

**PEMANFAATAN COMPUTER VISION PADA APLIKASI BELAJAR
MENYELESAIKAN RUBIK 3X3X3 BERBASIS ANDROID**

SKRIPSI

**M. ZAKI HARITS BURNAMA
191401106**



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

PEMANFAATAN COMPUTER VISION PADA APLIKASI BELAJAR
MENYELESAIKAN RUBIK 3X3X3 BERBASIS ANDROID

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Sarjana Ilmu
Komputer

M. ZAKI HARITS BURNAMA

191401106



PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024

PERSETUJUAN

Judul : Pemanfaatan Computer Vision pada Aplikasi Belajar Menyelesaikan Rubik 3x3x3 Berbasis Android

Kategori : SKRIPSI

Nama : M. Zaki Harits Burnama

Nomor Induk Mahasiswa : 191401106

Program Studi : SARJANA (S-1) ILMU KOMPUTER

Departemen : ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

Tanggal di uji dan dinyatakan lulus di Medan, 11 Juni 2024

Komisi Pembimbing

Pembimbing 1.

Dr. Ir. Elviawaty Muisa Zamzami S.T., M.T.,
M.M., IPU.
NIP. 197007162005012002

Pembimbing 2

Dewi Sartika Br Ginting S.Kom., M.Kom
NIP. 199005042019032023



Diketahui/ Disetujui oleh
Program Studi S-1 Ilmu Komputer

Ketua,

Dr. Amalia S.T., M.T.
NIP. 197812212014042001

PERNYATAAN

PEMANFAATAN COMPUTER VISION PADA APLIKASI BELAJAR
MENYELESAIKAN RUBIK 3X3X3 BERBASIS ANDROID

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 11 Juni 2024

M. Zaki Harits Burnama
191401106

PENGHARGAAN

Puji syukur penulis panjatkan atas kehadiran Allah SWT, karena rahmat dan karunia-Nya penulis dapat menyelesaikan skripsi ini dengan sebaik mungkin sebagai syarat memperoleh gelar Sarjana pada Program Studi S1 Ilmu Komputer Fasilkom-TI Universitas Sumatera Utara.

Dalam kesempatan ini penulis mengucapkan terima kasih kepada pihak-pihak yang telah banyak membantu serta memberi dukungan dalam penyusunan skripsi ini :

1. Allah SWT, berkat rahmat dan nikmat-Nya penulis diberikan kesehatan dan kekuatan dalam menyelesaikan skripsi ini.
2. Kedua orang tua penulis yang tersayang yang selalu mendukung dan mendoakan penulis hingga menyelesaikan skripsi ini.
3. Dr. Muryanto Amin, S.Sos. selaku Rektor Universitas Sumatera Utara
4. Dr. Maya Silvi Lydia. selaku M.Sc, Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
5. Dr. Amalia ST., M.T. selaku Ketua Prodi S-1 Ilmu Komputer Universitas Sumatera Utara.
6. Dr. Ir. Elviawaty Muisa Zamzami S.T., M.T., M.M., IPU. selaku Dosen Pembimbing I yang membimbing dan memberikan dukungan, saran dan kritik kepada penulis dalam penggerjaan skripsi.
7. Dewi Sartika Br Ginting S.Kom., M.Kom selaku Dosen Pembimbing I yang membimbing dan memberikan dukungan, saran dan kritik kepada penulis dalam penggerjaan skripsi.
8. Bapak Fauzan Nurahmadi S.Kom., M.Cs dan Bapak Amer Sharif S.Si., M.Kom selaku Dosem Pemimpin Akademik yang telah membimbing dan memberikan arahan dan saran kepada penulis selama menempuh perkuliahan.
9. Naafi Mahendra, Alfuadi Sakanaher, Fajar Shadiq Adhanai selaku teman yang memberikan suport.

10. Seluruh teman-teman mahasiswa/i S1 Ilmu Komputer Fasilkom-TI USU angkatan 2019, khususnya teman-teman Kom A 2019 yang tidak dapat disebutkan satu persatu.

Demikian skripsi ini dibuat, semoga para pembaca mendapatkan manfaat dari skripsi ini. Namun penulis sadar, dalam penyusunan skripsi, masih banyak ditemukan kekurangan yang perlu disempurnakan di kemudian hari. Oleh sebab itu penulis dengan sangat terbuka menerima kritik dan saran untuk menyempurnakan penyusunan laporan akhir.

Medan, 11 Juni 2024

M. Zaki Harits Burnama

ABSTRAK

Rubik 3x3x3 adalah salah satu *puzzle* klasik yang telah menarik minat banyak pemecah *puzzle* di seluruh dunia dan untuk dapat memainkannya membutuhkan keterampilan khusus dalam berpikir logis, strategi, dan visualisasi. Namun, bagi pemula proses memecahkan Rubik 3x3x3 seringkali membingungkan dan menantang. Penelitian ini bertujuan untuk mengembangkan sebuah aplikasi berbasis Android yang memanfaatkan teknologi *Computer Vision* untuk membantu pemula belajar dan memecahkan Rubik 3x3x3 dengan lebih mudah dan efektif. Produk dari penelitian ini adalah sebuah aplikasi yang memberikan materi belajar Rubik, mulai dari pengenalan hingga bagaimana menyelesaikan Rubik itu sendiri, dan fitur untuk memberikan tahap-tahap penyelesaian rubik asli milik pengguna. Memungkinkan pengguna untuk menyusun Rubik 3x3x3 sambil memperoleh pemahaman yang lebih mendalam tentang metode dan algoritma yang terlibat dalam menyelesaikan *puzzle* tersebut. Metode penyelesaian rubik yang digunakan adalah *Layer by Layer* yang merupakan salah satu cara terpopuler dalam menyelesaikan rubik, dan *Computer Vision* dimanfaatkan untuk memudahkan proses input data dari pengguna. Aplikasi ini dibuktikan mampu mencari algoritma untuk Rubik acak pengguna dengan tingkat keberhasilnya 100%. Dan tingkat akurasi pemindaian warna rubik pada kondisi cahaya cukup 99%. Hal ini membantu pemula yang ingin belajar Rubik 3x3x3 untuk dapat mengatasi kendala awal yang sering kali menghambat mereka. Dan kedepannya, aplikasi ini juga dapat menjadi alat yang efektif untuk pendidikan dalam memfasilitasi pembelajaran Rubik 3x3x3 di sekolah atau komunitas pemecah *puzzle*.

Kata kunci: Computer Vision, Rubik 3x3x3, Aplikasi Android, Pembelajaran, Pemecahan *Puzzle*.

Utilization of Computer Vision in an Android-based Learning Application for Solving the 3x3x3 Rubik's Cube

ABSTRACT

The Rubik's Cube 3x3x3 is a classic puzzle that has captivated puzzle solvers worldwide, demanding unique skills in logical thinking, strategy, and visualization. However, for beginners, the process of solving the Rubik's Cube 3x3x3 can be perplexing and challenging. This research aims to develop an Android-based application that utilizes Computer Vision technology to assist beginners in learning and solving the Rubik's Cube 3x3x3 more easily and effectively. The outcome of this research is an application that provides Rubik's Cube learning materials, ranging from introduction to solving techniques. It also features a function that provides step-by-step solutions for users' original Rubik's Cube configurations. This allows users to assemble the Rubik's Cube 3x3x3 while gaining a deeper understanding of the methods and algorithms involved in solving the puzzle. The Layer by Layer method, one of the most popular Rubik's Cube solving methods, is employed in the application. Computer Vision is utilized to facilitate the input process for users. The application has been proven capable of generating algorithms for users' random Rubik's Cubes with a 100% success rate. Additionally, the color scanning accuracy of the Rubik's Cube under sufficient lighting conditions is 99%. This aids beginners in overcoming the initial obstacles that often hinder their progress in learning the Rubik's Cube 3x3x3. Furthermore, this application has the potential to become an effective tool for educators in facilitating Rubik's Cube 3x3x3 learning in schools or puzzle-solving communities.

Keywords: Computer Vision, Rubik's Cube 3x3x3, Android Application, Learning, Puzzle Solving.

DAFTAR ISI

PERSETUJUAN.....	ii
PERNYATAAN.....	iii
PENGHARGAAN.....	iv
ABSTRAK.....	vi
ABSTRACT.....	vii
DAFTAR ISI.....	viii
DAFTAR TABEL.....	x
DAFTAR GAMBAR.....	xi
BAB 1.....	1
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah.....	3
1.3. Tujuan Penelitian.....	3
1.4. Batasan Masalah.....	3
1.5. Manfaat Penelitian.....	4
1.6. Metodologi Penelitian.....	4
1.7. Sistematika Penulisan.....	5
BAB 2.....	6
2.1. Rubik.....	6
2.2. Computer Vision.....	12
2.3. Pengajaran Berbantuan Komputer.....	14
2.4. Penelitian Terdahulu.....	15
BAB 3.....	16
3.1. Analisis Sistem.....	16
3.1.1. Analisis Masalah.....	16

3.1.2. Requirements Analysis.....	17
3.2. Pemodelan Sistem.....	18
3.2.1. Data Flow Diagram (DFD).....	18
3.2.2. Kamus Data.....	23
3.2.3. Spesifikasi Proses.....	24
3.2.4. Flowchart dan Pseudocode.....	31
3.3. Diagram Umum Sistem.....	69
3.4. File Structure Diagram.....	70
3.5. Perancangan Antarmuka.....	72
BAB 4.....	79
4.1. Spesifikasi Hardware dan Software.....	79
4.2. Implementasi Sistem.....	80
4.3. Pengujian Sistem.....	86
BAB 5.....	99
5.1. Kesimpulan.....	99
5.2. Saran.....	99
DAFTAR PUSTAKA.....	100

DAFTAR TABEL

Tabel 3.0 Kamus Data.....	23
Tabel 3.1 Spesifikasi Proses DFD level 1.....	24
Tabel 3.2 Spesifikasi Proses DFD level 1 Proses 2.....	24
Tabel 3.3 Spesifikasi Proses DFD level 2 Proses 1.1.....	25
Tabel 3.4 Spesifikasi Proses DFD level 2 Proses 1.2.....	25
Tabel 3.5 Spesifikasi Proses DFD level 2 Proses 2.1.....	25
Tabel 3.6 Spesifikasi Proses DFD level 2 Proses 2.2.....	26
Tabel 3.7 Spesifikasi Proses DFD level 3 Proses 1.1.....	26
Tabel 3.8 Spesifikasi Proses DFD level 3 Proses 1.2.....	27
Tabel 3.9 Spesifikasi Proses DFD level 3 Proses 2.1.....	27
Tabel 3.10 Spesifikasi Proses DFD level 3 Proses 2.2.....	27
Tabel 3.11 Spesifikasi Proses DFD level 3 Proses 2.3.....	28
Tabel 3.12 Spesifikasi Proses DFD level 3 Proses 2.4.....	28
Tabel 3.13 Spesifikasi Proses DFD level 3 Proses 2.5.....	29
Tabel 3.14 Spesifikasi Proses DFD level 3 Proses 2.6.....	29
Tabel 3.15 Spesifikasi Proses DFD level 3 Proses 2.7.....	30
Tabel 4.1 Pengujian Deteksi Warna Rubik.....	90
Tabel 4.2 Pengujian Algoritma Penyelesaian Rubik.....	91
Tabel 4.3 Pengujian Pengujian Oleh Pengguna.....	101

DAFTAR GAMBAR

Gambar 2.1 Centers.....	6
Gambar 2.2 Edges.....	7
Gambar 2.3 Corners.....	7
Gambar 2.4 Nama Sisi Rubik.....	7
Gambar 2.5 Sunflower.....	8
Gambar 2.6 White Cross.....	8
Gambar 2.7 First Layer.....	9
Gambar 2.8 Second Layer.....	9
Gambar 2.9 Yellow Cross.....	9
Gambar 2.10 OLL.....	9
Gambar 2.11 PLL.....	10
Gambar 2.12 Desain Penyimpanan Scanning.....	10
Gambar 2.13 Desain Penyimpanan Hasil Scanning Rubik.....	11
Gambar 2.14 Desain Penyimpanan Rubik Akhir.....	11
Gambar 2.15 Desain Penyimpanan Rubik Akhir V2.....	12
Gambar 2.12 HSV Color Space.....	13
Gambar 3.1 DFD level 0.....	18
Gambar 3.2 DFD level 1.....	19
Gambar 3.3 DFD level 2 Proses 1.....	20
Gambar 3.4 DFD level 2 Proses 2.....	20
Gambar 3.5 DFD level 3 Proses 2.1.....	21
Gambar 3.6 DFD level 3 Proses 2.2.....	22
Gambar 3.7 Flowchart Belajar Materi Rubik.....	31

Gambar 3.8 Flowchart Scan Rubik.....	32
Gambar 3.9 Flowchart Scan Sisi Rubik.....	33
Gambar 3.10 Flowchart Scan Sisi Rubik Lanj.....	34
Gambar 3.11 Flowchart Simpan Data Rubik Acak Sisi Kiri.....	35
Gambar 3.12 Flowchart Simpan Data Rubik Acak Sisi Atas.....	36
Gambar 3.13 Flowchart Simpan Data Rubik Acak Sisi Depan.....	37
Gambar 3.14 Flowchart Simpan Data Rubik Acak Sisi Belakang.....	38
Gambar 3.15 Flowchart Simpan Data Rubik Acak Sisi Kanan.....	39
Gambar 3.16 Flowchart Simpan Data Rubik Acak Sisi Bawah.....	40
Gambar 3.17 Flowchart Menerapkan Algoritma Sunflower 1 Bagian Bawah Menghadap Bawah.....	42
Gambar 3.18 Flowchart Menerapkan Algoritma Sunflower 2 Bagian Bawah Tidak Menghadap Bawah.....	43
Gambar 3.19 Flowchart Menerapkan Algoritma Sunflower 3 Bagian Tengah.....	44
Gambar 3.20 Flowchart Menerapkan Algoritma Sunflower 4 Bagian Atas Tidak Menghadap Atas.....	45
Gambar 3.21 Flowchart Menghitung Jumlah Putih Dibagian Atas.....	46
Gambar 3.22 Flowchart Mempersiapkan Tempat di Atas.....	47
Gambar 3.23 Flowchart Menerapkan Algoritma Cross.....	48
Gambar 3.24 Flowchart Menerapkan Algoritma Cross Lanj.....	49
Gambar 3.25 Flowchart Menerapkan Algoritma 1sr Layer / White Corner.....	50
Gambar 3.26 Flowchart Cek Corners Putih.....	51
Gambar 3.27 Flowchart Masukkan Corners.....	52
Gambar 3.28 Flowchart Masukkan Corners Lanj.....	53
Gambar 3.29 Flowchart Perbaiki Corners Yang Salah.....	56
Gambar 3.30 Flowchart Menerapkan Algoritma 2nd Layer.....	57

Gambar 3.31 Flowchart Cek Edges Bukan Kuning di Atas.....	58
Gambar 3.32 Flowchart Masukkan Edges.....	59
Gambar 3.33 Flowchart Masukkan Edges Lanj.....	60
Gambar 3.34 Flowchart Perbaiki Edges.....	61
Gambar 3.35 Flowchart Menerapkan Algoritma Yellow Cross.....	62
Gambar 3.36 Flowchart Cek Bentuk Edges Kuning.....	63
Gambar 3.37 Flowchart Jumlah Edges Kuning dengan Posisi Benar.....	64
Gambar 3.38 Flowchart Jumlah Corner Kuning dengan Posisi Benar.....	66
Gambar 3.39 Diagram Umum Sistem.....	69
Gambar 3.40 Diagram Struktur Berkas Materi Pengenalan.....	70
Gambar 3.41 Diagram Struktur Berkas Materi Tingkat Pemula.....	71
Gambar 3.42 Main Menu Screen.....	72
Gambar 3.43 Item Main Menu Screen.....	73
Gambar 3.44 Main Popup Menu Screen.....	73
Gambar 3.45 Level Menu Screen.....	74
Gambar 3.46 Item Level Menu Screen.....	74
Gambar 3.47 Materi Screen.....	75
Gambar 3.48 Item Materi Screen.....	76
Gambar 3.49 Scan Screen.....	76
Gambar 3.50 Solution Screen.....	77
Gambar 3.51 Item Solution Screen.....	78
Gambar 4.1 Main Menu Screen.....	80
Gambar 4.2 Item Main Menu Screen.....	81
Gambar 4.3 Main Popup Menu Screen.....	81
Gambar 4.4 Level Menu Screen.....	82

Gambar 4.5 Item Level Menu Screen.....	82
Gambar 4.6 Materi Screen.....	83
Gambar 4.7 Item Materi Screen.....	83
Gambar 4.8 Scan Screen.....	84
Gambar 4.9 Solution Screen.....	85
Gambar 4.10 Item Solution Screen.....	85

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Pada tahun 1974 seorang pria bernama Erno Rubik mematenkan sebuah *puzzle* mekanis berbentuk kubus yang telah memikat imajinasi jutaan orang di seluruh dunia. Erno Rubik lahir di tempat penampungan serangan udara rumah sakit Budapest di Hungaria selama Perang Dunia II, ibunya adalah seorang penyair dan ayahnya adalah seorang insinyur penerbangan yang mendirikan perusahaan untuk membangun glider. Rubik sendiri, jauh dari dunia matematika, justru mempelajari arsitektur dan desain di Akademi Seni dan Desain Terapan. Ia kemudian menjadi pengajar di sana dan mengajar desain interior. Menjelang 1982, "*Rubik's Cube*" sudah menjadi istilah yang familiar di masyarakat, dan bahkan tercantum dalam Kamus Bahasa Inggris Oxford. Lebih dari 100 juta *Rubik's Cube* telah terjual di seluruh dunia (Joyner, 2008).

Permainan ini dimainkan dengan cara menyusun enam sisi kubus yang berwarna sehingga tiap sisi menjadi satu warna dengan keterbatasan gerak yang hanya dapat diputar pada poros kubus. Sebelum algoritma penyelesaian Rubik ditemukan, Erno Rubik membutuhkan waktu satu bulan untuk menyelesaikan permainan yang ia ciptakan dan berpikiran bahwa ini merupakan permainan yang mustahil untuk diselesaikan. Rubik standar dikenal sebagai pembuat kombinasi dengan kerumitannya, ada 43.2×10^{18} kemungkinan berbeda yang dihasilkan dari proses pengacakan Rubik dan hanya 1 yang benar (Rohith et al., 2019).

Rubik menjadi salah satu permainan *puzzle* yang paling populer karena permainan yang menyenangkan dan menantang disaat yang bersamaan. Tak hanya itu, tanpa disadari permainan ini memberikan dampak positif terhadap pemainnya seperti dapat memperlambat penurunan fungsi otak dan dapat meningkatkan

perkembangan otak sehingga kinerja otak dapat berjalan dengan optimal, dan tak hanya itu bermain Rubik juga dapat melatih konsentrasi, melatih koordinasi otak dan tangan, menumbuhkan kepekaan warna terhadap anak, meningkatkan kecerdasan visual dan meningkatkan kreativitas dalam pemecahan masalah (Rismayani et al., 2020).

Penting untuk merasakan nikmatnya jika berhasil menyelesaikan rubik untuk pertama kalinya, namun proses belajar menyelesaikan sebuah Rubik merupakan perjalanan yang rumit dan cukup panjang, dan hasilnya akan terlihat jika berhasil mengikuti langkah tersebut dengan benar. Hal tersebut tergolong terlalu panjang untuk ukuran sebuah permainan yang biasanya hanya membutuhkan waktu beberapa menit untuk memahami aturan dan cara mainnya. Proses panjang yang harus diselesaikan secara sistematis mengakibatkan banyak pemula kehilangan arah dan putus asa untuk mempelajari Rubik. Sehingga dengan pendekatan yang salah, Rubik menjadi hal yang menakutkan bagi pemula.

Maka dibutuhkan alternatif cara belajar dan pemilihan metode belajar yang tepat, sehingga pengguna bisa merasakan setidaknya berhasil menyelesaikan rubik walaupun dengan mengikuti panduan. Sampai saat ini, masih belum ada penelitian yang membahas metode mana yang terbaik untuk pemula yang ingin belajar menyelesaikan rubik, namun hasil pencarian penulis di komunitas dan sosial media seperti *Instagram*, *Reddit*, *Quora*, *r/Cubbers*, *Ruwix*, *SpeedSolving* dan *JPerm* menemukan bahwa metode yang disarankan untuk pemula adalah *Beginner Method* atau nama lainnya *Layer by Layer Method* (LBL). Mereka menyarankan metode ini untuk pemula yang baru belajar rubik, dan jika sudah paham dan ingin meningkatkan permainan bisa belajar CFOP (*Cross F2L OLL PLL*).

Adanya tempat belajar dan alat bantu yang dapat membantu pemula untuk memahami Rubik dengan lebih mudah akan memberikan pengalaman yang berbeda ketika pemula belajar Rubik. Dengan demikian penulis membangun aplikasi yang menyediakan materi cara untuk menyelesaikan Rubik serta tutorial untuk menyelesaikan Rubik acak pengguna. Dengan memanfaatkan teknologi *Computer Vision* yang menggunakan kamera pada perangkat pengguna, aplikasi akan mampu menganalisis warna rubik pengguna sehingga memudahkan pengguna dalam proses *input* warna, tidak perlu memasukkan satu persatu sebanyak 54 kali, dan jika terjadi kesalahan *input* maka mudah untuk diperbaiki dibandingkan dengan aplikasi yang sudah ada tapi tidak memiliki fitur pemindaian. Dan berbeda dengan teknologi robot rubik, penelitian ini bertujuan untuk mengajari pengguna untuk menyelesaikan rubiknya sendiri, bukan diselesaikan oleh sebuah alat.

Teknologi *Computer Vision* saat ini marak digunakan diberbagai bidang kehidupan. *Computer Vision* merupakan program dan rancangan algoritma untuk memahami apa yang ada atau yang terjadi pada sebuah gambar (Solem, 2012). Teknologi ini dapat bekerja sebagaimana kerja mata manusia, dengan memanfaatkan kamera pada perangkat pengguna dan diajarkan untuk dapat mengidentifikasi warna dan posisinya pada Rubik pengguna. Dengan membangun program berdasarkan algoritma penyelesaian Rubik dan fundamental dari sebuah Rubik berupa bentuk, posisi, warna, aturan-aturan, bagian yang diputar dan tidak. Maka program dapat menyelesaikan Rubik acak dan mengeluarkan algoritma penyelesaian kepada pengguna. Sehingga pengguna mengetahui langkah-langkah cara menyelesaikan Rubik acaknya. Dengan adanya aplikasi dapat meningkatkan pengalaman dalam belajar menyelesaikan Rubik sehingga pemula tidak langsung menyerah ditengah jalan.

1.2. Rumusan Masalah

Kesulitan pemula dalam belajar menyelesaikan Rubik karena notasi dan algoritma yang rumit, dari permasalah tersebut dibutuhkan alat bantu yang dapat membuat pengalaman belajar Rubik menjadi terarah dan mudah dipahami dengan cara memberi materi pembelajaran yang mudah dipahami dan proses *input* datanya sederhana dan tidak rumit.

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membangun sebuah aplikasi yang dapat meningkatkan efisiensi dan efektivitas dalam belajar menyelesaikan permainan Rubik.

1.4. Batasan Masalah

Beberapa batasan masalah dalam penelitian sebagai berikut.

1. Program hanya dapat digunakan pada Rubik 3x3x3.
2. Program khusus menyelesaikan Rubik dengan warna “standar”.
3. Aplikasi yang dibangun hanya dapat digunakan pada perangkat Android.
4. Program tidak dapat mendeteksi rotasi atau gerak putaran pada Rubik.
5. Program menyelesaikan Rubik acak dengan metode *Layer by layer*.

1.5. Manfaat Penelitian

Manfaat dari penelitian ini sebagai berikut.

1. Menyediakan tempat untuk belajar menyelesaikan Rubik yang mudah diakses.
2. Memberikan pengalaman belajar menyelesaikan Rubik yang sulit dan kompleks menjadi lebih ringan dan menyenangkan.
3. Membuat belajar menyelesaikan Rubik lebih mudah dan terarah.
4. Dapat menjadi inspirasi untuk penelitian selanjutnya dalam pengembangan aplikasi pembelajaran menggunakan teknologi computer vision.

1.6. Metodologi Penelitian

Metode yang digunakan dalam penelitian ini adalah:

1. Studi Pustaka

Pada tahap ini, penulis mulai mencari referensi dari berbagai sumber melalui Buku, Jurnal, Artikel Ilmiah serta Makalah yang berhubungan dengan Computer Vision dan Algoritma dalam menyelesaikan Rubik.

2. Analisa dan Perancangan

Pada tahap ini, penulis melakukan analisis terhadap hal yang dibutuhkan dalam penelitian dan divisualisasikan dalam sebuah Diagram Alir/Flowchart Diagram Ishikawa.

3. Implementasi

Pada tahap ini, penulis membangun aplikasi dengan menggunakan *Android Studio* sesuai dengan diagram alir yang telah dirancang.

4. Pengujian

Pada tahap ini, penulis melakukan pengujian terhadap aplikasi dengan mengacak sebuah Rubik lalu aplikasi mengidentifikasi bentuk Rubik, output berupa rumus untuk menyelesaikan Rubik tersebut. Dikatakan berhasil apabila rumus tersebut berhasil menyusun Rubik acak tersebut.

5. Dokumentasi

Pada tahap ini, penulis melakukan dokumentasi terhadap penelitian yang telah dilakukan mulai dari tahap analisis sampai kepada pengujian sistem dalam bentuk skripsi.

1.7. Sistematika Penulisan

Penelitian ini tersusun menjadi 5 bab sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini terdiri dari penjelasan mengapa judul ini dipilih dan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi yang digunakan, serta susunan kepenulisan yang dijelaskan secara lengkap pada bab ini.

BAB 2 LANDASAN TEORI

Bab ini berisi mengenai studi pustaka yang penulis lakukan meliputi teori-teori relevan yang mendukung penelitian yang dilakukan penulis, seperti teori mengenai *computer vision* dan rubik serta penelitian yang berkaitan dengan pembangunan sistem pemecahan rubik.

BAB 3 ANALISA DAN PERANCANGAN SISTEM

Bab ini membahas tentang analisis masalah terkait penelitian dan perancangan pembangunan aplikasi sebagai penyelesaian dari masalah berdasarkan analisis yang dilakukan.

BAB 4 IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan menjelaskan penerapan hasil analisa dan perancangan yang sudah dijelaskan pada bab 3. Bab ini memiliki tujuan untuk membuktikan perancangan dan metode yang digunakan sesuai dengan yang diharapkan. Pada bab ini juga akan dijelaskan hasil dari proses uji sistem yang sudah dibuat.

BAB 5 KESIMPULAN DAN SARAN

Bab ini menjelaskan mengenai kesimpulan yang didapat setelah melakukan penelitian serta saran yang diharapkan akan membantu penelitian-penelitian selanjutnya.

BAB 2

LANDASAN TEORI

2.1. Rubik

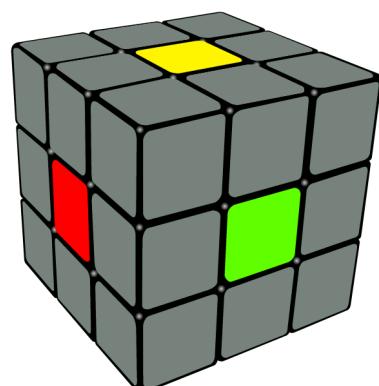
Rubik merupakan permainan *puzzle* yang diciptakan oleh arsitek dan profesor Hongaria Erno Rubik pada tahun 1974. *Puzzle* ini telah menjadi populer di seluruh dunia dan menarik perhatian para analisis matematika dan algoritmik. Rubik telah terbukti memiliki berbagai sifat yang membuatnya menjadi *puzzle* yang menarik untuk dipelajari, termasuk jumlah kemungkinan kombinasi lebih dari 43 triliun dan penggunaan teori grup dan kombinatorika untuk menemukan solusi optimal.

Rubik juga menarik peneliti di bidang psikologi kognitif untuk melakukan studi untuk mempelajari strategi pemecahan masalah dan kemampuan rotasi mental. Studi oleh Shepard dan Metzler menemukan bahwa memecahkan Rubik memerlukan rotasi mental, atau kemampuan untuk memanipulasi objek secara mental di ruang (Shepard & Metzler, 1971). Studi lain oleh Wai dkk menemukan bahwa orang yang menyelesaikan Rubik secara cepat lebih cenderung menggunakan algoritma dan kurang mengandalkan strategi coba-coba (Wai et al., 2009).

Rubik sendiri terdiri tersusun dari 26 keping kubus kecil yang dapat dikategorikan mencari tiga macam yaitu:

- a. *Center* (tengah rubik)

Centers adalah bagian rubik yang posisinya berada ditengah setiap sisi rubik. Pada rubik ada enam *centers* yang mana setiap warna memiliki satu *centers* dan setiap *centers* terdiri dari satu warna. Posisi *centers* rubik tidak berubah satu terhadap yang lain, misalkan *center* putih

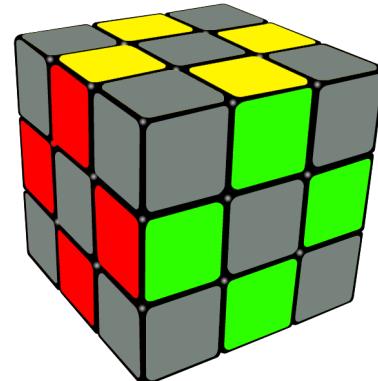


Gambar 2.1 *Centers*

akan selalu berseberangan dengan *center* kuning dan begitu pula *centers* yang lainnya. Dan untuk lebih jelas posisi *centers* dapat dilihat pada gambar 2.1.

b. *Edge* (tepi rubik)

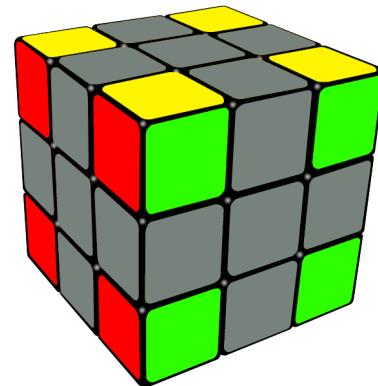
Edges adalah bagian rubik yang memiliki dua warna dalam satu kepingnya. Pada rubik ada 12 keping *edges* dan merupakan kepingan terbanyak pada rubik 3x3x3. Dan untuk lebih jelas posisi *edges* dapat dilihat pada gambar 2.2.



Gambar 2.2 *Edges*

c. *Corner* (sudut rubik)

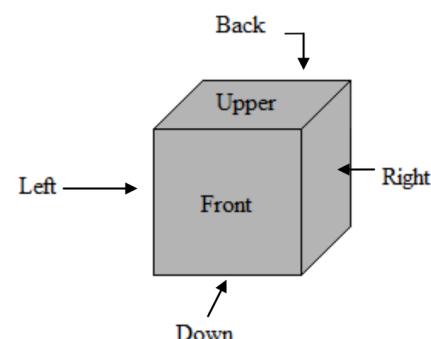
Corners adalah bagian rubik yang memiliki tiga warna dalam satu kepingnya. Pada rubik terdapat delapan *corners*. Seperti namanya kepingan ini terdapat pada setiap sudut rubik. Dan untuk lebih jelas posisi *corners* dapat dilihat pada gambar 2.3.



Gambar 2.3 *Corners*

Selain itu setiap sisi dari rubik memiliki nama dan dilabeli dengan huruf sebgain berikut:

- Depan / *Front* (F) merupakan sisi depan yang menghadap langsung ke pengguna,
- Belakang / *Back* (B) merupakan sisi belakang dan bagian yang paling sulit dilihat pengguna,
- Atas / *Upper* (U) merupakan sisi atas yang menghadap ke langit dan biasanya berwarna kuning,
- Bawah / *Down* (D) merupakan sisi bawah yang menghadap ke lantai dan biasanya disebut juga



Gambar 2.4 Nama Sisi Rubik

warna dasar. Bagian ini berwarna putih (warna yang beseberangan dengan warna atas yaitu kuning),

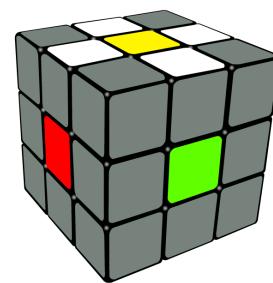
- Kiri / *Left* (L) merupakan sisi kiri dan tempat tangan kiri pengguna berada,
- Kanan / *Right* (R) merupakan sisi kanan dan tempat tangan kanan pengguna berada.

Dan dalam permainan rubik ada yang namanya **Notasi** yang merupakan representasi dari instruksi putaran pada sisi rubik. Notasi pada sisi rubik didasarkan pada setiap sisi rubik seperti yang diapaparkan di atas, ada enam sisi yang bisa diputar dan dilambangkan dengan huruf kapital yaitu F, B, U, D, R, L. Notasi huruf berarti putar sepertempat putaran sisi rubik searah jarum jam, misal notasi F berarti putar sisi depan rubik sepertempat putaran searah jarum jam. Lalu ada pula putara berlawanan arah jarum jam yang dilambangkan dengan tanda petik atau aksen setelah huruf, misal F' (dibaca F aksen) artinya putar sisi depan rubik seperempat putaran berlawanan arah jarum jam. Lalu ada putar setengah putaran yang dilambangkan dengan penambahan angka dua setelah huruf, misal F2 yang berarti putar sisi depan rubik setengah putaran dan bisa diputar kearah mana saja karena hasilnya sama saja.

