarkan pengujian *accuracy* dengan membandingkan penilaian sistem dengan penilaian 5 *responden* yang dilakukan melalui kuisioner di Google Form, didapatkan *accuracy* sebesar 91% seperti terlihat pada tabel 1. Dan *error* pada setiap *event*-nya dapat dilihat pada tabel 2. *Error* tersebut terjadi karena beberapa hal, yang pertama yaitu *error* karena *background*. Pada *library* OpenCV khususnya pada fungsi *createBackground SubtractorKNN()*, *background* terbentuk dari akumulasi *weighted average* dari beberapa frame sebelumnya (pada penelitian ini digunakan 5 frame untuk membuat *background*) yang membuat *background* tersebut bersifat dinamis. Dimana, algoritma dari metode *background subtraction* itu sendiri merupakan proses substraksi frame yang sekarang dengan *background* yang telah terbentuk. Hal tersebut menyebabkan beberapa frame di awal tidak menghasilkan *foreground mask* karena *background* belum terbentuk.

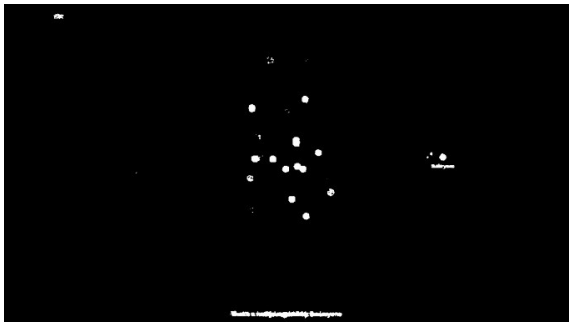
Tabel 2. *Accuracy* sistem setiap *event*

EVENT	TRUE	FALSE	ACCURACY
A	32	3	91,4%
B	75	10	88,2%
C	96	4	96%
D	62	8	88,6%
E	10	0	100%

Hal tersebut juga menyebabkan objek yang terdeteksi hanya objek bergerak saja, sehingga objek yang diam dalam beberapa frame tidak akan terdeteksi. *Error* tersebut biasanya terjadi pada kiper, dimana algoritma pada penelitian ini diterapkan sebuah *Region of Image* (ROI) dinamis yang mendeteksi pemain saja berdasarkan penyempitan ROI dari objek yang terdeteksi termasuk kiper sebesar 20 piksel. Jika hasil deteksi baik, *rectangle* akan terbentuk dari sisi terluar objek selain kiper. Sedangkan jika hasil deteksi terdapat *error*, *rectangle* di sisi kiri atau kanan akan menyempit sebesar 20 piksel jika kiper tidak terdeteksi. Atau kiper akan ikut terdeteksi juga jika kiper yang sudah diam tersebut mulai bergerak yang meninggalkan jejak dari *error* karena *background*.

Hal selanjutnya yang dapat menyebabkan *error* adalah karena nama pemain yang muncul pada *videogame*. *Error* sering terjadi saat kiper melakukan tendangan gawang yang membuat nama nya muncul dalam *videogame*. Ukuran dari nama tersebut melebihi 20 piksel sehingga nama dari kiper tersebut akan terdeteksi meskipun sudah dilakukan penyempitan sebesar 20 piksel seperti terlihat pada Gambar 14 dan Gambar 15.

Gambar 14. *Error* karena nama kiper (*current frame*)



Gambar 15. *Error* karena nama kiper (*foreground*)

Hal terakhir yang dapat menyebabkan *error* adalah karena *noise*. *Noise* biasanya muncul karena bayangan dan tidak sempurnanya pembuatan *background*. Namun *noise* yang berukuran kecil dapat diatasi dengan proses *blur* dan *thresholding*.

2. Waktu Komputasi

Dalam melakukan pengujian waktu komputasi keseluruhan yang dibutuhkan sistem. Dilakukan dengan 2 *platform* yang berbeda, yaitu dengan menggunakan laptop dan Google Colab yang memiliki spesifikasi berbeda. Hasil pengujian waktu komputasi dapat dilihat pada Gambar 16 dan Gambar 17.

