RANCANG BANGUN APLIKASI SIMULASI KEGIATAN P3K DENGAN ANIMASI 2D MENGGUNAKAN ALGORITMA *BOYER-MOORE*

Khaerul Manaf [1)], Ruqi Antika[2)]

Jurusan Sistem Informatika, Fakultas Teknik Universitas Sangga Buana
Jl.PHH Mustopa No.68 Kota Bandung 40124
khaerul.manaf@usbypkp.ac.id[1)]

Jurusan Teknik Informatika, Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Gunung Djati Bandung
Jl. A.H. Nasution No. 105 Kota Bandung 40614
ruqi.antika@student.uinsgd.ac.id[2)]

Abstrak

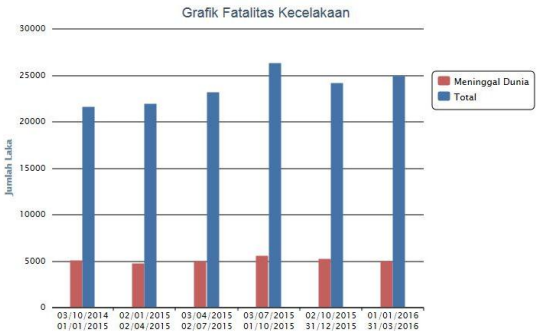
Kecelakaan merupakan hal yang lumrah yang dapat terjadi dimanapun dan kapanpun. Bahkan di Indonesia tingkat kecelakaan mengalami peningkatan yang cukup tinggi setiap tahunnya. Namun, bagaimana jika kecelakaan tersebut terjadi di depan mata kita sendiri dan tidak ada orang lain yang berada di lokasi kejadian, maka pada saat itulah kecakapan dalam memberikan pertolongan pertama pada kecelakaan(P3K) sangat dibutuhkan. Meski begitu, pengetahuan tentang pertolongan pertama pada kecelakaan sangat minim dimiliki oleh masyarakat Indonesia. Untuk membantu mempermudah masyarakat dalam memahami ilmu tentang pertolongan pertama pada kecelakaan maka dibuatnya aplikasi simulasi kegiatan P3K ini dengan berbasis *mobile* agar lebih mudah diakses juga lebih praktis dan menyajikan simulasi kegiatan P3K dengan animasi 2D agar lebih mudah dipahami. Untuk meningkatkan efesiensi aplikasi ini juga dilengkapi dengan fitur pencarian yang menerapkan algoritma *boyer-moore* untuk mencocokkan *string* yang di*input*kan. Dari hasil penelitian ternyata algoritma ini dapat diterapkan pada aplikasi dan cukup akurat dan cepat dalam melakukan pencocokkan *string*.

.

Kata kunci : Android, *Boyer-Moore*, P3K, Animasi 2D

1. PENDAHULUAN

Kecelakaan merupakan hal yang tidak pernah diduga, yang entah kapan dan dimananya terjadi. Kecelakaan dapat terjadi diruang tertutup ataupun terbuka. Hal tersebut tampak di negara kita Indonesia yang telah dikabarkan bahwa angka kecelakaan di Indonesia tergolong tinggi.



Gambar 1.1 Grafik Fatalitas Kecelakaan [1]

Melihat dari tingginya tingkat kecelakaan yang terjadi di negara kita, mengharuskannya kepada setiap masyarakat lebih waspada juga dapat lebih berhati-hati terhadap kecelakaan yang terjadi. Selain itu telah dinyatakan dalam pasal 531 Kitab Undang-undang Hukum Pidana (KUHP) yang menyebutkan bahwa "Barangsiapa menyaksikan sendiri ada orang di dalam keadaan bahaya maut, lalai memberikan atau mengadakan pertolongan kepadanya sedang pertolongan itu dapat diberikannya atau diadakannya dengan tidak akan menguatirkan, bahwa ia sendiri atau orang lain akan kena bahaya dihukum kurungan selama-lamanya tiga bulan atau denda sebanyak-banyaknya Rp. 4.500,-. Jika orang yang perlu ditolong itu mati, diancam dengan : KUHP 45, 165, 187, 304s, 478, 535, 566." Merujuk pada undang-undang tersebut maka perlunya pemahaman tentang materi pertolongan pertama

pada kecelakaan. Mengingat begitu banyaknya kecelakaan yang terjadi di Indonesia maka kesadaran akan tolong menolong akan menjadi percuma jika warga tidak memahami cara untuk menolong korban-korban tersebut.

