Implementasi Algoritma Kociemba pada Rekomendasi Penyelesaian Langkah Permainan Rubik

Halim Agung^{#1}, Jessica^{#2}, Rizky Januar^{#3}

[#]Program Studi Teknik Informatika Universitas Bunda Mulia

Jl. Lodan Raya No. 2 RT 012/RW 002. Kota Tua, Ancol, Pademangan, Kota Jakarta Utara, DKI Jakarta

¹hagung@bundamulia.ac.id

²jessicamega30@gmail.com

³rizkyjanuar2017@gmail.com

Abstrak— Implementasi algoritma kociemba pada rekomendasi penyelesaian langkah permainan rubik merupakan sebuah aplikasi yang dirancang dengan tujuan menghasilkan sebuah langkah dan kecepatan bermain yang efisien dan benar. Algoritma kociemba bekerja dengan *Two Phase*, fase 1 menggunakan *Iterative Deepening A** (IDA*) yaitu sistem akan mendeteksi dan membangun sebuah pola *state* dengan maksimal sebanyak 18 langkah. Fase 2 akan melakukan proses penyelesaian masalah dari pola tersebut dengan maksimal 12 langkah. Algoritma kociemba akan menghasilkan nilai maksimal perpindahan sebanyak 30 langkah. Sampel untuk penelitian ini adalah 10 mahasiswa berusia 20-24 tahun dengan membandingkan waktu bermain secara manual dengan waktu bermain secara sistem. Penelitian ini menyimpulkan bahwa implementasi algoritma kociemba pada rekomendasi penyelesaian langkah permainan rubik berhasil diterapkan dan mendapatkan rekomendasi langkah yang tepat serta efisien dengan waktu bermain yang sedikit. Hasil penelitian menunjukkan waktu rata-rata bermain secara manual adalah 40 menit. Sedangkan waktu bermain secara sistem adalah 10 detik dengan 22 langkah

Kata kunci— Rekomendasi, Rubik, Kociemba, *Two Phase*, IDA*

I. PENDAHULUAN

Rubik adalah sebuah permainan dalam bentuk teka-teki yang bertujuan untuk menyelesaikan sebuah masalah dalam bentuk tiga dimensi [1]. Permainan ini ditemukan oleh Erno Rubik, seorang oemahat yang ingin membantu muridnya untuk memahami permasalahan tiga dimensi. Erno Rubik membangun rubik selaha satu bulan dengan 54 stiker berwarna ke enam sisi rubik. Pada perlombaan World Cube Championship 2017 yang diselenggarakan di Paris, Perancis dimenangkan oleh Max Park dari Amerika Serikat dengan 5.87 seconds dengan ukuran kubus 3 x 3 x 3. Dalam pengembangannya, sudah terdapat beberapa jenis kubus

seperti kubus dengan dimensi 2 x 2 hingga ukuran kubus 7 x 7 [9].

Permainan Rubik mulai di minati oleh pasar dunia setelah mengikuti beberapa *International Toy's Fair* di berbagai negara, kemudian dilirik oleh Tom Kremer dan di komersialkan di pasar dunia.

Permainan Rubik umumnya sulit untuk menyelesaikan enam sisi yang terdapat pada rubik. Pemain yang baru mungkin akan kesulitan menyelesaikannya dan tidak banyak aplikasi yang menyediakan penyelesaian permainan rubik dengan efisien. Menurut Pankaj R. Chandre, Atul Raut, Riya, Shweta Terkay dan Jaswin Sehah melakukan pengujian dengan algoritma terkenal dengan hasil pengujian yaitu penyelesaian rubik dengan algoritma Basic Beginners dan menghubungkan algoritma Fridrich dengan 80 langkah [3]. Penelitian yang dilakukan oleh Ashutosh Tyagi dan Poonam Tyagi dengan judul "Genetic Algorithm Based Model for Rubik's Cube Solution" menghasilkan bahwa algoritma genetika adalah metode yang efektif untuk menentukan urutan langkah yang dibutuhkan untuk menyelesaikan rubik dengan berbagai pengacakan urutan rubik. Pemilihan kromosom yang lebih tinggi dari factor solusi rubik tahap awal menuntun ke solusi yang lebih baik [4]. Sedangkan penelitian "Solution for Rubik's Cube by Using Genetic Algorithm" oleh Kulkarni Sameer Anil dapat mereduksi hasil langkah dari 227 gerakan menyelesaikan rubik menjadi 107 langkah [5]. Penelitian Jay Hack dan Kevin Shutzberg dengan menggunakan metode k-means, Simple Linear Iterative Clustering dan algoritma kociemba memerlukan waktu yang banyak untuk mengoptimisasikan metode tersebut [6]. Peneliti Naik El-Sourani, Sascha Hauke dan Markus Borschbach dengan judul penelitin "An Evolutionary Approach For Solving the Rubik's Cube Incorporating Exact Methos" yang menggunakan 3 algoritma yaitu Thistlethwaite, Kociemba dan