Lalu untuk menyelesaikan sebuah rubik ada banyak cara yang bisa kita pelajari, namun untuk penelitian ini penulis akan menggunakan metode *Layer by layer* yang merupakan cara yang cukup sederhana dan banyak digunakan oleh orang-orang yang baru belajar bermain rubik. Cara ini ada tujuh tahapan yang perlu dilalui yaitu:

1. *Sunflower*

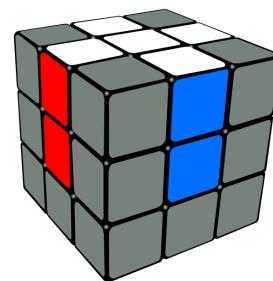
Pada tahap ini tujuan kamu adalah memposisikan keempat edges kuning pada center putih seperti pada gambar 2.5. Untuk tiga edges akan mudah dilakukan namun untuk edges keempat akan sedikit sulit.



Gambar 2.5
Sunflower

2. *White Cross*

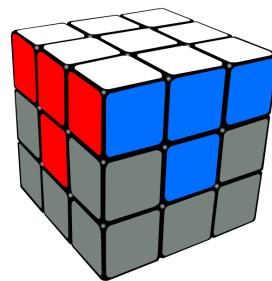
Pada tahap ini bertujuan untuk meletakkan keempat edges putih pada center putih dan warna lain edges berada pada center yang tepat sesuai gambar 2.6.



Gambar 2.6 *White Cross*

3. First Layer/ White Corner

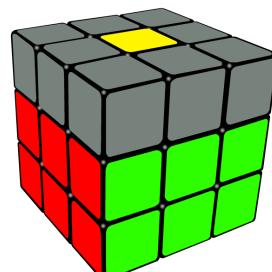
Pada tahap ini bertujuan setiap corners yang memiliki warna putih pada tempat yang tepat, sehingga ketiga warna pada corners berada pada center yang tepat, dan dilakukan pada semua corners seperti pada gambar 2.7.



Gambar 2.7 First Layer

4. Second Layer

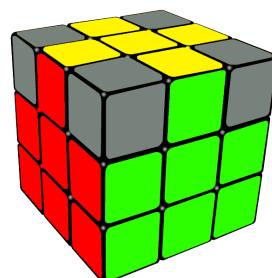
Pada tahap ini kamu akan meletakkan edges pada tempat yang tepat, sehingga rubik akan seperti gambar 2.8.



Gambar 2.8 Second Layer

5. Yellow Cross

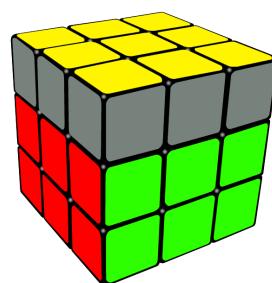
Pada tahap ini kamu akan meletakkan edges kuning pada center kuning, sehingga rubik akan seperti gambar 2.9.



Gambar 2.9 Yellow Cross

6. OLL

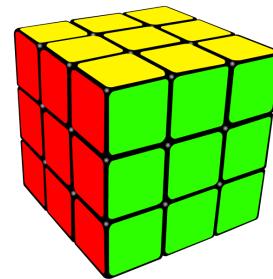
Pada tahap ini tujuannya adalah menyelesaikan warna kuning (warna puncak) seperti gambar 2.10. Pada tahap ini ada banyak algoritma yang disediakan namun dapat dipermudah dengan menggunakan algoritma yang sama berulang-ulang. Pada tahap ini ada banyak algoritma yang disediakan namun dapat dipermudah dengan menggunakan algoritma yang sama berulang-ulang.



Gambar 2.10 OLL

7. PLL

Ini merupakan tahap akhir, bagian atas rubik semunya sudah berwarna kuning namun belum pada posisi yang tepat, kita perlu merubah itu sehingga menjadi selesai seperti gambar 2.11. Pada tahap ini ada banyak algoritma yang disediakan namun dapat dipermudah dengan menggunakan algoritma yang sama berulang-ulang hingga selesai atau hanya posisi edges saja yang salah.



Gambar 2.11 PLL

Dalam menerjemahkan konsep rubik ini dapat dipahami oleh komputer, penulis merancang rekayasa rubik. Secara sederhana ada tiga tahapan yang dibutuhkan dalam menerjemahkan rubik nyata pengguna menjadi digital yang nanti sistem akan menganalisis rubik tersebut untuk mencari penyelesaiannya. Tahapan yang dilakukan dalam menerjemahkan rubik nyata menjadi digital dapat dilakukan sebagai berikut:

1. Scanning rubik sisi rubik dan disimpan sementara dalam bentuk *array* [3][3], hal ini dilakukan karena dalam satu sisi rubik berukuran 3x3 sehingga memiliki sembilan warna pada satu sisi dan masih disimpan dalam bentuk dua dimensi. Bentuk rancangan data dapat divisualisasi seperti gambar 2.12 berikut:

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

Gambar 2.12 Desain Penyimpanan Scanning

2. Lalu disimpan dalam bentuk *array* [6][3][3] dikarenakan rubik memiliki enam sisi, array baru ini diperlukan untuk dapat menampung data semua sisi rubik dan di tahap ini menyimpan data rubik dalam bentuk enam buah data sisi dua dimensi. Bentuk rancangan data dapat divisualisasi seperti gambar 2.13 berikut:

1,0,0	1,0,1	1,0,2
1,1,0	1,1,1	1,1,2
1,2,0	1,2,1	1,2,2

0,0,0	0,0,1	0,0,2
0,1,0	0,1,1	0,1,2
0,2,0	0,2,1	0,2,2

2,0,0	2,0,1	2,0,2
2,1,0	2,1,1	2,1,2
2,2,0	2,2,1	2,2,2

3,0,0	3,0,1	3,0,2
3,1,0	3,1,1	3,1,2
3,2,0	3,2,1	3,2,2

4,0,0	4,0,1	4,0,2
4,1,0	4,1,1	4,1,2
4,2,0	4,2,1	4,2,2

5,0,0	5,0,1	5,0,2
5,1,0	5,1,1	5,1,2
5,2,0	5,2,1	5,2,2

Gambar 2.13 Desain Penyimpanan Hasil *Scanning* Rubik

3. Terakhir disimpan dalam bentuk *array* [3][3][3] dikarenakan rubik bebentuk tiga dimensi dan berdimensi 3x3x3, maka data harus disimpan sebagaimana rubik itu berbentuk. Bentuk rancangan data dapat divisualisasi seperti gambar 2.14 berikut:

0,0,0	1,0,0	2,0,0
0,0,1	1,0,1	2,0,1
0,0,2	1,0,2	2,0,2

0,1,0	1,1,0	2,1,0
0,1,1	1,1,1	2,1,1
0,1,2	1,1,2	2,1,2

0,2,0	1,2,0	2,2,0
0,2,1	1,2,1	2,2,1
0,2,2	1,2,2	2,2,2

Gambar 2.14 Desain Penyimpanan Rubik Akhir

Jika data divisualisasi dalam bentuk jaring-jaring rubik, dapat dilihat pada gambar 2.15

0,2,0	1,2,0	2,2,0
0,1,0	1,1,0	2,1,0
0,0,0	1,0,0	2,0,0

0,2,0	0,1,0	0,0,0
0,2,1	0,1,1	0,0,1
0,2,2	0,1,2	0,0,2

0,0,0	1,0,0	2,0,0
0,0,1	1,0,1	2,0,1
0,0,2	1,0,2	2,0,2

2,0,0	2,1,0	2,2,0
2,0,1	2,1,1	2,2,1
2,0,2	2,1,2	2,2,2

2,2,0	1,2,0	0,2,0
2,2,1	1,2,1	0,2,1
2,2,2	1,2,2	0,2,2

0,0,2	1,0,2	2,0,2
0,1,2	1,1,2	2,1,2
0,2,2	1,2,2	2,2,2

Gambar 2.15 Desain Penyimpanan Rubik Akhir V2

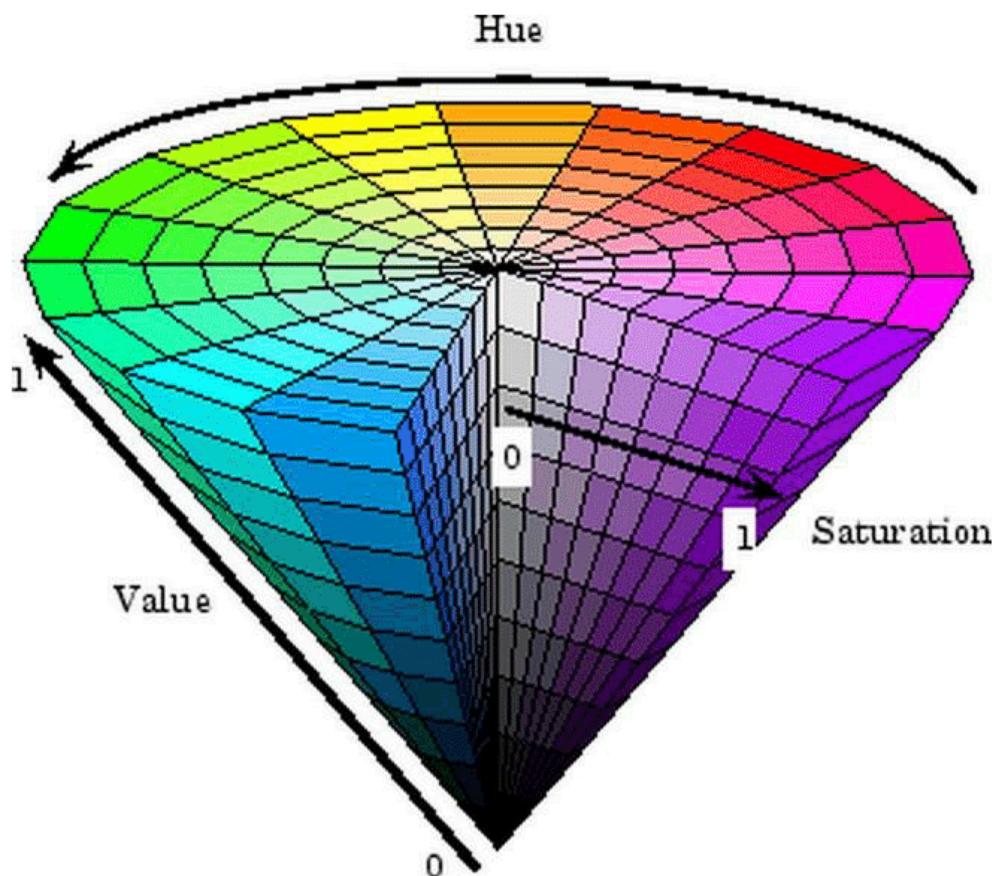
2.2. Computer Vision

Computer vision merupakan program dan rancangan algoritma untuk memahami apa yang ada atau yang terjadi pada sebuah gambar (Solem, 2012). Menurut buku "*Computer Vision: Algorithms and Applications*" karya Richard Szeliski, *computer vision* melibatkan pengembangan algoritma dan model matematika yang memungkinkan komputer untuk mengenali dan menafsirkan objek, adegan, dan pola dalam data visual, seperti gambar dan video. Hal ini melibatkan penggunaan berbagai teknik seperti pengolahan gambar, *pattern recognition*, dan *machine learning* (Szeliski, 2010).

Sederhananya seperti memberi komputer sebuah mata sehingga mampu untuk memahami dan menganalisis keadaan di dunia nyata seperti manusia. Contoh penggunaan teknologi ini yang sering dijumpai disekitar kita seperti *face recognition* yang banyak terdapat pada smartphone yang mampu mengenali wajah seseorang dan tilang elektronik yang mendeteksi pelanggaran lalu lintas dan merekam plat kendaraan yang melanggar.

Computer vision sudah berkembang ke berbagai bidang kehidupan termasuk robotika, kedokteran, pengawasan atau pengamanan, dan hiburan. Dalam robotika, *computer vision* memungkinkan robot untuk menavigasi dan berinteraksi dengan lingkungannya. Dalam kedokteran, *computer vision* digunakan untuk tugas seperti analisis gambar medis dan

diagnosis. Dalam keamanan dan pengawasan, *computer vision* biasa digunakan untuk tugas seperti pengenalan wajah dan pelacakan objek. Dalam hiburan, *computer vision* digunakan untuk membuat efek khusus dalam film dan video game (Ponce & Forsyth, 2012).



Gambar 2.12 HSV Color Space

Dalam perancangan sistem ini, penulis memanfaatkan *library* OpenCV untuk melakukan deteksi warna rubik atau *Image Recognition*. Warna dideteksi dengan melakukan *Image Processing* pada gambar yang didapat pada *Camera Preview*, lalu mengubah gambar dari *One Color Space* menjadi *HSV Color Space*. Dengan memiliki nilai HSV pada gambar kita dapat mengklasifikasikan warna pada gambar tersebut.

Sebelumnya sudah sedikit disinggung mengenai OpenCV, OpenCV sendiri adalah C++ *library* untuk *real-time computer vision* dan awalnya dikembangkan oleh Intel namun saat ini dikelola oleh Willow Garage. *Library* ini *open source* dan diluncurkan di bawah lisensi BSD, yang artinya bebas digunakan untuk belajar maupun komersial (Solem, 2012, 239).

2.3. Pengajaran Berbantuan Komputer

Pengajaran berbantuan komputer (PBK) atau *computer assisted teaching* (CAT) merupakan suatu metode yang memanfaatkan komputer dalam proses belajar mengajar. PBK sudah berkembang sangat pesat terutama di masa pandemi yang menerapkan cara belajar mengajar baru menggunakan berbagai aplikasi dan perangkat pendukung. Dan yang terbaru sebuah sekolah di China sudah menerapkan teknologi kecerdasan buatan (AI) untuk membantu pengajar mengetahui tingkat fokus dan pemahaman siswanya.

Menurut buku "*Handbook of Research on Educational Communications and Technology*", pengajaran berbantuan komputer dapat berupa bentuk apa saja seperti tutorial berbasis komputer, simulasi, game pendidikan, kursus online, dan presentasi multimedia. Ini dapat digunakan untuk mengajar berbagai mata pelajaran, mulai dari matematika dan ilmu pengetahuan hingga bahasa dan studi sosial (Spector et al., 2007).

Pengajaran berbantuan komputer dirancang untuk meningkatkan motivasi, dan hasil pembelajaran siswa dengan memberikan pengalaman belajar yang interaktif, personal, dan adaptif. Hal ini memungkinkan siswa untuk bekerja dengan kecepatan mereka sendiri, menerima umpan balik segera, dan mengakses berbagai sumber belajar, termasuk teks, gambar, audio, dan video. Secara umum, pengajaran berbantuan komputer ini memberi dampak positif dalam dunia pendidikan. Sebuah meta-analisis dari penelitian tentang efektivitas PBK menemukan bahwa hal itu berhubungan dengan tingkat pencapaian yang lebih tinggi dan keterlibatan siswa yang lebih besar (Tamim et al., 2011).

2.4. Penelitian Terdahulu

1. Penelitian yang dilakukan oleh (Liu et al., 2019) dengan judul “*Color Recognition for Rubik's Cube Robot*” menemukan terjadinya pergeseran warna atau *color drifting* yang kerap terjadi ketika menggunakan metode offline yakni Scatter balance & extreme learning machine (SB-ELM), hal ini dapat disebabkan oleh pencahayaan, perbedaan kamera, material Rubik, dan pemudaran warna Rubik. Peneliti membandingkan metode ini dengan metode online yakni (Weak label hierachic propagation (WLHP) dan Dynamic weight label propagation (DWLP)), dan menemukan bahwa metode online lebih baik dengan pembuktian dilakukan pada lingkungan dunia nyata dan memuaskan sebagian besar pengguna.
2. Berdasarkan hasil penelitian yang dilakukan oleh (Lyu et al., 2022) dengan judul “*Q-learning and traditional methods on solving the pocket Rubik's cube*” yang membandingkan kinerja antara *reinforcement learning (Q-Learning)* dengan tiga metode tradisional (LP formulation, Breadth-first search, Dijkstra's algorithm) menghasilkan bahwa metode tradisional dapat menyelesaikan permasalahan secara efisien dan jaminan 100% efektif sedangkan Q-learning tidak menghasilkan jaminan 100% berhasil dan membutuhkan waktu yang sangat lama untuk menjadi optimal. Namun Q-learning ini dapat diperluas untuk menyelesaikan Rubiks 3x3x3 yang mana metode tradisional tidak dapat melakukannya.
3. Berdasarkan jurnal “*Autonomous Rubik's Cube Solver Bot*” setelah peneliti menyelesaikan penelitiannya tak hanya manusia yang kesulitan dalam memecahkan *puzzle* Rubik, namun robot jauh lebih sulit dalam memecahkannya. Dibutuhkan *image processing, robot kinetics* dan pengembangan algoritma dalam membangun robot ini. (Rohith et al., 2019).

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis Sistem

Pada bagian ini, akan dilakukan analisis mendalam terhadap sistem yang akan dikembangkan, dengan fokus pada pemahaman masalah yang ingin diatasi dan kebutuhan yang harus dipenuhi.

3.1.1. Analisis Masalah

Tahap analisis masalah merupakan tahap yang dilakukan untuk mengetahui dan menganalisa masalah, serta mencari solusi dari permasalahan tersebut. Sehingga bagian ini, akan dilakukan analisis terhadap permasalahan yang ingin dipecahkan melalui pembuatan aplikasi belajar menyelesaikan Rubik 3x3x3 berbasis Android dengan pemanfaatan computer vision. Salah satu pendekatan yang dilakukan untuk menganalisa masalah adalah dengan *Five Whys Analysis* (Analisa Lima Kenapa).

Analisis 5 Whys (5 WHYS) adalah sebuah metode *interrogatif iteratif* yang digunakan untuk menggali akar penyebab dari suatu masalah. Dengan kata lain, teknik ini membantu untuk menelusuri lapisan demi lapisan permasalahan hingga mencapai penyebab utamanya dengan cara terus-menerus menanyakan "Kenapa?" sebanyak lima kali atau lebih. Berikut adalah 5 WHYs penelitian ini.

Kenapa orang kesulitan belajar menyelesaikan Rubik?

1. Karena Rubik memiliki banyak kombinasi dan langkah yang rumit.
2. Kurangnya panduan dan tutorial yang mudah dipahami.
3. Orang-orang tidak memiliki waktu dan kesabaran untuk belajar menyelesaikan Rubik.

Kenapa Rubik memiliki banyak kombinasi dan langkah yang rumit?

1. Rubik 3x3x3 memiliki 43 quintillion kemungkinan kombinasi.
2. Algoritma untuk menyelesaikan Rubik membutuhkan banyak langkah dan rumus.

Kenapa kurangnya panduan dan tutorial yang mudah dipahami?

1. Tutorial dan panduan sering kali menggunakan bahasa yang rumit dan istilah teknis.
2. Tutorial dan panduan tidak selalu menunjukkan langkah-langkah secara visual.
3. Tutorial dan panduan tidak interaktif dan tidak dapat memberikan umpan balik kepada pengguna.

Kenapa orang-orang tidak memiliki waktu dan kesabaran untuk belajar menyelesaikan Rubik?

1. Belajar menyelesaikan Rubik membutuhkan waktu dan dedikasi.
2. Proses belajar Rubik bisa membingungkan dan membuat frustasi.
3. Orang-orang lebih memilih aktivitas lain yang lebih mudah dan cepat.

Kenapa belajar menyelesaikan Rubik membutuhkan waktu dan dedikasi?

1. Butuh waktu untuk memahami dan membiasakan diri dengan notasi Rubik.
2. Mempelajari algoritma dan langkah-langkah yang panjang menyelesaikan Rubik membutuhkan waktu.
3. Mengatasi frustasi dan kebingungan selama proses belajar membutuhkan kesabaran.

Akar Masalah (Root Cause):

Berdasarkan analisis 5 Whys, terdapat beberapa faktor yang menyebabkan orang kesulitan belajar menyelesaikan Rubik. Faktor-faktor tersebut antara lain:

- a. Kompleksitas Rubik
- b. Kurangnya panduan dan tutorial yang mudah dipahami
- c. Kurangnya waktu dan kesabaran

3.1.2. Requirements Analysis

Bagian ini akan menganalisis terhadap kebutuhan yang harus dipenuhi oleh aplikasi belajar menyelesaikan Rubik 3x3x3 berbasis Android dengan pemanfaatan computer vision.

3.1.2.1. Functional Requirements

Pada penelitian ini, *functional requirements* (kebutuhan fungsional) merupakan kebutuhan yang harus dimiliki oleh aplikasi sehingga dapat menjalankan fungsi dasarnya dan

menghasilkan *output* yang diinginkan. Kebutuhan fungsional dari penelitian dapat dilihat pada daftar detail fitur yang harus dimiliki oleh aplikasi, seperti:

1. Materi pembelajaran mulai dari pengenalan rubik hingga cara menyelesaiakannya,
2. Pendekripsi warna rubik milik pengguna,
3. Algoritma penyelesaian dari rubik acak pengguna,
4. Panduan langkah demi langkah cara menyelesaikan rubik acak pengguna,
5. Dan visualisasi untuk memudahkan pengguna menyelesaikan rubiknya.

3.1.2.2. Non-Functional Requirements

Pada penelitian ini, *non-functional requirements* (kebutuhan non-fungsional) dari sebuah aplikasi adalah kriteria yang dapat dijadikan tolak ukur terhadap aplikasi. Kebutuhan non-fungsinal dari penelitian ini adalah sebagai berikut:

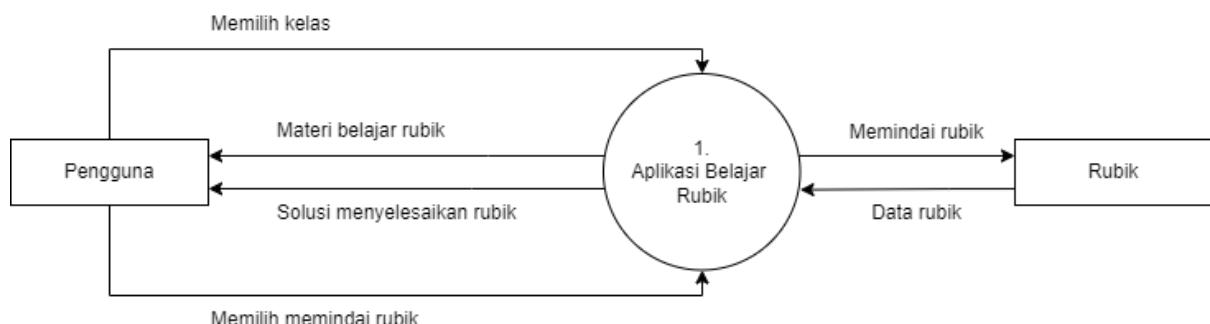
1. Spesifikasi performa aplikasi, termasuk responsivitas, kecepatan deteksi, dan akurasi.
2. Kebutuhan tampilan antarmuka yang intuitif dan mudah digunakan oleh pengguna.

3.2. Pemodelan Sistem

Dalam sub-bab ini, akan dilakukan pemodelan sistem untuk memberikan gambaran lebih rinci tentang bagaimana aplikasi akan bekerja dan berinteraksi dengan pengguna.

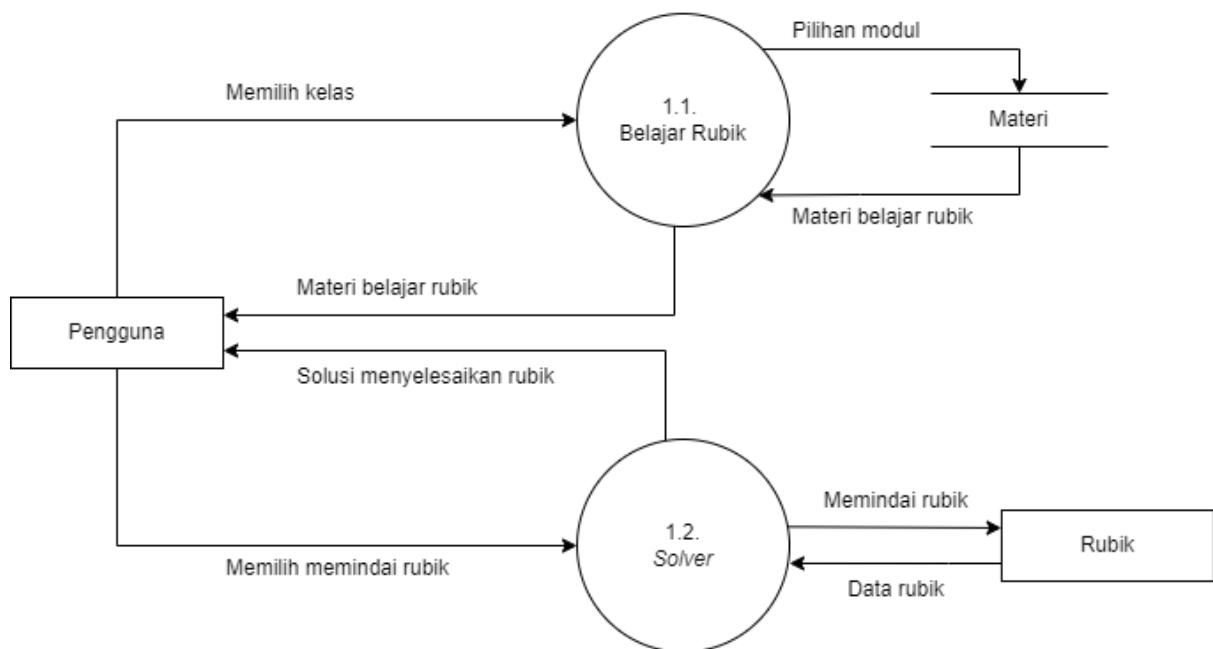
3.2.1. Data Flow Diagram (DFD)

Data Flow Diagram (DFD) adalah teknik pemodelan sistem yang menggambarkan aliran data bergerak dalam suatu sistem. DFD menggunakan simbol-simbol untuk menggambarkan proses, data, dan penyimpanan data.



Gambar 3.1 DFD level 0

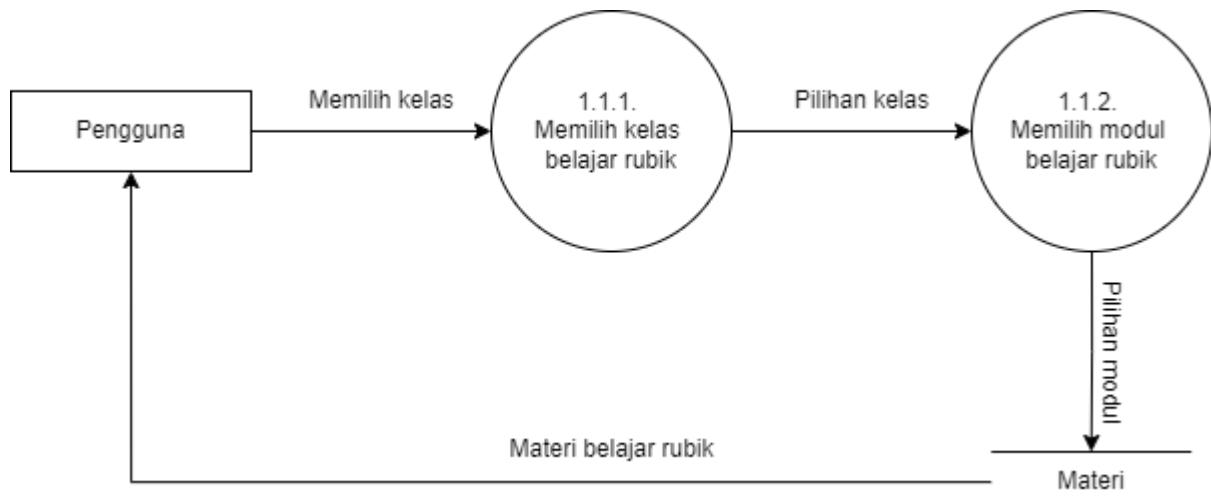
Gambar 3.2 merupakan bentuk DFD level 0 yang menjelaskan aliran dari aplikasi belajar Rubik. Dapat dilihat ada satu proses yaitu proses Aplikasi Belajar Rubik. Di proses ini terjadi interaksi dengan dua entitas yaitu entitas Pengguna dan entitas Rubik. Interaksi dengan pengguna memiliki dua *input* yaitu input memilih kelas dan milih memindai kelas, serta memiliki dua *output* yaitu materi belajar rubik dan solusi menyelesaikan rubik. Interaksi dengan rubik memiliki satu *input* yaitu memindai rubik dan memiliki satu *output* yaitu data rubik.



Gambar 3.2 DFD level 1

Gambar 3.2 merupakan bentuk DFD level 1 yang menjelaskan aliran pada proses aplikasi Belajar Rubik. Dapat dilihat ada dua proses yaitu:

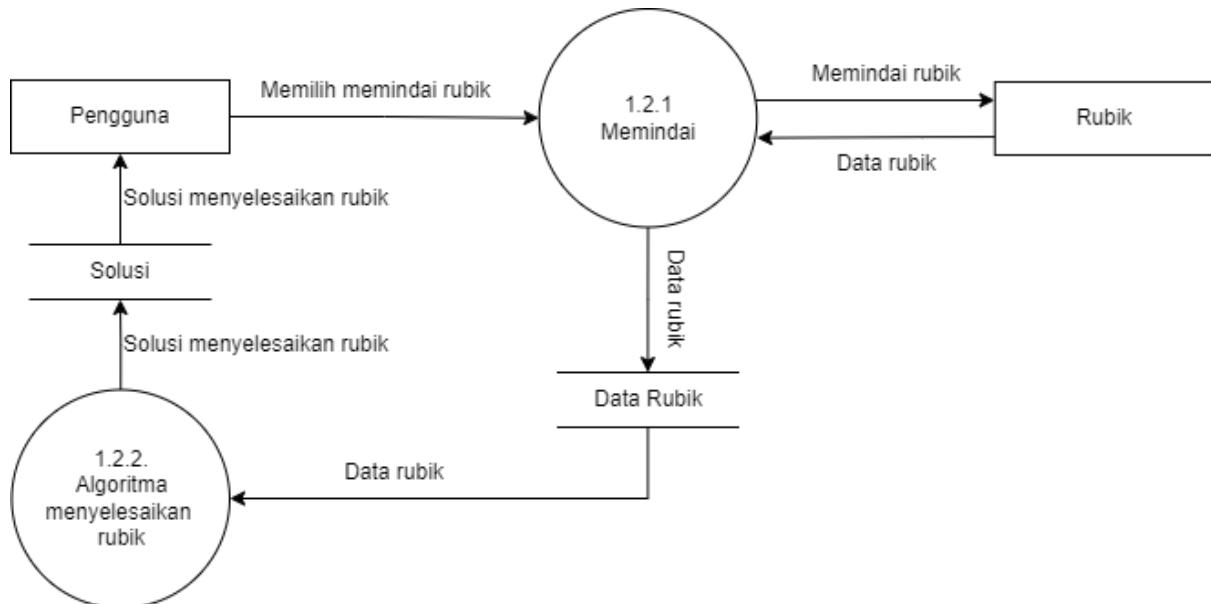
- Belajar Rubik: Proses ketika pengguna ingin belajar, dari input pilihan pengguna lalu data diambil di database yang selanjutnya ditampilkan ke pengguna.
- *Solver*: Proses untuk mencari solusi dari rubik acak pengguna dengan memindai rubik tersebut.