Hal itu terkuak dalam penelitian yang dilakukan oleh Banu Setyo Adi pada mahasiswa Program Kelanjutan Studi D2 Pendidikan Guru Sekolah Dasar (PGSD) di Klaten tentang pemahamannya terhadap P3K dari 38 responden yang ia teliti sebanyak 10 responden atau 26,32% yang memahami tentang tindakan Pertolongan Pertama Pada Kecelakaan sedangkan 28 responden lainnya atau 73,68% yang tidak memahami tindakan pertolongan pertama pada kecelakaan.[2]

Maka hal ini yang memicu pengembang untuk membuat aplikasi tentang P3K dan agar lebih mudah dimengerti penyampaian materi disajikan dalam bentuk animasi 2dimensi.

Dengan Penggunaan Algoritma *Boyer-Moore* yang diterapkan dalam fitur pencarian merupakan sebuah solusi yang sangat tepat, karena dapat mempermudah user dalam melakukan pencarian judul kecelakaan. Algoritma *Boyer-Moore* ini dianggap cukup akurat dan cepat.

2. TINJAUAN PUSTAKA

2.1 Pertolongan Pertama Pada Kecelakaan

Pertolongan Petama Pada Kecelakaan (P3K) adalah upaya pertolongan dan perawatan sementara terhadap korban kecelakaan sebelum mendapat pertolongan yang lebih sempurna dari dokter atau paramedik. Ini berarti pertolongan tersebut bukan sebagai pengobatan atau penanganan yang sempurna, tetapi hanyalah berupa pertolongan sementara yang dilakukan oleh petugas P3K (petugas medik atau orang awam) yang pertama kali melihat korban.[3]

Pemberian pertolongan harus secara cepat dan tepat dengan menggunakan sarana dan prasarana yang ada di tempat kejadian. Tindakan P3K yang dilakukan dengan benar akan mengurangi cacat atau penderitaan dan bahkan menyelamatkan korban dari kematian, tetapi bila tindakan P3K dilakukan tidak baik malah bisa memperburuk akibat kecelakaan bahkan menimbulkan kematian.

2.2 Algoritma *Boyer-Moore*

Algoritma *Boyer-Moore* dikembangkan oleh Bob Boyer dan J. Strother Moore pada tahun 1977. Pada berbagai literatur dan referensi, algoritma ini menjadi standar *benchmark* dalam pencarian *string*.[18] Berbeda dengan *brute force* dan KMP yang membandingkan karakter *pattern* mulai dari kiri ke kanan, *Boyer-Moore* melakukan hal yang sebaliknya, yaitu membandingkan karakter *pattern* dari kanan ke kiri.

Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Tidak seperti algoritma pencarian *string* yang ditemukan sebelumnya, algoritma *Boyer-Moore* mulai mencocokkan karakter dari sebelah kanan *pattern*. Ide di balik algoritma ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Langkah-langkah algoritma *Boyer-Moore* [19]:

1. Buat tabel pergeseran *string* yang dicari (S) dengan pendekatan *Match Heuristic* (MH) dan *Occurence Heuristic* (OH), untuk menentukan jumlah pergeseran yang akan dilakukan jika mendapat karakter tidak cocok pada proses pencocokan dengan *string* (T).

2. Jika dalam proses pembandingan terjadi ketidakcocokan antara pasangan karakter pada S dan karakter pada T, pergeseran dilakukan dengan memilih salah satu nilai pergeseran dari dua tabel analisa *string*, yang memiliki nilai pergeseran paling besar.

3. Dua kemungkinan penyelesaian dalam melakukan pergeseran S, jika sebelumnya belum ada karakter yang cocok adalah dengan melihat nilai pergeseran hanya pada tabel *occurence heuristic* :

    a) Jika karakter yang tidak cocok tidak ada pada S maka pegeseran adalah sebanyak jumlah karakter pada S dan jika karakter yang tidak cocok ada pada S, maka banyaknya pergeseran bergantung dari nilai pada tabel.

    b) Jika karakter pada teks yang sedang dibandingkan cocok dengan karakter pada S, maka posisi karakter pada S dan

T diturunkan sebanyak 1 posisi, kemudian lanjutkan dengan pencocokan pada posisi tersebut dan seterusnya. Jika kemudian terjadi ketidakcocokan karakter S dan T, maka pilih nilai pergeseran terbesar dari dua tabel analisa *pattern* yaitu nilai dari tabel *match heuristic* dan nilai tabel *occurence heuristic* dikurangi dengan jumlah karakter yang telah cocok.