Rokicki menyatakan bahwa algoritma tersebut efisien akan tetapi sulit dipahami oleh manusia [7]. Penelitian “*Autonomous Rubik's Cube Solver Using Image Processing*” oleh Harshad Sawhey, Sakshi Sinha, Anurag Lohia, Prashant Jalan dan Priyanka Harlalka menyatakan bahwa algoritma kociemba menjelaskan efisiensi dalam menetapkan setiap langkah dalam penyelesaian rubik [8]. Peneliti Chichi Rizka Gunawan, Ahmad Ihsan dan Munawir dengan judul penelitian “Optimasi Penyelesaian Permainan Rubik's Cube Menggunakan Algoritma IDA* dan Brute Force” yang menggunakan 2 algoritma yaitu Algoritma IDA* dan Brute Force menyatakan bahwa algoritma IDA* kurang efektif jika digunakan untuk menyelesaikan masalah rubik karena rubik dapat diselesaikan lebih cepat dari kemampuan manusia dan Brute Force memiliki kemampuan yang kalah cepat jika dibandingkan dengan algoritma IDA* karena membutuhkan waktu yang lama untuk mendapatkan solusi [10].

Permasalahan permainan rubik adalah bagaimana mendapatkan solusi dengan langkah yang tepat dan cepat. Dalam perkembangan saat ini, ketersediaan aplikasi yang dapat mempermudah pengguna atau pemain rubik sangat minim dan permainan tersebut sulit dipecahkan bagi pemain pemula. Sehingga penelitian berupa implementasi algoritma kociemba dalam rekomendasi langkah pada aplikasi permainan rubik merupakan sebuah solusi yang baik untuk menghasilkan langkah yang tepat

II. TINJAUAN PUSTAKA

A. Rubik

Kubus rubik adalah sebuah permainan dalam bentuk teka-teki yang dibuat oleh pemahat dan profesor dari Budapest di Hungaria yaitu Erno Rubik pada tahun 1974. Erno Rubik awalnya ingin membantu muridnya untuk memahami permasalahan tiga dimensi. Erno Rubik membangun magic cube nya membutuhkan waktu lebih dari satu bulan untuk menyelesaikannya serta menambahkan 54 stiker berwarna ke enam sisi dari rubik. Pada tahun 1970-an, permainan rubik mulai dikenal dan menarik minat akan tetapi ekspor dan impor di Hungaria memiliki kontrol yang cukup ketat. Kemudian, Erno Rubik memperkenalkan penemuannya di International Toy's Fair [1][15].

Magic Cube disertakan dalam beberapa International Toy's Fair di London, Paris dan Nurnberg. Pada pameran di Nurnberg, permainan teka-teki tersebut dilirik oleh Tom Kremer dan ingin mengkomersialkan kubus tersebut di pasar dunia. Tom Kremer meyakinkan perusahaan Ideal Toy Corp dan hasilnya “Magic Cube” diluncurkan secara global pada tahun 1980 [1]. Terhitung Januari 2009, sebanyak 350 juta buah kubus telah terjual diseluruh dunia [2]. Erno Rubik membuat sebuah perlombaan pada 5 Juni 1982 di Budapest, Hungaria. Pada saat itu, yang memenangkan perlombaan dengan menyelesaikan rubik tersebut adalah Minh Thai dari Amerika Serikat dengan waktu 22.95 seconds. Pada perlombaan World Cube Championship 2017 yang

diselenggarakan di Paris, Perancis dimenangkan oleh Max Park dari Amerika Serikat dengan 5.87 seconds dengan ukuran kubus 3 x 3 x 3.