Berdasarkan data pada Gambar 16 dan Gambar 17, dapat dibuat tabel seperti terlihat pada Tabel 3. Dengan manggunakan persamaan 2 maka dari data hasil pengujian pada tabel 3 dapat dihitung banyaknya frame yang dapat diproses sistem dalam satu detik atau *Frames per Second* (FPS) dari setiap *sample* yang diuji dan *average* FPS dari sistem yang diperlihatkan pada Tabel 4 dan Tabel 5.

```
Waktu eksekusi Sample 1:
0 jam 28 menit 20.53426504135132 detik
Total frame: 18005

Waktu eksekusi Sample 2:
1 jam 5 menit 54.474183082580566 detik
Total frame: 45006

Waktu eksekusi Sample 3:
1 jam 37 menit 33.75661301612854 detik
Total frame: 63980
```

Gambar 16. Waktu komputasi pada laptop

```
Waktu Eksekusi Sample 1 di Google Colab:
0 jam 10 menit 51.40208172798157 detik
Total frame: 18005

Waktu Eksekusi Sample 2 di Google Colab:
0 jam 26 menit 51.81241512298584 detik
Total frame: 45006

Waktu Eksekusi Sample 3 di Google Colab:
0 jam 38 menit 24.692660331726074 detik
Total frame: 63980
```

Gabar 17. Waktu komputasi pada Google Colab

Tabel 3. Data hasil pengujian waktu komputasi

VIDEO	DURASI	TOTAL FRAME	WAKTU KOMPUTASI	
			LAPTOP	GOOGLE COLAB
Sample 1	10 menit	18005	28 menit 21 detik	10 menit 51 detik
Sample 2	25 menit	45006	1 jam 5 menit 54 detk	26 menit 52 detik
Sample 3	35 menit 32 detik	63980	1 jam 37 menit 34 detik	38 menit 25 detik

Tabel 4. FPS dari sistem

VIDEO	FRAME RATE	
	LAPTOP	GOOGLE COLAB
Sample 1	10,6 fps	27,6 fps
Sample 2	11,4 fps	27,9 fps
Sample 3	10,9 fps	27,6 fps

Tabel 5. *Average* FPS dari sistem

PLATFORM	AVERAGE FRAME RATE
LAPTOP	11 fps
GOOGLE COLAB	27,2 fps

Pada parameter kedua ini, terdapat 2 buah *platform* dengan spesifikasi yang berbeda agar menjadi pertimbangan sistem selanjutnya yang mengacu pada penelitian ini dalam hal spesifikasi yang digunakan. Pengujian ini dilakukan perhitungan waktu komputasi oleh laptop dan Google Colab dari 3 sampel video yang sama. Dari kedua *platform* yang digunakan, *platform* Google Colab jauh lebih unggul dalam hal komputasi, karena menggunakan GPU yang lebih baik untuk mengolah citra. Jika dilihat dari *average frame rate* pada Tabel 5, Google Colab dapat memproses 27,2 frame dalam waktu 1 detik. Sedangkan laptop hanya dapat memproses 11 frame dalam waktu 1 detik. Hal tersebut membuktikan bahwa Google Colab unggul 147,3% dalam hal komputasi.

Pada penelitian ini, parameter waktu yang diproses untuk memproses frame dari video dengan FPS tertentu untuk mendeteksi pemain dalam pertandingan sepakbola sangatlah penting. Berdasarkan percobaan yang dilakukan, waktu yang dibutuhkan sistem ini pada laptop untuk video berdurasi 25 menit adalah 1 jam 5 menit 54 detik. Sedangkan pada Google Colab untuk video dengan durasi 25 menit dibutuhkan waktu 26 menit 52 detik. Berdasarkan hasil tersebut dapat dianalisis bahwa *platform* Google Colab yang memiliki spesifikasi lebih baik daripada laptop dapat mengeksekusi sistem lebih cepat dan memiliki *average frame rate* 27,2 fps yang mendekati *frame rate* video inputnya yaitu 30 fps. Hal ini menjadi sebuah pertimbangan yang baik dalam menggunakan *proccessor* yang baik apabila nantinya akan dilakukan pendeteksian secara *real-time* menggunakan sistem ini.

