BAB III

ANALISIS SISTEM

Bab ini berisi penjelasan analisis proses pengolahan, analisis permasalahan, pihak-pihak yang terlibat dalam proyek, analisis *requirements*, dan analisis algoritma. Analisis *requirement* mencakup *functional requirements* dan *non-functional requirements.* Kebutuhan dibuatnya sistem *"Scanner Project"* diketahui setelah dilakukan perbincangan bersama Bapak Dadan Hardianto, selaku salah satu dosen Fakultas Ilmu Komputer Universitas Indonesia. Dari perbincangan ini, diketahui bahwa Fakultas Ilmu Komputer UI mempunyai bagian/divisi yang menangani pengelolaan Lembar Isian Komputer (LIK) SPMB.

Analisis alir proses pengolahan dilakukan pada awal proyek mahasiswa untuk memberi gambaran seberapa besar ruang lingkup sistem yang dikembangkan. Analisis ini digunakan untuk menentukan tujuan pengembangan sistem dan berbagai fungsi yang dapat ditawarkan maupun dikembangkan dalam sistem ini.

Analisis dilakukan berdasarkan *requirements gathering* yang merupakan hasil wawancara dengan Bapak Dadan Hardianto. Berdasarkan hasil wawancara tersebut diperoleh gambaran alir proses pengolahan. Dari gambaran rincian alir proses pengolahan yang didapat, *System Analyst* merumuskan *functional requirements* dan *non-functional requirements* dari sistem. Berdasarkan *requirements* inilah kerangka fungsional sistem *"Scanner Project"* akan dibangun.

3.1 Analisis Alir Proses Pengolahan

Dalam pengelolaan Lembar Isian Komputer (LIK), dilakukan beberapa tahapan. Awalnya LIK yang telah diisi/dihitamkan oleh peserta SPMB dikumpulkan dan disusun oleh masing-masing panitia SPMB berdasarkan jenis program yang dipilih (program IPA, IPS atau IPC). Setelah formulir-formulir LIK tersebut terkumpul berdasarkan jenis program masing-masing, tiap formulir LIK dimasukkan satu persatu ke alat pemindai khusus *Optical Mark Recognition*

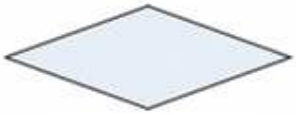
(OMR) yang dapat memindai dan mengekstrak informasi sekaligus dari LIK yang dihitamkan/diisi.

Setiap LIK dipindai dengan OMR, kemudian hasil pemindaian akan tampil pada monitor beserta hasil deteksi tempat lingkaran-lingkaran pada formulir LIK yang telah diisi/dihitamkan. Hasil deteksi tersebut merupakan jawaban dari setiap pertanyaan di SPMB, UMB, dan SNMPTN ataupun informasi-informasi yang berhubungan dengan peserta SPMB, UMB, dan SNMPTN. Adapun alir proses pengolahan LIK dengan OMR akan dijelaskan dengan mennggunakan Diagram Alir (*Flowchart*).

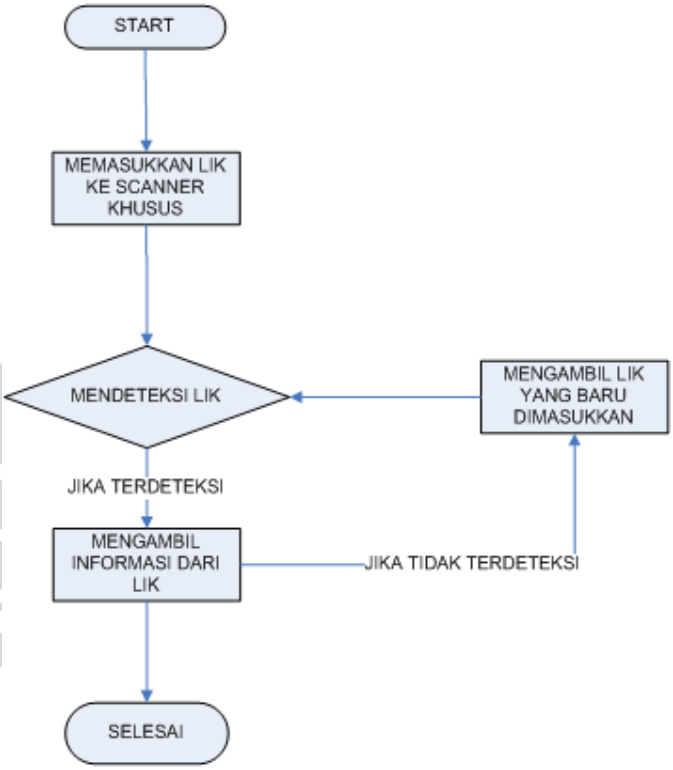
Diagram alir (*Flowchart*) adalah penyajian yang sistematis tentang proses dan logika dari kegiatan penanganan informasi [11].

Untuk menggambarkan Diagram Alir, tim pengembang menggunakan aplikasi Microsoft Office Visio 2003 yang menyediakan notasi-notasi standar pembuatan Diagram Alir. Tabel berikut berisi notasi Diagram Alir beserta penjelasan singkat dari notasi-notasi yang digunakan.