Gambar 3.3 DFD level 2 Proses 1

Gambar 3.3 merupakan bentuk DFD level 2 Proses 1 yang menjelaskan aliran pada aplikasi ketika pengguna memilih Belajar Rubik. Dapat dilihat ada dua proses yaitu:

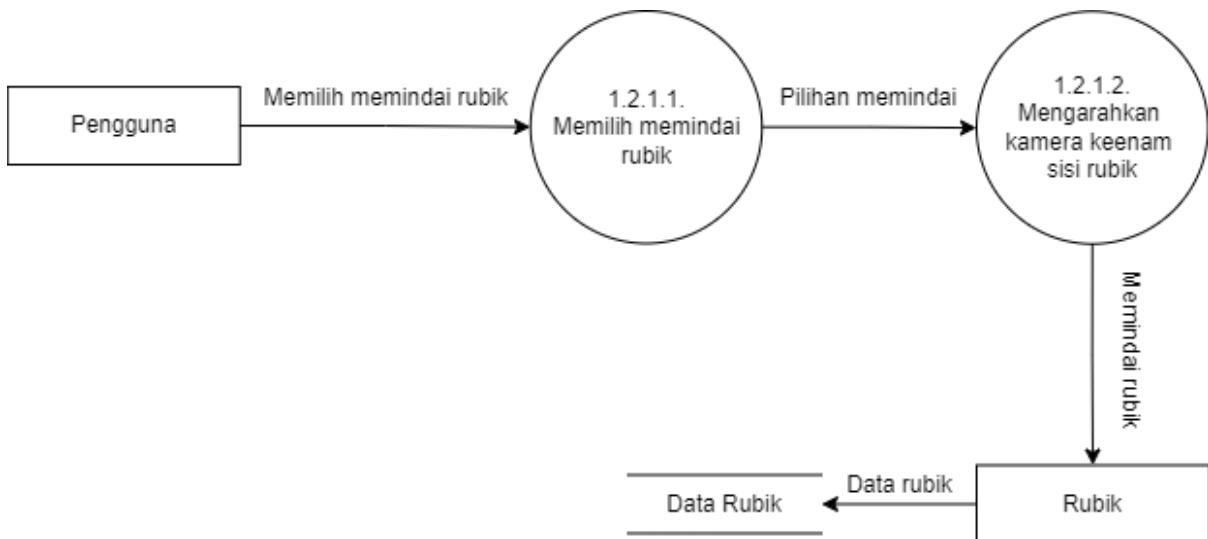
- Memilih kelas belajar Rubik: Ada beberapa pilihan materi untuk dipelajari.
- Memilih modul: Dari setiap kelas terdiri dari beberapa modul, dan pilihan modul dikirim ke database Materi untuk mendapatkan materi yang dibutuhkan.



Gambar 3.4 DFD level 2 Proses 2

Gambar 3.4 merupakan bentuk DFD level 2 Proses 2 yang menjelaskan aliran pada aplikasi ketika memilih *Solver*. Di dalam proses ini ada dua proses yaitu:

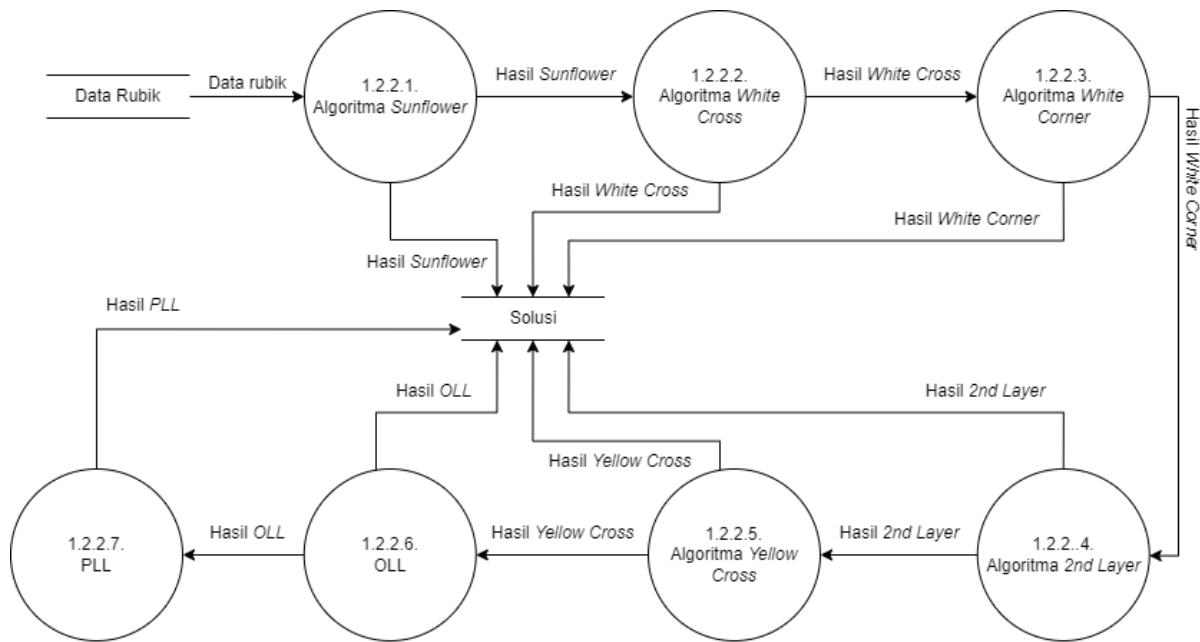
- Memindai: Merupakan proses ketika pengguna memindai rubik untuk mendapatkan data warna dan posisi rubik yang selanjutnya disimpan pada Data Rubik.
- Algoritma menyelesaikan rubik: Berisi algoritma penyelsaian rubik menggunakan data rubik acak untuk diselesaikan.



Gambar 3.5 DFD level 3 Proses 2.1

Gambar 3.5 merupakan bentuk DFD level 3 Proses 2.1 yang menjelaskan aliran pada aplikasi ketika pengguna memilih pemindaian. Dapat dilihat ada dua proses yaitu:

- Memilih menu pemindaian untuk memulai pemindaian
- Mengarahkan kamera *smartphone* kearah enam sisi rubik untuk dipindai



Gambar 3.6 DFD level 3 Proses 2.2

Gambar 3.5 merupakan bentuk DFD level 3 Proses 2.1 yang menjelaskan aliran pada aplikasi ketika pemindaian telah dilakukan dan lanjut menjacari solusi penyelesaian. Dapat dilihat ada tujuh proses yaitu:

- Algoritma *Sunflower* merupakan tahap awal dari algorima untuk membentuk *edges* putih pada sisi kuning
- Algoritma *White Cross* merupakan algoritma untuk menyusun *edges* putih pada sisi putih
- Algoritma *White Corner* tahap untuk untuk menyelesaikan sisi putih
- Algoritma *2nd Layer* sesuai namanya tahapan ini untuk menyelesaikan lapisan kedua pada rubik dengan memasukkan *edges* yang bukan kuning ke tempat yang tepat
- Algoritma *Yellow Cross* dibentuk dengan cara memasukkan *edges* kuning pada sisi kuning
- Algoritma *OLL* bertujuan untuk menyelesaikan sisi warna kuning
- Algoritma *PLL* merupakan tahap akhir yang bertujuan menyelesaikan sisi atas rubik yang belum terselesaikan

3.2.2. Kamus Data

Kamus data adalah sebuah dokumen yang berisi definisi dan deskripsi terperinci tentang elemen data yang digunakan dalam suatu sistem.

Tabel 3.0 Kamus Data

No	Nama	Definisi	Tipe Data
1	Memilih kelas	Pilihan kelas dari pengguna	Char
2	Memilih memindai rubik	Pilihan menuju menu memindai dari pengguna	Char
3	Memindai rubik	Data warna satu sisi pada rubik	Array
4	Data rubik	Data warna rubik	Array
5	Materi belajar rubik	Materi pelajaran cara bermain rubik	ArrayList
6	Solusi menyelesaikan rubik	Langkah-langkah menyelesaikan rubik	Array
7	Pilihan modul	Pilihan modul dari pengguna	Char
8	Pilihan memindai	Pilihan memindai dari pengguna	Char
9	Hasil <i>Sunflower</i>	Langkah membuat <i>Sunflower</i>	Varchar
10	Hasil <i>White Cross</i>	Langkah membuat <i>White Cross</i>	Varchar
11	Hasil <i>White Corner</i>	Langkah membuat <i>White Corner</i>	Varchar
12	Hasil <i>2nd Layer</i>	Langkah membuat <i>2nd Layer</i>	Varchar
13	Hasil <i>Yellow Cross</i>	Langkah membuat <i>Yellow Cross</i>	Varchar
14	Hasil OLL	Langkah membuat OLL	Varchar
15	Hasil PLL	Langkah membuat PLL	Varchar

3.2.3. Spesifikasi Proses

Spesifikasi proses adalah dokumen yang menjelaskan secara rinci tentang suatu proses dalam sistem. Dokumen ini berisi informasi tentang sumber, input, output dan tujuan.

Tabel 3.1 Spesifikasi Proses *DFD level 1*

Proses	Keterangan
No Proses	1.1
Nama Proses	Belajar Rubik
Sumber	Pengguna
Input	Memilih kelas, Materi belajar rubik
Output	Materi belajar rubik, Pilihan modul
Tujuan	Pengguna
Logika Proses	Mulai Pengguna memilih menu belajar Materi diambil di database Materi dikirim ke sistem Aplikasi belajar rubik Selesai

Tabel 3.2 Spesifikasi Proses *DFD level 1* Proses 2

Proses	Keterangan
No Proses	1.2
Nama Proses	<i>Solver</i>
Sumber	Pengguna
Input	Memilih memindai rubik, Data rubik
Output	Solusi menyelesaikan rubik, Memindai rubik
Tujuan	Pengguna
Logika Proses	Mulai Pengguna memilih menu pemindaian rubik Pengguna memindai rubik acaknya menggunakan kamera handphone Selesai

Tabel 3.3 Spesifikasi Proses *DFD level 2* Proses 1.1

Proses	Keterangan
No Proses	1.1.1
Nama Proses	Memilih kelas belajar Rubik
Sumber	Pengguna
Input	Memilih kelas
Output	Pilihan kelas
Tujuan	Memilih modul belajar rubik
Logika Proses	Mulai Pengguna memilih kelas yang ingin dipelajari Selesai

Tabel 3.4 Spesifikasi Proses *DFD level 2* Proses 1.2

Proses	Keterangan
No Proses	1.1.2
Nama Proses	Memilih modul belajar rubik
Sumber	Memilih kelas belajar Rubik
Input	Pilihan kelas
Output	Pilihan modul
Tujuan	Materi
Logika Proses	Mulai Pengguna memilih modul materi yang ingin dipelajari Selesai

Tabel 3.5 Spesifikasi Proses *DFD level 2* Proses 2.1

Proses	Keterangan
No Proses	1.2.1
Nama Proses	Memindai
Sumber	Pengguna, Rubik
Input	Memilih memindai rubik, Data Rubik

Output	Data Rubik, Memindai Rubik
Tujuan	Algoritma menyelesaikan rubik
Logika Proses	<p>Mulai</p> <p>Pengguna memilih menu pemindaian rubik</p> <p>Pengguna memindai rubik acaknya menggunakan kamera handphone</p> <p>Selesai</p>

Tabel 3.6 Spesifikasi Proses DFD level 2 Proses 2.2

Proses	Keterangan
No Proses	1.2.2
Nama Proses	Algoritma menyelesaikan rubik
Sumber	Memindai
Input	Data Rubik
Output	Solusi menyelesaikan rubik
Tujuan	Pengguna
Logika Proses	<p>Mulai</p> <p>Sistem menerima data rubik acak pengguna</p> <p>Memasukkan data ke rubik ke dalam algoritma penyelesaian</p> <p>Mengirimkan solusi untuk ditampilkan ke pengguna</p> <p>Selesai</p>

Tabel 3.7 Spesifikasi Proses DFD level 3 Proses 1.1

Proses	Keterangan
No Proses	1.2.1.1
Nama Proses	Memilih pemindaian rubik
Sumber	Pengguna
Input	Memilih memindai rubik
Output	Pilihan memindai
Tujuan	Mengarahakan kamera ke enam sisi rubik
Logika Proses	<p>Mulai</p> <p>Pengguna memilih fitur pemindaian</p> <p>Selesai</p>

Tabel 3.8 Spesifikasi Proses DFD level 3 Proses 1.2

Proses	Keterangan
No Proses	1.2.1.2
Nama Proses	Mengarahakan kamera ke enam sisi rubik
Sumber	Memilih memindai rubik
Input	Pilihan memindai
Output	Memindai rubik
Tujuan	Rubik
Logika Proses	Mulai Kamera diarahkan ke enam sisi rubik Menyimpan data warna rubik Selesai

Tabel 3.9 Spesifikasi Proses DFD level 3 Proses 2.1

Proses	Keterangan
No Proses	1.2.2.1
Nama Proses	Algoritma <i>Sunflower</i>
Sumber	Data Rubik
Input	Data Rubik
Output	Hasil solusi Algoritma <i>Sunflower</i>
Tujuan	Algoritma <i>White Cross</i> , Solusi
Logika Proses	Mulai Memasukkan data ke Algoritma <i>Sunflower</i> Mengirimkan data hasil Algoritma <i>Sunflower</i> ke Algoritma <i>White Cross</i> Selesai

Tabel 3.10 Spesifikasi Proses DFD level 3 Proses 2.2

Proses	Keterangan
No Proses	1.2.2.2
Nama Proses	Algoritma <i>White Cross</i>

Sumber	Algoritma <i>Sunflower</i>
Input	Hasil solusi Algoritma <i>Sunflower</i>
Output	Hasil solusi Algoritma <i>White Cross</i>
Tujuan	Algoritma <i>White Corner</i> ; Solusi
Logika Proses	<p>Mulai</p> <p> Memasukkan data ke Algoritma <i>White Cross</i></p> <p> Mengirimkan data hasil Algoritma <i>White Cross</i> ke Algoritma <i>White Corner</i></p> <p>Selesai</p>

Tabel 3.11 Spesifikasi Proses DFD level 3 Proses 2.3

Proses	Keterangan
No Proses	1.2.2.3
Nama Proses	Algoritma <i>White Corner</i>
Sumber	Algoritma <i>White Cross</i>
Input	Hasil solusi Algoritma <i>White Cross</i>
Output	Hasil solusi Algoritma <i>White Corner</i>
Tujuan	Algoritma <i>2nd Layer</i> ; Solusi
Logika Proses	<p>Mulai</p> <p> Memasukkan data ke Algoritma <i>White Corner</i></p> <p> Mengirimkan data hasil Algoritma <i>White Corner</i> ke Algoritma <i>2nd Layer</i></p> <p>Selesai</p>

Tabel 3.12 Spesifikasi Proses DFD level 3 Proses 2.4

Proses	Keterangan
No Proses	1.2.2.4
Nama Proses	Algoritma <i>2nd Layer</i>
Sumber	Algoritma <i>White Corner</i>
Input	Hasil solusi Algoritma <i>White Corner</i>

Output	Hasil solusi Algoritma <i>2nd Layer</i>
Tujuan	Algoritma <i>Yellow Cross</i> , Solusi
Logika Proses	<p>Mulai</p> <p> Memasukkan data ke Algoritma <i>2nd Layer</i></p> <p> Mengirimkan data hasil Algoritma <i>2nd Layer</i> ke Algoritma <i>Yellow Cross</i></p> <p>Selesai</p>

Tabel 3.13 Spesifikasi Proses DFD level 3 Proses 2.5

Proses	Keterangan
No Proses	1.2.2.5
Nama Proses	Algoritma <i>Yellow Cross</i>
Sumber	Algoritma <i>2nd Layer</i>
Input	Hasil solusi Algoritma <i>2nd Layer</i>
Output	Hasil solusi Algoritma <i>Yellow Cross</i>
Tujuan	Algoritma OLL, Solusi
Logika Proses	<p>Mulai</p> <p> Memasukkan data ke Algoritma <i>Yellow Cross</i></p> <p> Mengirimkan data hasil Algoritma <i>Yellow Cross</i> ke Algoritma OLL</p> <p>Selesai</p>

Tabel 3.14 Spesifikasi Proses DFD level 3 Proses 2.6

Proses	Keterangan
No Proses	1.2.2.6
Nama Proses	Algoritma OLL
Sumber	Algoritma <i>Yellow Cross</i>
Input	Hasil solusi Algoritma <i>Yellow Cross</i>
Output	Hasil solusi Algoritma OLL
Tujuan	Algoritma PLL, Solusi
Logika Proses	<p>Mulai</p> <p> Memasukkan data ke Algoritma OLL</p>

	Mengirimkan data hasil Algoritma OLL ke Algoritma PLL Selesai
--	--

Tabel 3.15 Spesifikasi Proses *DFD level 3* Proses 2.7

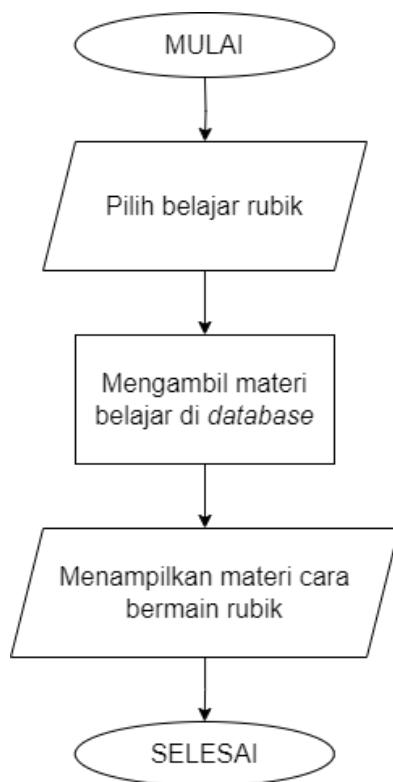
Proses	Keterangan
No Proses	1.2.2.7
Nama Proses	Algoritma PLL
Sumber	Algoritma OLL
Input	Hasil solusi Algoritma OLL
Output	Hasil solusi Algoritma PLL
Tujuan	Solusi
Logika Proses	Mulai Memasukkan data ke Algoritma PLL Mengirimkan data hasil semua Algoritma ke Algoritma <i>Solver Rubik</i> Selesai

3.2.4. Flowchart dan Pseudocode

Flowchart atau bagan alur adalah diagram dari sebuah alur kerja atau proses yang terdiri dari serangkaian langkah-langkah, tindakan dan keputusan yang diambil dalam suatu sistem atau program. Setiap langkah digambarkan dalam bentuk diagram yang berbentuk bangun datar dua dimensi dan dihubungkan dengan garis atau arah panah. Sedangkan *Pseudocode* atau kode semu adalah representasi informal dari logika program komputer.

a. Proses belajar materi melalui app

Proses bagaimana pengguna mengakses materi untuk belajar menyelesaikan rubik dapat dilihat pada Gambar 3.7, pengguna hanya perlu memilih menu belajar lalu memilih materi yang ingin dipelajari.

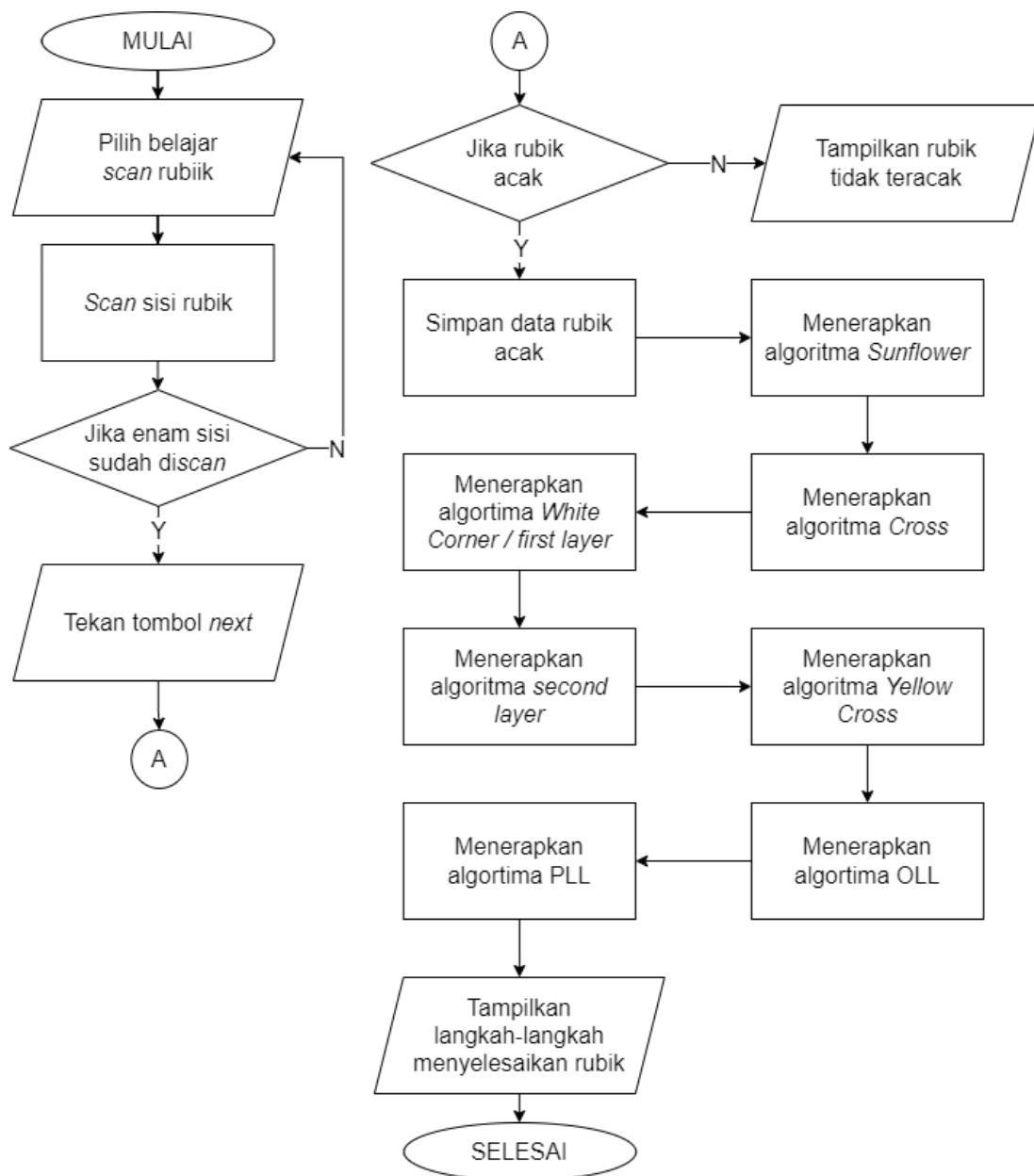


Gambar 3.7 Flowchart Belajar Materi Rubik

b. Proses aplikasi mendeteksi rubik dan mencari algoritma penyelesaiannya

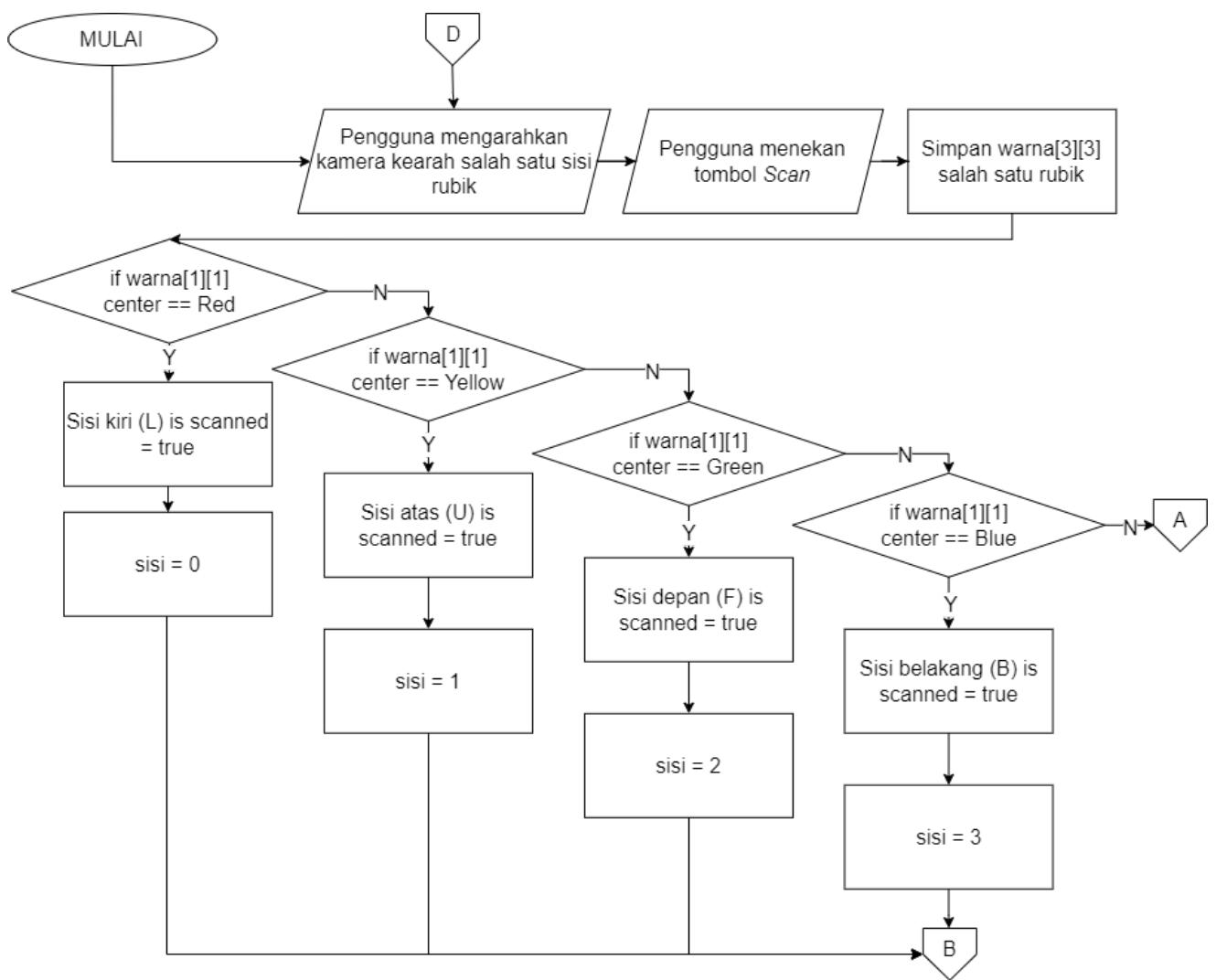
Untuk proses bagaimana program membaca pemindaian rubik dan mencari solusi penyelesaian rubik dapat dilihat pada bagian ini.

Pada Gambar 3.8 adalah Flowchart gambaran umum bagaimana aplikasi mempu mencari algoritma penyelesaian terhadap rubik acak pengguna. Langkah dimulai ketika pengguna memilih menu *Scan Rubik*, lalu pengguna perlu melakukan *Scan* keseluruhan sisi rubik. Jika semua sisi *discan* secara benar maka sistem mulai masuk ke program untuk mencari algoritma penyelesaian rubik.

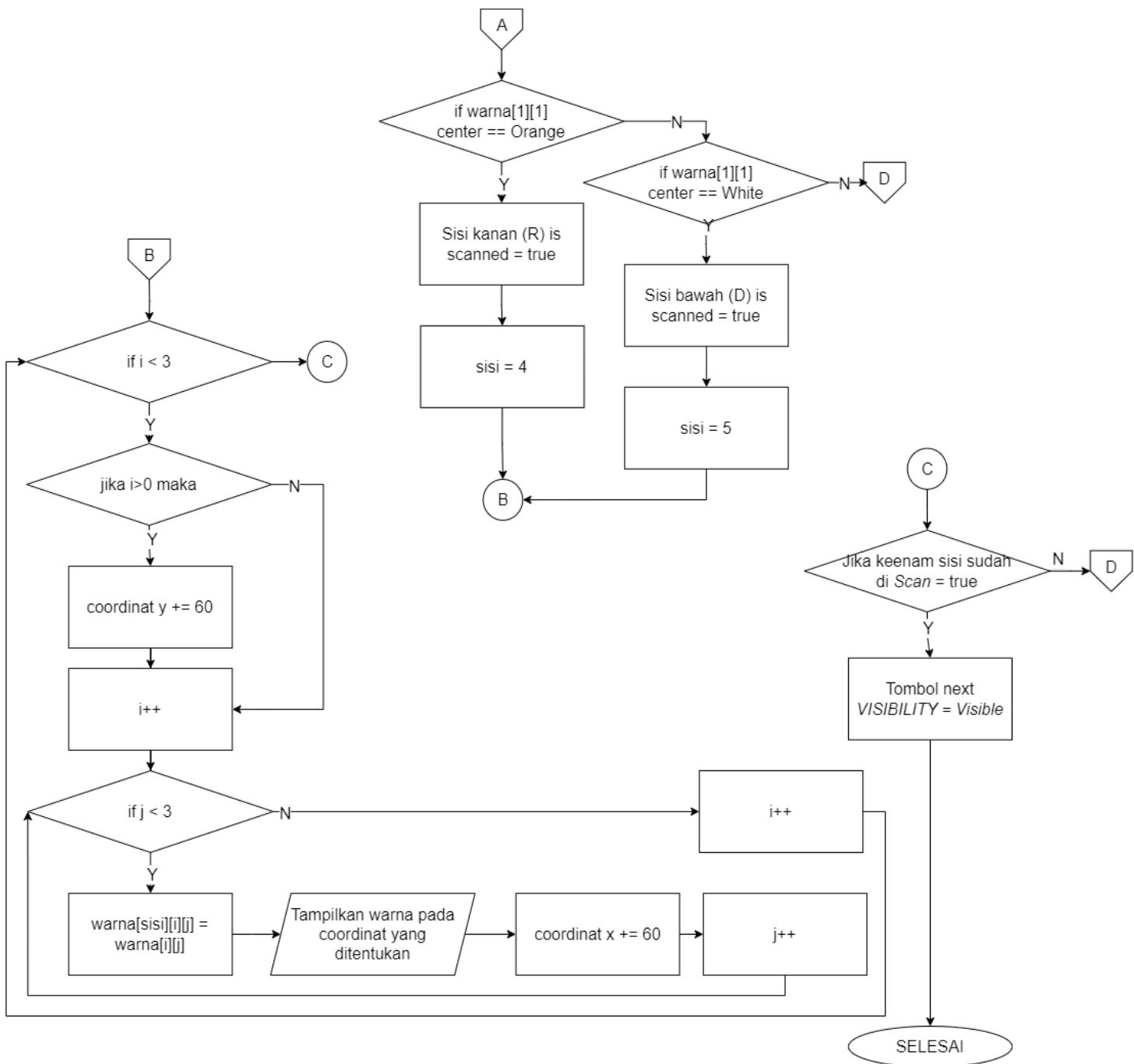


Gambar 3.8 Flowchart Scan Rubik

Pada Gambar 3.9 dan Gambar 3.10 merupakan Flowchart proses deteksi warna kesemua sisi rubik serta menampilkan hasil deteksinya di layar. Untuk memastikan satu sisi sudah dipindai hal yang dapat dijadikan patokan adalah *centers* dari rubik, jika semua warna *centers* sudah dipindai maka sama artinya semua sisi rubik telah dipindai. Dan untuk menampilkan hasil pemindaian dapat digambar pada canvas sesuai koordinat yang didapat berdasarkan warna apa *center* yang dipindai tersebut.

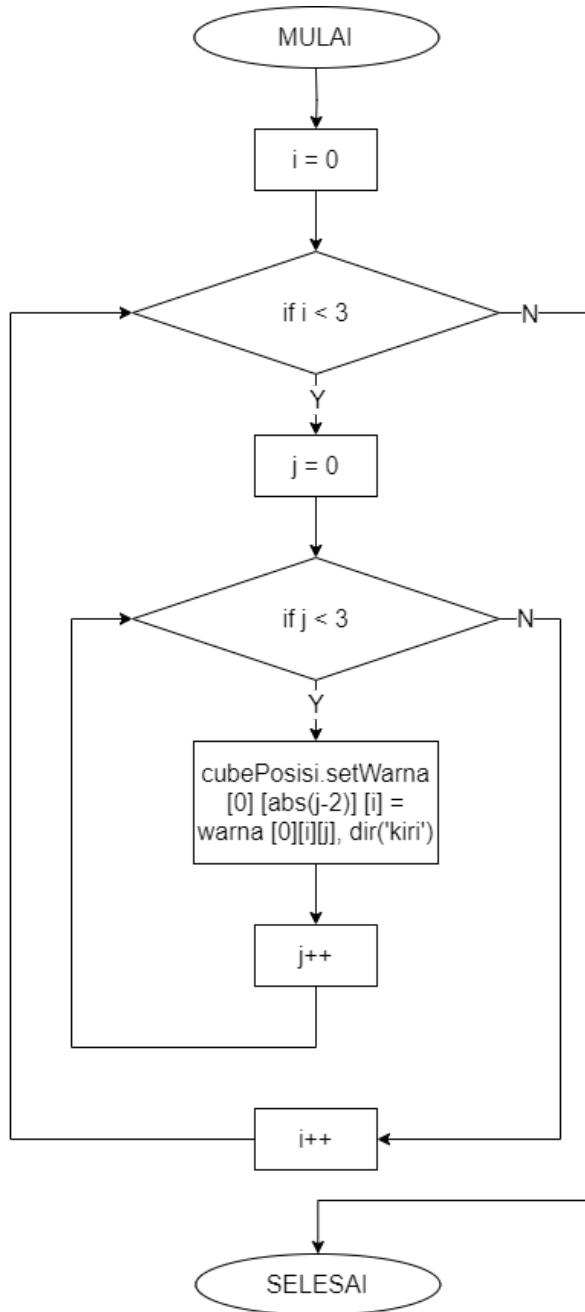


Gambar 3.9 Flowchart Scan Sisi Rubik



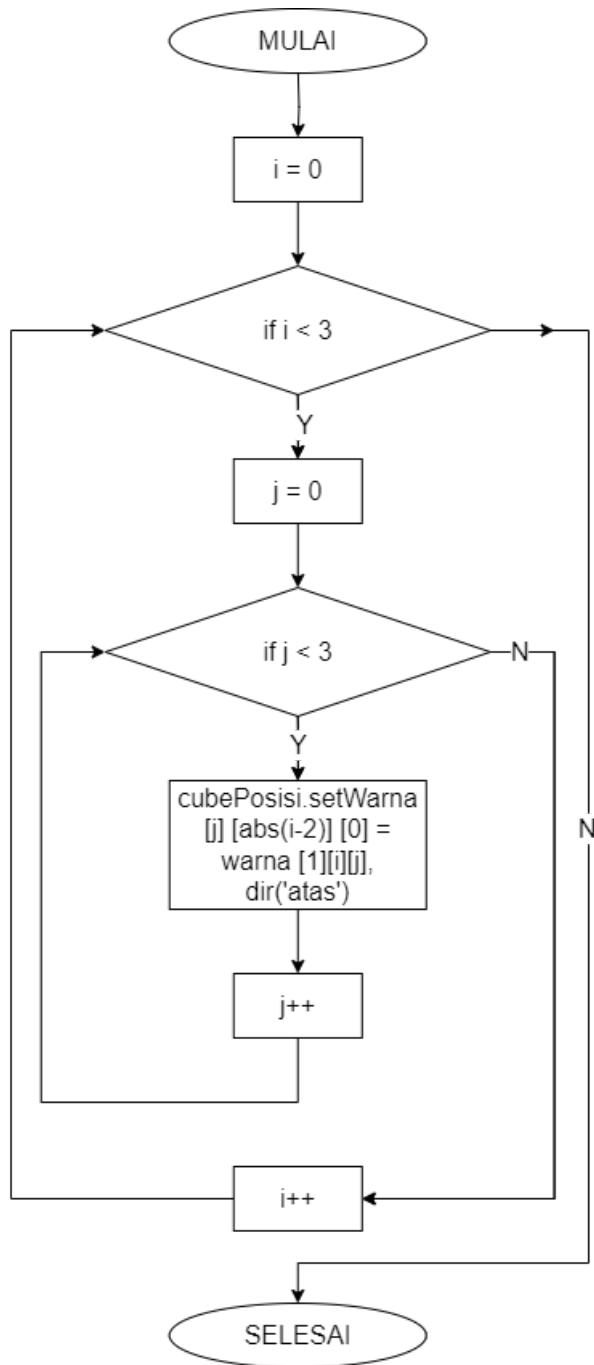
Gambar 3.10 Flowchart *Scan* Sisi Rubik Lanj

Gambar 3.11 merupakan Flowchart proses penyimpanan data sisi kiri rubik hasil pemindaian. Disimpang dengan menggunakan *nested loop* sebanyak tiga kali karena satu sisi rubik terdapat enam kotak berwarna.