Dalam pengembangannya, sudah terdapat beberapa jenis kubus seperti kubus dengan dimensi 2 x 2 hingga ukuran kubus 7 x 7 [3]. Kubus rubik memiliki tiga macam bagian yaitu :

1. Pusat (*center*)
Terdapat enam bagian *center*, setiap kotak memiliki satu warna tertentu dan tidak dapat merubah posisinya terhadap satu sama lain.
2. Ujung (*corner*)
Terdapat 8 bagian *corner* yang memiliki kombinasi dari tiga warna.
3. Tepi (*edges*)
Terdapat 12 bagian *edges* yang memiliki dua kombinasi warna
Kemudian rubik memiliki notasi yaitu F (*Front*), B (*Back*), L (*Left*), R (*Right*), U (*Up*), D (*Down*).

B. Algoritma Kociemba

Algoritma kociemba ini diciptakan oleh Herbert Kociemba untuk memperbaiki algoritma *Thistlethwaite* yang ada dengan menurunkan jumlah grup perantara menjadi dua:

- a. $G_0 = (U, D, L, R, F, B)$
- b. $G_1 = (U, D, L_2, R_2, F_2, B_2)$
- c. $G_2 = \{1\}$

Sebagaimana dengan algoritma *Thistlethwaite*, ia akan mencari spasi coset dari $G_1 \setminus G_0$ yang benar untuk mengelompokkan kubus kedalam grup G_1 . Berikutnya dia mencari solusi optimal untuk *Group G1*. Pencarian di G_0 dan G_1 keduanya dilakukan dengan metode yang senilai dengan IDA*. Pencarian di G_1 memerlukan paling banyak 12 gerakan dan pencarian di G_1 paling banyak 18 gerakan. Michael Reid menunjukkan pada tahun 1995 dengan membuat solusi suboptimal yang membawa kubus ke grup G_1 dan mencari solusi pendek di G_1 , biasanya akan menghasilkan solusi keseluruhan yang lebih singkat. [2]

Uraian berikut ini dimaksudkan untuk memberikan ide dasar tentang bagaimana algoritma ini bekerja. 6 permukaan yang berbeda dari Cube disebut Up (U), Down (D), Right (R), Left (L), Front (F) dan Back (B). Sementara U menunjukkan seperempat putaran permukaan Up 90 derajat searah jarum jam, U² menunjukkan 180 putaran derajat dan U' menunjukkan seperempat putaran 90 derajat berlawanan arah jarum jam. Deretan seperti UDR 'D2 dari gerakan Cube disebut manuver. [2]

Jika mengubah permukaan kubus yang telah diselesaikan dan tidak menggunakan gerakan R, R', L, L', F, F', B dan B' maka hanya akan menghasilkan subset tertentu dari semua posisi kubus yang dimungkinkan. Subset ini dilambangkan dengan $G_1 = \langle U, D, R_2, L_2, F_2, B_2 \rangle$. Dalam Subset ini, orientasi corner dan edges tidak dapat diubah. Artinya, orientasi edges atau corner di lokasi tertentu selalu sama dan empat sisi di

UD-slice (lapisan antara permukaan U dan D) tetap terisolasi di slice itu.

Pada Fase 1, algoritma mencari manuver yang akan mengubah kubus teracak menjadi G1. Artinya, orientasi corner dan edge harus dibatasi dan edge di UD-slice harus ditransfer ke bagian tersebut. Dalam ruang abstrak ini, Gerakan hanya mengubah sumbu *triple* (x, y, z) ke tiga posisi lainnya (x', y', z'). Semua kubus dari G1 memiliki *triplet* yang sama (x0, y0, z0) dan ini adalah *state* yang dituju dalam fase 1. [2]