Tabel 3. 1 Notasi Diagram Alir

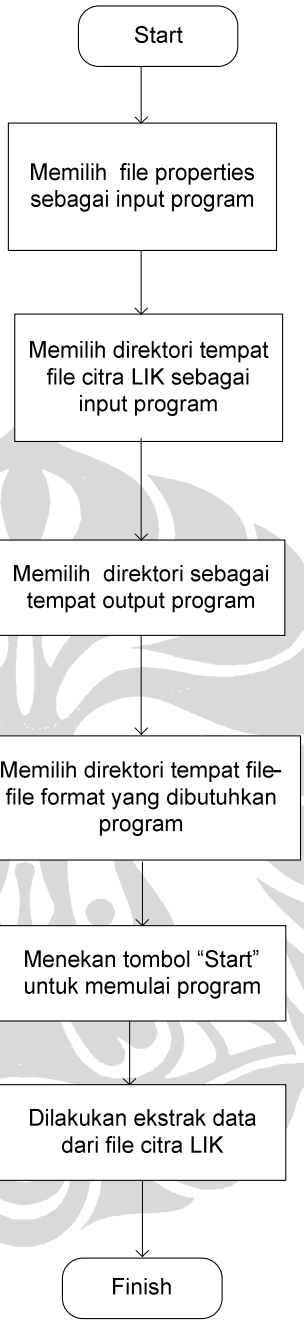
Notasi	Nama	Deskripsi
	*Terminator*	Lambang dimulai dan berakhirnya suatu aliran proses.
	*Process*	Lambang yang menunjukkan terjadinya suatu proses
	*Decision*	Lambang yang menyatakan perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	*Document*	Dokumen atau berkas yang diakses.

Berikut ini adalah Diagram Alir yang menggambarkan proses pengolahan LIK dengan OMR.



Gambar 3. 1 Diagram alir proses pengolahan LIK dengan OMR

Pada sistem "*Scanner Project*" yang kami kembangkan, terdapat beberapa perubahan dalam alir proses pengolahannya. Pertama, LIK dipindai terlebih dahulu dengan alat pemindai standar sehingga didapatkan format citra digital LIK. Lalu citra digital LIK dijadikan *input* pada sistem "*Scanner Project*" untuk dilakukan ekstraksi. Sedangkan pada OMR, pengekstrakan dilakukan bersamaan dengan pemindaian. Adapun alir proses pengolahan dari sistem "*Scanner Project*" digambarkan pada diagram di bawah ini:



Gambar 3. 2 Diagram Alir proses pengolahan "*Scanner Project*"

Diagram di atas menggambarkan tahapan-tahapan yang harus dilakukan sebelum mengekstrak citra LIK. Sebelum mulai mengekstrak LIK, pengguna harus memilih berkas properties yang diperlukan dalam pemrosesan LIK, memilih direktori tempat disimpannya citra-citra LIK yang akan diekstrak, memilih direktori tempat output program, dan memilih direktori tempat disimpannya

berkas-berkas lain yang diperlukan dalam mengekstrak LIK. Setelah itu barulah pengguna dapat menjalankan program.

3.2 *Stakeholders*

Stakeholders dari pengembangan sistem *"Scanner Project"* ini adalah:

1. Bapak Dadan Hardianto, selaku pembimbing proyek mahasiswa ini. Beliau juga adalah koordinator untuk pemeriksaan formulir LIK SPMB.
2. Pihak pengelola pemeriksaan formulir LIK SPMB, baik para *data entry* untuk SPMB ataupun bagian IT Fakultas Ilmu Komputer Universitas Indonesia.

3.3 Analisis Permasalahan

Selama ini pemeriksaan LIK di Fakultas Ilmu Komputer Universitas Indonesia menggunakan alat pemindai berteknologi OMR (*Optical Mark Reader*). OMR dapat melakukan pemeriksaan LIK dengan cepat. Namun OMR juga memiliki beberapa kekurangan, yaitu:

a. Diperlukan biaya yang mahal untuk pencetakan LIK dan pembelian alat pemindai OMR sehingga hanya pihak tertentu saja yang dapat menggunakannya.
b. Diperlukan kertas dengan ketebalan tertentu dalam pencetakan LIK.
c. Diperlukan alat tulis khusus yang digunakan untuk pengisian LIK.
d. Walaupun dibutuhkan, sistem dengan teknologi OMR tidak tepat untuk diterapkan pada institusi berskala kecil
e. Posisi formulir LIK harus tepat dan tidak boleh terbalik, karena jika posisi LIK kurang tepat atau terbalik maka OMR tidak akan bisa melakukan pemeriksaan pada LIK.
f. Pada teknologi OMR, ekstraksi data dilakukan bersamaan dengan pemindaian. Pemindaian tersebut dilakukan dengan bantuan penanda tambahan pada tepi LIK (*timing track*). Apabila *timing track* tidak tercetak, formulir LIK rusak, terlipat ataupun kotor, maka formulir LIK tidak dapat terdeteksi. Hal ini pernah terjadi pada saat pemeriksaan SPMB tahun 2005, dimana *timing track* tidak tercetak dengan sempurna,

akibatnya puluhan ribu lembar LIK tidak dapat dideteksi, dan untuk menanggulangi masalah ini maka operator terpaksa menyempurnakan *timing track* dengan cara manual. *Timing track* yang rusak juga bisa terjadi karena kelengahan operator (misalnya formulir LIK terkena tumpahan minuman), sehingga LIK tidak dapat diperiksa lagi.

g. Apabila kertas LIK yang masuk ke dalam alat pemindai OMR menempel, maka akan terjadi kesalahan deteksi, akibatnya data yang didapat tidak valid, sehingga perlu dilakukan pengulangan pemindaian.

3.4 Analisis Kebutuhan

Proses analisis kebutuhan dilakukan oleh tim pengembang berdasarkan analisis alir proses pengolahan dan permasalahan yang telah dipaparkan sebelumnya. *"Scanner Project"* yang dikembangkan diharapkan dapat mengatasi permasalahan-permasalahan yang ada. Berdasarkan hasil analisis yang dilakukan, kebutuhan sistem secara garis besar terbagi menjadi dua, yaitu kebutuhan fungsional (*functional requirements*) dan kebutuhan nonfungsional (*non-functional requirements*).