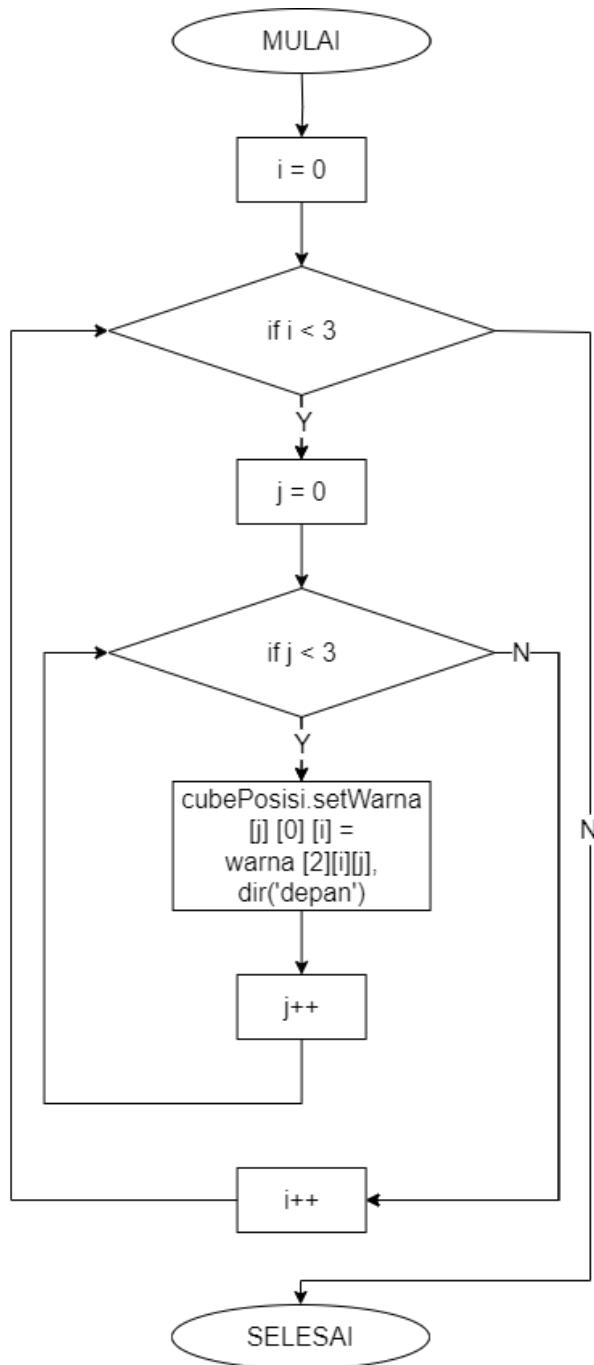
Gambar 3.11 Flowchart Simpan Data Rubik Acak Sisi Kiri

Gambar 3.12 merupakan Flowchart proses penyimpanan data sisi atas rubik hasil pemindaian. Disimpang dengan menggunakan *nested loop* sebanyak tiga kali karena satu sisi rubik terdapat enam kotak berwarna.



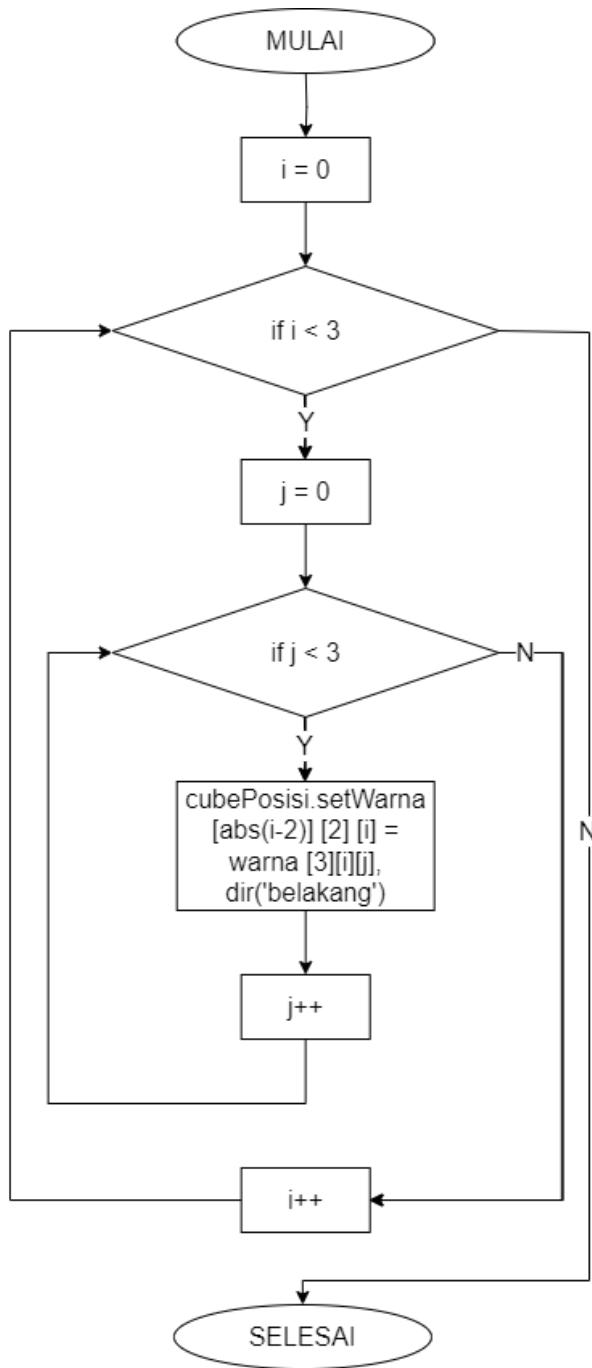
Gambar 3.12 Flowchart Simpan Data Rubik Acak Sisi Atas

Gambar 3.13 merupakan Flowchart proses penyimpanan data sisi depan rubik hasil pemindaian. Disimpang dengan menggunakan *nested loop* sebanyak tiga kali karena satu sisi rubik terdapat enam kotak berwarna.



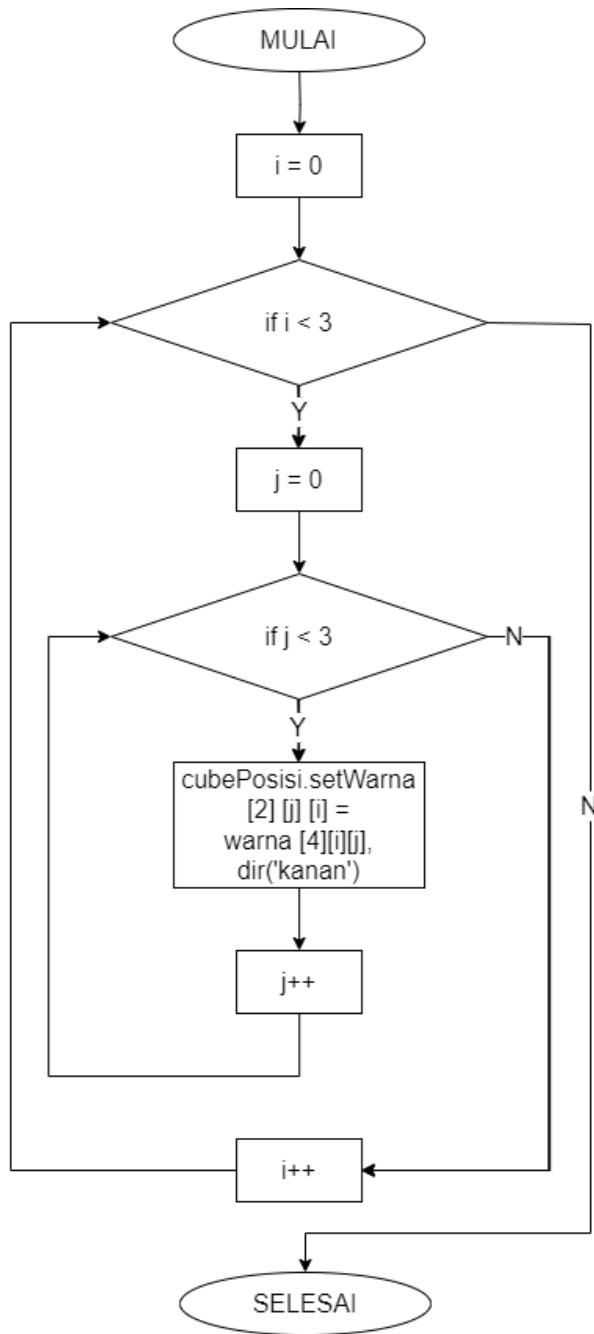
Gambar 3.13 Flowchart Simpan Data Rubik Acak Sisi Depan

Gambar 3.14 merupakan Flowchart proses penyimpanan data sisi belakang rubik hasil pemindaian. Disimpang dengan menggunakan *nested loop* sebanyak tiga kali karena satu sisi rubik terdapat enam kotak berwarna.



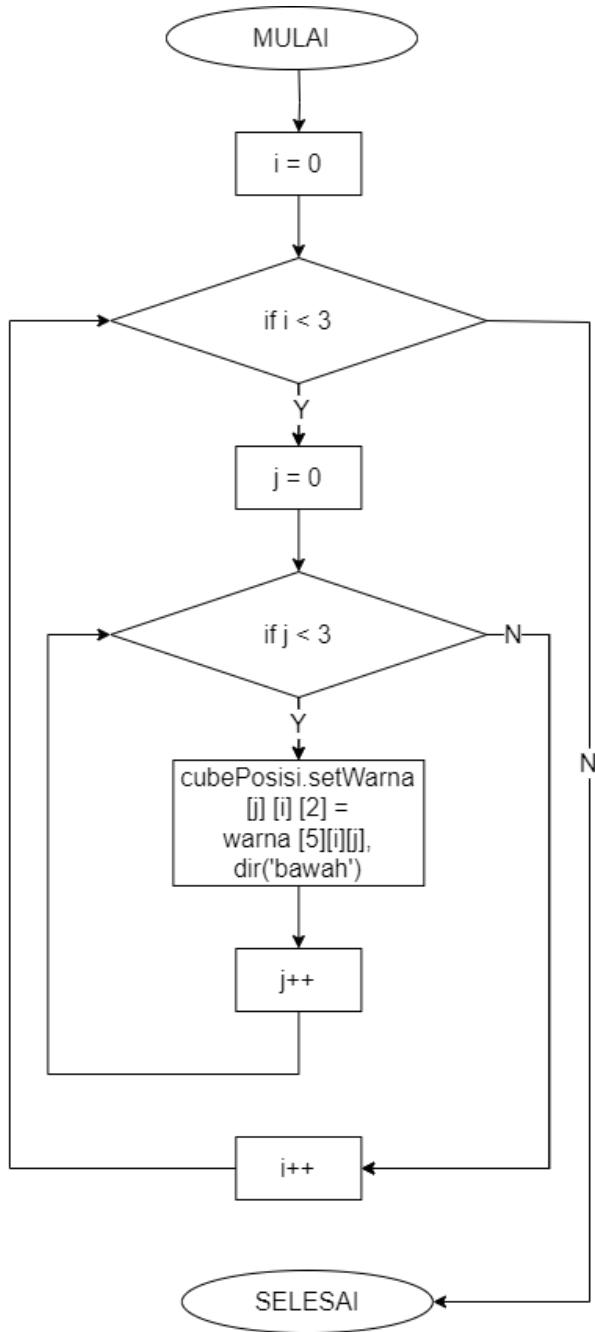
Gambar 3.14 Flowchart Simpan Data Rubik Acak Sisi Belakang

Gambar 3.15 merupakan Flowchart proses penyimpanan data sisi kanan rubik hasil pemindaian. Disimpang dengan menggunakan *nested loop* sebanyak tiga kali karena satu sisi rubik terdapat enam kotak berwarna.



Gambar 3.15 Flowchart Simpan Data Rubik Acak Sisi Kanan

Gambar 3.16 merupakan Flowchart proses penyimpanan data sisi bawah rubik hasil pemindaian. Disimpang dengan menggunakan *nested loop* sebanyak tiga kali karena satu sisi rubik terdapat enam kotak berwarna.



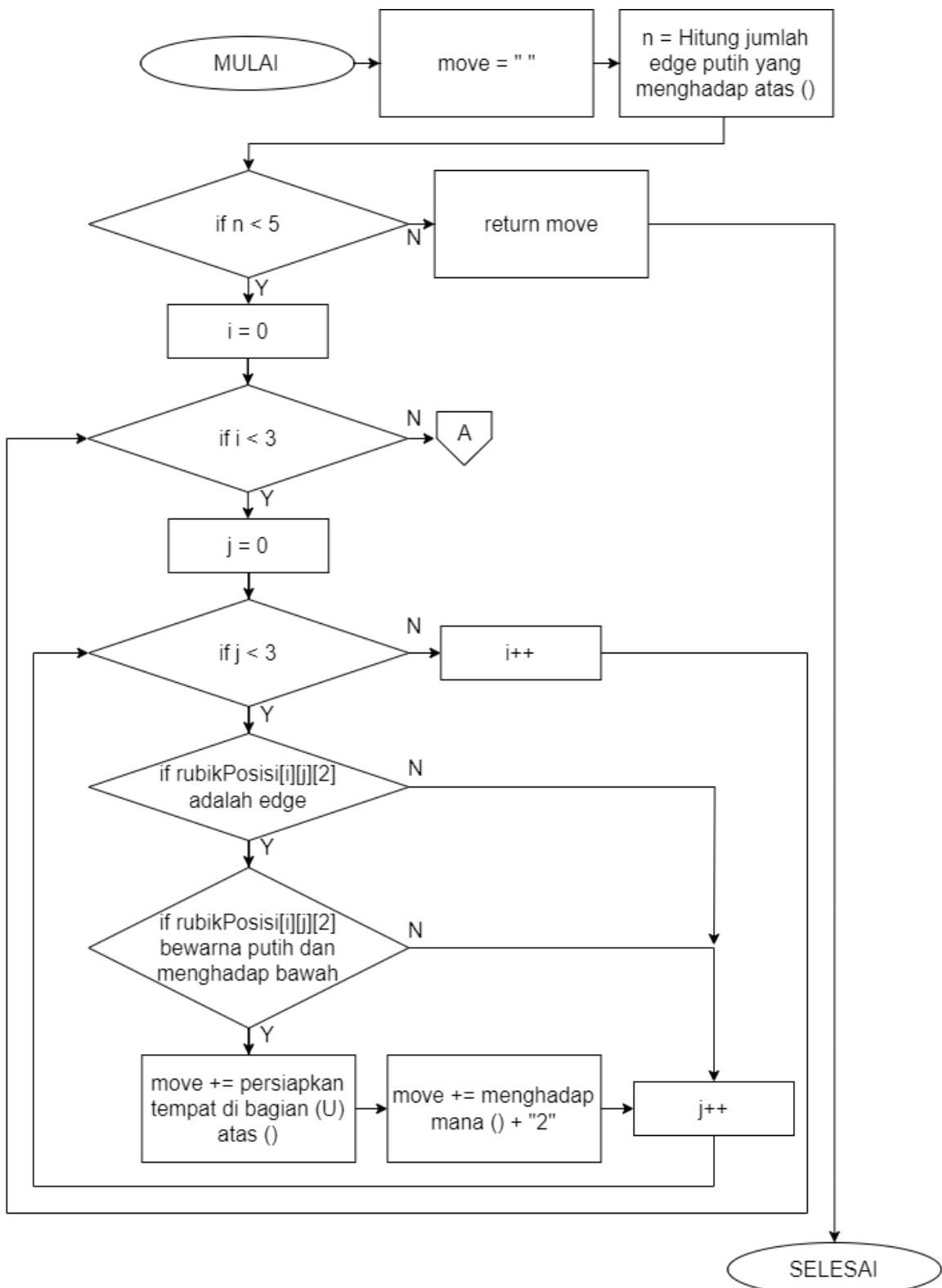
Gambar 3.16 Flowchart Simpan Data Rubik Acak Sisi Bawah

Flowchart sudah masuk ke tahap mencari algoritma penyelesaian dari rubik. Gambar 3.17 sampai 3.20 merupakan tahap pertama dari algoritma pertama *sunflower*. Pada Gambar 3.17 adalah hal pertama yang perlu dilakukan yaitu memeriksa apakah *edges* yang berwarna putih yang menghadap atas jumlahnya kurang dari lima, jika tidak kita bisa langsung ke tahap selanjutnya karena *sunflower*nya sudah terbentuk. Jika iya maka dicari *edges* putih yang menghadap bawah, jika bertemu maka lakukan gerakan mempersiapkan tempat untuk *edge* itu masuk lalu gerak memasukan *edge* berdasarkan menghadap mana warna putih pada *edges* tersebut ditambah dua.

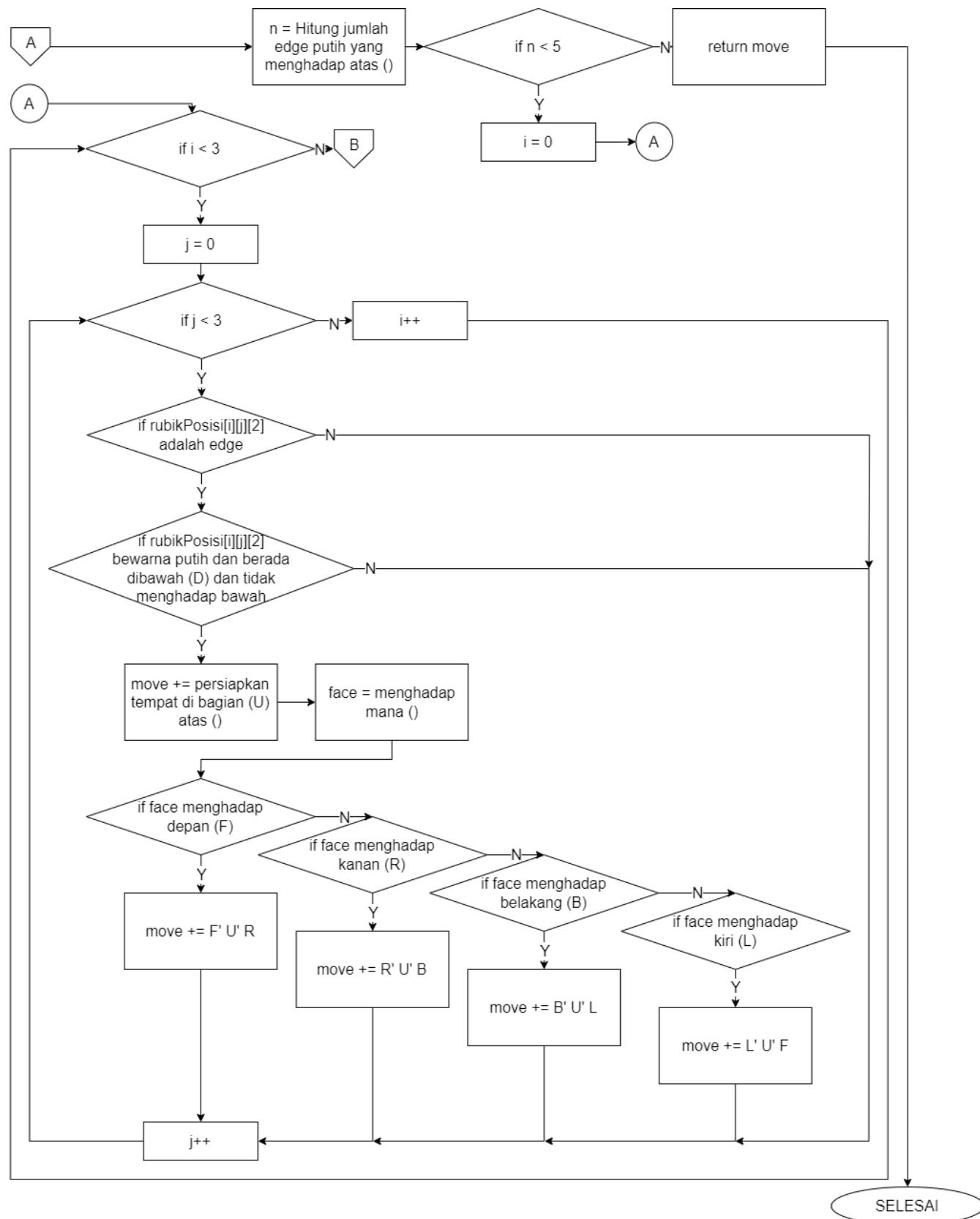
Untuk Gambar 3.18 merupakan lanjutan dari flowchart Gambar 3.17 dengan langkah pertama sama dengan sebelumnya yaitu memeriksa apakah *edges* yang berwarna putih yang menghadap atas jumlahnya kurang dari lima. Jika iya maka cari *edges* berwarna putih yang berada di bawah namun tidak menghadap bawah. Jika bertemu maka lakukan gerakan mempersiapkan tempat untuk *edge* itu masuk jika periksa menghadap mana *edge* putih tersebut. Jika menghadap depan maka langkahnya adalah $F' U' R$, jika menghadap kanan maka langkahnya adalah $R' U' B$, jika menghadap belakang maka langkahnya adalah $B' U' L$, jika menghadap kiri maka langkahnya adalah $L' U' F$.

Untuk Gambar 3.19 merupakan lanjutan dari langkah sebelumnya dengan langkah pertama memeriksa *edges* yang berwarna putih yang menghadap atas jumlahnya kurang dari lima. Jika iya maka cari *edges* berwarna putih pada bagian tengah (E) rubik. Lalu periksa jika posisi dan arah warna putih dari *edge* tersebut dan buat gerakan berdasarkan hal itu.

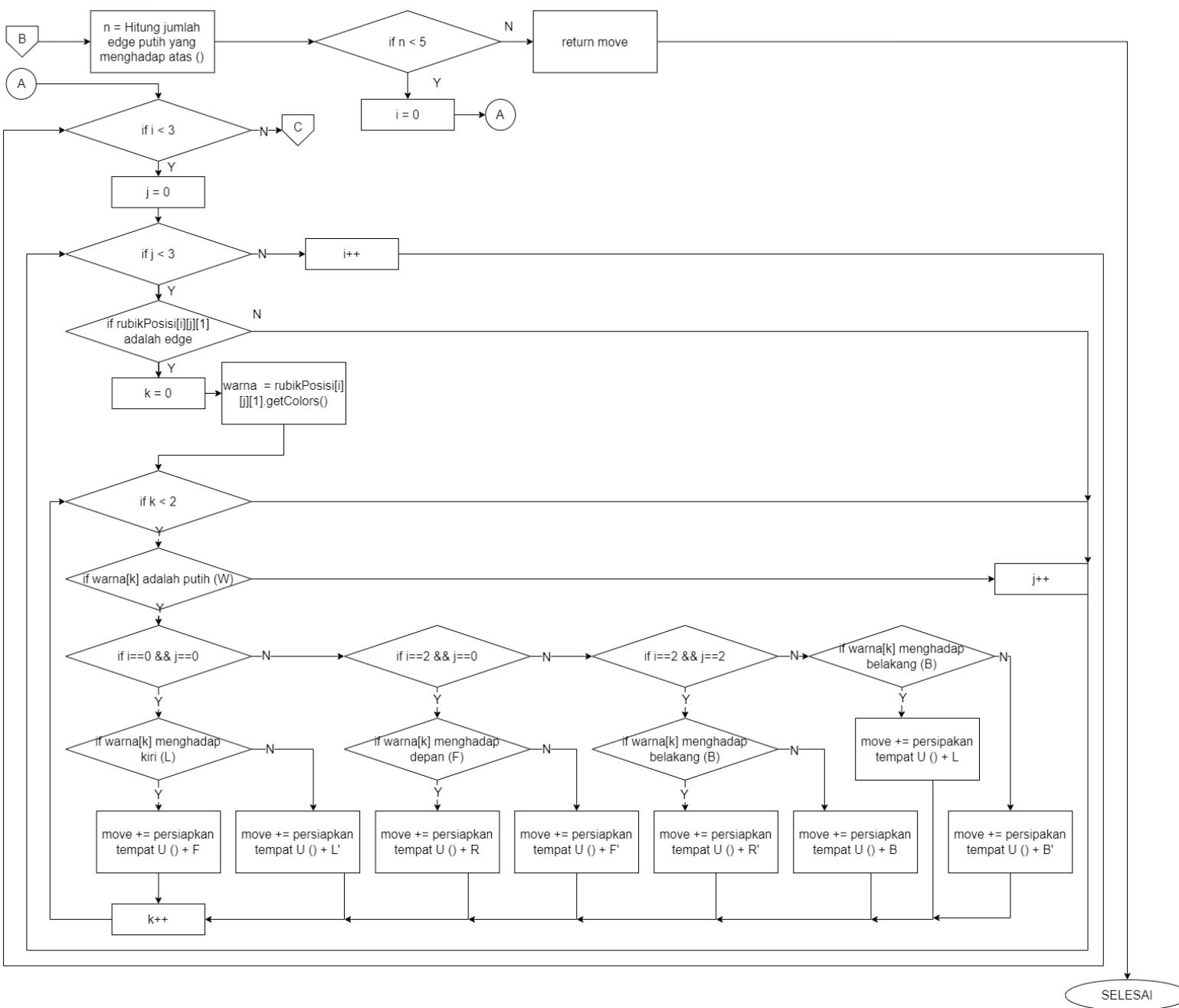
Untuk Gambar 3.20 merupakan langkah terakhir di tahap *sunflower* ini, pertama memeriksa *edges* yang berwarna putih yang menghadap atas jumlahnya kurang dari lima. Lalu buat gerakan berdasarkan arah *edge* putih.