Untuk menemukan state tujuan ini akan digunakan algoritma pencarian yang disebut Iterative Deepening A* dengan fungsi heuristic lowerbound (IDA*). Dalam kasus Rubik Cube, ini berarti bahwa pengulangan dilakukan melalui semua manuver yang bertambah panjang. Fungsi heuristic h1 (x, y, z) memperkirakan untuk masing-masing state kubus (x, y, z) berapa jumlah langkah yang diperlukan untuk mencapai state tujuan. Sangat penting bagi fungsi untuk tidak memperkirakan jumlah langkah ini secara berlebihan. [2]

Dalam pencarian diperlukan untuk memberikan jumlah pasti langkah yang diperlukan untuk mencapai keadaan tujuan pada Fase 1. Heuristik ini memungkinkan pruning sewaktu membuat manuver, hal ini penting jika kita tidak ingin menunggu untuk waktu yang sangat lama sebelum *state* tujuan tercapai. Fungsi heuristic h1 adalah tabel pencarian berbasis memori dan memungkinkan *pruning* hingga 12 gerakan [2].

Pada fase 2 algoritma mengembalikan kubus di subgroup G1, hanya menggunakan gerakan dari subkelompok ini. Ini mengembalikan permutasi dari 8 *corner*, permutasi dari 8 edges permukaan U dan D, dan permutasi dari 4 edges di UD-slice. Fungsi h2 heuristic (a, b, c) hanya memperkirakan jumlah langkah yang diperlukan untuk mencapai goal state, karena ada terlalu banyak elemen yang berbeda dalam G1. Algoritma ini tidak berhenti ketika sebuah solusi pertama ditemukan, namun terus mencari solusi yang lebih pendek dengan menjalankan fase 2 dari solusi suboptimal dari fase 1. Sebagai contoh, jika solusi pertama memiliki 10 gerakan pada fase 1 yang diikuti oleh 12 gerakan dalam fase 2, solusi kedua bisa memiliki 11 gerakan pada fase 1 dan hanya 5 gerakan dalam fase 2. Panjang manuver di fase 1 meningkat dan panjang manuver di fase 2 menurun. Jika panjang fase 2 mencapai nol, solusinya adalah optimal dan algoritma berhenti. [2].

III. ANALISIS DAN RANCANGAN

A. Metode Waterfall

Metode yang digunakan adalah metode waterfall. Model *System Development Life Cycle* ini sering disebut model sekuensial model atau alur hidup klasik. Model *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dari analisis, desain, pengkodean, pengujian dan perawatan atau *maintenance* [11].

1. Analisis Kebutuhan :

Tahapan ini bertujuan untuk pengumpulan data dan kebutuhan untuk mengetahui spesifikasi kebutuhan perangkat lunak agar dapat dipahami dan disesuaikan oleh kebutuhan *user*.

2. Desain dan Perancangan :

Tahapan ini adalah proses yang berfokus pada desain pembuatan program termasuk skruktur data, arsitektur perangkat lunak, *user interface*, dan juga *coding*. Tahap ini adalah translasi dari analisis ke representasi desain agar dapat diimplementasikan menjadi program tahap selanjutnya

3. Implementasi :

Tahap ini adalah translasi dari desain dan perancangan ke dalam sebuah program. Penelitian ini menggunakan bahasa pemrograman Java dengan menggunakan *game engine Unity*.

4. Pengujian :

Tahap ini berfokus pada perangkat lunak dari segi logika dan fungsional dan memastikan semua bagian sudah diuji dengan tujuan untuk meminimalkan kesalahan serta memastikan *output* sesuai dengan kebutuhan *user*.

5. Perawatan atau maintenance :

Tahap ini bertujuan untuk melakukan perubahan jika *user* menemukan sebuah kesalahan dan tidak terdeteksi saat pengujian.

B. Usecase Diagram

Usecase Diagram digunakan untuk bagaimana *user* berinteraksi dengan program yang dibuat [11][14]. Dalam aplikasi ini, *user* dapat melakukan input 3D rubik untuk mendapatkan penyelesaian permainan rubik, *input* lainnya adalah *capturing* yaitu mengambil gambar dari sisi-sisi rubik untuk mendapatkan penyelesaian rubik. Selain itu, *user* sebagai pemain dapat melakukan permainan rubik dalam aplikasi tersebut. *Usecase Diagram* pada aplikasi ini dapat dilihat pada Gambar 1.