3.4.1 Kebutuhan Fungsional

"Scanner Project" dibuat dengan tujuan memfasilitasi proses pemeriksaan LIK sehingga didapatkan informasi dari LIK tersebut. Sistem ini diharapkan dapat membantu semua aktivitas dari pihak-pihak yang terkait dalam proses pengolahan LIK. Kebutuhan fungsionalitas sistem ini disimpulkan dari hasil wawancara kami dengan *stakeholder*, serta informasi-informasi yang diperoleh dari sumber lainnya. Kebutuhan fungsional dalam sistem ini adalah sebagai berikut:

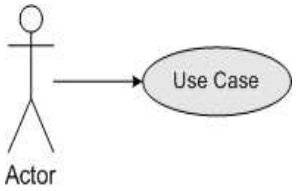
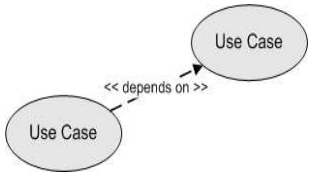
1. Ekstraksi berkas citra digital LIK (Formulir *Batch Control Sheet*, formulir Pendaftaran, dan formulir Jawaban)
2. Membuat desain format kerangka LIK
3. Menyimpan hasil desain format kerangka LIK

Untuk memaparkan kebutuhan fungsional dari sistem secara jelas dan sistematis, tim pengembang menggunakan *Use-Case Diagram*. *Use-Case*

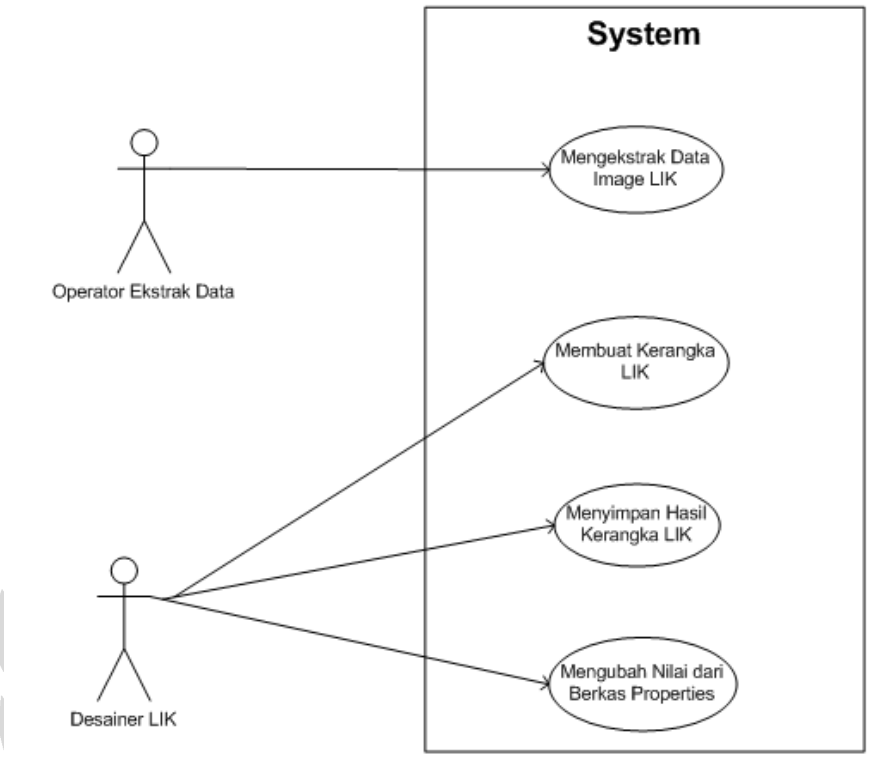
Diagram adalah sebuah diagram yang menggambarkan interaksi dan keterhubungan antara sistem yang akan dibuat dengan sistem-sistem eksternal lainnya, termasuk *user* yang akan menggunakan sistem tersebut [WH04]. Sementara *Use-Case* adalah sebuah skenario atau peristiwa bisnis dimana sistem harus memberikan suatu respon yang ditentukan. *Use-Case* mencakup analisis berorientasi objek. Namun, penggunaannya menjadi hal umum dalam banyak metodologi untuk analisis dan perancangan sistem [WHI06].

Untuk menggambarkan *Use-Case Diagram*, tim pengembang menggunakan aplikasi Microsoft Office Visio 2003 yang menyediakan notasi-notasi standar pembuatan *Use-Case*. Tabel berikut berisi notasi *Use-Case Diagram* beserta penjelasan singkat dari notasi-notasi yang digunakan.

Tabel 3. 2 Notasi *Use-Case Diagram*

Notasi	Nama	Deskripsi
	*Use-Case*	Lambang dari suatu fungsi pada sistem. *Use-Case* terdiri dari skenario yang dapat dilakukan oleh pengguna terhadap sistem untuk mencapai tujuan dari suatu alir proses pengolahan.
	*Actor*	Pengguna yang menginisiasi terjadinya skenario pada *Use-Case* untuk menjalankan suatu alir proses pengolahan pada sistem.
	*Association*	Notasi yang melambangkan suatu bentuk interaksi dan relasi antara *Actor* dan *Use-Case*, dimana *Actor* menggunakan atau memicu berjalannya suatu fungsi di dalam *Use-Case* tersebut.
	*Dependency Relation*	Notasi yang melambangkan relasi ketergantungan antar *Use-Case*, dimana *Use-Case* yang menunjuk, bergantung pada *Use-Case* yang ditunjuk.

Kebutuhan fungsional dari sistem "*Scanner Project*" pada Lembar SPMB ini digambarkan oleh *Use-Case Diagram* pada gambar berikut:



Gambar 3. 3 *Use-Case Diagram*

Actor yang berperan dalam sistem ini ada dua yaitu Desainer LIK dan Operator Ekstrak Data. Desainer LIK memiliki peranan untuk membuat format kerangka LIK, menyimpannya dan mengubah nilai dari berkas *properties*. Tujuannya adalah untuk membuat berkas format yang digunakan oleh program untuk mengekstrak data. Desainer LIK dapat membuat format kerangka LIK yang baru. Setelah membuat format kerangka LIK yang baru secara interaktif dengan tampilan GUI, sistem akan otomatis men-*generate* desain tersebut menjadi berkas berekstensi txt yang akan digunakan sebagai format kerangka dalam mengekstrak data dari formulir LIK.