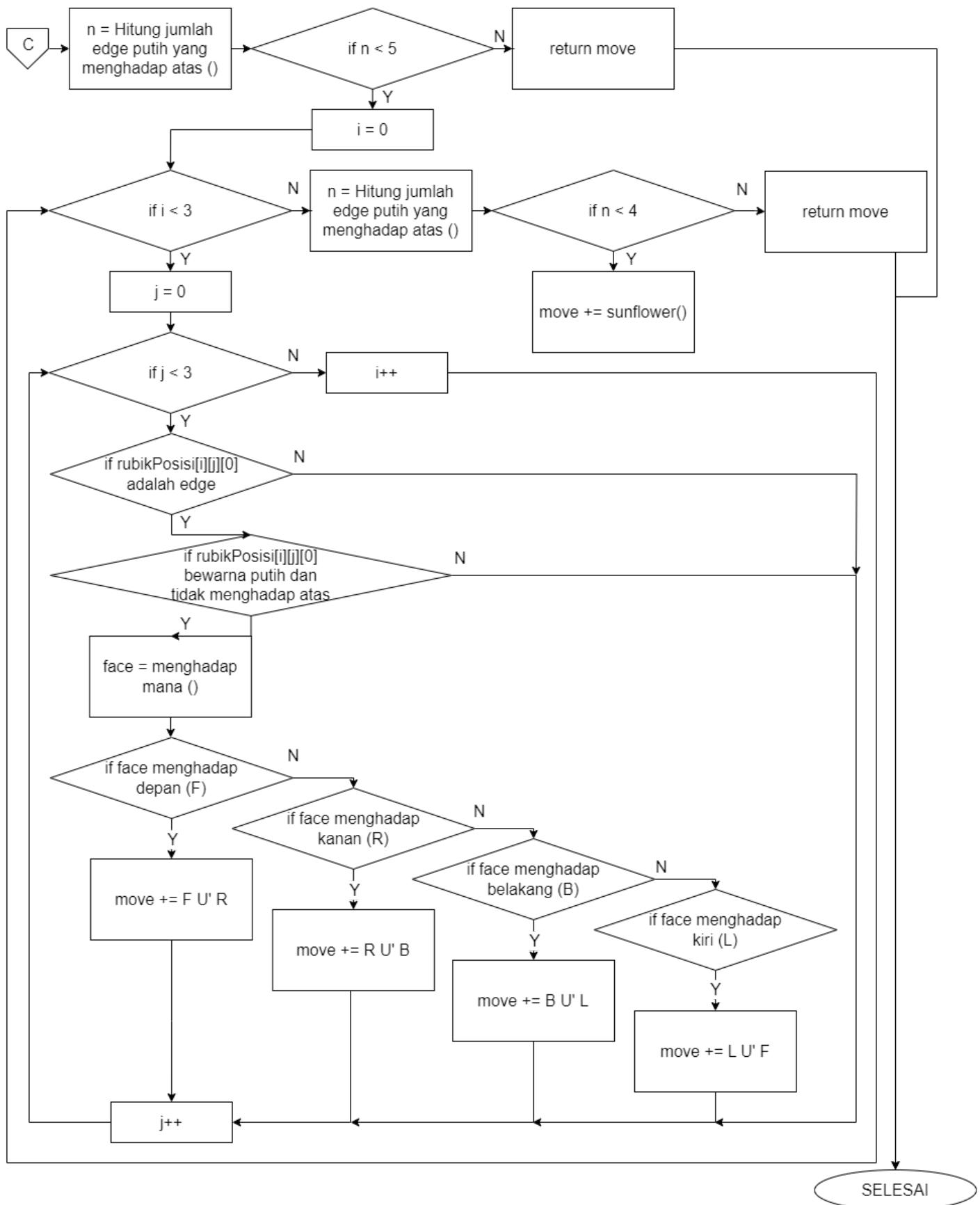
Gambar 3.17 Flowchart Menerapkan Algoritma *Sunflower 1* Bagian Bawah Menghadap Bawah



Gambar 3.18 Flowchart Menerapkan Algoritma *Sunflower 2* Bagian Bawah Tidak Menghadap Bawah

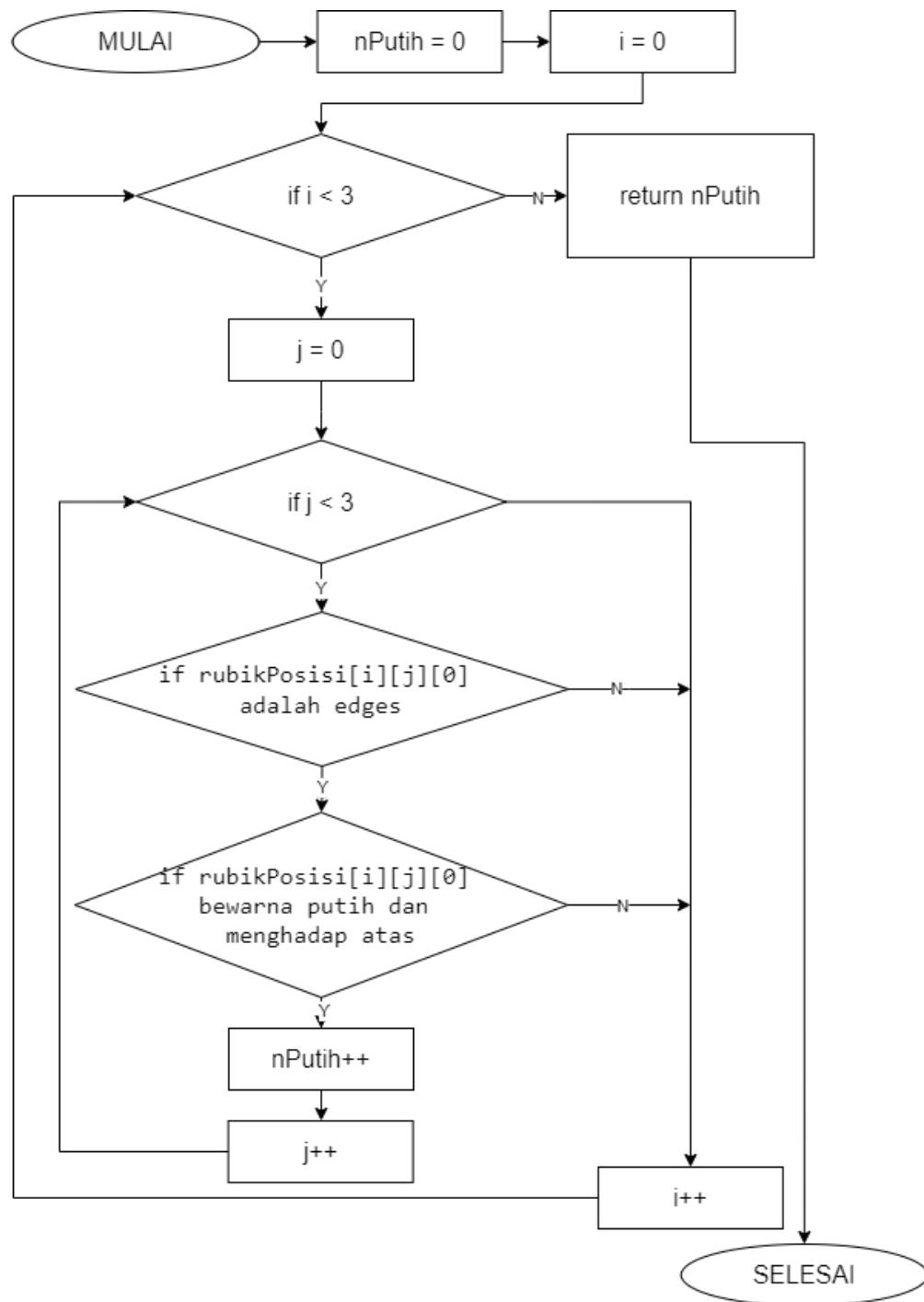


Gambar 3.19 Flowchart Menerapkan Algoritma *Sunflower 3* Bagian Tengah



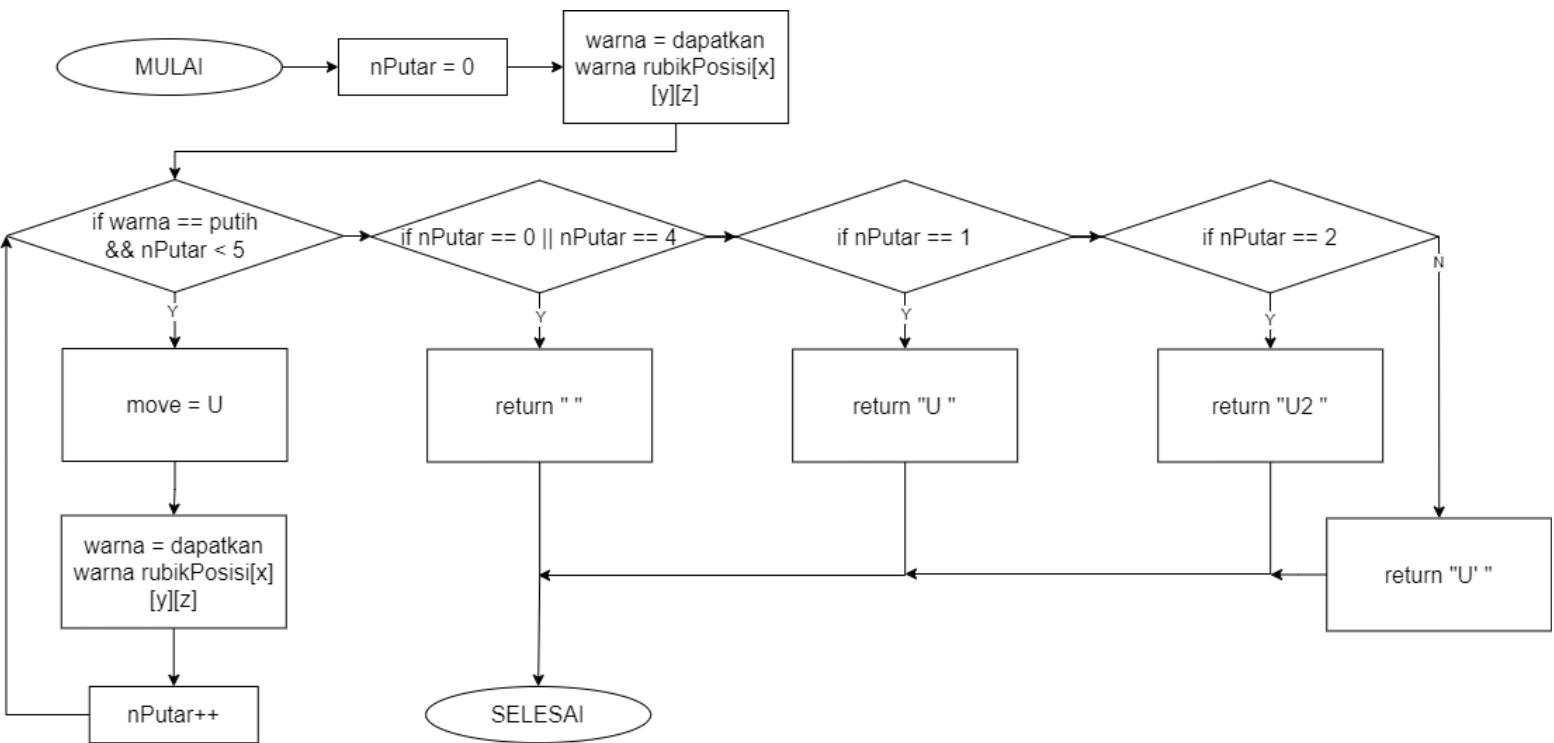
Gambar 3.20 Flowchart Menerapkan Algoritma *Sunflower 4* Bagian Atas Tidak Menghadap Atas

Gambar 3.21 merupakan flowchart yang banyak digunakan pada tahap sunflower yang befungsi untuk mencari banyak edges berwarna putih yang berada di sisi atas dan menghadap atas. Tujuannya dari mencari jumlah edges putih ini untuk mengetahui apakah tahap sunflower masih perlu dilakukan atau dapat dilanjutkan ke tahap selanjutnya.



Gambar 3.21 Flowchart Menghitung Jumlah Putih Dibagian Atas

Dan untuk gambar 3.22 merupakan flowchart untuk mempersiapkan tempat untuk edges yang tepat masuk. Pertama lakukan perulangan while selama edge posisi atas depan memiliki warna putih dan putaran belum sampai lima kali, maka lakukan gerakan U. Jika putarannya adalah nol atau empat maka tidak perlu ada gerakan, jika putarannya satu maka gerakannya adalah U, jika putarannya dua maka gerakannya U2 dan jika tidak maka gerakannya U'.



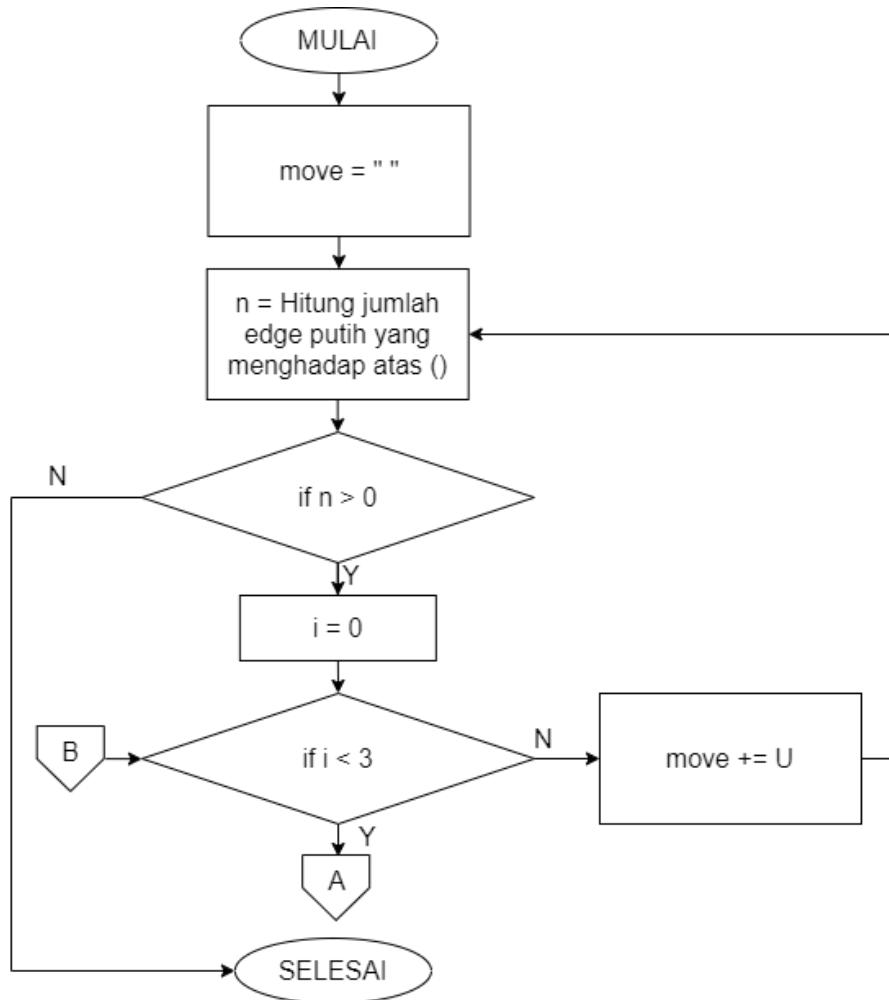
Gambar 3.22 Flowchart Mempersiapkan Tempat di Atas

Gambar 3.23 sampai gambar 3.24 merupakan algoritma kelanjutan dari *Sunflower* yaitu algoritma *Cross*.

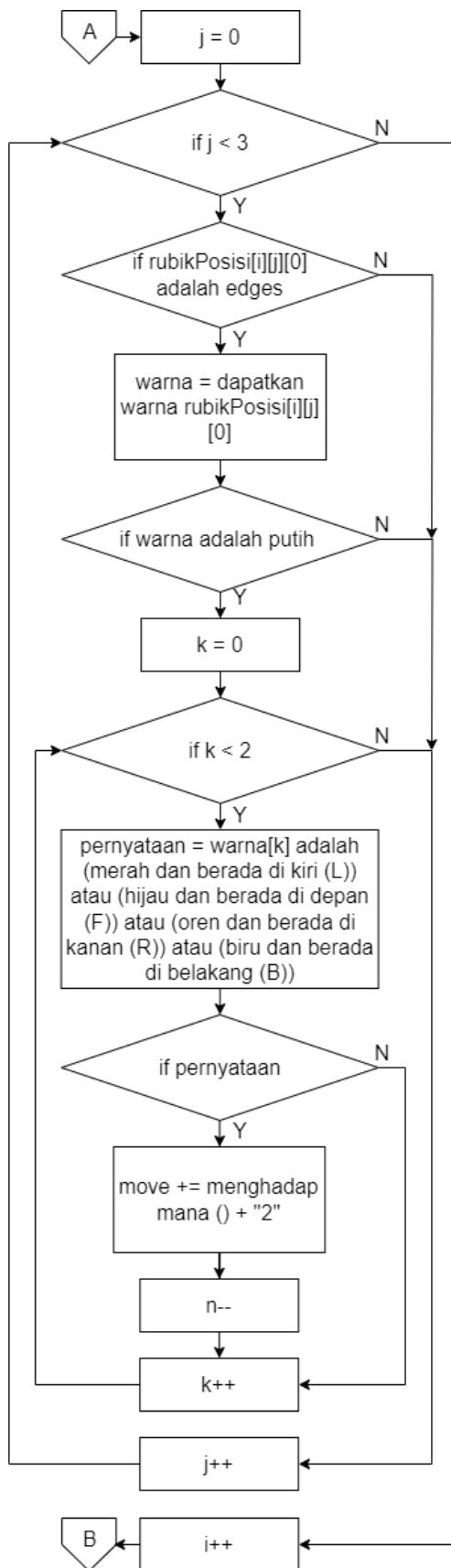
Gambar 3.23 merupakan flowchart tahap selanjutnya yaitu membuat cross atau tanda tambah pada bagian putih rubik. Langkah pertama dengan menghitung jumlah edges putih yang menghadap atas, jika ada maka lanjut jika tidak maka program selesai.

Gambar 3.24 merupakan lanjutan dari algoritma *cross*. Langkah selanjutnya yang dilakukan adalah dengan melakukan *nested loop* sebanyak tiga kali, lalu dicek apakah rubik merupakan *edge* dan berwarna putih, jika iya maka lakukan perulangan ketiga sebanyak dua kali untuk memeriksa apakah *edge* berwarna merah dan berada

di kiri, atau warna hijau dan berada di depan, atau berwarna oren dan berada di kanan, atau warna biru dan berada di belakang. Jika benar maka lakukan gerakan arah mana *edge* putih tersebut ditambah dua. Kenapa dilakukan dua kali perluangan di akhir, karena untuk memeriksa seluruh sisi suatu *edges*, karena satu *edge* memiliki dua buah sisi.

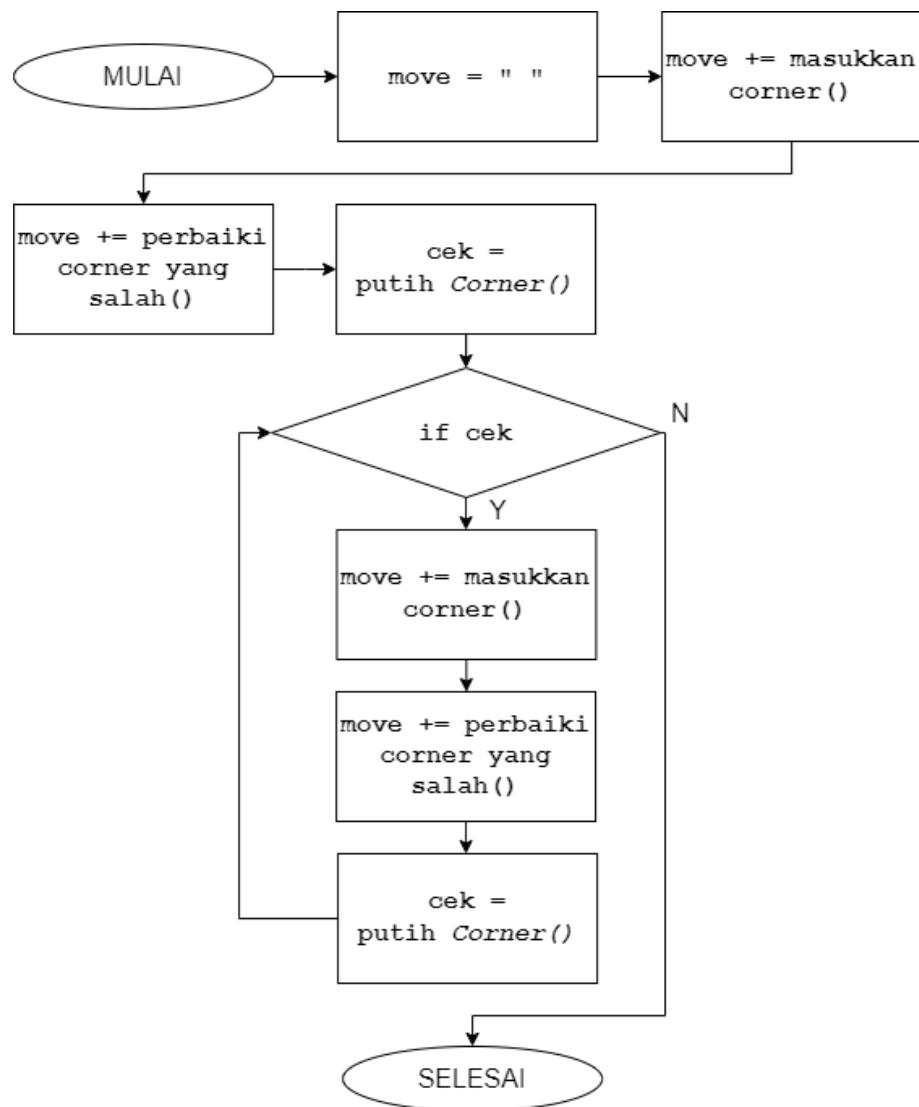


Gambar 3.23 Flowchart Menerapkan Algoritma *Cross*



Gambar 3.24 Flowchart Menerapkan Algoritma *Cross Lanj*

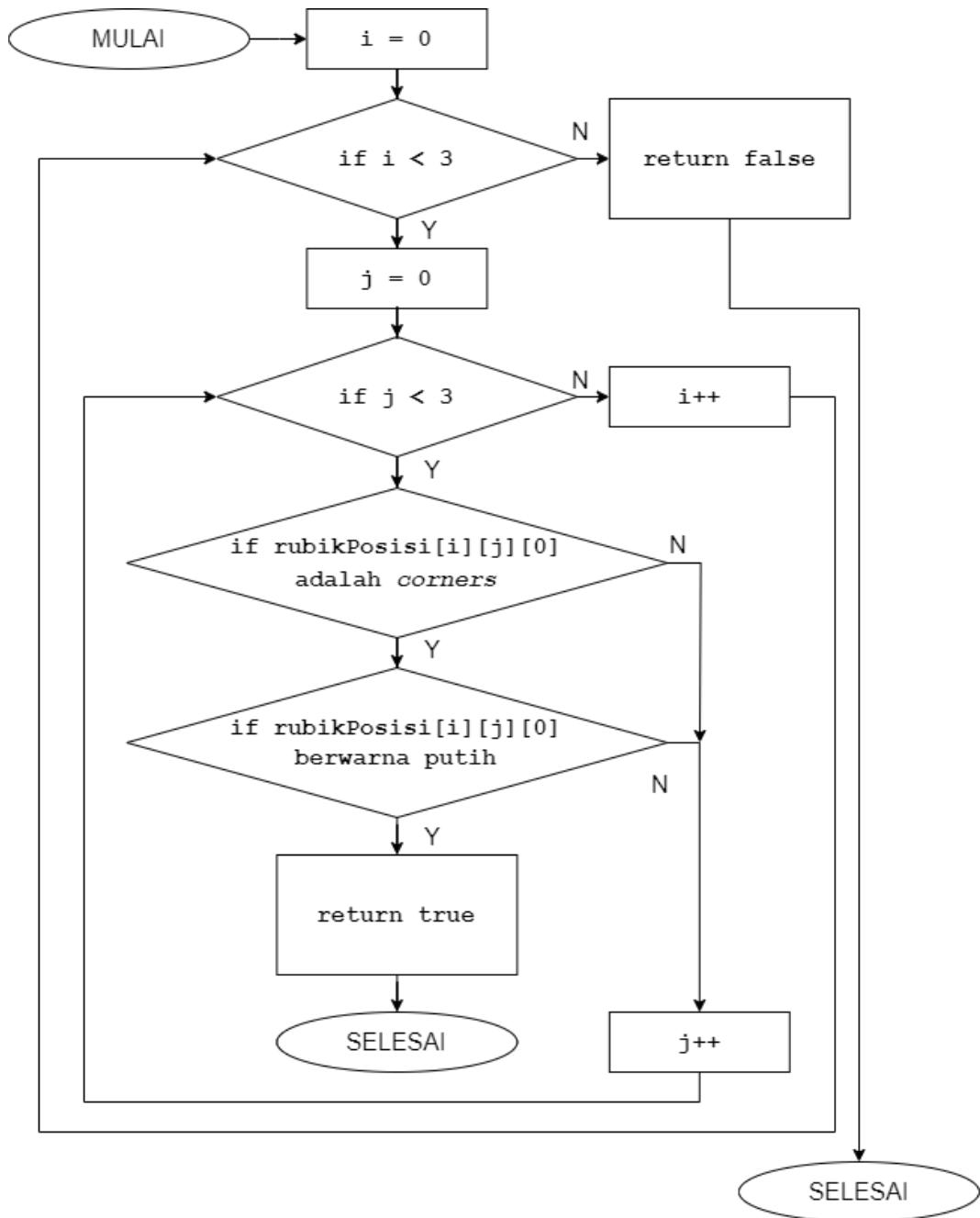
Gambar 3.25 adalah flowchart tahapan ketiga yaitu algoritma menyelesaikan lapisan pertama rubik dengan cara memasukkan seluruh corner yang berwarna putih ke posisi yang tepat. Langkah pertama dengan melakukan algoritma memasukkan corner yang tidak berada ke posisi yang tepat, langkah selanjutnya memperbaiki corner yang sudah berada di bawah namun posisinya tidak tepat. Lalu di cek apakah masih ada corner berwarna putih yang berada pada lapisan atas rubik, jika iya maka lakukan kedua langkah tadi kembali, jika tidak maka selesai.



Gambar 3.25 Flowchart Menerapkan Algoritma *1sr Layer / White Corner*

Gambar 3.26 merupakan flowchar algoritma pengecekan apakah masih adakah corner berwarna putih pada lapisan atas rubik. Pertama lakukan nested loop tiga kali karena setiap layer rubik terdiri dari tiga keping, lalu cek kepingan pada

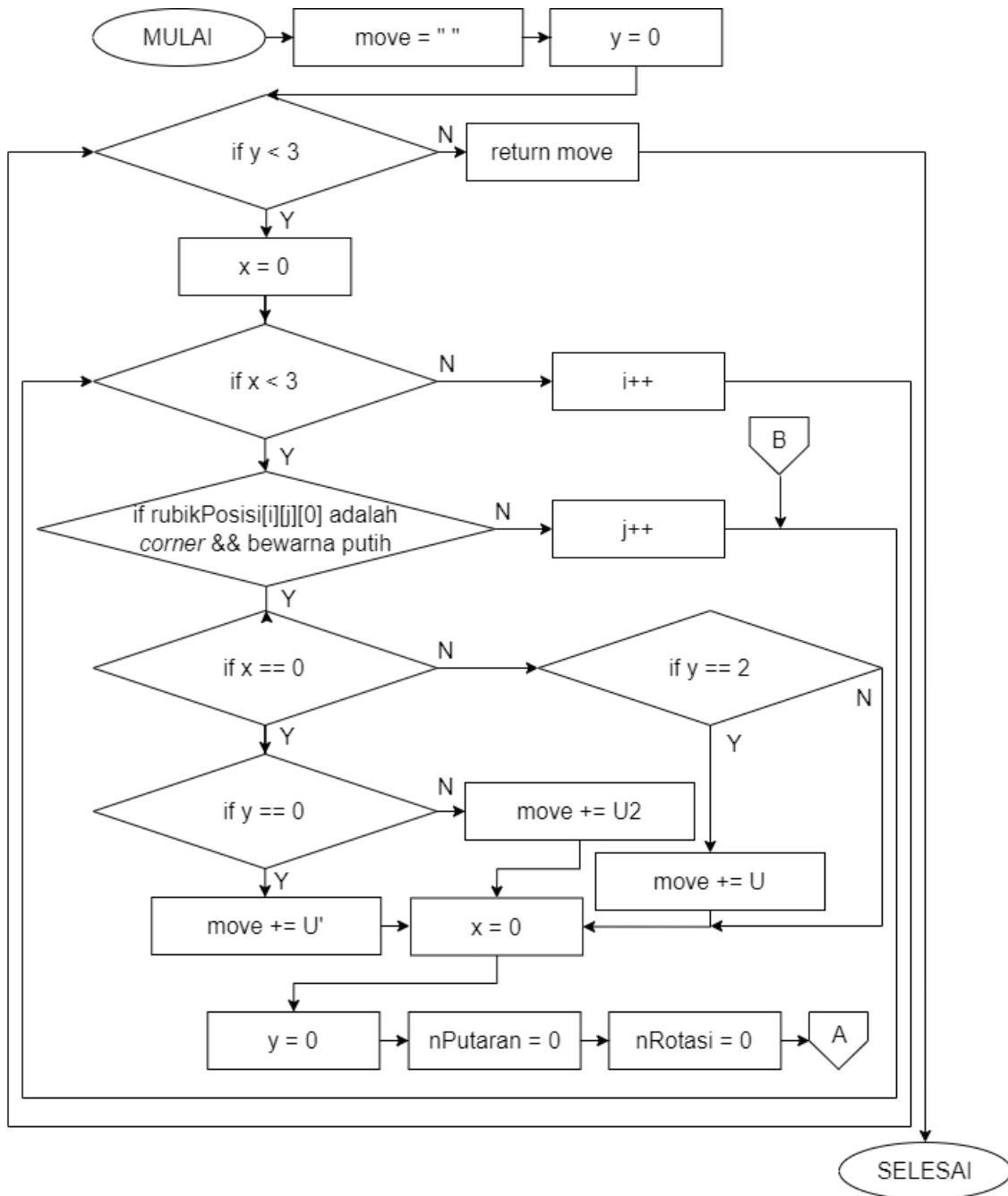
layer atas, jika corner dan berwarna putih maka return true, jika tidak bertemu maka return false.



Gambar 3.26 Flowchart Cek *Corners* Putih

Gambar 3.27 merupakan flowchart untuk memasukkan *corners* ke posisi yang tepat. Pertama lakukan *nested loop* tiga kali pada *layer* atas dan dicek apakah kepingan rubik merupakan *corner* dan berwarna putih. Jika *true* maka dicek lagi apakah *x* adalah 0 (merupakan sisi kiri), jika iya maka dicek apakah *y* adalah 0

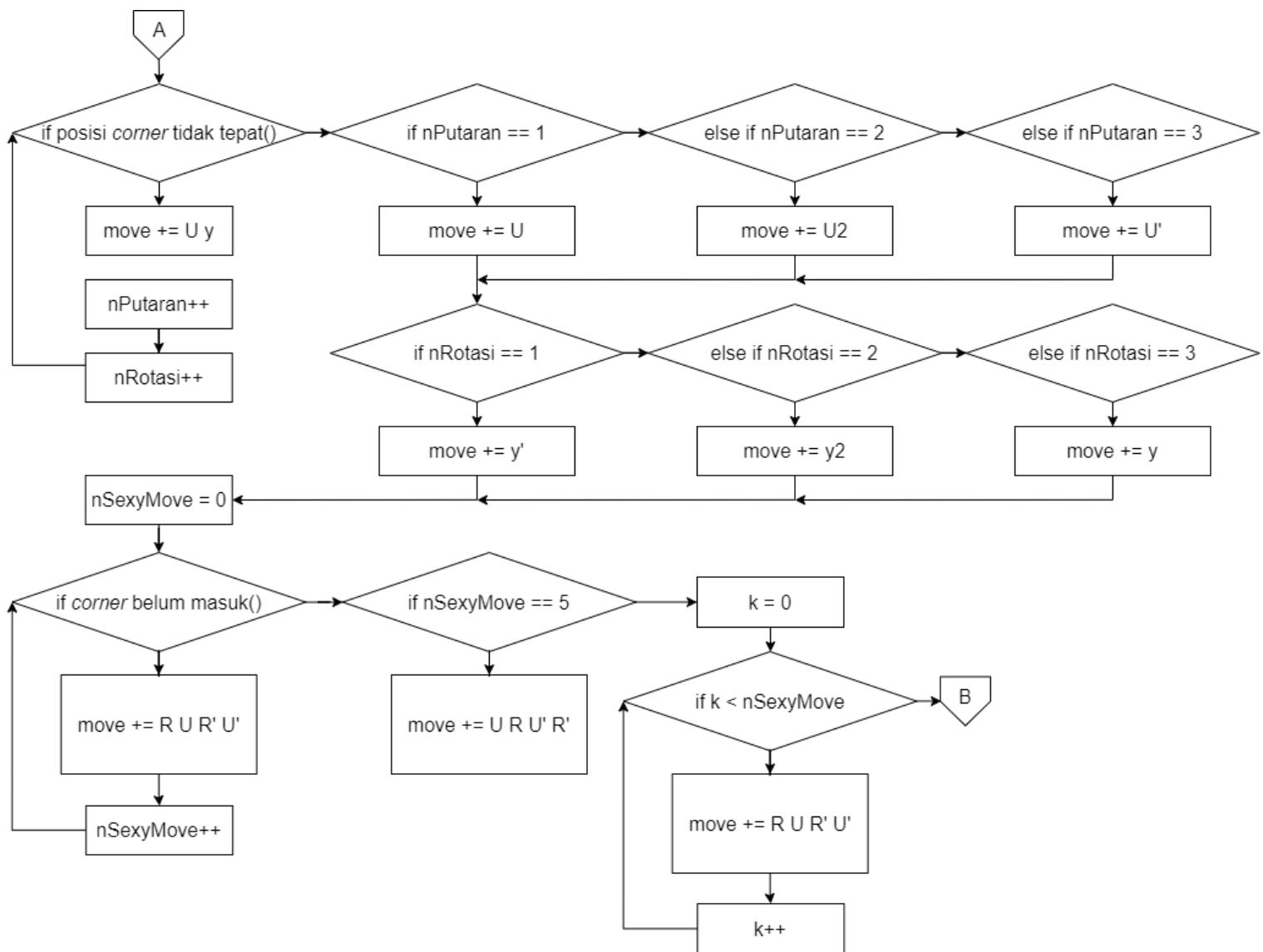
(merupakan sisi depan) , jika iya maka gerakan adalah U' , jika tidak maka U2. jika x bukan 0 dan y adalah dua (merupakan sisi belakang) maka gerakan adalah U, jika tidak maka tidak perlu gerakan. lalu nilai x, y, jumlah putaran dan rotasi perlu diatur menjadi nol kembali.



Gambar 3.27 Flowchart Masukkan *Corners*

Gambar 3.28 merupakan flowchart lanjutan dari algoritma memasukkan *corners*. Tahap selanjutnya adalah memeriksa apakah posisi dari *corner* sudah tidak

tepat, jika iya maka lakukan gerakan $U + y$, nilai putaran dan rotasi bertambah, dan lakukan sampai posisi corner tepat. Lalu periksa jumlah putaran yang dilakukan sampai posisi *corner* benar. Jika satu maka gerakan adalah U , jika dua maka $U2$ dan jika tiga maka U' . Lalu periksa jumlah rotasi yang dilakukan, jika satu maka y' , jika dua maka $y2$, jika tiga maka y . Lalu jika posisi tepat maka masukkan rubik dengan rubik dengan gerakan *sexy move* ($R U R' U'$) sampai masuk dan dihitung berapa kali dilakukan, jika dilakukan lima kali maka gerakan adalah $U R U' R'$. Lalu kembali ke perulangan untuk lanjut menggerjakan kepingan rubik selanjutnya.