4. Jika semua karakter telah cocok, artinya S telah ditemukan di dalam T, selanjutnya geser *pattern* sebesar 1 karakter.

5. Lanjutkan sampai akhir *string* T.

2.3 Animasi 2 Dimensi

Animasi adalah urutan *frame* yang ketika diputar dalam rangka dengan kecepatan yang cukup, dapat menyajikan gambar bergerak lancar seperti sebuah *film* atau *video*. Animasi dapat juga diartikan dengan menghidupkan gambar, sehingga anda perlu mengetahui dengan pasti setiap detail karakter anda, mulai dari tampak (depan, belakang dan samping) detail muka si karakter dalam berbagai ekspresi (normal, diam, marah, senyum, ketawa, kesal, dan lainnya.) lalu pose/gaya khas karakter bila sedang melakukan kegiatan tertentu yang menjadi ciri khas si karakter tersebut.

Animasi 2 dimensi adalah penciptaan gambar bergerak dalam lingkungan dua dimensi. Hal ini dilakukan dengan urutan gambar berturut-turut, atau "*frame*", yang mensimulasikan gerak oleh setiap gambar menunjukkan berikutnya dalam perkembangan bertahap langkah-langkah.[23]

Jenis animasi 2 dimensi adalah jenis yang memiliki sifat *flat* secara visual. Bila dilihat dari teknis pembuatannya terdapat dua cara, yaitu manual dan komputer. [24]

Teknik animasi manual atau yang biasa disebut dengan *cell animation* adalah teknik animasi yang paling lama usianya. Teknik animasi ini memungkinan *animator* untuk membuat gambar pada lembaran *celuloid* (lembar trasparan) yang berlapis-lapis. karena kemajuan teknologi sekarang *animator* tidak lagi membuat animasi tradisional ini dengan lembaran *celuloid*, tapi bisa dengan menggunakan kertas biasa yang nanti akan di pindai (*scan*) lalu di warna dengan menggunakan komputer.

Teknik animasi 2D komputer adalah teknik animasi yang dibuat dengan menggunakan bantuan komputer (*hardware/software*) dan tetap mengandalkan kemampuan menggambar lembar demi lembar. Sehingga yang membedakan antara *traditional animation* dengan 2D *Computer generated imagery* (GIF) adalah medianya.

3. METODE PENELITIAN

3.1 Analisis Algoritma *Boyer-Moore*

Pencocokan string dengan algoritma boyer-moore ini dibagi menjadi 2 tahap. Tahap pertama yaitu preprocessing, pada tahap ini dibangunnya tabel *bad-character* dan juga tabel *good-suffix*. Dari tabel ini akan didapatkan nilai-nilai pergeseran pada masing-masing karakter. Nilai yang didapat akan digunakan untuk melakukan pergeseran pada proses pencarian. Tahap kedua yaitu proses algoritma *boyer-moore*, pada tahap ini akan mulai mencocokkan antara kata kunci yang dimasukkan atau pattern dengan kata yang terdapat dalam database atau disebut juga dengan teks. Pencocokkan akan dimulai dari karakter paling kanan dari *pattern*, jika cocok maka akan mundur untuk mencocokkan pada karakter berikutnya. Jika ditemukan karakter yang tidak cocok maka, pada saat inilah nilai dari tabel *bad-character* dan *good-suffix* digunakan. Nilai bad-character dari karakter teks yang dicocokkan akan dibandingkan dengan nilai *good-suffix* dari karakter pattern yang dicocokkan, lalu diambil nilai terbesar untuk melakukan pergeseran pada *pattern*.

Diilustrasikan bagaimana penerapan algoritma *boyer-moore* pada aplikasi yang akan dibuat, dengan menggunakan kata yang akan dicari 'LINTAH' sebagai *pattern* dalam basis data 'GIGITAN LINTAH' sebagai *text.*

Pattern : LINTAH
Text : GIGITAN LINTAH

Tahap awal dari proses algoritma ini yaitu membuat tabel *BmBc* juga *BmGs* untuk mendapatkan nilai yang akan digunakan dalam pergeseran pada pencarian karakter yang sesuai.