Desainer LIK juga dapat mengubah nilai dari berkas *properties*, seperti tipe formulir LIK yang akan di ekstrak datanya (Jawaban atau Pendaftaran), tingkat kehitaman, mengubah nama direktori tempat penyimpanan berkas-berkas *input* yang digunakan dalam proses untuk mengekstrak data, mengubah posisi *skunkmark* dan nilai-nilai lainnya yang semuanya terdapat di dalam berkas *properties*.

Operator Ekstrak Data berperan untuk mengekstrak data dari formulir LIK yang telah dipindai menjadi berkas citra dengan format BMP *black & white* 1 bit. Operator Ekstrak Data dapat mengekstrak data apabila format kerangka formulir LIK yang ada dan digunakan oleh sistem sesuai dengan formulir LIK yang akan diekstrak.

Untuk mendapatkan *input* formulir LIK yang berupa berkas citra dengan format BMP *black & white* 1 bit, terlebih dahulu formulir LIK dipindai dengan alat pemindai standar. Tugas ini dilaksanakan oleh Operator Pemindai untuk memindai data berupa formulir LIK sehingga menjadi citra digital dengan format BMP *black & white* 1 bit. Operator pemindai memiliki peranan yang cukup penting yaitu memindai formulir LIK yang nantinya akan menjadi *input* pada sistem "*Scanner Project*", namun operator pemindai tidak berinteraksi langsung dengan sistem.

3.4.2 Kebutuhan Non-Fungsional

Selain kebutuhan fungsional, terdapat aspek lain yaitu kebutuhan non-fungsional. Kebutuhan non-fungsional yang diharapkan ada pada sistem antara lain:

1. Tampilan antar muka yang baik serta tidak menyulitkan *user* dalam menggunakan sistem. Oleh karena itu, dalam perancangan GUI tersebut, perlu diperhatikan prinsip konsistensi dari segi penggunaan jenis dan ukuran huruf, serta posisi link untuk navigasi yang tidak berubah-ubah.
2. Waktu kerja dari sistem yang relatif cepat. Waktu kerja sistem "*Scanner Project*" diharapkan relatif jauh lebih cepat dibandingkan

dengan waktu kerja pengekstrakan LIK dengan menggunakan OMR yaitu 2 detik per lembar.

3.5 Analisis Algoritma

Dalam pengembangan sistem "*Scanner Project*" ini, kami menganalisa dan membandingkan beberapa algoritma untuk mendapatkan algoritma yang paling efektif dan efisien untuk diimplementasikan pada sistem "*Scanner Project*". Algoritma yang kami bandingkan ini sudah kami jelaskan sebelumnya pada subbab 2.6 Tinjauan Pustaka.

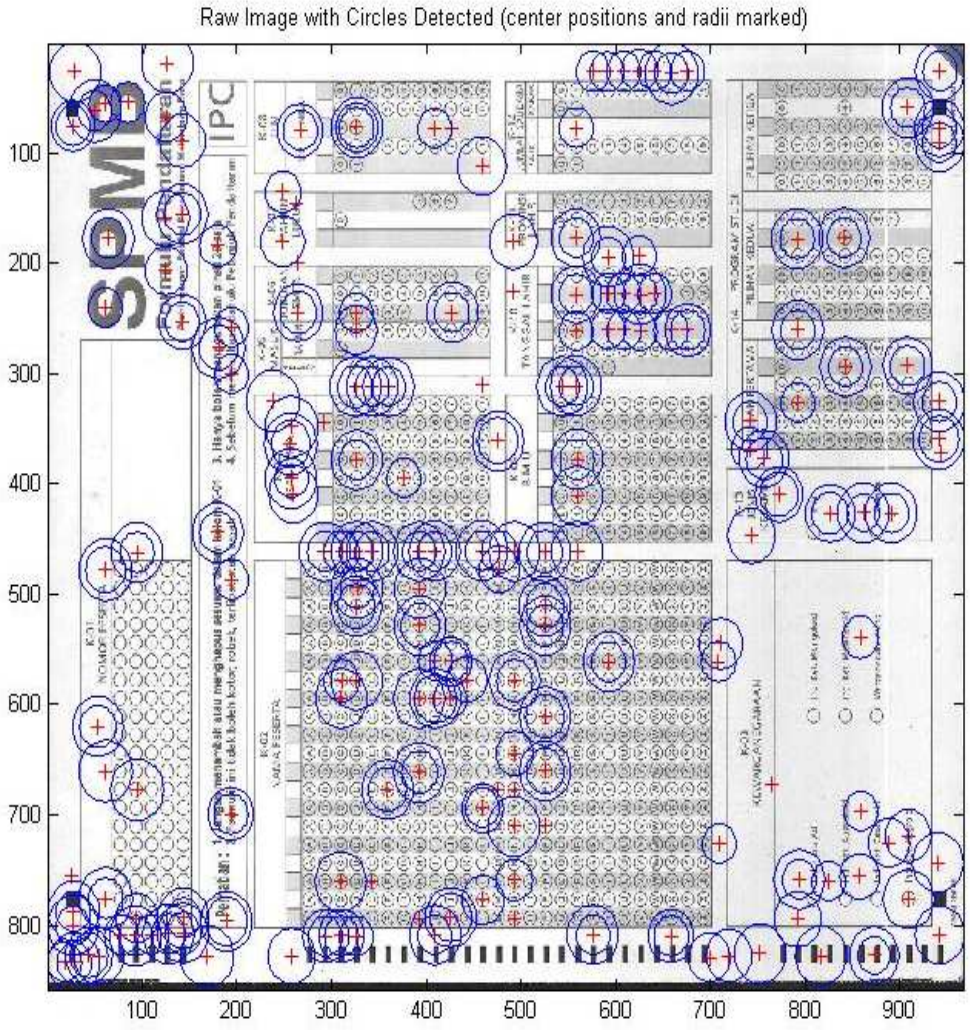
Kami melakukan analisa terhadap 6 algoritma, yaitu:

- Algoritma *Circle Hough Transform*
- Algoritma Titik Tengah
- Algoritma *Circle Detection*
- Algoritma Pusat Massa dan Titik Berat
- Algoritma Deteksi *Timing Track* dan *Skunk Mark*
- Algoritma Deteksi *Skunk Mark*

Adapun analisa yang kami lakukan yaitu dengan cara menguji coba enam algoritma di atas untuk mendeteksi lingkaran pada LIK SPMB, dan dari uji coba algoritma yang kami lakukan, didapatkan hasil sebagai berikut:

3.5.1 Algoritma *Circle Hough Transform*

Setelah dilakukan uji coba algoritma *Circle Hough Transform* terhadap LIK SPMB, didapatkan hasil pendeteksian LIK SPMB seperti terlihat pada gambar di bawah ini:



Gambar 3. 4 Hasil Pendeteksian Lingkaran dengan Algoritma *Circle Hough Transform* pada LIK

Dari gambar di atas terlihat bahwa algoritma ini tidak dapat mendeteksi lingkaran pada LIK SPMB dengan baik, karena gambar-gambar yang bentuknya melengkung dan hampir menyerupai lingkaran juga dideteksi sebagai lingkaran oleh algoritma ini (seperti terlihat pada huruf S, P, B pada

kata SPMB pada gambar di atas). Hal ini menunjukkan bahwa algoritma ini tidak efektif untuk digunakan pada sistem "*Scanner Project*" ini.

3.5.2 Algoritma Titik Tengah

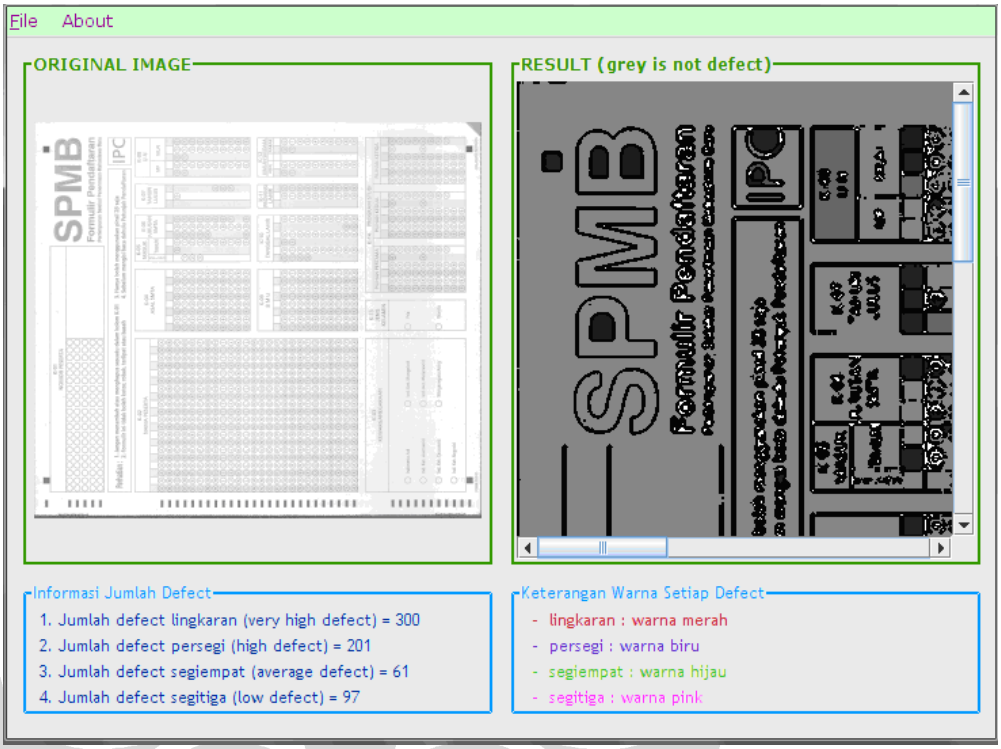
Algoritma ini tidak cocok untuk diterapkan pada sistem "*Scanner Project*" karena waktu yang dibutuhkan untuk mendeteksi lingkaran pada LIK SPMB sangat lama. Hal itu kami ketahui setelah kami uji coba algoritma Titik Tengah ini untuk mendeteksi lingkaran pada LIK SPMB, untuk mendapatkan hasil dari satu buah LIK dibutuhkan waktu 15 menit 52 detik. Dari hal tersebut kami simpulkan akan dibutuhkan waktu yang sangat lama untuk mendeteksi banyak lembar LIK. Dengan demikian, kami menyimpulkan bahwa algoritma ini tidak efektif untuk diterapkan pada sistem "*Scanner Project*".

3.5.3 Algoritma *Circle Detection*

Sama seperti algoritma Titik Tengah, algoritma *Circle Detection* ini tidak cocok untuk diterapkan pada sistem "*Scanner Project*" karena waktu yang dibutuhkan untuk mendeteksi lingkaran pada LIK SPMB sangat lama. Dibutuhkan waktu lebih dari 15 menit untuk mendeteksi lingkaran pada satu lembar LIK SPMB. Dengan demikian, kami menyimpulkan bahwa algoritma ini tidak efektif untuk diterapkan pada sistem "*Scanner Project*".

3.5.4 Algoritma Pusat Massa dan Titik Berat

Ketika algoritma ini kami implementasikan untuk mendeteksi lingkaran pada sebuah citra digital LIK, terjadinya kesalahan dalam pendeteksian jumlah lingkaran pada citra LIK tersebut. Hal ini dapat dilihat pada gambar berikut:



Gambar 3. 5 Hasil pendeteksian Lingkaran pada LIK dengan Algoritma Pusat Massa dan Titik Berat

Pada gambar di atas terlihat dalam kolom Informasi Jumlah *Defect* (pada percobaan ini lingkaran, persegi, segiempat, segitiga dianggap sebagai *defect*) diketahui bahwa jumlah lingkaran yang ada pada formulir SPMB berjumlah 300 buah, padahal dalam 1 formulir SPMB tersebut terdapat 838 lingkaran. Dengan demikian, kami menyimpulkan bahwa algoritma ini tidak efektif untuk diterapkan pada sistem "*Scanner Project*".