Gambar. 1 Usecase Diagram

C. Analisis Algoritma

Terdapat dua fase untuk implementasi algoritma kociemba dalam permainan rubik yaitu:

1. Fase 1:

Fase pertama yaitu algoritma akan melakukan pengacakan dengan posisi pada G1 yaitu {F2, B2, R, L, U2, D2 }, untuk mencapai pengacakan tersebut melakukan algoritma *scramble* dimana hanya mengubah posisi (x,y,z) menjadi (x',y',z'). Kemudian algoritma tersebut akan meningkatkan jumlah langkah. Fungsi dari heuristik $h1(x,y,z)$ melakukan perkiraan mengenai bagian dari rubik tersebut (x,y,z) dan memprediksi jumlah gerakan untuk mencapai tujuan tersebut. Dalam fase ini kemungkinan pergerakan dibatasi sebanyak 12 gerakan.

2. Fase 2:

Fase ini mengembalikan rubik menjadi subgroup G1. Untuk mengembalikan rubik tersebut dilakukan permutasi dari 8 *corner* dan 8 *edges* bagian U dan permutasi bagian D serta permutasi dari 4 UD-Slice. Algoritma ini tidak berhenti ketika

sebuah solusi pertama ditemukan, namun terus mencari solusi yang lebih pendek dengan menjalankan fase 2 dari solusi suboptimal dari fase 1. Sebagai contoh, jika solusi pertama memiliki 10 gerakan pada fase 1 yang diikuti oleh 12 gerakan dalam fase 2, solusi kedua bisa memiliki 11 gerakan pada fase 1 dan hanya 5 gerakan dalam fase 2. Panjang manuver di fase 1 meningkat dan panjang manuver di fase 2 menurun. Jika panjang fase 2 mencapai nol, solusinya adalah optimal dan algoritma berhenti.

D. Flowchart

Bagan alir program (*flowchart*) adalah suatu bagan yang menggambarkan arus logika dari data yang akan diproses dalam suatu program dari awal sampai akhir. Bagan alir terdiri dari simbol – simbol yang mewakili fungsi-fungsi langkah program dan garis alir (*flow line*) menunjukan urutan dari simbol - simbol yang akan dikerjakan [12]

Gambar. 2 Flowchart Sistem

Proses yang terjadi pada gambar 2 adalah user melakukan command atau perintah dimana *user* dapat memilih 2 *button* yaitu *play* atau *solve*. Jika user memilih perintah *play* maka *user* akan bermain dengan rubik yang disediakan didalam aplikasi. Kemudian sistem akan melakukan pengacakan dengan menggunakan algoritma *scramble* jika *user* memilih *scramble* dan pemain akan disediakan rubik secara acak. Apabila pemain memilih *solve* maka prosesnya yang terjadi adalah melakukan masukan (*input*) secara manual dengan *facelet color* yang sudah disediakan. pilihan tersebut akan diselesaikan menggunakan algoritma kociemba, jika sudah melakukan proses tersebut maka akan menghasilkan keluaran (*output*) berupa langkah-langkah untuk menyelesaikan permainan rubik.

IV. HASIL DAN PEMBAHASAN

A. Hasil dan Pembahasan

Pada teori-teori yang digunakan dan algoritma yang digunakan. Aplikasi ini dibangun dengan pemrograman berbasis *desktop* dan menggunakan bahasa java [13] dan dibuat untuk memudahkan pengguna atau pemain rubik dapat menyelesaikan rubik dimensi 3x3 dengan langkah yang tepat. Berikut merupakan hasil dari tampilan antarmuka aplikasi. Pada Gambar 3 merupakan tampilan yang berisi perintah *play* dan *solve* yang bertujuan dengan permainan rubik.

Gambar. 3 Tampilan Home

Pada Gambar 3 merupakan tampilan dari *button solve* aplikasi permainan rubik yang berisi *button scramble*, *button solve*, dan *input* persegi yang mendeskripsikan dua dimensi dari rubik serta informasi bagian dari rubik tersebut dan terdapat *pallette* warna yang terdiri dari warna putih, merah, hijau, oranye, dan biru. Kemudian terdapat *string solution* yang berisi solusi dalam bentuk *string* yang menunjukkan langkah-langkah pergerakan permainan rubik hingga rubik terselesaikan.