Gambar 3.28 Flowchart Masukkan *Corners Lanj*

Pseudocode di bawah untuk mengetahui posisi *corner* sudah tepat sebelum dimasukkan. pertama pertiksa apakah warna putih menghadap atas atau tidak. Jika iya maka *return* warna kepingan menghadap kanan == warna *center* depan && warna kepingan menghadap depan == warna *center* kanan. Jika tidak maka *return* warna kepingan menghadap kanan == warna *center* depan || warna kepingan menghadap depan == warna *center* kanan.

```

FUNCTION whiteCornerPrepared()
    DECLARE whiteUp BOOLEAN
    SET whiteUp TO FALSE

    IF NOT (cubiePos[2][0][0].isCornerCubie() AND cubiePos[2][0][0]
→.getDirOfColor('W') == 'A')
        RETURN FALSE
    END IF

    IF cubiePos[2][0][0].getDirOfColor('W') == 'U'
        SET whiteUp TO TRUE
    END IF

    IF whiteUp
        RETURN cubiePos[2][0][0].getColorOfDir('R') == cubiePos[1][0][1]
→.getColors()[0].getColor() AND cubiePos[2][0][0]
→.getColorOfDir('F') == cubiePos[2][1][1].getColors()[0].getColor()
    ELSE
        RETURN cubiePos[2][0][0].getColorOfDir('R') == cubiePos[2][1][1]
→.getColors()[0].getColor() OR
            cubiePos[2][0][0].getColorOfDir('F') == cubiePos[1][0][1]
→.getColors()[0].getColor()
    END IF
END FUNCTION

```

Pseudocode di bawah untuk memeriksa apakah *centers* sudah masuk. pertama periksa kepingan apakah *centers* atau tidak, jika iya maka *return* warna

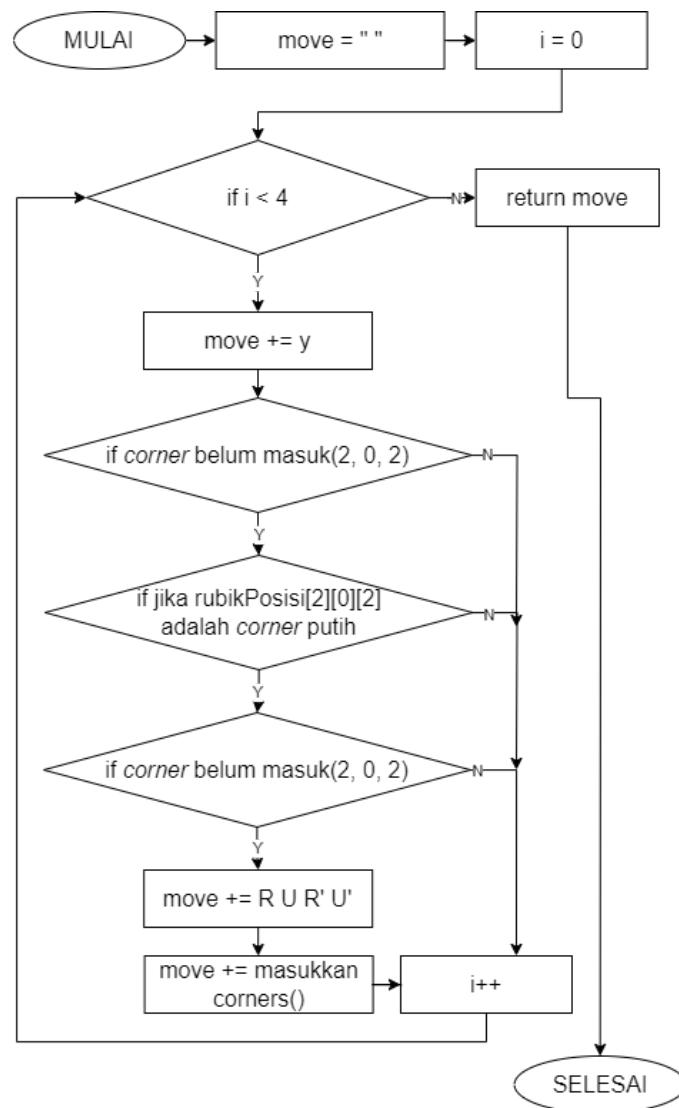
kepingan menghadap bawah adalah putih && warna depan == warna *center* depan && warna kanan == warna *center* kanan. Jika tidak maka *return false*.

```

FUNCTION cornerInserted(x, y, z)
    IF cubiePos[x][y][z].isCornerCubie()
        RETURN cubiePos[x][y][z].getColorOfDir('D') == cubiePos[1][1][2]
        →.getColors()[0].getColor() AND
            cubiePos[x][y][z].getColorOfDir('F') == cubiePos[1][0][1].getColors()[0].getColor()
        AND
            cubiePos[x][y][z].getColorOfDir('R') == cubiePos[2][1][1].getColors()[0].getColor()
        ELSE
            RETURN FALSE
        END IF
    END FUNCTION

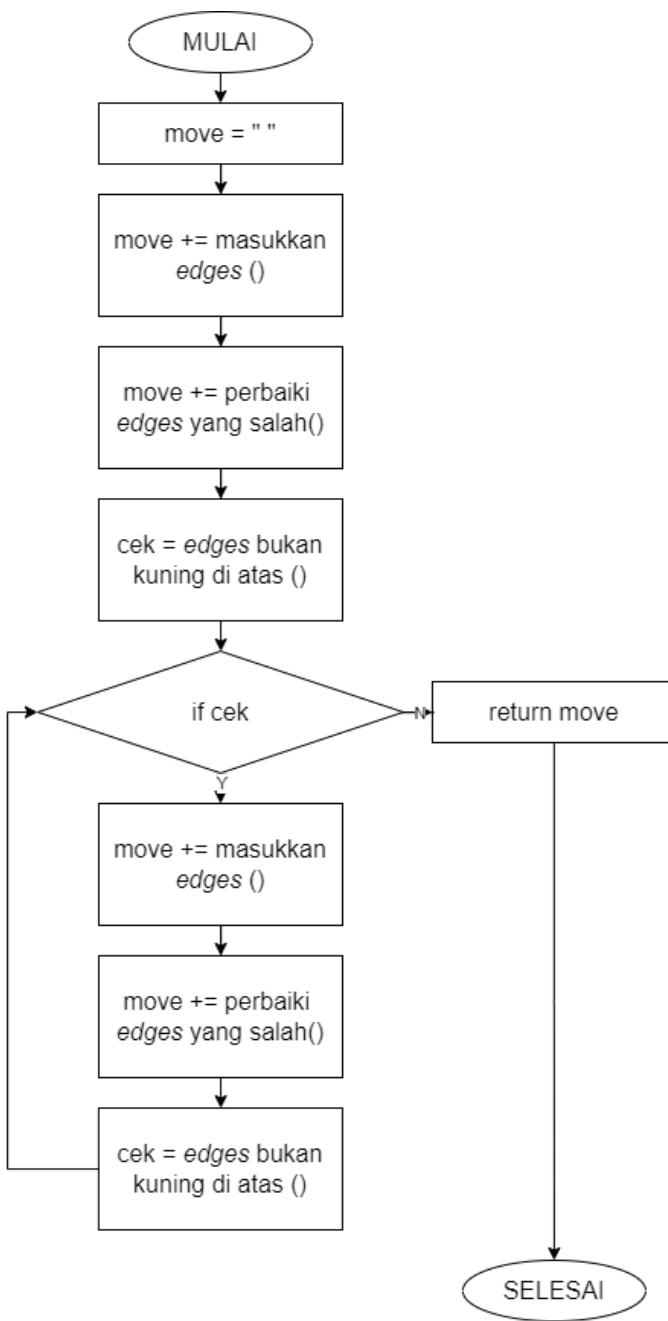
```

Gambar 3.29 merupakan flowchart untuk memperbaiki *corners* yang sudah masuk namun posisinya tidak tepat. Pertama lakukan perulangan sebanyak empat kali karena satu layer memiliki empat keping *corners*. Lakukan rotasi y setiap perulangan untuk memutar rubik. Pertama periksa apakah corner sudah masuk dengan fungsi sebelumnya, lalu jika salah maka periksa apakah *corner* berwarna putih, jika benar lakukan *sexy move* satu kali untuk mengeluarkan kepingan tersebut lalu lakukan panggil fungsi memasukkan *corner*.



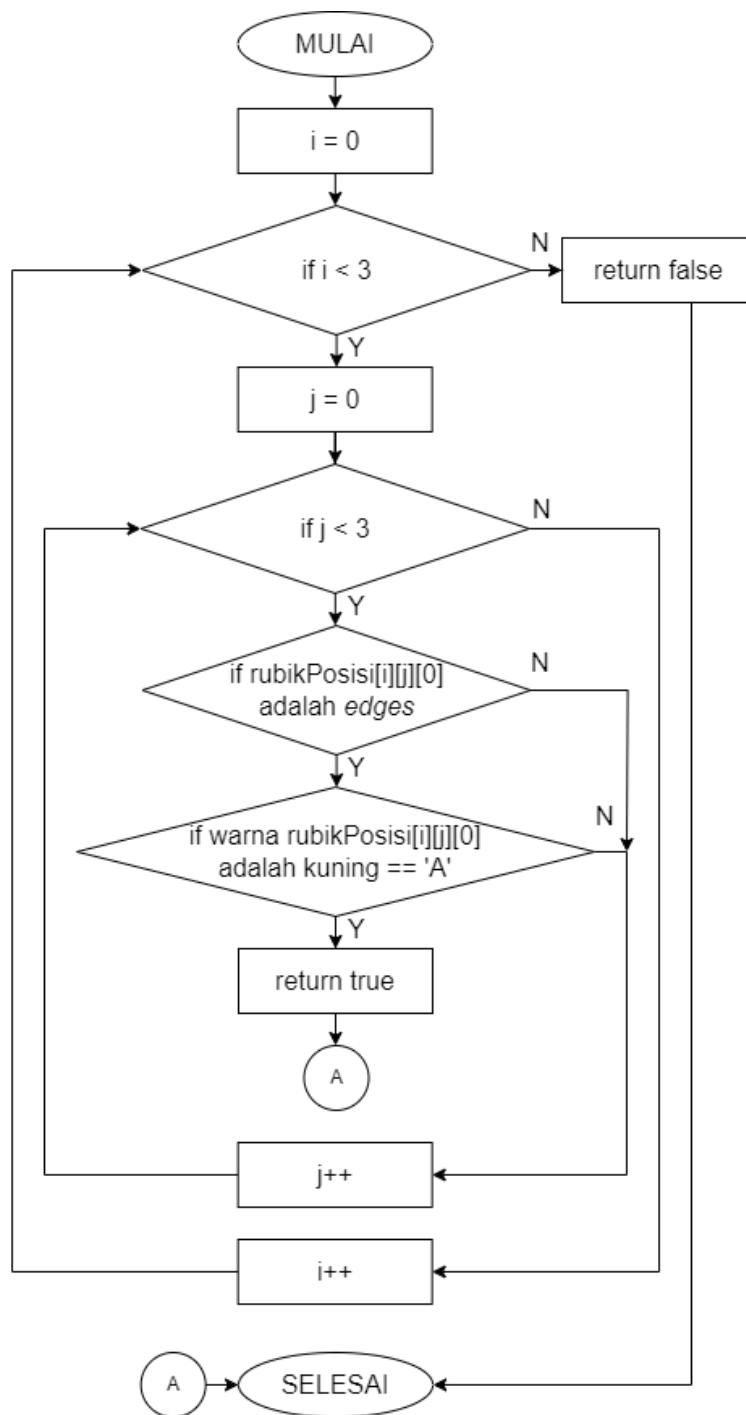
Gambar 3.29 Flowchart Perbaiki *Corners* Yang Salah

Gambar 3.30 merupakan flowchart algoritma untuk menyelesaikan *layer* kedua. Langkah pertama lakukan algoritma memasukkan *edges*, lalu lakukan algoritma perbaiki *edges* yang salah, lalu cek apakah masih ada *edges* yang bukan berwarna kuning di *layer* atas, jika iya maka lakukan ketiga langkah sebelumnya kembali, jika tidak maka selesai.



Gambar 3.30 Flowchart Menerapkan Algoritma 2nd Layer

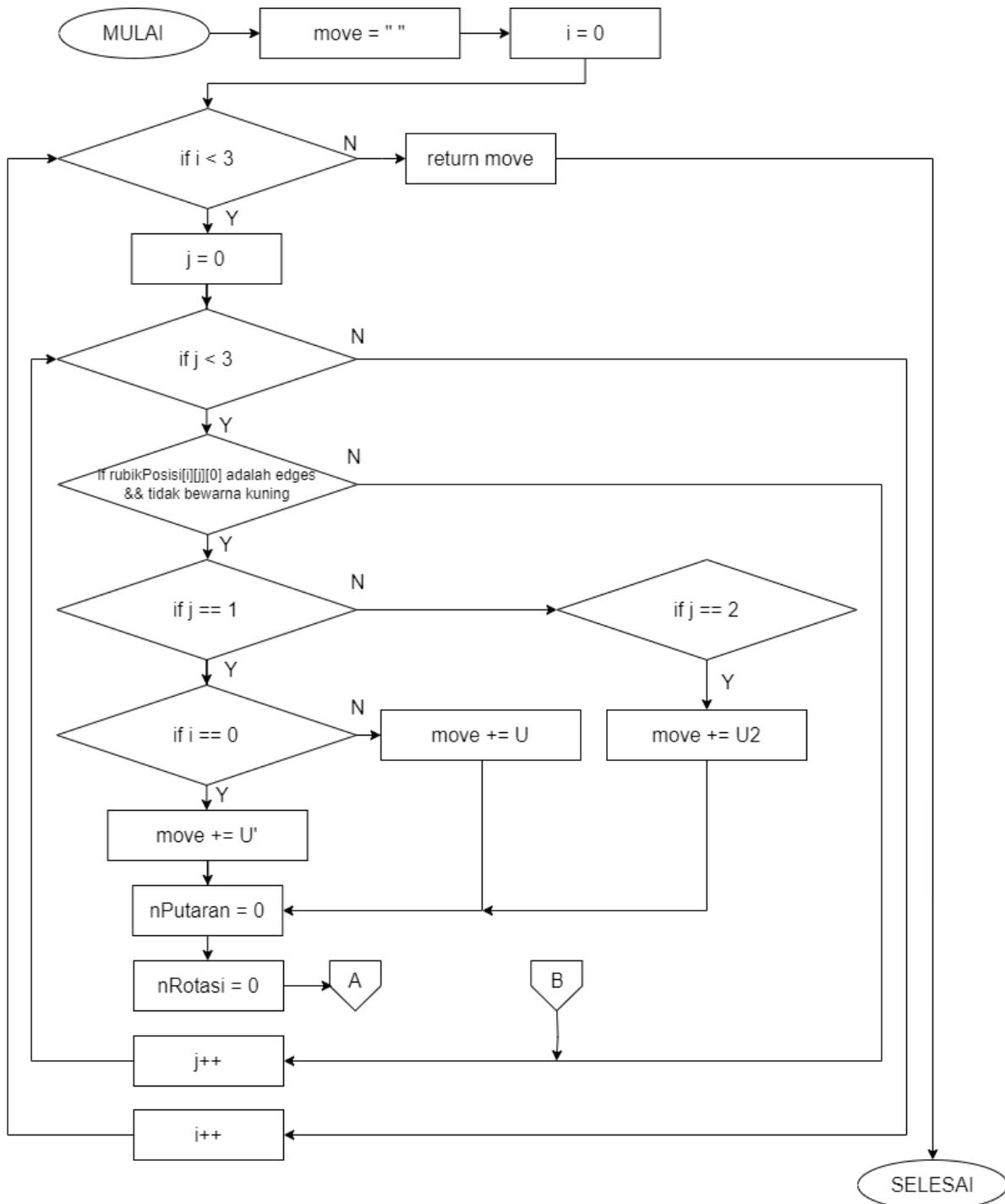
Gambar 3.31 merupakan flowchart untuk memeriksa apakah masih ada *edges* yang tidak berwarna kuning di *layer* atas. Hal ini untuk memeriksa jika tidak ada *edges* yang tidak berwarna kuning di atas itu artinya semua *edges* sudah berada pada level yang tepat. Pertama lakukan *nested loop* tiga kali untuk memeriksa seluruh keping di *layer* atas. Lalu periksa apakah keping tersebut *edges*, jika iya periksa apakah ada yang tidak berwarna kuning, jika ada *return true*, jika tidak *return false*.



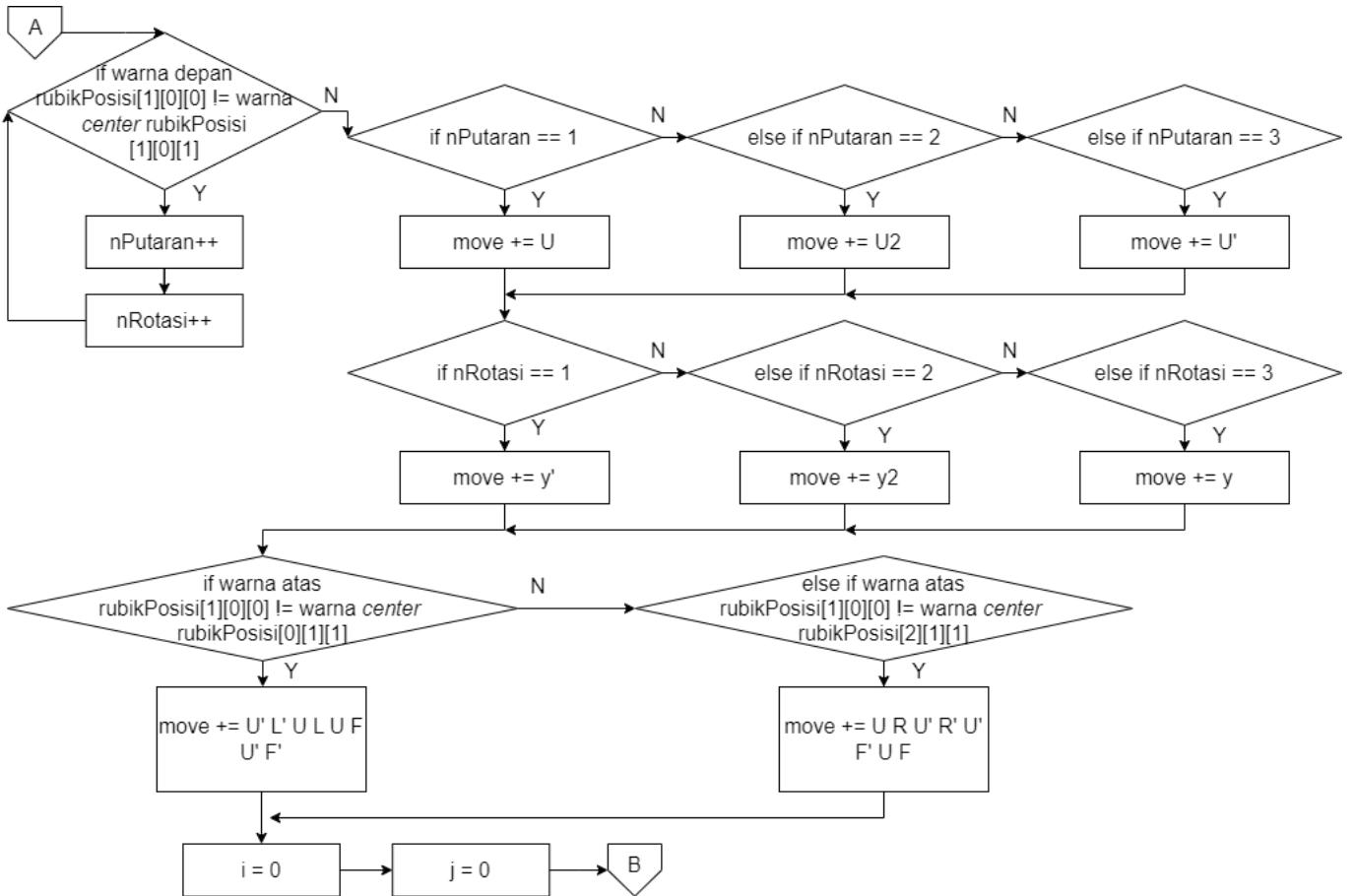
Gambar 3.31 Flowchart Cek *Edges* Bukan Kuning di Atas

Gambar 3.32 dan Gambar 3.33 merupakan flowchart untuk memasukkan *edges* ke posisi yang tepat. Peratama lakukan *nested loop* sebanyak tiga kali untuk memeriksa *layer* atas rubik untuk mencari *edges* yang tidak berwarna kuning. Jika *edges* yang diingikan berada di sisi kiri maka gerakan adalah U', jika di kanan maka U, jika di belakang maka U2, jika di depan makan tidak perlu gerakan. Lalu periksa

apakah warna depan dari *edge* tersebut sama dengan warna *center* nya, jika sudah sama tidak perlu gerakan, jika tidak sama putar U sampai sesuai dengan warna *center* dan rotasi berdasarkan kebalikan gerakan U yang dilakukan. Terakhir periksa warna atas dari *edge* jika warnanya sama dengan *center* di kanan, maka gerakannya adalah U' L' U L U F U' F', jika tidak maka U R U' R' U' F' U F.

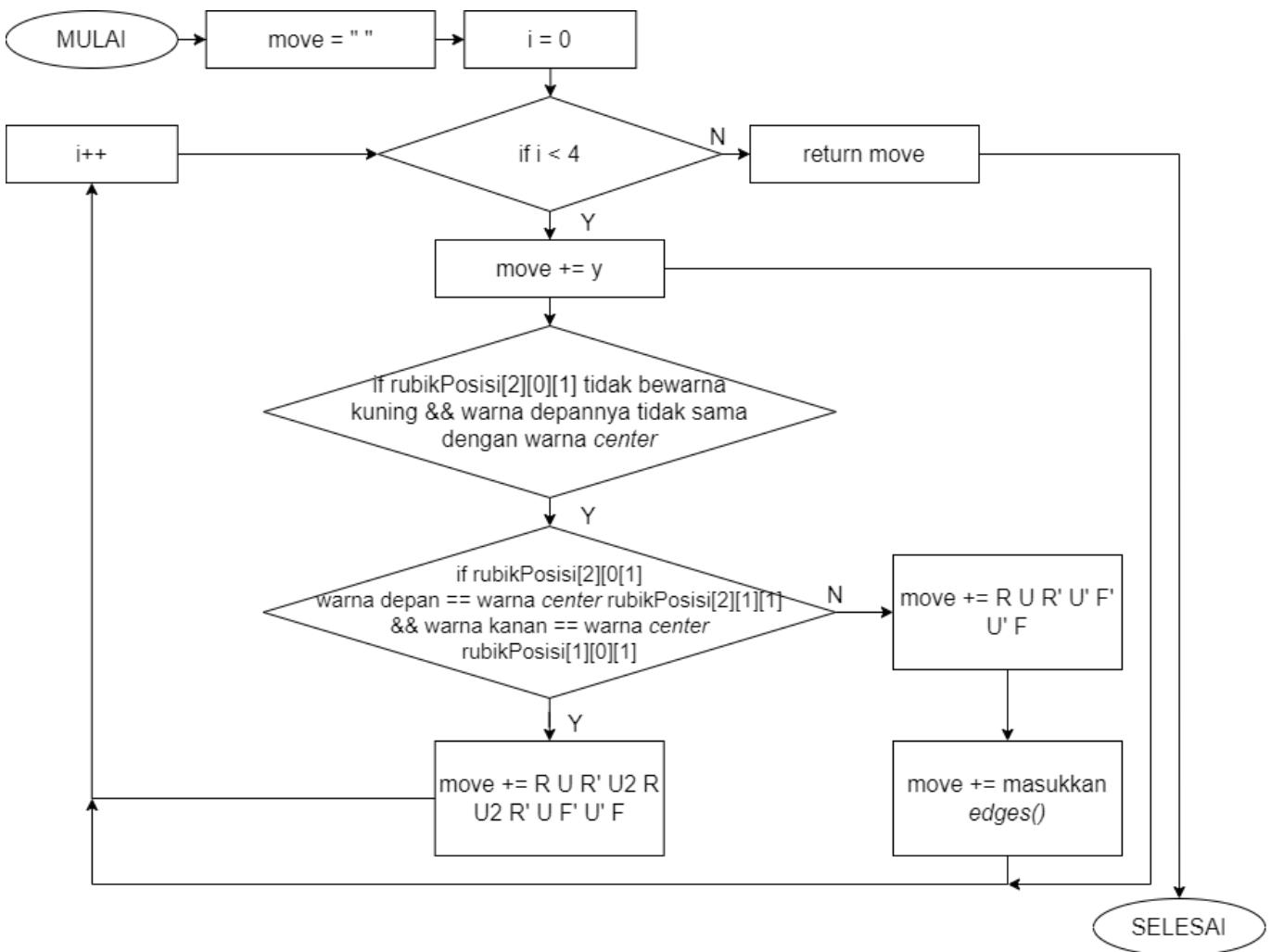


Gambar 3.32 Flowchart Masukkan *Edges*



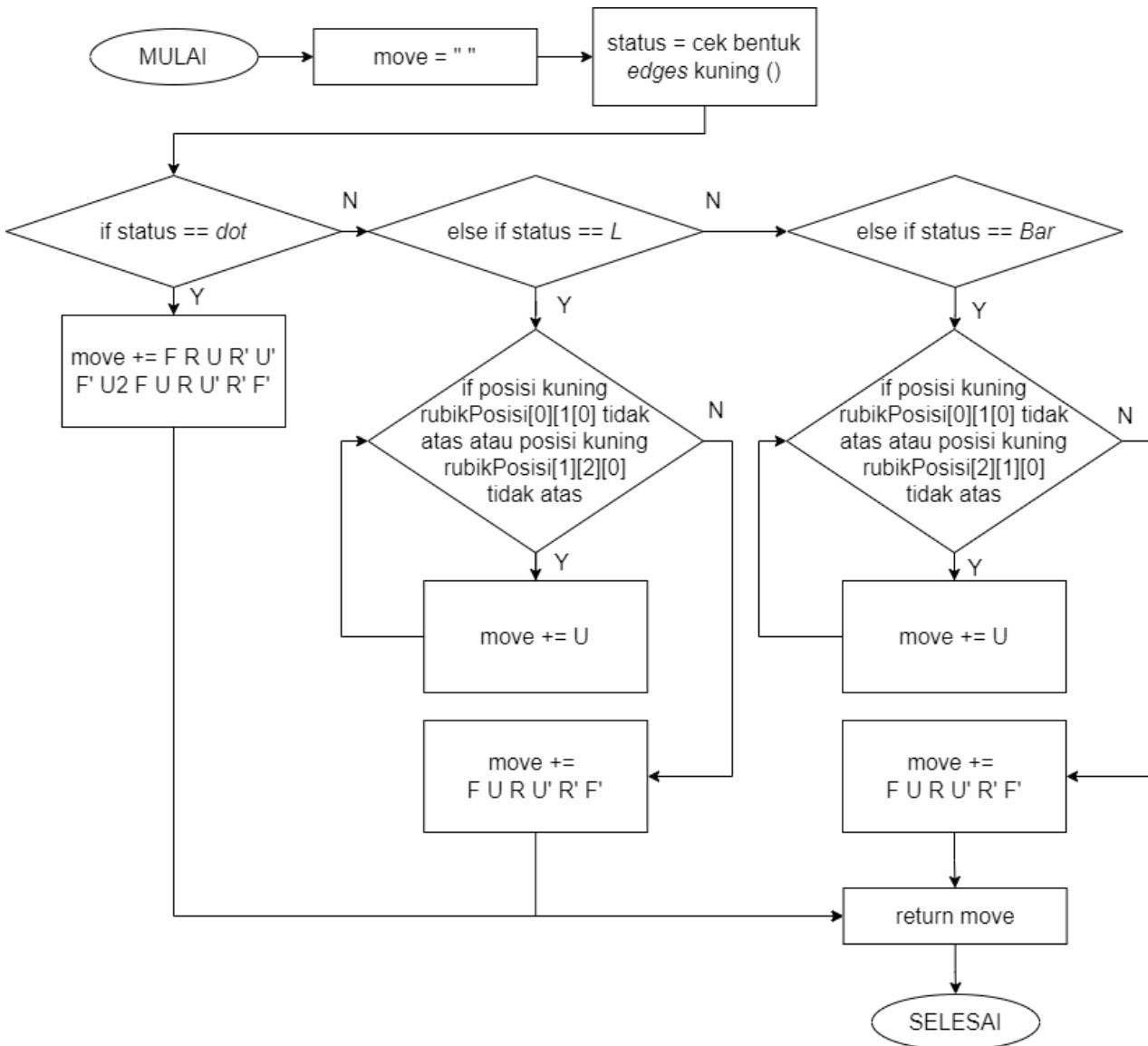
Gambar 3.33 Flowchart Masukkan *Edges* Lanj

Gambar 3.34 merupakan flowchar dari algoritma memperbaiki *edges* yang posisinya tidak tepat. Pertama yang dilakukan melakukan *loop* empat kali untuk memeriksa keempat *edges* di *layer* kedua, lalu dicek apakah keping benar atau tidak, jika tidak benar lalu diperiksa lagi apakah tempatnya sudah tepat namun kepingan terbalik atau posisinya benar-benar salah. Jika terbalik maka lakukan gerakan R U R' U2 R U2 R' U F' U' F', jika tidak maka lakukan gerak R U R' U' F' U' F lalu lakukan kembali masukkan *edges*.



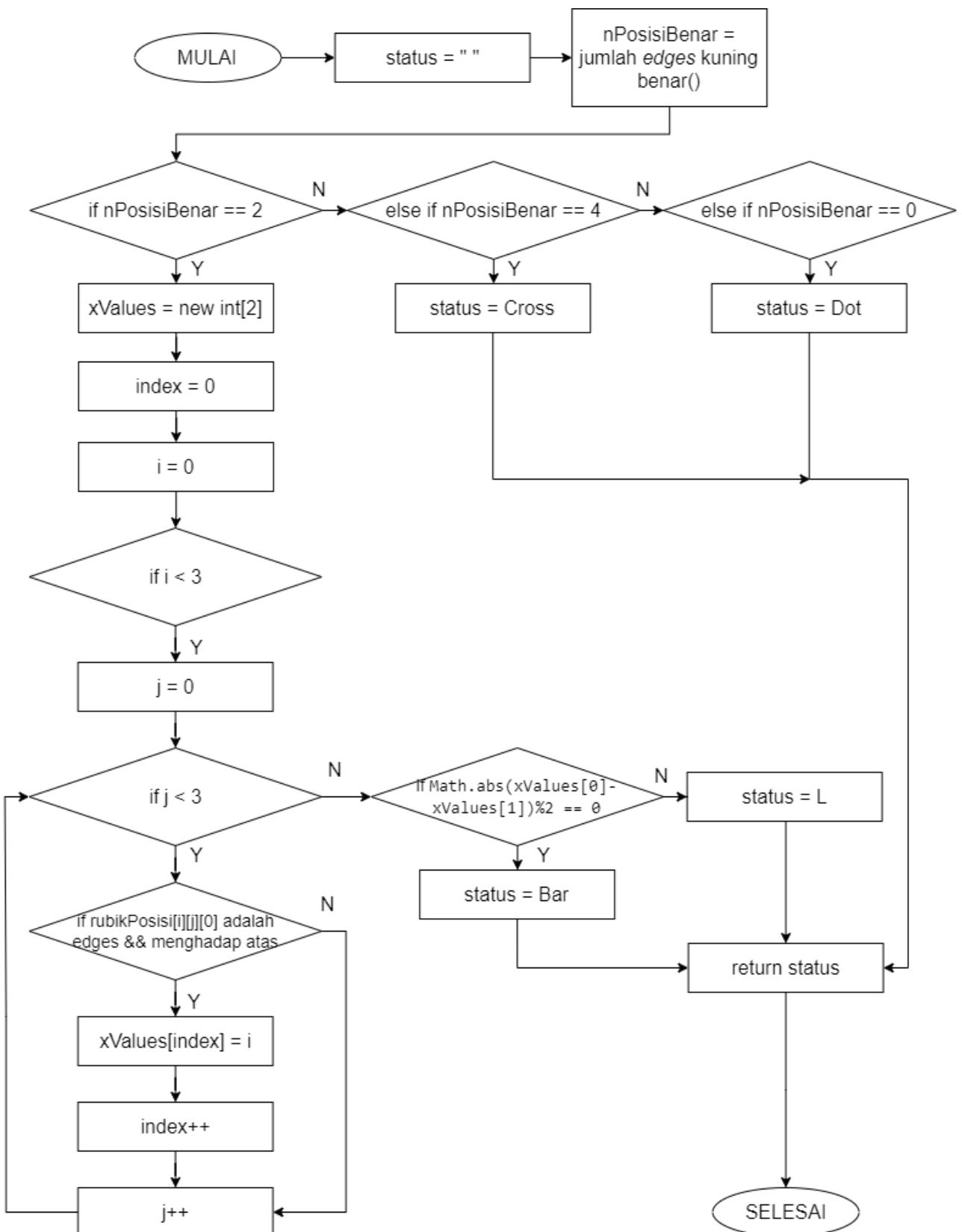
Gambar 3.34 Flowchart Perbaiki *Edges*

Gambar 3.35 merupakan flowchart algoritma membuat tanda tambah berwarna kuning atau pada *layer* atas. Langkah pertama memeriksa bentuk *edges* kuning saat ini. Jika berbentuk *dot* (tidak ada *edges* kuning yang menghadap atas) maka lakukan gerakan F R U R' U' F' U2 F U R U' R' F'. Jika berbentuk L maka periksa orientasi dari bentuk L tersebut, jika belum pada posisi yang tepat putar U sampai posisinya tepat, jika sudah tepat maka lakukan gerakan F U R U' R' F'. Jika berbentuk garis maka periksa apakah garisnya melintang secara horizontal, jika tidak maka putar U, jika tidak maka lakukan gerakan F U R U' R' F'.