3.5.5 Algoritma Deteksi *Timing Track* dan *Skunk Mark*

Algoritma ini merupakan algoritma yang dibuat khusus untuk mendeteksi lingkaran pada formulir LIK yang memiliki *timing track* dan *skunk mark* seperti pada formulir LIK SPMB, sehingga algoritma ini cukup efektif digunakan untuk mengekstrak data pada formulir SPMB. Namun algoritma ini masih memiliki ketergantungan kepada *timing track*, sementara itu salah satu tujuan dari proyek mahasiswa ini adalah menghilangkan ketergantungan

terhadap *timing track*, sehingga kami mengembangkan lagi algoritma ini sehingga tidak lagi bergantung pada *timing track.*

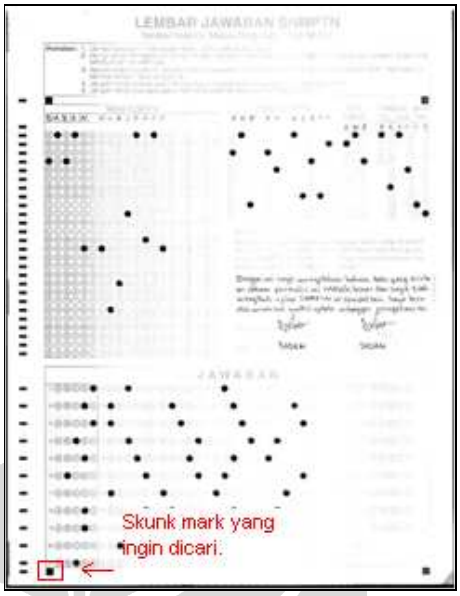
3.5.6 Algoritma Deteksi *Skunk Mark*

Algoritma ini adalah algoritma yang khusus dibuat untuk mendeteksi lingkaran pada lembar LIK yang memiliki *skunk mark*.

Prinsip kerja algoritma ini mirip dengan algoritma deteksi *timing track* dan *skunk mark*, yaitu dengan membuat matriks (untuk lebih jelasnya lihat subbab 2.6.6). Tetapi pada algoritma ini *timing track* tidak digunakan lagi sebagai referensi untuk membuat matriks, hanya *skunk mark* yang dideteksi dan digunakan sebagai referensi pembuatan matriks.

Berikut ini adalah tahapan-tahapan dalam implementasi algoruitma ini untuk mengekstrak LIK. LIK yang diperiksa terdiri dari formulir *Batch Control Sheet* (BCS), Jawaban (JWB), dan Pendaftaran (PDF). Sebuah formulir BCS yang bersesuaian akan selalu diekstrak pertama kali sebelum memeriksa formulir-formulir JWB ataupun PDF yang bersesuaian dengan BCS tersebut. Langkah-langkah implementasi algoritma ini untuk mengekstrak tiap-tiap formulir tersebut adalah sama. Berikut adalah penjelasan tahap-tahap algoritma "Deteksi *Skunk Mark*".

1. **Deteksi *skunk mark* pada pojok kiri bawah LIK**
   - Langkah awal pada algoritma ini adalah mendeteksi *skunk mark* yang berada pada pojok kiri bawah LIK. Agar lebih jelas, dapat dilihat pada gambar di bawah ini.

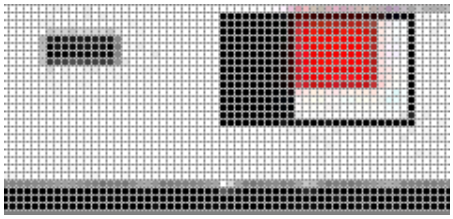


Gambar 3. 6 *Skunk mark* pada pojok kiri bawah LIK

- Untuk mendeteksi *skunk mark*, pertama-tama dibuat batasan area pencarian *skunk mark* LIK.

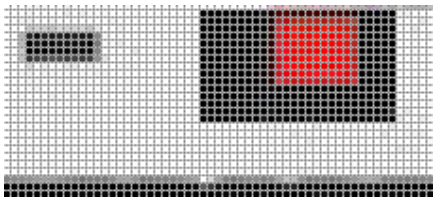
Gambar 3. 7 Area pencarian *skunk mark*

- Lalu dibuat *vector* untuk menampung hasil pencarian. Kemudian dilakukan *looping* seluas area batasan yang dibuat. *Looping* dilakukan untuk menemukan piksel berwarna hitam. Jika ditemukan piksel berwarna hitam (pada ilustrasi gambar di bawah, kotak diberi warna merah), maka dilakukan pemeriksaan untuk mengidentifikasi kotak hitam *skunk mark.*



Gambar 3. 8 *Looping* hingga *skunk mark* ditemukan

- *Looping* dilanjutkan hingga area pencarian *skunk mark* selesai ditelusuri.



Gambar 3. 9 *Looping* hingga selesai

- Jika ukuran kotak hitam tersebut sesuai, maka kotak tersebut dianggap sebagai *skunk mark.* Kemudian hasil deteksi tanda baca ditaruh dalam *vector*.

2. Deteksi *skunk mark* pada pojok kanan bawah LIK

- Setelah kotak hitam *skunk mark* pada pojok kiri bawah telah ditemukan, maka selanjutnya dilakukan pencarian kotak *skunk mark* pada pojok kanan bawah LIK.

Gambar 3. 10 *Skunk mark* pada pojok kanan bawah LIK

- Pertama-tama pencarian dilakukan dengan menentukan batasan area pencarian *skunk mark.* Area tersebut adalah area piksel yang dianggap memiliki kemungkinan terdapat *skunk mark.* Setelah itu, langkah-langkah selanjutnya sama seperti pada pencarian *skunk mark* pada pojok kiri bawah yang telah dijelaskan di atas.