IV. KESIMPULAN

Berdasarkan hasil pengujian dan analisis dari penelitian ini dapat diambil kesimpulan bahwa algoritma deteksi yang diterapkan dalam sistem ini memiliki akurasi penilaian *event* yang terjadi dalam pertandingan sepakbola sebesar 91% dari sampel

video berdurasi 1 menit yang diuji dengan jumlah *responden* adalah 5 orang. Namun pada pendeteksian dalam sistem ini masih terdapat *error* yang awam terjadi karena kesalahan pada *background.* Dari pengujian waktu komputasi yang dilakukan oleh laptop dan Google Colab dengan 3 buah sampel video input 30 fps, dapat diketahui bahwa laptop dapat mengolah 11 frame dalam 1 detik. Sedangkan Google Colab dapat mengolah 27,2 frame dalam 1 detik. Dengan kata lain, Google Colab mampu mengolah frame 147,3% lebih cepat daripada laptop. Jika dilihat dari spesifikasi *proccessor*, Google Colab lebih unggul dari laptop. Dari hal tersebut dapat disimpulkan bahwa semakin baik *proccessor* yang digunakan, maka semakin cepat pemrosesan yang dilakukan. Untuk pengembangan selanjutnya video uji pertandingan yang sebenarnya dapat digunakan untuk mendapatkan hasil yang lebih akurat. Algoritma pendeteksian pergerakan bola dapat ditambahkan untuk mendapatkan hasil statistik yang lebih variatif, dibandingkan hanya menghitung *ball possession* pada sebuah pertandingan.

REFERENSI

[1] Sarumpaet, Permainan Besar, Departemen Pendidikan dan Kebudayaan, 1992.

[2] A. A. Musthofa, "PENGEMBANGAN BUKU PANDUAN MENCATAT STATISTIK PERTANDINGAN SEPAKBOLA," 2020. [Online]. Available: http://eprints.uny.ac.id/id/eprint/68710. [Accessed 21 April 2021].

[3] M. Nasir, A. and M. Arhami, "Sistem Pendeteksi Dini Kebakaran Menggunakan Colour Image Processing dan Raspberry Pi," *Prosiding Seminar Nasional Politeknik Negeri Lhokseumawe,* vol. 2, no. 1, pp. 226-231, 2018.

[4] N. N. Putri, "APLIKASI PENDETEKSI OBJEK BERGERAK PADA IMAGE SEQUENCE DENGAN METODE BACKGROUND SUBSTRACTION," *Jurnal Teknologi Rekayasa,* vol. 21, no. 3, pp. 162-172, 2016.

[5] D. Prihatmoko and A. K. Zyen, "SISTEM PENDETEKSI GERAK BERBASIS WEB MENGGUNAKAN METODE BACKGROUND SUBSTRACTION," *Jurnal DISPROTEK,* vol. 6, no. 1, pp. 20-25, 2015.

[6] H. S. Hutasoit and I. Krisnadi, "SISTEM PENDETEKSI GERAK BERBASIS TELEGRAM PADA GEDUNG MENGGUNAKAN METODE TEMPLATE MATCHING," 2014. [Online]. Available: https://www.academia.edu/44257856. [Accessed 21 April 2021].

[7] R. P. Mandala Putra, F. Thalib and M. Lamsani, "PENGAMANAN RUANG BRANKAS MENGGGUNAKAN KAMERA PENDETEKSI GERAK BERBASIS RASPBERRY PI DENGAN PENYIMPANAN OTOMATIS KE GMAIL DAN DROPBOX," *Jurnal Informatika dan Komputer,* vol. 21, no. 3, pp. 37-44, 2016.

[8] E. R. Siagian, "IMPLEMENTASI METODE BAYES KEAMANAN GEDUNG UNTUK MENDETEKSI GERAK," *Jurnal INFOTEK,* vol. 1, no. 1, pp. 29-32, 2016.

[9] D. A. Irawati, "PENGEMBANGAN APLIKASI PENGENALAN PLAT NOMOR KENDARAAN RODA DUA PADA AREA PARKIR," in *Prosiding Seminar Nasional Sains dan Teknologi*, Jakarta, 2015.

[10] A. Christian, S. Hesinto and A. , "Rancang Bangun Website Sekolah Dengan Menggunakan Framework Bootstrap (Studi Kasus SMP Negeri 6 Prabumulih)," *Jurnal SISFOKOM,* vol. 7, no. 1, pp. 22-27, 2018.

[11] I. Sholikhat, H. E. Rosyadi and D. M. Putri, "Rancang Bangun Website yang Berorientasi Video Sebagai Sarana Media Informasi di SMK YP 17-2 Malang," *ILKOMNIKA: Journal of Computer Science and Applied Informatics,* vol. 1, no. 1, pp. 15-23, 2019.

[12] A. W. Kusuma, M. Sarosa and L. D. Mustafa, "RANCANG BANGUN DAN ANALISA MEDIA VIDEO STREAMING PADA JARINGAN 3G DAN 4G," *Jurnal JARTEL,* vol. 7, no. 2, pp. 14-22, 2018.