Gambar. 4 Button Solve

Pada Gambar 4 merupakan tampilan dari *button solve* aplikasi permainan rubik yang berisi *button scramble*, *button solve*, dan *input* persegi yang mendeskripsikan dua dimensi dari rubik serta informasi bagian dari rubik tersebut dan terdapat *pallette* warna yang terdiri dari warna putih, merah, hijau, oranye, dan biru. Kemudian terdapat *string solution* yang berisi solusi dalam bentuk *string* yang menunjukkan langkah-langkah pergerakan permainan rubik hingga rubik terselesaikan.

Gambar. 5 Button Scramble

Pada Gambar 5 merupakan perintah dari *scramble* dimana sistem akan melakukan pengacakan.

Gambar 6. Button Solve

Pada Gambar 6 merupakan hasil dari perintah *solve* yang menghasilkan langkah pada *string solution*.

State	Step	Gambar
Langkah ke-1	U2
Langkah ke-2	D
Langkah ke-3	B2
Langkah ke-4	L
Langkah ke-5	F'

Langkah ke-6	L2
Langkah ke-7	B'
Langkah ke-8	L'
Langkah ke-9	D'

State	Step	Gambar
Langkah ke-10	B
Langkah ke-11	R2
Langkah ke-12	L2
Langkah ke-13	D

Langkah ke-14	R2
Langkah ke-15	D'
Langkah ke-16	F2
Langkah ke-17	B2

State	Step	Gambar
Langkah ke-18	D'
Langkah ke-19	R2
Langkah ke-20	D2
Langkah ke-21	Finish

Tabel 1 merupakan hasil dari *input* warna yang dimasukan oleh pemain, langkah maksimal yang didapatkan adalah 21 langkah.

B. Pengujian

Pengujian terhadap aplikasi ini adalah apakah aplikasi yang sudah dirancang dapat menghasilkan langkah dan waktu bermain yang lebih cepat dan melakukan perbandingan dengan bermain secara manual dan secara sistem. Pengujian ini dilakukan kepada mahasiswa berusia 20-24 tahun. Pengujian ini dilakukan dengan 10 orang bermain rubik dan *system*. Hasil pengujian bermain secara manual terdapat pada Tabel 2. Sedangkan, hasil pengujian bermain dengan skema yang sama secara sistem terdapat pada Tabel 1.

TABEL II
TABEL HASIL PENGUJIAN MANUAL

Pemain	Waktu(h:m:s)	Hasil Permainan
Pemain ke-1	0:10:55	Berhasil
Pemain ke-2	0:22:58	Gagal
Pemain ke-3	0:05:08	Berhasil
Pemain ke-4	0:43:50	Gagal
Pemain ke-5	0:36:54	Gagal
Pemain ke-6	0:17:33	Berhasil
Pemain ke-7	0:23:55	Gagal
Pemain ke-8	1:02:33	Gagal
Pemain ke-9	0:12:32	Berhasil
Pemain ke-10	0:21:22	Gagal

Pada Tabel 2 sebanyak 7 orang dari 10 orang yang diuji gagal menyelesaikan permainan rubik dengan dimensi 3x3 dan 3 orang berhasil menyelesaikan permainan rubik dengan waktu lebih dari 10 menit.

TABEL III
TABEL HASIL PENGUJIAN MELALUI SISTEM

Waktu untuk menghasilkan solusi	10.11 s
Waktu untuk menyelesaikan	10.15 s
Jumlah langkah	22 langkah

Pada Tabel 3 dengan posisi rubik yang sama dengan pengujian secara manual, sistem dapat menghasilkan 22 langkah dengan waktu penyelesaian 10.15 detik serta 10.11 detik untuk menghasilkan langkah tersebut.