3. Pemeriksaan kemiringan LIK

- Setelah kedua kotak hitam *skunk mark* ditemukan, kemudian dilakukan pemeriksaan apakah LIK tersebut miring atau tidak.

- Pemeriksaan kemiringan dilakukan dengan membandingkan posisi piksel pada titik pojok kiri atas kotak *skunk mark* antara *skunk mark*

pada pojok kiri bawah LIK dengan *skunk mark* pada pojok kanan LIK.

Gambar 3. 11 Pemeriksaan titik pojok kiri atas masing-masing *skunk mark*

- Jika miring, maka LIK tersebut dirotasi agar menjadi lurus kembali. Batas kemiringan LIK yang ditoleransi maksimal sebesar 5°. Untuk kemiringan lebih dari 5°, biasanya kotak *skunk mark* sudah tidak sempurna lagi (terpotong), sehingga tidak memungkinkan untuk diekstrak dan harus dipindai ulang.

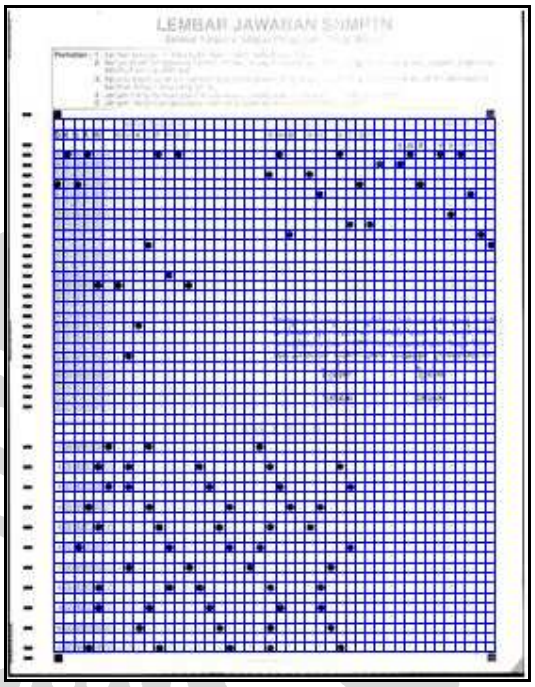
4. Pembuatan matriks *boolean*

- Setelah dilakukan pemeriksaan kemiringan LIK, kemudian dibuat matriks *boolean*. Matriks ini mengacu pada *skunk mark* yang berada pada pojok kiri bawah LIK. *Skunk mark* yang berada pada pojok kanan bawah LIK hanya digunakan dalam tahap pemeriksaan kemiringan LIK.

- Selain mengacu pada *skunk mark* yang berada pada pojok kiri bawah LIK, matriks yang akan dibuat juga mengacu pada berkas *properties*. Ada tiga jenis berkas *properties*, yaitu berkas *properties* untuk JWB, PDF dan Operator. Tidak ada berkas *properties* khusus untuk BCS, karena berkas *properties* untuk BCS sudah termasuk dalam berkas *properties* untuk JWB atau PDF. Hal ini dikarenakan BCS dengan sendirinya sudah terkait dengan formulir-formulir JWB atau PDF. Penjelasan dan keterangan berkas-berkas yang terkait dalam proses ekstraksi (termasuk berkas *properties*) dapat dilihat pada lampiran.

- Titik awal pembuatan matriks berada di atas *skunk mark* yang terdapat pada pojok kiri bawah LIK. Matriks dibuat ke arah atas dan kanan dari titik awal pembuatan matriks. Pada matriks tersebut akan

dibuat sejumlah baris dan kolom dengan besar setiap sel yang telah ditentukan sebelumnya. Informasi jumlah baris dan kolom yang akan dibuat pada matriks tersebut dapat dilihat pada berkas *properties*. Berikut adalah ilustrasi matriks yang dibuat pada lembar LIK.



Gambar 3. 12 Matriks *boolean* 2 dimensi

5. Pemeriksaan nilai *boolean* dari sel (grid)

- Setelah matriks selesai dibuat, dilakukan pemeriksaan untuk mendapatkan nilai *boolean* dari setiap sel (grid). Berdasarkan berkas *properties*, dapat diketahui nilai *MinBlackFullness*. *MinBlackFullness* adalah persentase minimum luas area piksel yang berwarna hitam dalam suatu sel (grid) sehingga dianggap telah diisi (dihitamkan). Jika persentase luas piksel yang berwarna hitam pada sel (grid) tersebut memenuhi nilai *MinBlackFullness*, maka sel bernilai 1, jika tidak maka bernilai 0.

6. Pengelompokan area sel (grid) matriks dan pemetaan *value*

- Setelah nilai *boolean* matriks selesai diperiksa, dilakukan pengelompokan area sel (grid) menurut Berkas format yang bersesuaian. Pada contoh ini berkas yang bersesuaian adalah berkas "FormatJWB08.txt" (untuk lebih jelasnya dapat dilihat pada lampiran). Berdasarkan berkas "FormatJWB08.txt", matriks dikelompokan menjadi beberapa area yaitu Nama Peserta, Nomor Peserta, Kode Naskah, Tanggal Lahir, dan Kolom Jawaban.

- Lalu nilai dari matriks *boolean* tersebut (0 atau 1) dipetakan ke elemen *value* yang telah didefinisikan pada berkas format yang bersesuaian. Sebagai contoh, pada area Nama Peserta, elemen *value* yang didefinisikan adalah ABCDEFGHIJKLMNOPQRSTUVWXYZ. Sehingga jika pada area Nama Peserta pengisi menghitamkan kolom pertama, baris pertama, maka huruf pertama dari nama pengisi (nama peserta) *value*-nya adalah A, demikian seterusnya.

7. Penulisan berkas *output*

- Setelah didapat semua *value*, dari setiap kelompok area sel (grid), hasilnya akan ditulis ke dalam sebuah berkas *Output*. Berkas *output* ini berisi hasil pembacaan sejumlah lembar Jawaban atau Formulir yang terdapat dalam satu *batch* yang bersesuaian. Satu baris pada berkas *output* mewakili hasil pembacaan sebuah lembar Jawaban atau Pendaftaran.