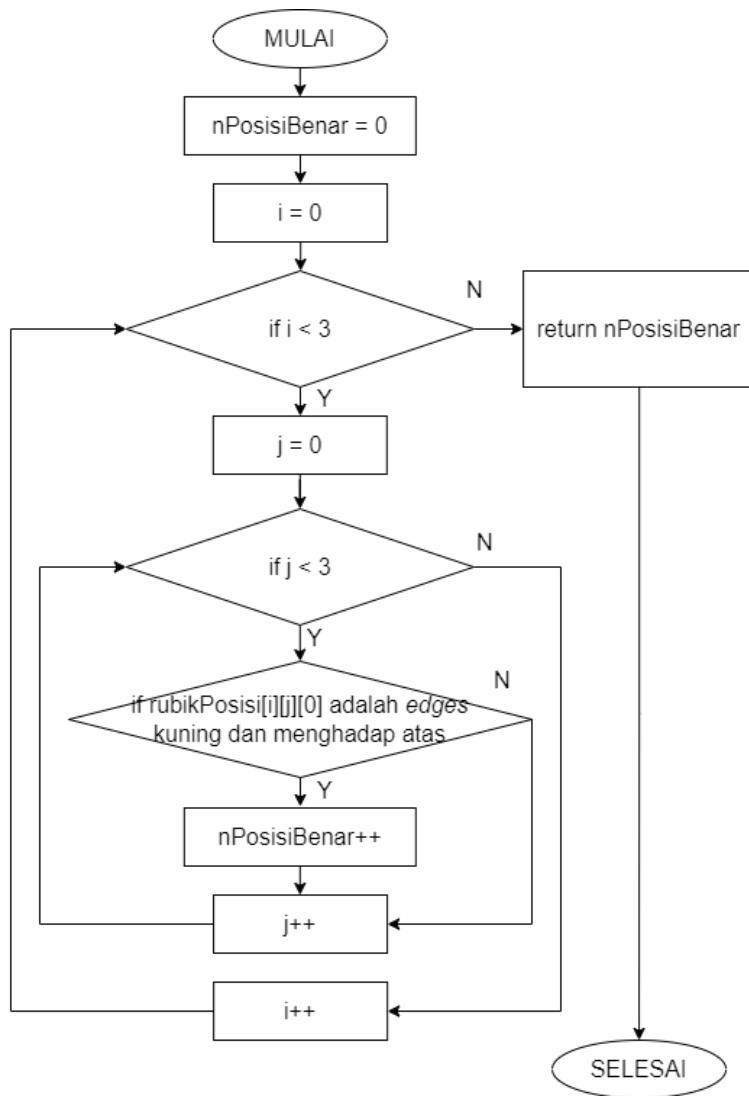
Gambar 3.35 Flowchart Menerapkan Algoritma *Yellow Cross*

Gambar 3.36 merupakan flowchart untuk memeriksa bentuk dari *edges* kuning. Langkah pertama dengan menghitung jumlah *edges* kuning yang menghadap ke atas. Jika berjumlah empat maka berbentuk *Cross*, jika berjumlah nol maka berbentuk *Dot*, dan jika berjumlah dua maka ada dua kemungkinannya yaitu berbentuk garis atau *L*.



Gambar 3.36 Flowchart Cek Bentuk Edges Kuning

Gambar 3.37 adalah flowchart untuk menghitung jumlah *edges* kuning yang menghadap atas. Langkah pertama yang dilakukan adalah melakukan *nested loop* tiga kali untuk memeriksa seluruh *layer* atas jika kepingan adalah *edges* berwarna kuning dan menghadap atas jumlah *edges* kuning benar bertambah satu.



Gambar 3.37 Flowchart Jumlah *Edges* Kuning dengan Posisi Benar

Pseudocode di bawah untuk menerepkan algoritma OLL (*Orientation Last Layer*), yaitu algoritma agar semua warna pada *layer* paling atas yang menghadap atas berwarna kuning atau satu warna dengan menyusun *corners* agar tepat posisinya. Untuk tahap ini hanya akan menggunakan satu rumus *sune algorithm* yaitu R U R' U R U2 R'. Langkah pertama yang perlu dilakukan adalah menghitung jumlah *corner* yang sudah benar. Jika berjumlah empat maka semua *corners* sudah tepat dan dapat

lanjut ke algoritma selanjutnya. Jika tidak maka periksa jumlah *corners* yang benar, jika nol maka periksa apakah warna kuning pada kepingan rubik kiri depan apakah menghadap kiri, jika iya lakukan rumus *sune* lalu kembali hitung jumlah *corners* benarnya dan kembali periksa jumlahnya, jika tidak maka putar U lalu kembali hitung jumlah warna *corners* benarnya dan kembali periksa jumlahnya. Jika satu periksa apakah warna kuning pada kepingan rubik kiri depan apakah menghadap atas jika iya lakukan rumus *sune*, jika tidak putar U dan kembali periksa. Jika dua periksa apakah warna kuning pada kepingan rubik kiri depan apakah menghadap depan jika iya lakukan rumus *sune*, jika tidak putar U dan kembali periksa. Lalukan hal ini berulang sampai *corners* benarnya berjumlah empat.

```

FUNCTION orientLastLayer()
DECLARE moves STRING
DECLARE numOriented INTEGER
SET numOriented TO numYellowCornersOriented()

WHILE numOriented != 4

    IF numOriented == 0
        WHILE cubiePos[0][0][0].getDirOfColor('Y') != 'L'
            moves = moves CONCAT performMoves("U ")
        END WHILE
        moves = moves CONCAT performMoves("R U R' U R U2 R' ")

    ELSE IF numOriented == 1
        WHILE cubiePos[0][0][0].getDirOfColor('Y') != 'U'
            moves = moves CONCAT performMoves("U ")
        END WHILE
        moves = moves CONCAT performMoves("R U R' U R U2 R' ")

    ELSE IF numOriented == 2
        WHILE cubiePos[0][0][0].getDirOfColor('Y') != 'F'
            moves = moves CONCAT performMoves("U ")
        END WHILE
        moves = moves CONCAT performMoves("R U R' U R U2 R' ")
    
```

```

END IF

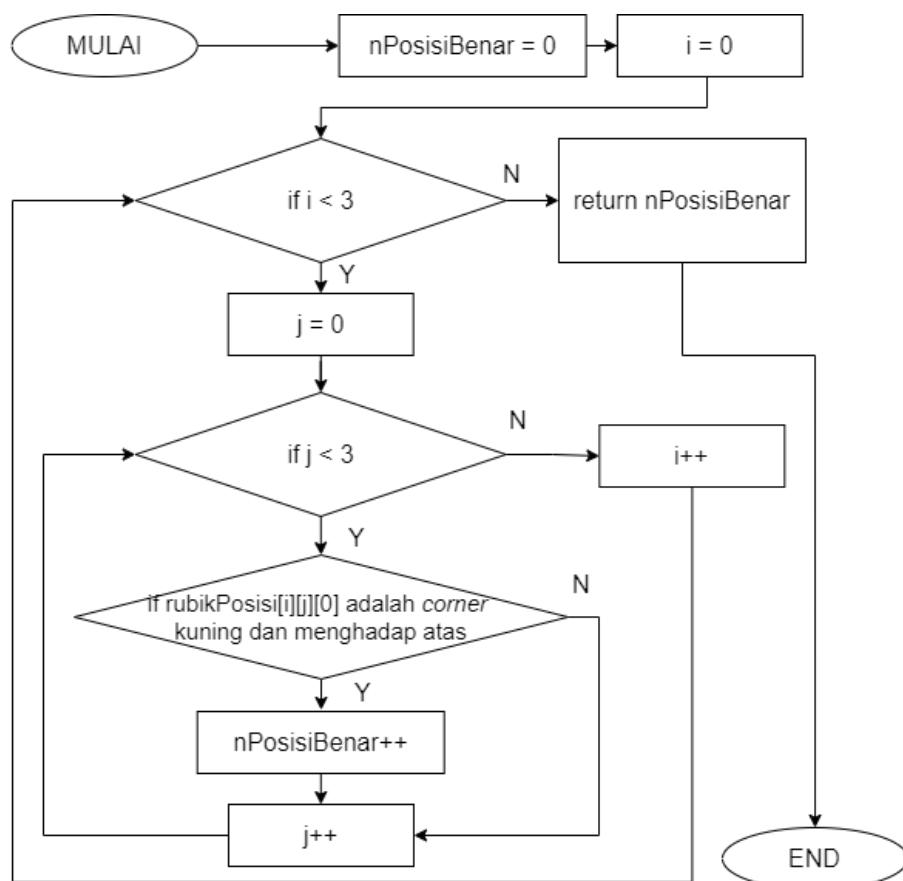
SET numOriented TO numYellowCornersOriented()

END WHILE

RETURN optimizeMoves(moves)
END FUNCTION

```

Gambar 3.38 merupakan flowchart untuk menghitung jumlah *corners* benar, yang warnanya kuningnya menghadap atas. Langkah pertama lakukan adalah *nested loop* tiga kali untuk memeriksa seluruh *layer* atas, lalu periksa jika kepingan merupakan *edges* kuning dan menghadap atas maka jumlah *corners* benar bertambah satu.



Gambar 3.38 Flowchart Jumlah *Corner* Kuning dengan Posisi Benar

Pseudocode di bawah merupakan algoritma PLL (*Permutation of Last Layer*) yang berfungsi untuk menyusun rubik bagian atas namun tidak menghadap atas. Hal pertama yang dilakukan adalah melakukan perulangan sebanyak empat kali untuk memeriksa *corner* yang telah selesai dengan mencocokkan warna *corners*, jika warna *corner* pada kiri depan sama dengan warna *corner* kanan depan maka jumlah benar bertambah satu lalu putar y jika tidak putar y. Lalu periksa jumlah *coorners* benarnya, jika nol maka lakukan gerakan R' F R' B2 R F' R' B2 R2 dan jumlah benar menjadi satu lalu lanjut ke percabangan jika selanjutnya. Jika satu maka periksa apakah warna belakang kepingan pada kiri belakang sama dengan warna belakang kepingan kanan belakang, jika tidak maka lanjut jumlah *edges* benar adalah nol, jika iya maka putar U, jika tidak maka lakukan gerakan R' F R' B2 R F' R' B2 R2 dan lanjut jumlah *edges* benar adalah nol. Langkah selanjutnya lakukan perluangan sebanyak empat kali untuk memeriksa *edges* yang benar dengan mencocokkan warna *corner* kiri depan sama dengan warna *edge* tengah depan, jika iya maka jumlah *edges* benar bertambah satu lalu putar y, jika tidak putar y. Lalu periksa jika jumlah *edges* benar adalah nol maka lakukan gerakan R2 U R U R' U' R' U' R' U R' lalu lanjut ke percabangan selanjutnya. Jika jumlah *edges* benar adalah satu maka periksa apakah warna belakang kepingan pada kiri belakang sama dengan warna belakang kepingan tengah belakang, jika tidak maka putar U sampai benar, jika iya maka periksa apakah warna depan kepingan tengah depan sama dengan warna kiri kepingan kiri depan, jika iya maka lakukan gerakan R2 U R U R' U' R' U' R' U R', jika tidak maka lakukan gerakan R U' R U R U R' U' R' U' R2. Terakhir periksa apakah warna depan rubik kiri depan sama dengan warna *center* nya, jika tidak maka putar U sampai benar, jika iya maka rubik telah selesai.

```

FUNCTION permuteLastLayer()
    DECLARE moves STRING
    DECLARE numHeadlights INTEGER
    DECLARE numSolved INTEGER

    SET numHeadlights TO 0
    FOR i FROM 0 TO 3
        turn("y")
    
```

```

IF cubiePos[0][0][0].getColorOfDir('F') == cubiePos[2][0][0].getColorOfDir('F')
    SET numHeadlights TO numHeadlights + 1
END FOR

IF numHeadlights == 0
    moves = moves CONCAT performMoves("R' F R' B2 R F' R' B2 R2 ")
    SET numHeadlights TO 1
ELSE IF numHeadlights == 1
    WHILE cubiePos[0][2][0].getColorOfDir('B') != cubiePos[2][2][0]
        →.getColorOfDir('B')
        moves = moves CONCAT performMoves("U ")
    END WHILE
    moves = moves CONCAT performMoves("R' F R' B2 R F' R' B2 R2 ")
END IF

SET numSolved TO 0
FOR i FROM 0 TO 3
    turn("y")
    IF cubiePos[0][0][0].getColorOfDir('F') == cubiePos[1][0][0].getColorOfDir('F')
        SET numSolved TO numSolved + 1
    END FOR

    IF numSolved == 0
        moves = moves CONCAT performMoves("R2 U R U R' U' R' U' R' U R' ")
        SET numSolved TO 1
    ELSE IF numSolved == 1
        WHILE cubiePos[0][2][0].getColorOfDir('B') != cubiePos[1][2][0]
            →.getColorOfDir('B')
            moves = moves CONCAT performMoves("U ")
        END WHILE
        IF cubiePos[1][0][0].getColorOfDir('F') == cubiePos[0][0][0].getColorOfDir('L')
            moves = moves CONCAT performMoves("R2 U R U R' U' R' U' R' U R' ")
        ELSE
            moves = moves CONCAT performMoves("R U' R U R U R U' R' U' R2 ")
        END IF
    END IF
END IF

```

```

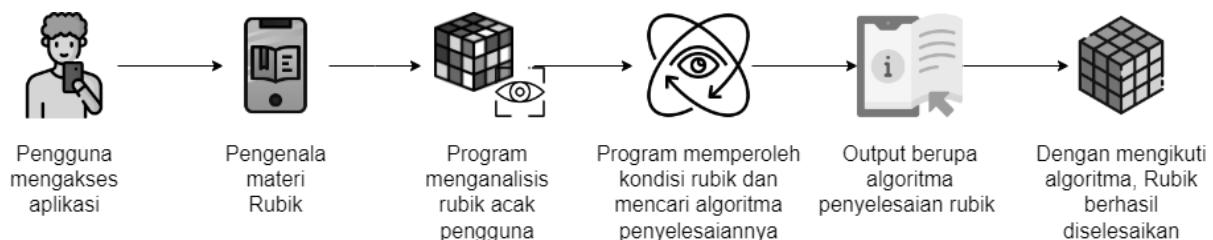
    WHILE cubiePos[0][0][0].getColorOfDir('F') != cubiePos[1][0][1]
        →.getColors()[0].getColor()
        moves = moves CONCAT performMoves("U ")
    END WHILE

    RETURN optimizeMoves(moves)
END FUNCTION

```

3.3. Diagram Umum Sistem

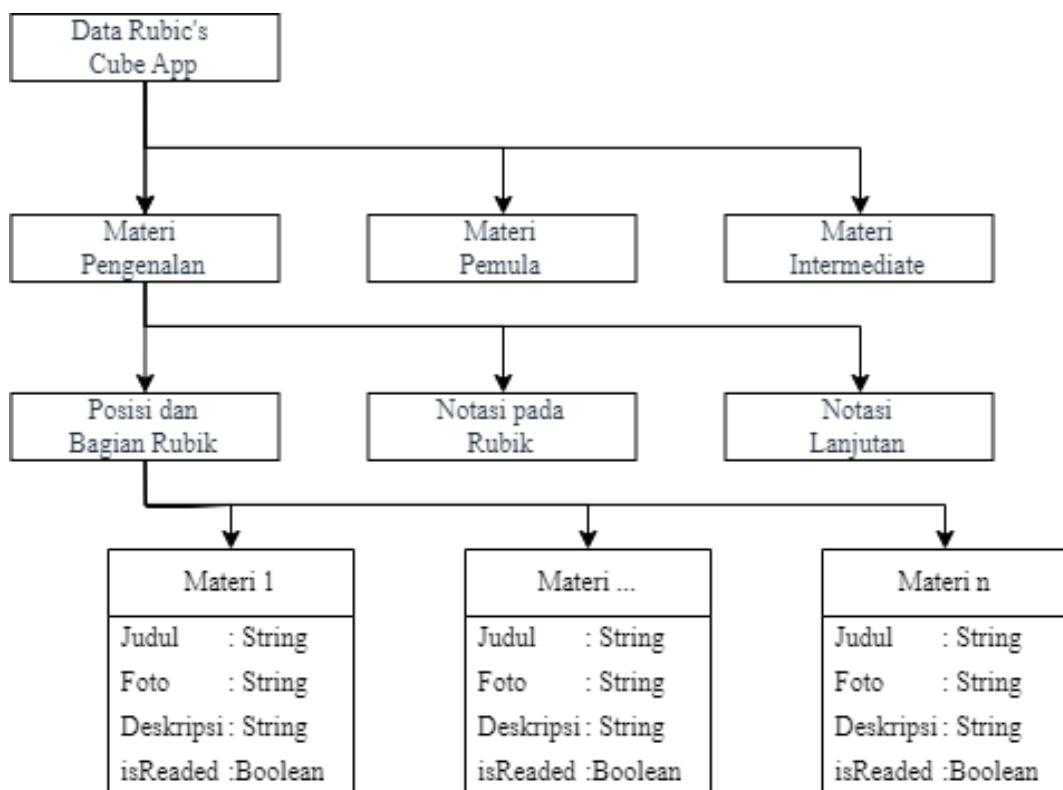
Ketika pengguna pertama kali menggunakan aplikasi, hal pertama yang diperoleh adalah pengenalan dan materi dasar tentang Rubik. Pada materi dasar akan diajari mengenai pengetahuan dasar seperti urutan warna, warna dasar, notasi pada Rubik, nama dari setiap bagian Rubik, urutan penyelesaian Rubik beserta istilahnya. Pada modul selanjutnya sudah mulai belajar cara menyelesaikan Rubik. Fitur selanjutnya aplikasi akan menganalisis Rubik pengguna lalu setelah mengetahui kondisi Rubik, sistem akan memberi output berupa penjelasan dan algoritmanya. Setelah pengguna selesai mengikuti semua modul maka Rubik berhasil diselesaikan. Hal ini dapat digambarkan dalam bentuk diagram seperti gambar 3.39.



Gambar 3.39 Diagram Umum Sistem

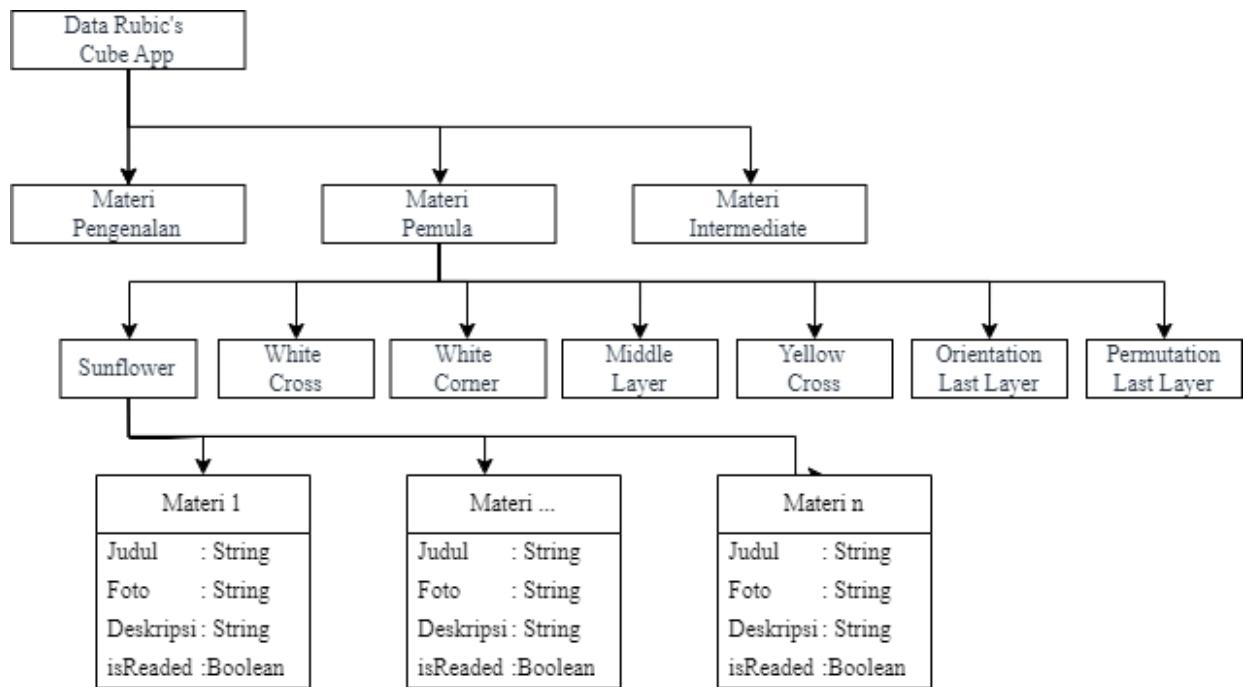
3.4. Diagram Struktur Berkas

Sistem ini dirancang menggunakan *database* Firebase. Rancangan *file structure diagram* untuk *database* dapat dilihat pada gambar 3.40 dan gambar 3.41. Gambar 3.40 merupakan perancangan *file structure* untuk menampung data materi pengenalan rubik yang berisi tentang nama bagian-bagian rubik dan notasi rubik. Setiap materi berisi empat data diantaranya Judul yang menampung nama materi, Foto yang menampung nama foto yang ingin ditampilkan, Deskripsi berisi data yang menjelaskan materi tersebut dan *IsReaded* yang menampung data *Boolean* sebagai penanda apakah materi ini sudah dibaca atau belum.



Gambar 3.40 Diagram Struktur Berkas Materi Pengenalan

Gambar 3.41 merupakan perancangan *file structure* untuk menampung data materi pemula yang berisi cara menyelesaikan rubik dengan menggunakan metode *Layer by Layer* (LBL). Pada Materi ini seberisi tujuh sub materi yang berisi tahap-tahap menyelesaikan rubik diantaranya *Sunflowe*, *White Cross*, *White Corner*, *Middle Layer*, *Yellow Cross*, *Orientation Last Layer* dan *Permutation Last Layer* yang masing-masing penjelasan dan tujuan dari setiap tahap dapat dilihat pada landasan teori rubik.



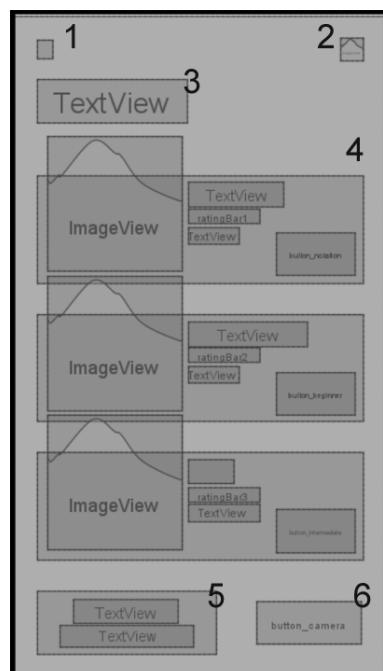
Gambar 3.41 Diagram Struktur Berkas Materi Tingkat Pemula

3.5. Perancangan Antarmuka

Antarmuka atau *User interface* adalah elemen yang sangat penting karena dapat memengaruhi pengalaman pengguna secara keseluruhan. Dalam membangun perancangan antar muka tahap awalnya adalah membuat *wireframe*. *Wireframe* adalah struktur dasar dan tata letak untuk mengetahui gambaran kasar dari suatu halaman dan menonjolkan fungsionalitas aplikasi sebelum ditambahkan warna dan gambar. Seperti namanya, *Wireframe* dibuat hanya dengan menggunakan garis-garis dan bentuk dasar saja.

A. Main Menu Screen

Pada Gambar 3.42 dapat dilihat rancangan digital dari *Main Menu Screen*, yang merupakan rancangan untuk tampilan menu aplikasi.



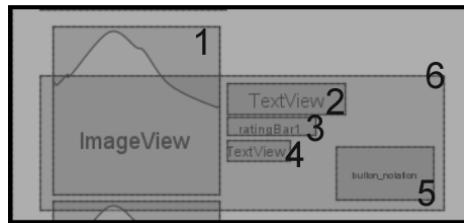
Gambar 3.42 Main Menu Screen

Keterangan:

1. *TextView*: menampilkan *greeting* “Hi”.
2. *ImageView*: menampilkan tampilan ikon *about*.
3. *TextView*: menampilkan kalimat perintah “Pilih Level”.
4. *LinearLayout*: menampilkan list level yang tersedia.
5. *CardView*: menampung tulisan *progress* belajar terakhir dan berfungsi juga sebagai tombol untuk menuju materi terakhir yang dibaca.
6. *ButtonView*: menuju halaman *ScanActivity*.

B. Item Main Menu Screen

Pada Gambar 3.43 merupakan rancangan digital dari *Item Main Menu Screen* yaitu desain dari *items* pilihan menu.



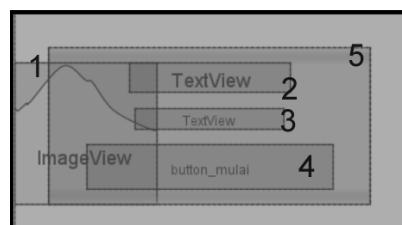
Gambar 3.43 Item Main Menu Screen

Keterangan:

1. *ImageView*: menampilkan gambar karakter.
2. *TextView*: menampilkan nama level.
3. *RatingBar*: menampilkan tingkat kesulitan level.
4. *TextView*: menampilkan deskripsi level.
5. *ButtonView*: tombol untuk menuju langsung ke halaman level.
6. *CardView*: membungkus setiap *view* di atas.

C. Main Popup Menu Screen

Pada Gambar 3.44 merupakan rancangan digital dari *Main Popup Menu Screen* yaitu rancangan tampilan ketika salah satu pilihan menu di klik.



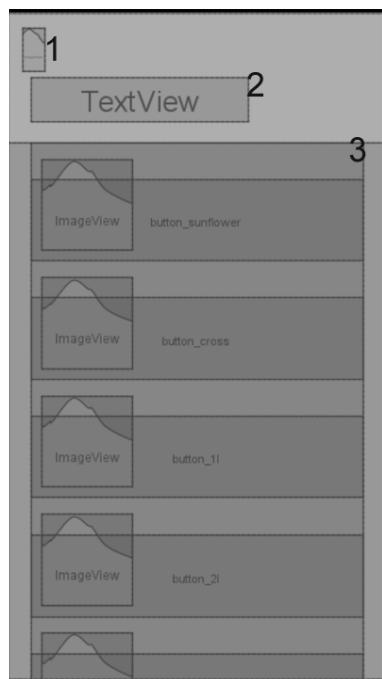
Gambar 3.44 Main Popup Menu Screen

Keterangan:

1. *ImageView*: menampilkan gambar karakter level yang dipilih.
2. *TextView*: menampilkan nama level yang dipilih.
3. *TextView*: menampilkan tulisan deskripsi.
4. *ButtonView*: menampilkan tombol menuju halaman level yang dipilih.
5. *CardView*: membungkus semua *view* di atas.

D. Level Menu Screen

Pada Gambar 3.45 merupakan rancangan digital dari *Level Menu Screen* yaitu tampilan yang muncul ketika pengguna memilih belajar pada menu awal.



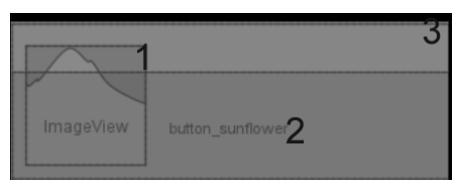
Gambar 3.45 Level Menu Screen

Keterangan:

1. *ImageButton*: tombol untuk kembali.
2. *TextView*: menampilkan tulisan level yang dipilih.
3. *RecyclerView*: menampung *item* materi yang tersedia.

E. Item Level Menu Screen

Pada Gambar 3.46 merupakan rancangan digital dari *Item Level Menu Screen* merupakan rancangan tampilan *items* dari setiap pilihan menu level.



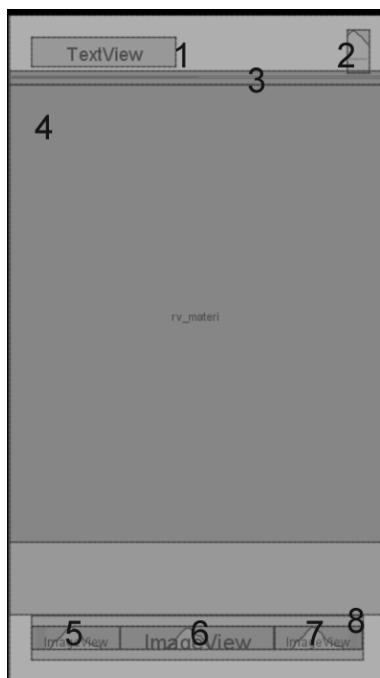
Gambar 3.46 Item Level Menu Screen

Keterangan:

1. *ImageView*: menampilkan gambar representasi dari materi.
2. *TextView*: menampilkan judul dari materi.
3. *CardView*: membungkus semua *view*.

F. Materi *Screen*

Pada Gambar 3.47 merupakan rancangan digital dari Materi *Screen* yaitu ketika salah satu dari menu level diklik.



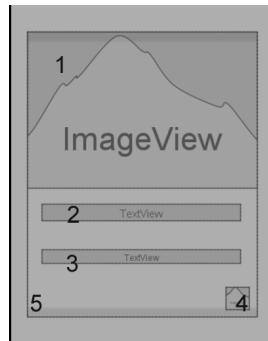
Gambar 3.47 Materi *Screen*

Keterangan:

1. *TextView*: menampilkan tulisan materi yang dipilih.
2. *ImageView*: tombol untuk menutup halaman materi.
3. *ProgressBar*: menampilkan progress belajar pada satu materi.
4. *RecyclerView*: menampung list untuk menampilkan data.
5. *ImageView*: menampilkan ikon panah ke kiri untuk tombol kembali.
6. *ImageView*: menampilkan ikon *play* atau *pause*.
7. *ImageView*: menampilkan ikon panah ke kanan untuk tombol *next*.
8. *CardView*: membungkus ikon *previos*, *play/pause*, *next*.

G. Item Materi Screen

Pada Gambar 3.48 merupakan rancangan digital dari *Item Materi Screen* yaitu tampilan *items* dari tampilan materi.



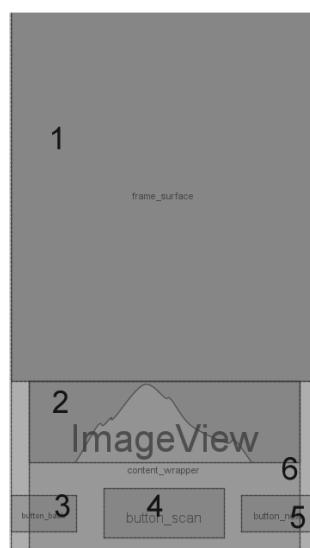
Gambar 3.48 Item Materi Screen

Keterangan:

1. *ImageView*: menampilkan gambar terkait materi.
2. *TextView*: menampilkan nama dari materi.
3. *TextView*: menampilkan isi dari materi.
4. *ImageView*: menampilkan ikon apakah materi telah dibaca.
5. *CardView*: membungkus semua *view* di atas.

H. Scan Screen

Pada Gambar 3.49 merupakan rancangan digital dari *Scan Screen* yaitu rancangan tampilan ketika memindai rubik acak pengguna.



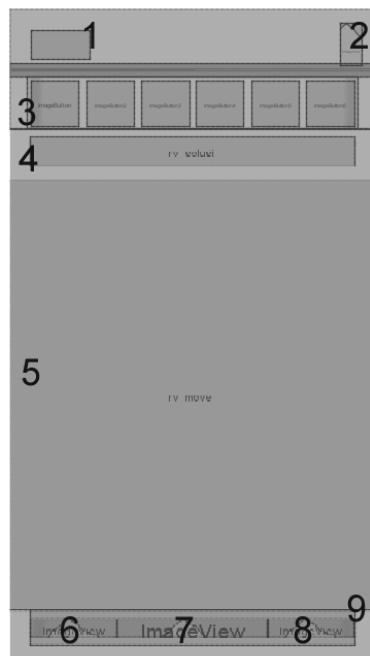
Gambar 3.49 Scan Screen

Keterangan:

1. *CameraPreview*: menampilkan tampilan dari kamera.
2. *ImageView*: menampilkan hasil *scan* dari kamera.
3. *ButtonView*: menampilkan tombol untuk kembali.
4. *ButtonView*: menampilkan tombol untuk melakukan *scan*.
5. *ButtonView*: menampilkan tombol untuk mencari algoritma penyelesaian.
6. *CardView*: membungkus tampilan hasil *scan*.

I. Solution Screen

Pada Gambar 3.50 merupakan rancangan digital dari *Solution Screen* yaitu tampilan ketika rubik berhasil dipindai.



Gambar 3.50 Solution Screen

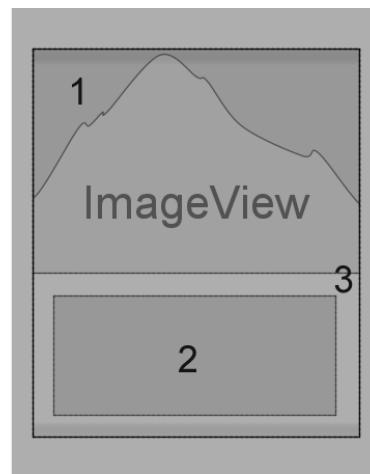
Keterangan:

1. *TextView*: menampilkan nama tahapan penyelesaian.
2. *ImageView*: menampilkan ikon ‘X’ sebagai tombol *close*.
3. *LinearLayout* dan *ButtonView*: menampilkan kumpulan tombol tahapan penyelesaian.
4. *RecyclerView*: menampilkan algoritma penyelesaian rubik.
5. *RecyclerView*: menampilkan visualisasi dari algoritma penyelesaian rubik.
6. *ImageView*: menampilkan ikon panah ke kiri untuk tombol kembali.