Tabel 3.1 Tabel *BmBc*

Tabel *BmBc*						
Index	0	1	2	3	4	5

Pattern	L	I	N	T	A	H
BmBc	5	4	3	2	1	0

Untuk mendapatkan nilai pada tabel *BmBc* maka lakukan pencacahan mulai dari posisi terakhir yaitu ‘H’ sampai ke posisi awal yaitu ‘L’. Jika karakter belum pernah ditemukan pada posisi sebelumnya maka diberikan nilai dengan berurutan sampai karakter paling awal. Seperti halnya pada tabel diatas, karakter ‘H’ diberi nilai 0, lalu mundur pada karakter ‘A’ diberi nilai 1 dan seterusnya sampai karakter terakhir yaitu ‘L’ diberi nilai 5.

Tabel 3.2 Tabel *BmGs*

Tabel *BmGs*						
Index	0	1	2	3	4	5
Pattern	L	I	N	T	A	H
BmGs	6	6	6	6	6	1

Setelah membuat tabel *BmBc* dilanjut dengan menentukan nilai pada tabel *BmGs*. Untuk menentukan nilai pada tabel ini ketentuannya yaitu, dimulai dari indeks paling awal yaitu indeks 0 dengan karakter ‘L’ karena tidak ditemukan pola ‘INTAH’ pada sebelah kiri indeks maka diberi nilai panjang dari *pattern* yaittu 6, berikutnya menggeser pada indeks 1 yaitu karakter ‘I’ karena tidak ditemukan juga pola ‘NTAH’ pada sebelah kiri dari indeks 1 maka diberi nilai 6. Hingga seterusnya pada kasus yang sama sampai pada indeks terakhir, karena merupakan indeks terakhir maka secara *default* diberi nilai 1.

Setelah mencari nilai dari tabel *BmBc* dan *BmGs*, selanjutnya yaitu mensejajarkan *pattern* dan teks lalu mencocokkan pattern dengan teks dengan melihat karakter *pattern* dari paling kanan.

Tabel 3.3 Iterasi Algoritma *Boyer-Moore*

Iterasi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Bc	Gs
	G	I	G	I	T	A	N		L	I	N	T	A	H		
1	L	I	N	T	A	H									1	1
2		L	I	N	T	A	H								3	1
3					L	I	N	T	A	H					4	1
4									L	I	N	T	A	H		

Pada iterasi ke-1, *pattern* sejajar dengan teks paling kiri. Lalu melihat pada karakter terakhir dari *pattern* yaitu ‘H’ bersejajaran dengan karakter ‘A’ pada teks karena kedua karakter tidak sesuai, maka dilihat nilai Bc dari karakter pada teks yaitu ‘A’ mempunyai nilai 1 dalam tabel. Sedangkan Gs melihat dari karakter pada *pattern* yaitu ‘H’ mempunyai nilai 1 dalam tabel. Karena nilai yang dimiliki Bc dan Gs sama maka dilakukan *pattern* melakukan pergeseran ke sebelah kanan 1 langkah.

Pada iterasi ke-2, setelah *pattern* menggeser 1 langkah maka karakter paling akhir yaitu ‘H’ akan bersejajaran dengan ‘N’ pada teks. Karena kedua karakter tidak sesuai, maka dilihat nilai Bc dan Gs dari kedua karakter tersebut. Yaitu Bc dari karakter ‘N’ = 3 dan Gs dari karakter ‘H’ = 1, dikarenakan nilai karakter tidak sama maka yang diambil untuk nilai pergeseran yaitu nilai terbesar yaitu 3. Lalu *pattern* melakukan pergeseran sebanyak 3 langkah.

Pada iterasi ke-3, setelah *pattern* menggeser 3 langkah. Karakter yang bersejajaran dengan ‘H’ yaitu karakter ‘I’ pada teks. Nilai Bc dari karakter ‘I’ = 4 dan nilai Gs dari karakter ‘H’ = 1. Maka, diambil nilai pergeserannya yaitu 4 langkah ke sebelah kanan *pattern*.