V. KESIMPULAN

A. Kesimpulan

Implementasi algoritma kociemba pada aplikasi rekomendasi penyelesaian langkah permainan rubik berhasil diterapkan dan dapat digunakan dengan baik. Pemain rubik mendapatkan hasil berupa langkah yang efisien untuk menyelesaikan permainan rubik dan memberikan langkah yang paling sedikit. Hasil tersebut dapat dilihat dari pengujian dengan manual dan melakukan dengan sistem yang dilakukan terhadap 10 pemain rubik dengan pengacakan rubik yang sama. Pengujian tersebut menghasilkan bahwa sebagian besar

pemain rubik tidak dapat menyelesaikan rubik dengan ukuran 3x3 dan pemain yang dapat menyelesaikan rubik tersebut membutuhkan waktu lebih dari 10 menit. Apabila sistem melakukan penyelesaian dibutuhkan waktu paling lama 1 menit. Hal tersebut sudah diujikan dan terbukti.

B. Saran

Untuk pengembangan aplikasi rekomendasi penyelesaian langkah rubik maka peneliti memberikan saran untuk pengembangan aplikasi yang lebih baik yaitu:

1. Perlunya penggunaan algoritma yang lebih baik untuk menghasilkan rekomendasi langkah yang lebih sedikit dan benar dengan waktu proses yang lebih cepat.
2. Implementasi dalam membentuk aplikasi, disarankan untuk menggunakan berbasis android untuk memudahkan pemain dalam memanfaatkan aplikasi.
3. Penggunaan kamera untuk mendeteksi warna pada sisi rubik akan sangat memudahkan pemain dalam menyelesaikan permainan rubik.

REFERENSI

- [1] Rubik. <https://eu.rubiks.com/about/> . diakses 9 Oktober 2018
- [2] Kociemba, Herbert. *Cube Explorer*. 2015. <http://kociemba.org/cube.htm> . diakses pada 9 oktober 2018.
- [3] Chandre, Pankaj R., Raut, Atul., Riya, Terkay, Shweta., Seha, Jaswin. 2017. Rubik's Cube Solver.
- [4] Tyagi, Ashutosh., Tyagi Poonam. 2011. GEN-R : Genetic Algorithm Based Model for Rubik's Cube Solution Generator.
- [5] Anil, Sameer Kulkarni Anil. 2015. Solution For Rubik's Cube by Using Genetic Algorithm.
- [6] Hack, Jay. Shutzberg, Kevin. 2015. Rubik's Cube Localization, Face Detection, and Interactiv Solving.
- [7] El-Sourani, Nail., Hhauke, Sasscha., Borschbach, Markus. 2015. An Evolutionary Approach For Solving the Rubik's Cube Incorporating Exact Methods.
- [8] Sawhney, Harshad., Sinha, Sakshi., Lohia, Anurag., Jalan, Prashant., Harlalka, Priyanka. 2013. Autonomous Rubik's Cube Using Image Processing.
- [9] World Cube Championship. <https://www.worldcubeassociation.org/competitions/WC2017>. Diakses 9 Oktober 2018
- [10] Gunawan, C.R., Ihsan, A., Munawir. 2018. Optimasi Penyelesaian Permainan Rubik's Cube Menggunakan Algoritma IDA* dan Brute Force. Jurnal Infomedia. Vol 3. No 1. P-ISSN: 2527-9858, E-ISSN: 2548-1180.
- [11] Dennis, Alan, et al. 2013. System Analysis and Design with UML 4th Edition, John Wiley & Sons.
- [12] Donal Riosza, 10 Mei 2013. Penngenaln Flowchart (Definisi dan Simbol Flowchart). <http://sites.google.com/site/pagentukilmu2/tutorial/pemrograman-c/pengenalnflowchartdefinisidansimbolfowchart> . Diakses 9 Oktober 2018
- [13] Raharjo, Budi., Heryanto, Imam., Haryono, Arif. 2010. Mudah Belajar Java. Bandung : Informatika
- [14] Nugroho, Adi. 2004. Pemrograman Berorientasi Objek. Bandung: Informatika.
- [15] Joyner, D. 2002. Adventures in group theory: Rubik's Cube, Merlin's machine, and Other Mathematical Toys. Baltimore: Johns Hopkins University Press. p. 7.