7. *ImageView*: menampilkan ikon *play* atau *pause*.
8. *ImageView*: menampilkan ikon panah ke kanan untuk tombol *next*.
9. *CardView*: membungkus ikon *previos*, *play/pause*, *next*.

J. Item Solution Screen

Pada Gambar 3.51 merupakan rancangan digital dari *Item Solution Screen* yaitu tampilan dari setiap *items* tampilan solusi.



Gambar 3.51 Item Solution Screen

Keterangan:

1. *ImageView*: menampilkan gambar dari tiap notasi rubik.
2. *TextView*: menampilkan notasi.
3. *CardView*: membungkus semua *view* di atas.

BAB 4

IMPLEMENTASI DAN PEMBAHASAN

4.1. Spesifikasi Hardware dan Software

Untuk membangun aplikasi belajar menyelesaikan rubiks 3x3x3 dibutuhkan *software* dan *hardware* yang mendukung untuk membangun aplikasi ini. Pada penelitian ini, penulis menggunakan *software* dan *hardware* dengan spesifikasi sebagai berikut .

4.1.1. *Hardware* (Perangkat Keras)

Berikut adalah spesifikasi *hardware* yang digunakan penulis untuk membangun sistem dalam penelitian ini:

1. Processor AMD Athlon 300U
2. Radeon Vega Mobile Gfx 2.40 GHz
3. Installed RAM 8,00 GB
4. System type 64-bit operating system, x64-based processor
5. SSD 256GB

4.1.2. *Software and Tools* (Perangkat lunak dan alat-alat)

Berikut adalah spesifikasi *software dan tools* yang digunakan penulis untuk membangun sistem dalam penelitian ini:

1. Android Studio Flamingo | 2022.2.1
2. Java
3. Firebase
4. OpenCV
5. Emulator Android

4.1.3. Smartphone (Telepon pintar)

Berikut adalah spesifikasi *smartphone* yang digunakan penulis untuk membangun sistem dalam penelitian ini:

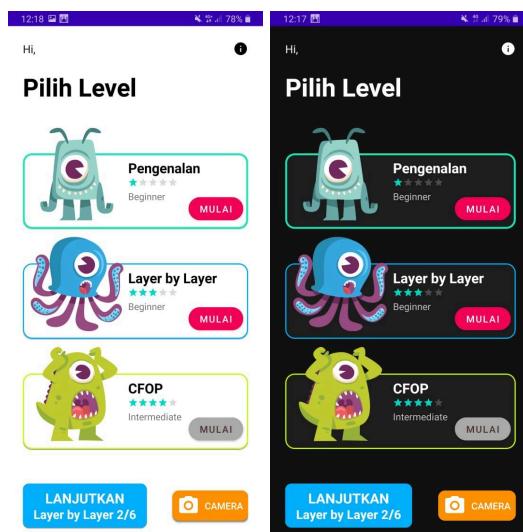
1. Samsung Galaxy A8 Star,
2. Layar 6.3" 2220x1080 Super AMOLED
3. Kamera 16-MP f/1.7 (Utama) + OIS24-MP f/1.7 + Selfie 24-MP f/2
4. Prosesor Snapdragon 660 SoC4GB
5. Memori RAM64GB PenyimpananMicroSDXC
6. Baterai 3700 mAh

4.2. Implementasi Sistem

Merupakan tahap realisasi rancangan sistem yang telah didesain untuk diimplementasikan secara nyata untuk digunakan oleh pengguna.

4.2.1. Main Menu Screen

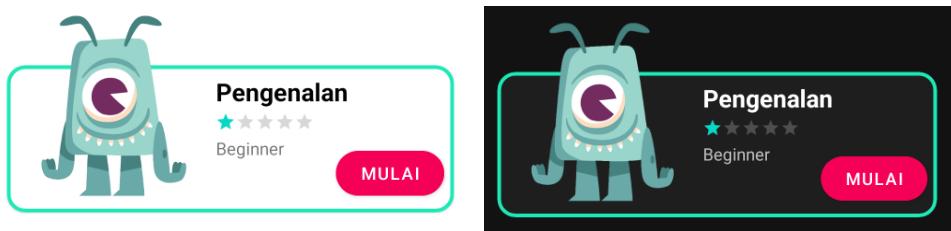
Gambar 4.1 merupakan tampilan menu awal dimana pengguna memilih ingin melakukan apa di aplikasi ini. Pada menu terdapat pilihan untuk belajar yang memiliki tiga materi yaitu materi pengenalan rubik, materi *Layer by layer*, materi CFOP (*Cross*, *F2L*, *OLL*, *PLL*). Ada juga pilihan lanjutkan yang merupakan *shortcut* untuk menuju materi terakhir yang dipelajari. Dan terakhir ada menu untuk melakukan pemindaian terhadap rubik untuk dicarikan algoritma penyelesaian rubik acak pengguna.



Gambar 4.1 Main Menu Screen

4.2.2. Item Main Menu Screen

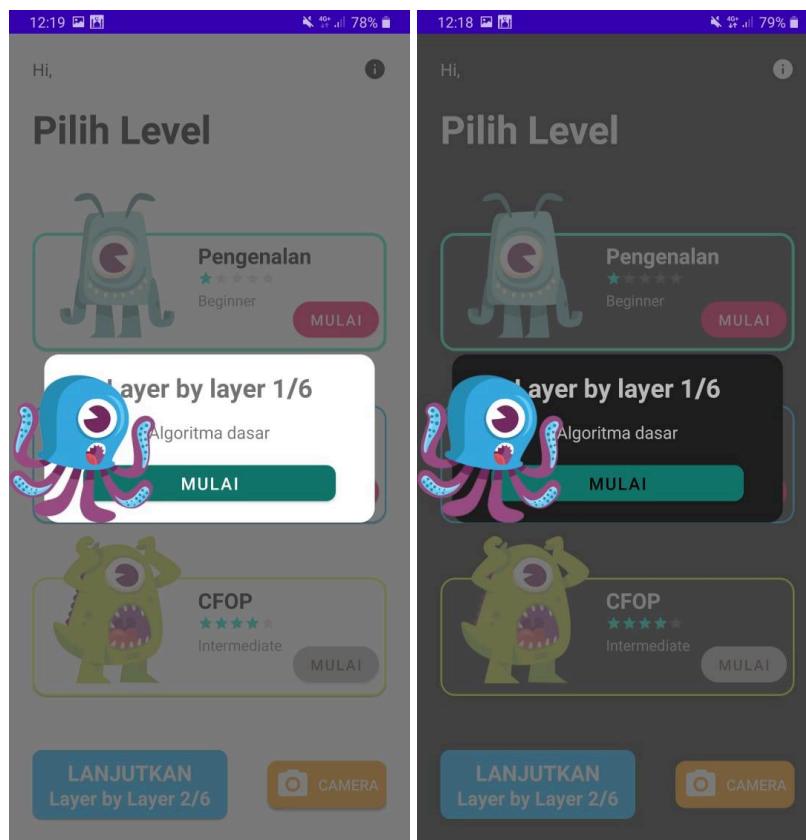
Gambar 4.2 merupakan tampilan dari setiap *item* pada pilihan menu awal. Jadi setiap pilihan di menu, tampilan memiliki *Item Main Menu Screen*. Dan dapat ditampilkan dalam *light mode* atau *dark mode*.



Gambar 4.2 Item Main Menu Screen

4.2.3. Main Popup Menu Screen

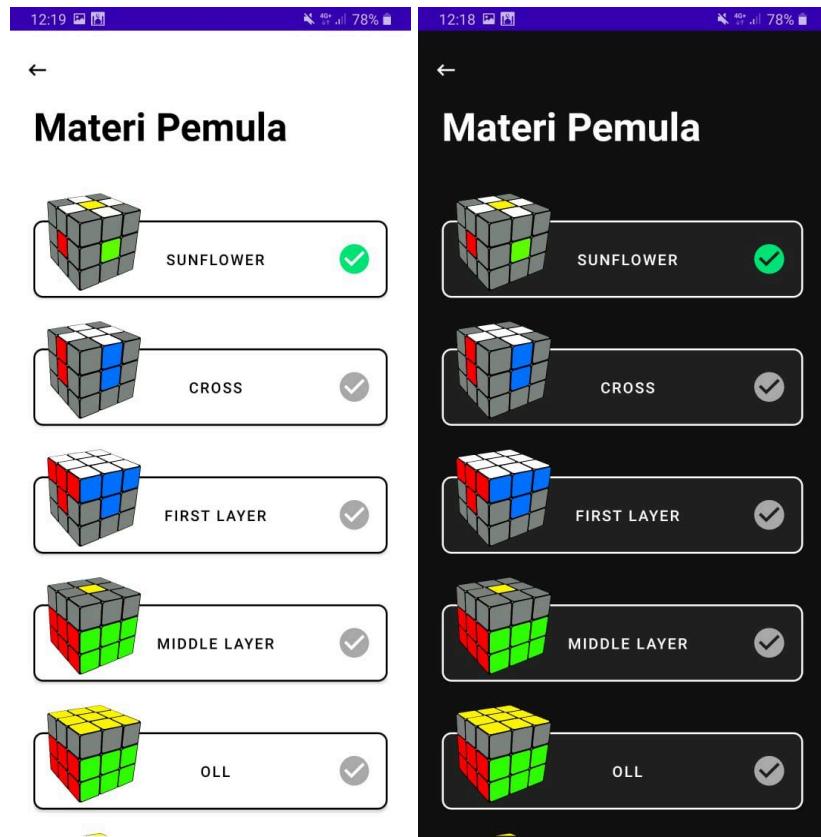
Gambar 4.3 merupakan tampilan yang muncul ketika salah satu menu dipilih. Dan dapat ditampilkan dalam *light mode* atau *dark mode*.



Gambar 4.3 Main Popup Menu Screen

4.2.4. Level Menu Screen

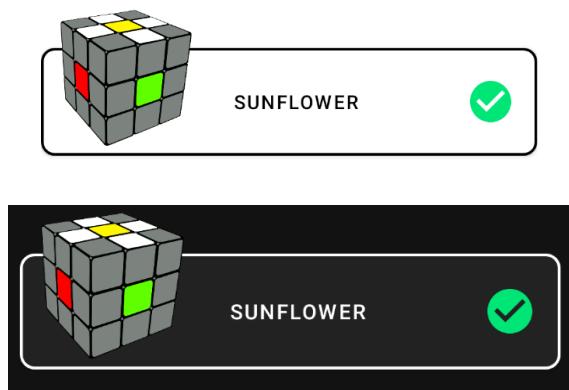
Gambar 4.4 merupakan tampilan menu yang muncul ketika salah satu materi dipilih. Dan dapat ditampilkan dalam *light mode* atau *dark mode*.



Gambar 4.4 Level Menu Screen

4.2.5. Item Level Menu Screen

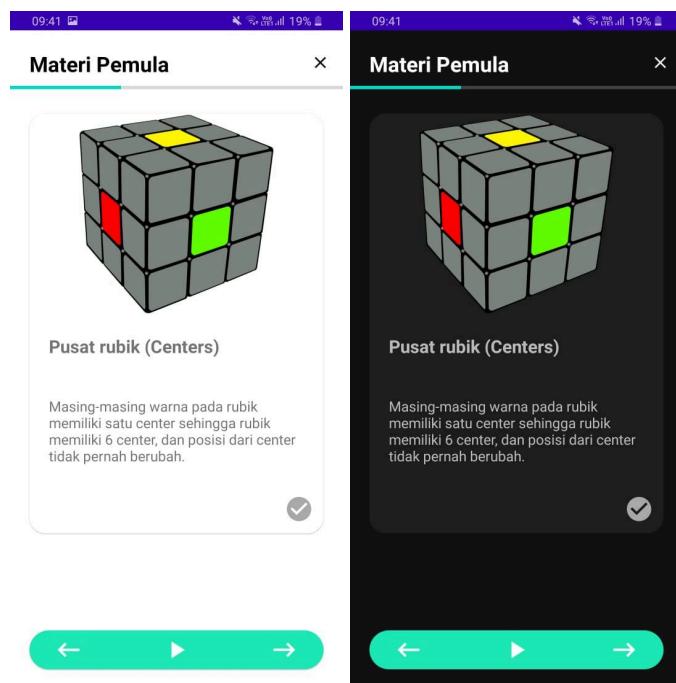
Gambar 4.5 merupakan tampilan dari setiap item pada pilihan menu materi. Dan dapat ditampilkan dalam *light mode* atau *dark mode*.



Gambar 4.5 Item Level Menu Screen

4.2.6. Materi Screen

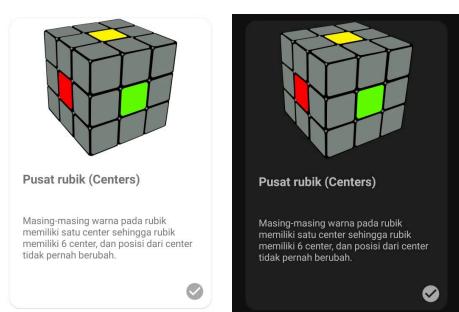
Gambar 4.6 merupakan tampilan materi yang muncul ketika pengguna mulai mempelajari suatu materi yang tersedia pada pilihan menu. Pada tampilan ini berisi judul dari materi, progress belajar pada materi tersebut, konten dari materinya serta ada tiga tombol yang memiliki fungsi diantaranya untuk lanjut ke materi selanjutnya, kembali ke materi sebelumnya, dan materi otomatis bergerak ke materi selanjutnya. Dan dapat ditampilkan dalam *light mode* atau *dark mode*.



Gambar 4.6 Materi Screen

4.2.7. Item Materi Screen

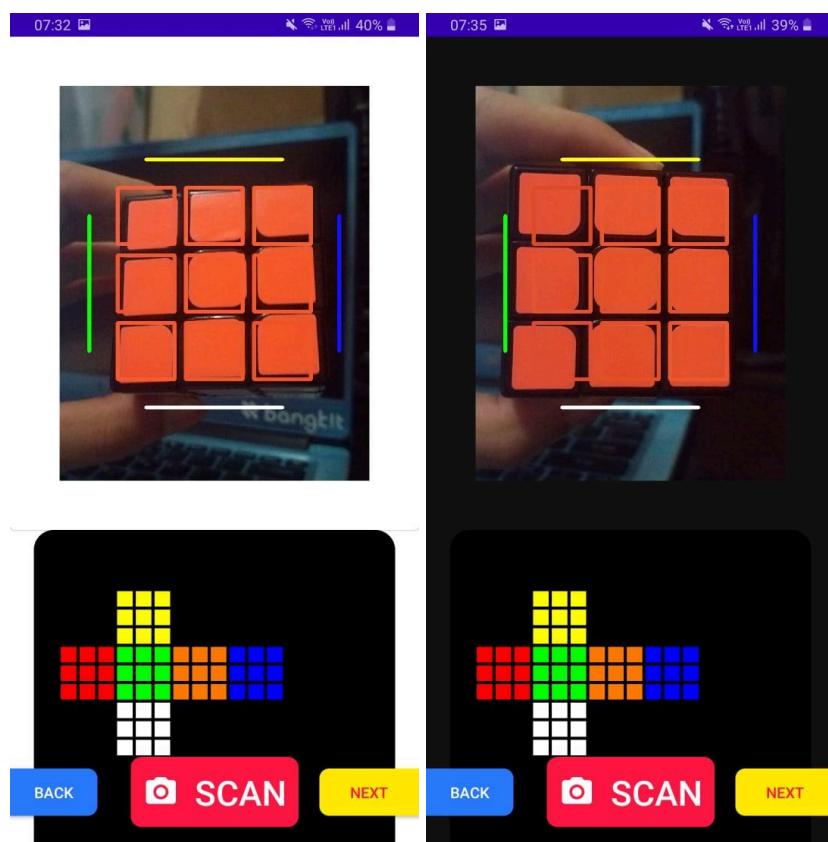
Gambar 4.7 merupakan tampilan penampung konten dari setiap materi. Dan dapat ditampilkan dalam *light mode* atau *dark mode*.



Gambar 4.7 Item Materi Screen

4.2.8. Scan Screen

Gambar 4.8 merupakan tampilan dari fitur scanning rubik pada bagian atas terdapat preview camera yang memiliki guide bagaimana posisi dan penempatan dari rubik ketika dipindai. Di bawahnya terdapat tampilan hasil pemindaian yang berfungsi sebagai pedoman pengguna apakah hasil pemindaian benar dengan kondisi sebenarnya, serta bagian paling bawah terdapat tiga tombol yang masing-masingnya memiliki fungsi berbeda diantaranya tombol paling kiri untuk kembali ke menu awal, tombol tengah untuk pindah rubik, dan tombol paling kanan untuk next mencari algoritma penyelesaian dari rubik acak tersebut.

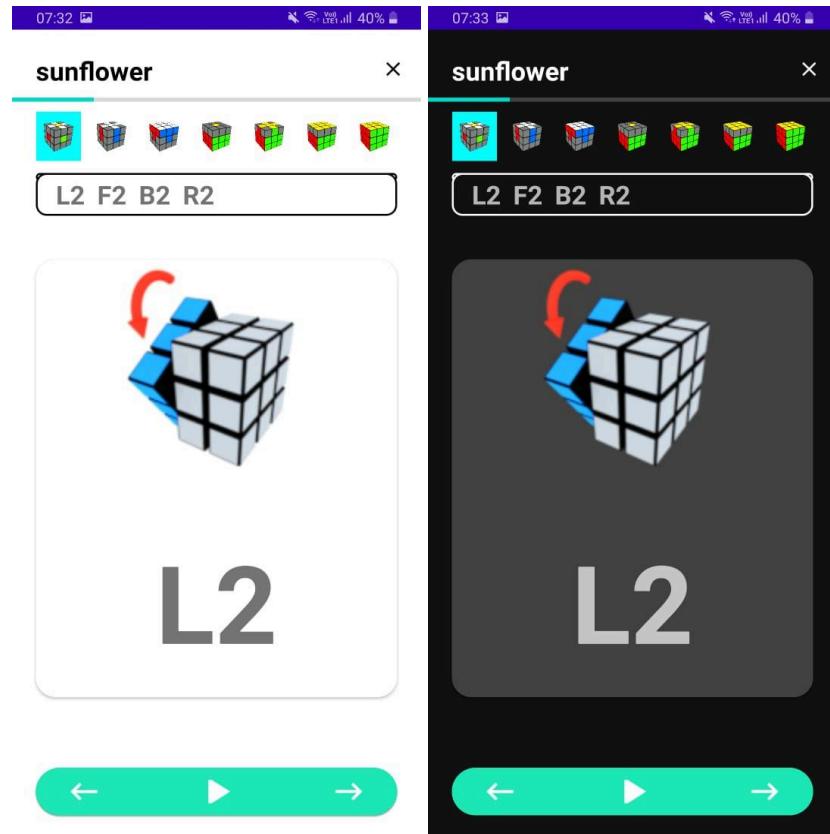


Gambar 4.8 Scan Screen

4.2.9. Solution Screen

Gambar 4.9 merupakan tampilan yang menampilkan algoritma untuk menyelesaikan rubik acak pengguna. pada bagian atas terdapat pilihan tahap penyelesaian, di bawahnya merupakan algoritma penyelesaian pada tahap yang dipilih, di bawahnya pedoman cara melakukan algoritmanya, dan paling bawah

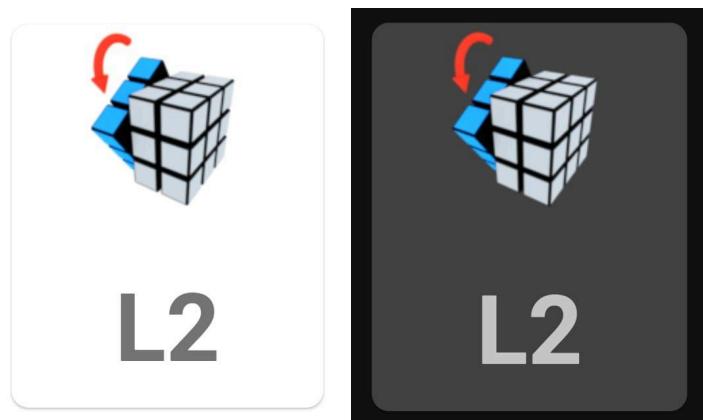
terdapat tiga tombol yang berfungsi sebagai kembali (sebelumnya), otomatis, dan selanjutnya. Dan dapat ditampilkan dalam *light mode* atau *dark mode*.



Gambar 4.9 Solution Screen

4.2.10. Item Solution Screen

Gambar 4.10 merupakan tampilan item dari setiap konten langkah penyelesaian rubik. Dan dapat ditampilkan dalam *light mode* atau *dark mode*.



Gambar 4.10 Item Solution Screen

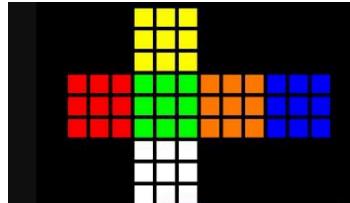
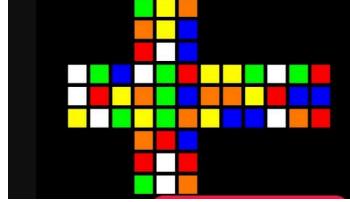
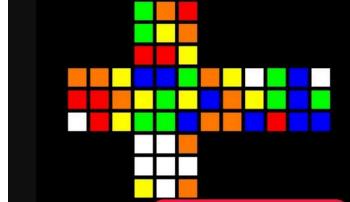
4.3. Pengujian Sistem

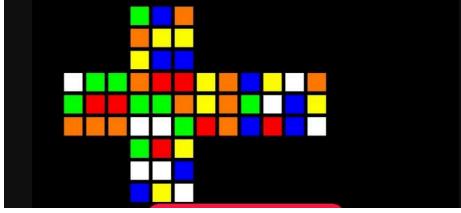
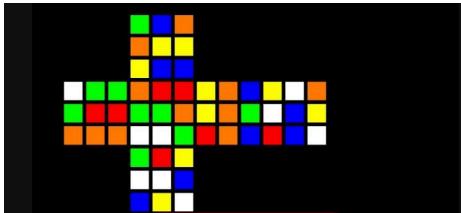
Pengujian sistem adalah proses mengevaluasi apakah sistem yang dibuat memenuhi kebutuhan, spesifikasi, dan persyaratan yang telah ditentukan.

4.3.1. Pengujian deteksi warna rubik

Pengujian deteksi warna rubik bertujuan untuk mencari keakuratan sistem dalam memindai warna rubik.

Tabel 4.1 Pengujian Deteksi Warna Rubik

No.	Hasil Deteksi	Keadaan Rubik	Kecocokan
1			54/54
2			54/54
3			54/54

4			52/54
5			54/54

4.3.2. Pengujian algoritma penyelesaian

Pengujian algoritma penyelesaian bertujuan untuk mencari tahu apakah algoritma yang dihasilkan oleh sistem berhasil untuk menyelesaikan rubik acak pengguna.

Tabel 4.2 Pengujian Algoritma Penyelesaian Rubik

No.	Tahap	Algoritma	Hasil Penerapan Algoritma	Keberhasilan
1	Mulai	-		Berhasil

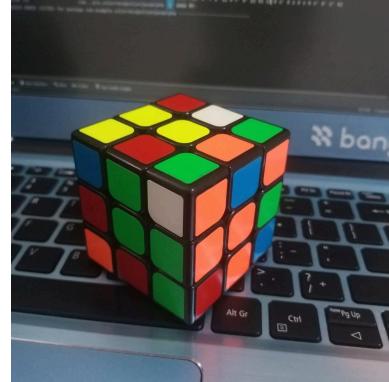
	Sunflower F' U' R L B F2		Berhasil
	Cross U2 L2 F2 B2 R2 U		Berhasil
	1st Layer y' R U R' y' R U R' U' R U R' U' R U R' U2 y2 U R U'R' U' y R U R' U' R U R' U' R U R' U'		Berhasil
	2nd Layer U2 y2 U' L' U L U F U' F' U2 y2 U' L' U L U F U' F' R U R' U2 R U2 R' U F' U' F		Berhasil

	Yellow Cross	U F U R U' R' F'		Berhasil
	OLL	R U R' U R U2 U R' U R U2 R'		Berhasil
	PLL	U R2 U R U R' U' R' U' R' U R' U2		Berhasil
2	Mulai	-		Berhasil

	Sunflower	$U F2 B' U' L F U' R$ $U B' U' L$		Berhasil
	Cross	$B2 R2 U L2 U2 F2 U$		Berhasil
	1st Layer	$U y2 U R U' R' U2$ $y2 R U R' U' R U R'$ $U' R U R' U' y R U$ $R' U y2 U R U' R'$		Berhasil
	2nd Layer	$R U' R' U' F' U F U$ $L' U L U F U' F'$		Berhasil
	Yellow Cross	-		Berhasil

	OLL	R U R' U R U2 R' U R U R' U R U2 R' U2 R U R' U R U2 R'		Berhasil
	PLL	U' R' F R' B2 R F' R' B2 R2 U2 R U' R U R U R U' R' U' R2 U2		Berhasil
3	Mulai	-		Berhasil
	Sunflower	U L U' R L U' F L U' F		Berhasil
	Cross	U L2 F2 B2 R2 U		Berhasil

	1st Layer	$\begin{aligned} & U2 R U R' U' y \quad R U \\ & R' U2 y U R U' R' U' \\ & y R U R' U y2 R U R' \\ & U' y' \end{aligned}$		Berhasil
	2nd Layer	$\begin{aligned} & U y' U' L' U L U F U' \\ & F' y' U' L' U L U F U' \\ & F' U' y U R U' R' U' F' \\ & U F U y2 U' L' U L U \\ & F U' F' \end{aligned}$		Berhasil
	Yellow Cross	$U' F U R U' R' F'$		Berhasil
	OLL	$\begin{aligned} & R U R' U R U2 R' U \\ & R U R' U R U2 R' U2 \\ & R U R' U R U2 R' \end{aligned}$		Berhasil
	PLL	$\begin{aligned} & U R' F R' B2 R F' R' \\ & B2 R2 U2 R U' R U R \\ & U R U' R' U' R2 U2 \end{aligned}$		Berhasil

4	Mulai	-		Berhasil
	Sunflower	L2 U L' U' R'		Berhasil
	Cross	L2 U F2 R2 U B2 U		Berhasil
	1st Layer	R U R' y R U R' U y2 R U R' U' R U R' U' R U R' U' y' R U R' U' R U R' U' y		Berhasil

	2nd Layer	y2 U' L' U L U F U' F' U y U R U' R' U' F' U F U' y U R U' R' U' F' U F y2 U R U' R' U' F' U F		Berhasil
	Yellow Cross	F R U R' U' F'		Berhasil
	OLL	U R U R' U R U2 R' U R U R' U R U2 R' U2 R U R' U R U2 R'		Berhasil
	PLL	R' F R' B2 R F' R' B2 R2 U2 R U' R U R U R U' R' U' R2		Berhasil

5	Mulai	-		Berhasil
	Sunflower	R2 B' U' L U L2 U2 B' U' L		Berhasil
	Cross	U2 L2 F2 B2 R2 U		Berhasil
	1st Layer	U2 y U R U' R' U2 y2 U R U' R' U2 y U R U' R'		Berhasil

	2nd Layer U y' U' L' U L U F U' F' U R U' R' U' F' U F y2 U' L' U L U F U' F' U y U' L' U L U F U' F'		Berhasil
	Yellow Cross F U R U' R' F'		Berhasil
	OLL U R U R' U R U2 R' R U R' U R U2 R'		Berhasil
	PLL R' F R' B2 R F' R' B2 R2 U R' F R' B2 R F' R' B2 R2 U' R2 U R U R' U' R' U' R' U R' U		Berhasil

4.3.3. Pengujian oleh pengguna

Dilakukan pengujian aplikasi kepada pengguna yang tidak bisa bermain rubik, dan berikut hasil pengujinya.

Tabel 4.3 Pengujian Oleh Pengguna

Tahap	Algoritma	Hasil Penerapan Algoritma	Keberhasilan
Mulai	-		-
Sunflower	R2 F U B' R U' B		Berhasil
Cross	F2 R2 U B2 U2 L2 U		Berhasil
1st Layers	U y2 U R U' R' U2 y2 U R U' R' y R U R' U y2 R U R' U' R U R' U' R U R' U y2 R U R' U' R U R' U' R U R' U' y'		Berhasil

2nd Layers	$U' y' U' L' U L U F U' F' U' y'$ $U' L' U L U F U' F' y2 U' L' U$ $L U F U' F' R U R' U2 R U2 R'$ $U F' U' F$		Berhasil
Yellow Cross	-	-	Berhasil
OLL	$U' R U R' U R U2 U R' U R U2$ $R' U2 R U R' U R U2 R'$		Berhasil
PLL	$U R' F R' B2 R F' R' B2 R2 U'$ $R U' R U R U R U' R' U' R2 U2$		Berhasil

Pengguna berhasil menyusun rubik mengikuti langkah-langkah yang diberikan oleh aplikasi, berikut komentar dari penguji “Aplikasi intuitif dan tampilannya menarik, materi yang diberikan cukup mudah untuk dipahami. Fitur penyelesain rubik menggunakan kamera cukup bagus dan mudah diikuti.”

BAB 5

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari penelitian dan pengembangan yang dilakukan terhadap aplikasi belajar rubik 3x3x3 dengan memanfaatkan *computer vision*, kesimpulan yang didapatkan adalah:

1. Dari tujuh kali percobaan yang dilakukan penyusunan rubik dengan metode *Layer by layer* pada aplikasi, dibuktikan memiliki 100% tingkat keberhasilan dan mampu menyelesaikan semua kondisi rubik acak pengguna dengan memasukkan kondisi rubik pengguna kedalam aplikasi secara benar.
2. Penerapan *Computer Vision* pada aplikasi memiliki tingkat akurasi 99% dalam menganalisis warna setiap sisi rubik pada kondisi cahaya yang cukup, sehingga hal ini memudahkan proses penginputan kondisi rubik kedalam aplikasi.
3. Menjadi salah satu cara belajar menyelesaikan rubik yang praktis dan bisa digunakan kapan saja dan dimana saja.

5.2. Saran

Beberapa saran untuk pengembangan dan penelitian lebih lanjut:

1. Meningkatkan akurasi analisis warna rubik perlu dilakukan, terutama dalam kondisi cahaya yang rendah.
2. Meningkatkan konten pembelajaran seperti penambahan video atau animasi untuk meningkatkan pengalaman dan pemahaman pengguna selama menggunakan aplikasi ini.
3. Menambahkan fitur suara untuk memandu pengguna mengikuti langkah yang dihasilkan aplikasi.
4. Memperluas jenis variasi warna rubik untuk dideteksi, diluar warna “standar” rubik.

DAFTAR PUSTAKA

- Joyner, D. (2008). *Adventures in Group Theory: Rubik's Cube, Merlin's Machine, and Other Mathematical Toys* (2nd ed.). Johns Hopkins University Press.
- Liu, S., Jiang, D., Feng, L., Wang, F., Feng, Z., Liu, X., Guo, S., & Li, B. (2019, 1 11). Color Recognition for Rubik's Cube Robot. <https://doi.org/10.48550/arXiv.1901.03470>
- Lyu, Z., Liu, Z., Khojandi, A., & Yu, A. J. (2022, 9). Q-learning and traditional methods on solving the pocket Rubik's cube. *Computers & Industrial Engineering*, 171, 108452. <https://doi.org/10.1016/j.cie.2022.108452>
- Ponce, J., & Forsyth, D. (2012). *Computer Vision: A Modern Approach*. Pearson.
- Rismayani, Hidayah, T. N., Hasanah, U., & Indriani, L. (2020). Pengaruh Permainan Rubik Terhadap Perkembangan Otak Manusia. 2, 485 - 490.
- Rohith, S. P., Mohammed, A. S., Jayasankar, S., & Harikrishnan, M. (2019, 5 3). Autonomous Rubik's Cube Solver Bot. *International Journal of Scientific Research and Engineering Development*, 2(3), 146 - 151.
- Shepard, R. N., & Metzler, J. (1971). Mental rotation of three-dimensional objects. *Science*, 171(3972), 701-703. <https://doi.org/10.1037/a0016127>
- Solem, J. E. (2012). *Programming Computer Vision with Python: Tools and Algorithms for Analyzing Images*. O'Reilly Media.
- Spector, M. J., van Merriënboer, J., Merrill, M. D., & Driscoll, M. P. (Eds.). (2007). *Handbook of Research on Educational Communications and Technology: A Project of the Association for Educational Communications and Technology*. Taylor & Francis.
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
- Tamim, R. M., Borokhovski, E., Abrami, P. C., & Schmid, R. F. (2011). What forty years of research says about the impact of technology on learning: A second-order meta-analysis and validation study. *Review of educational research*, 81(1), 4-28. [10.3102/0034654310393361](https://doi.org/10.3102/0034654310393361)
- Wai, J., Lubinski, D., & Benbow, C. P. (2009). Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifies its

importance. *Journal of Educational Psychology*, 101(4), 817-835.
<https://doi.org/10.1037/a0016127>