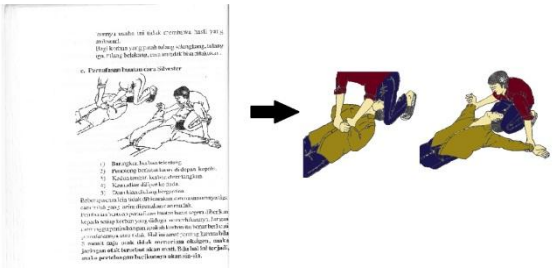
Pada iterasi ke-4, setelah *pattern* melakukan pergeseran sebanyak 4 langkah. Maka akan dilihat kesesuaian karakter dari ujung paling kanan *pattern* yaitu ‘H’ bersejajaran dengan karakter ‘H’ pada teks karena sesuai, maka mundur pada karakter berikutnya ‘A’ bersejajaran dengan karakter ‘A’ pada teks lalu mundur dan menyesuaikan sampai karakter paling awal dari *pattern*. Jika ternyata *pattern* sesuai maka indeks awal pattern pada teks yaitu 8 akan disimpan dalam *array* untuk memberitahukan letak pattern yang sesuai dengan teks. Lalu jika berada pada ujung teks maka proses pencarian berakhir. Namun, jika belum berada pada ujung teks akan melakukan pencarian kembali sampai pada ujung teks.

3.2 Perancangan Animasi 2 Dimensi

Animasi ini menggunakan teknik *cell*, yang mana proses pembuatannya terbentuk dari lembaran-lembaran *frame* yang membentuk *frame* animasi tunggal. Setisp lembarannya mewakili objek-objek yang berbeda yang dapat diisi dengan efek animasi agar terlihat lebih mulus yaitu dengan menentukan kemunculan setiap objek dengan menggunakan *keyframe*. Adapun proses *tweening* untuk membuat sebuah pergerakan objek tanpa perlu mengatur objek pada setiap *keyframe*nya.

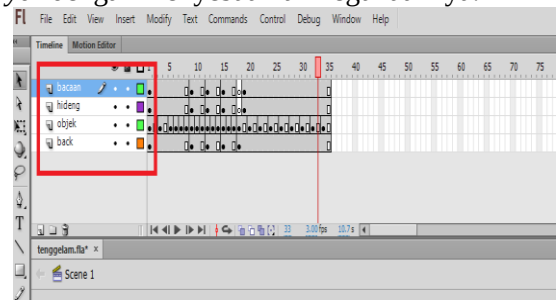
Proses pembuatan animasi ini dapat dimulai dengan menyiapkan objek yang akan digunakan

baik itu menggambar manual lalu dilanjutkan dengan proses *scanning* agar menjadi bentuk *softfile*, lalu *tracing* dan *colouring* untuk menyempurnakan objek yang dibuat atau bisa dengan memanfaatkan gambar yang ada dengan menyesuaikan alur cerita pada animasi yang dibuat.



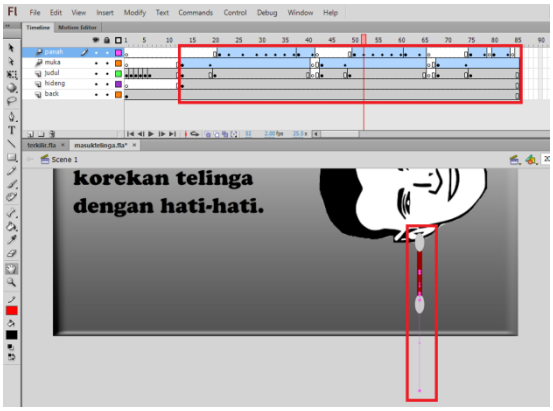
Gambar 3.1 Proses Pembuatan Objek

Setelah menyiapkan objek yang dilakukan yaitu menempatkan masing-masing objek pada *layer* berbeda dan menempatkannya pada urutan *layer* dengan menyesuaikan kegunaannya.



Gambar 3.2 Objek pada Layer Berbeda

Objek yang dimasukkan pada masing-masing *layer*pun perlu disesuaikan dengan waktunya kapan objek tersebut harus masuk dalam *frame* juga kapan objek tersebut harus menghilang dalam *frame*. Pada bagian ini menggunakan *keyframe* untuk mengatur letak objek ditempatkan sesuai waktunya. Dapat juga mengguanakan efek tweening untuk menciptakan gerakan dinamis dari objek agar terlihat lebih mulus.



Gambar 3.3 Penempatan *Keyframe* dan Efek *Tweening*

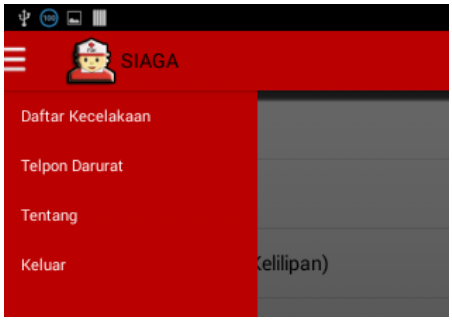
Setelah pengaturan penempatan objek juga penambahan efek dan animasi sesuai dengan alur cerita yang ingin disampaikan maka dilanjutkan dengan tahap akhir yaitu mengeksport *file* tersebut dalam bentuk .GIF dan diberi nama sesuai dengan yang terdapat dalam *database* data P3K yang akan digunakan pada aplikasi. Lalu menyimpan *file* animasi tersebut pada folder *asset* didalam proyek.

4. PEMBAHASAN

4.1 Implementasi

Gambar 4.1 *Splash Screen*

Gambar diatas merupakan tampilan *splashscreen* dari aplikasi simulasi kegiatan P3K. Tampilan ini akan tampil di awal aplikasi sesaat ketika aplikasi dijalankan.



Gambar 4.2 Menu

Pada tampilan *Menu* ini terdapat disebelah kiri halaman utama atau pada daftar kecelakaan. Menu ini dapat dibuka dengan menarik ujung layar dari kiri ke kanan, atau dengan menekan garis tiga atau *icon navdrawer* yang terletak disebelah nama aplikasi.

Gambar 4.3 Daftar Data Kecelakaan

Pada gambar diatas menampilkan jendela yang berisikan daftar data kecelakaan yang terdapat dalam basis data. Pada jendela ini juga *user* dapat melakukan pencarian dengan menekan *icon* pencarian yang terdapat pada pojok kanan atas dan mengisikan kata kunci yang diinginkan pada *bar*. Jika kata sudah dimasukkan lalu di tekan *enter*, maka *sistem* akan mulai melakukan pencarian dengan menggunakan algoritma *boyer-moore*.

Gambar 4.4 Pencarian Pada Bar

Pada gambar 4.4 merupakan tampilan ketika pencarian sudah berlangsung dan menampilkan waktu yang ditempuh untuk menjalankan algoritma *boyermoore* juga jumlah data yang ditemukan. Lalu akan ditampilkan pula daftar judul yang ditemukan pada pencarian.

Gambar 4.5 Keterangan

Pada tampilan ini berisikan keterangan yang bersangkutan dari judul kecelakaan yang sebelumnya telah dipilih. Pada jendela ini pun berisikan animasi yang mensimulasikan solusi atau penanganan terhadap judul kecelakaan tersebut.

4.2 Pengujian

Pengujian algoritma *boyer-moore* dilakukan untuk menguji kesesuaian pencarian kata dengan kata dalam *database* dan juga untuk menguji kecepatan yang dibutuhkan untuk mencari kata yang di*input*kan.

Pengujian yang didasarkan dengan kesesuaian kata yang dimasukkan dengan kata dalam *database* menggunakan 10 kata uji yang

umum digunakan. Berikut hasil pengujian dari keakuratan algoritma *boyer-moore*:

Tabel 4.1 Tabel Pengujian Kesesuaian kata

No	Kata Uji	Judul yang ditemukan		Pada *Database*	
		Judul	Jumlah	Sesuai	Tidak
1	Gatal	-	-	√	
2	Gigit	1. Gigitan Anjing (Rabies) 2. Gigitan Binatang atau Cakaran 3. Gigitan Lintah 4. Gigitan Lipan/Kelabang	5	√	
		5. Gigitan Ular			
3	Jatuh	-	-	√	
4	Keguguran	1. Keguguran atau Abortus	1	√	
5	Luka	1. Luka Bakar 2. Luka Goresan atau Tersayat 3. Luka Memar 4. Pendarahan Akibat Luka	4	√	
6	Mimisan	1. Pendarahan hidung atau Mimisan	1	√	
7	Nyamuk	1. Sengatan Serangga (Nyamuk, Kutu, Agas, Lalat)	1	√	
8	Pingsan	1. Pingsan 2. Pingsan Karena Diabetes 3. Pingsan Karena Panas Matahari 4. Pingsan Karena Shock 5. Pingsan Karena Tempat Panas 6. Pingsan Karena Tenggelam atau Tersedak	6	√	
9	Sakit	-	-	√	
10	Terkilir	1. Terkilir/Keseleo	1	√	

Tabel diatas menunjukkan bahwa dari 10 kata uji yang dijadikan sebagai kata kunci pencarian 7 diantaranya terdapat dalam *database* dan sisanya tidak terdapat dalam *database*. Namun, proses pencariannya sesuai dengan apa yang ada dalam *database* dan kesepuluh kata pengujian tersebut sesuai dengan apa yang ada didalam database tersebut sehingga dapat disimpulkan bahwa algoritma ini dapat mencari kata dengan keakuratan 100%.

Sedangkan pengujian yang didasarkan dengan jumlah *pattern* dengan menguji dari sudut pandang waktu yang dibutuhkan dalam setiap pencarian yang diujikan. Pengujian ini dibagi menjadi kedalam 3 bagian, panjang karakter 0-5, panjang karakter 6-10 dan panjang karakter 11-15.

Tabel 4.2 Rata-rata Keseluruhan Pengujian

Waktu	**Panjang Karakter**		
	0-5	6-10	11-16
Waktu ke-1	**2.4 ms**	**1.8 ms**	**2.0 ms**
Waktu ke-2	**1.9 ms**	**2.5 ms**	**2.0 ms**
Waktu ke-3	**1.8 ms**	**1.9 ms**	**2.5 ms**
Rata-rata	**2.03 ms**	**2.06 ms**	**2.16 ms**

Dari tabel diatas dapat kita simpulkan bahwa panjang karakter yang dicari tidak terlalu berpengaruh terhadap waktu pencarian yang dibutuhkan. Meskipun terdapat selisih waktu yang cukup tipis namun melakukan pencarian dengan waktu rata-rata **2.08ms** secara keseluruhan dianggap cukup efesien dan tidak memakan waktu yang cukup lama sehingga algoritma *boyer-moore* cukup layak diterapkan pada aplikasi ini.

5. KESIMPULAN

5.1 Kesimpulan

1. Algoritma *boyer-moore* dapat diterapkan dengan hasil yang akurat dan cepat dalam pencarian *string* pada aplikasi yang dibuat dengan mengambil kata kunci yang dimasukkan sebagai *pattern* maka algoritma akan mencocokkan dengan teks yang sesuai dengan *database*.
2. Animasi kegiatan simulasi P3K dibuat dengan bantuan *tools Adobe Flash* menghasilkan gambaran kegiatan yang perlu dilakukan penolong terhadap korban yang dihadapinya ini dapat mendukung dari hal yang diungkapkan melalui teori.

5.2 Saran

1. Algoritma *boyer-moore* hanya dapat mencari *string* yang sama persis dengan kata dalam database. Maka penulis menyarankan agar menambahkan algoritma yang dapat membandingkan kata kunci yang mendekati dengan kata di dalam database.
2. Aplikasi ini kedepan dapat menggunakan animasi 3D untuk mensimulasikan kegiatan P3K agar lebih jelas dan menarik.

DAFTAR PUSTAKA

[1] Korlantas. "Grafik Fatalitas Kecelakaan". 2016. <URL: http://www.korlantas-irsms.info/graph/accidentTypeTable> Diakses pada tanggal 07 Juni 2016 pukul 09.37 WIB

[2] Adi, Banu Setyo (2012). "Pemahaman

guru tentang pertolongan pertama pada kecelakaan". Vol 3 no.1. Hal 88-101.
[3] Machfoedz, Irham dkk., 2005. "Pertolongan pertama pada kecelakaan di rumah dan di tempat kerja". FITRAMAYA. Yogyakarta.
[4] Munir, Rinaldy. 2011. "Algoritma dan Pemrograman Dalam Bahasa *Pascal* dan C". Informatika Bandung. Bandung.