- Urutan informasi hasil pembacaan LIK yang ditampilkan pada berkas *output* mengacu pada berkas string.txt. Pada contoh ini, berkas yang digunakan adalah berkas "JWB08-string.txt".

- Penamaan berkas *output* juga mengikuti aturan tertentu, untuk lebih jelasnya format penamaan berkas *output* dapat dilihat pada lampiran.

Setelah membandingkan beberapa algoritma di atas, kami mengambil kesimpulan bahwa algoritma inilah yang paling efektif untuk mendeteksi

lingkaran isian yang dihitamkan pada formulir LIK yang memiliki penanda *skunk mark* pada pinggiran formulir seperti pada formulir SPMB, UMB dan SNMPTN. Algoritma inilah yang selanjutnya akan kami terapkan pada sistem "*Scanner Project*".

3.6 Analisa Alir Proses Algoritma Deteksi *Skunk Mark*

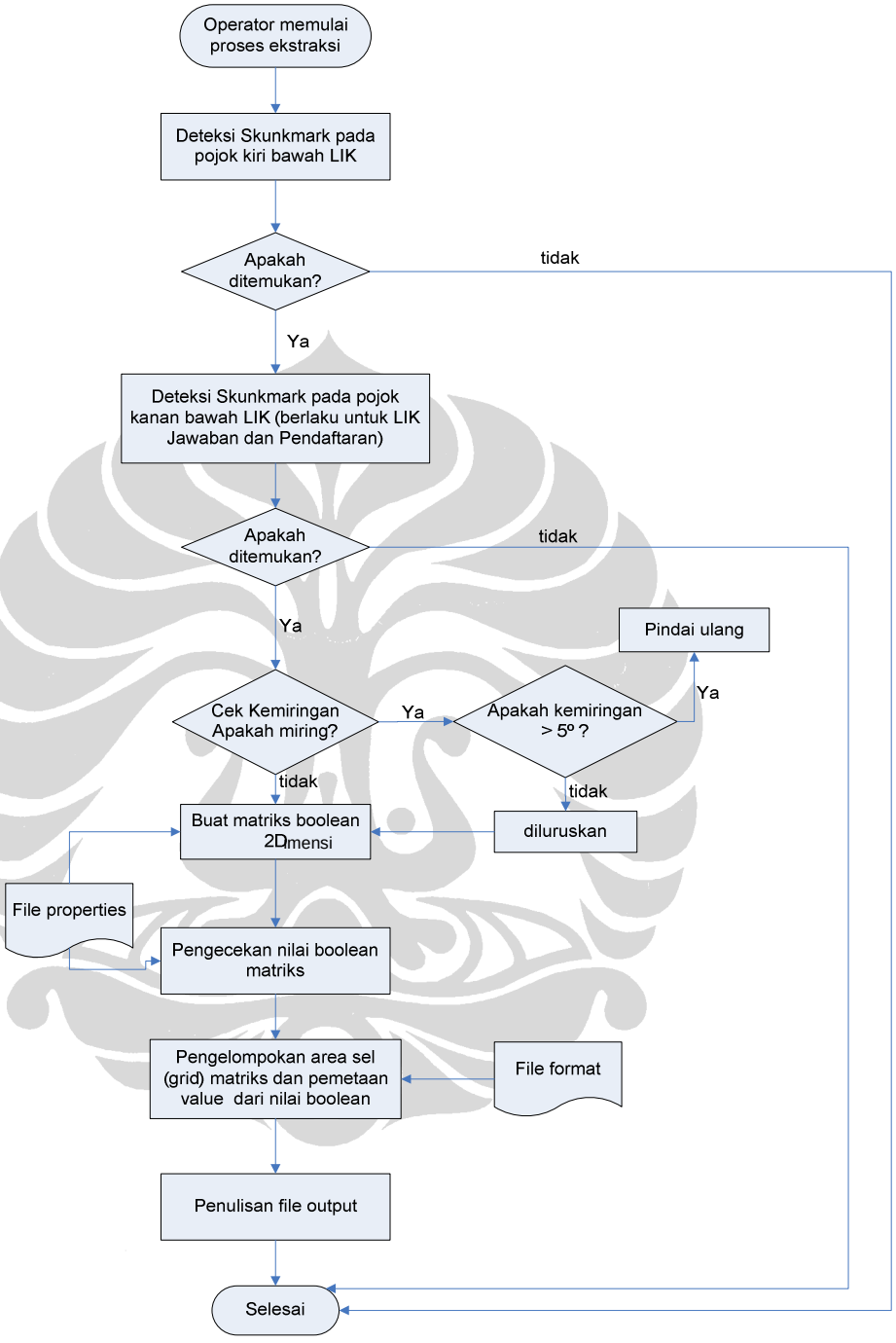
Untuk lebih memahami alur proses ekstrak data citra LIK dengan algoritma "Deteksi *Skunk Mark*", akan dijelaskan dengan menggunakan diagram alir (*flowchart*).

3.6.1 Alur Proses Ekstrak Data secara Umum

Dalam proses ekstrak data dengan menerapkan algoritma "Deteksi *Skunk Mark*", terdapat 5 proses utama, yaitu:

- Proses Deteksi *Skunk Mark*
  - Deteksi *skunk mark* pada pojok kiri bawah LIK (untuk BCS, JWB, dan PDF)
  - deteksi *skunk mark* pada pojok kanan bawah LIK (untuk JWB dan PDF)
- Proses Pembuatan Matriks *Boolean* 2 Dimensi
- Proses Pemeriksaan Nilai *Boolean* Matriks
- Proses Pengelompokkan Area Sel Matriks dan Pemetaan *Value*
- Proses Penulisan Berkas *Output*

Berikut ini adalah Diagram Alir proses ekstrak data secara umum.

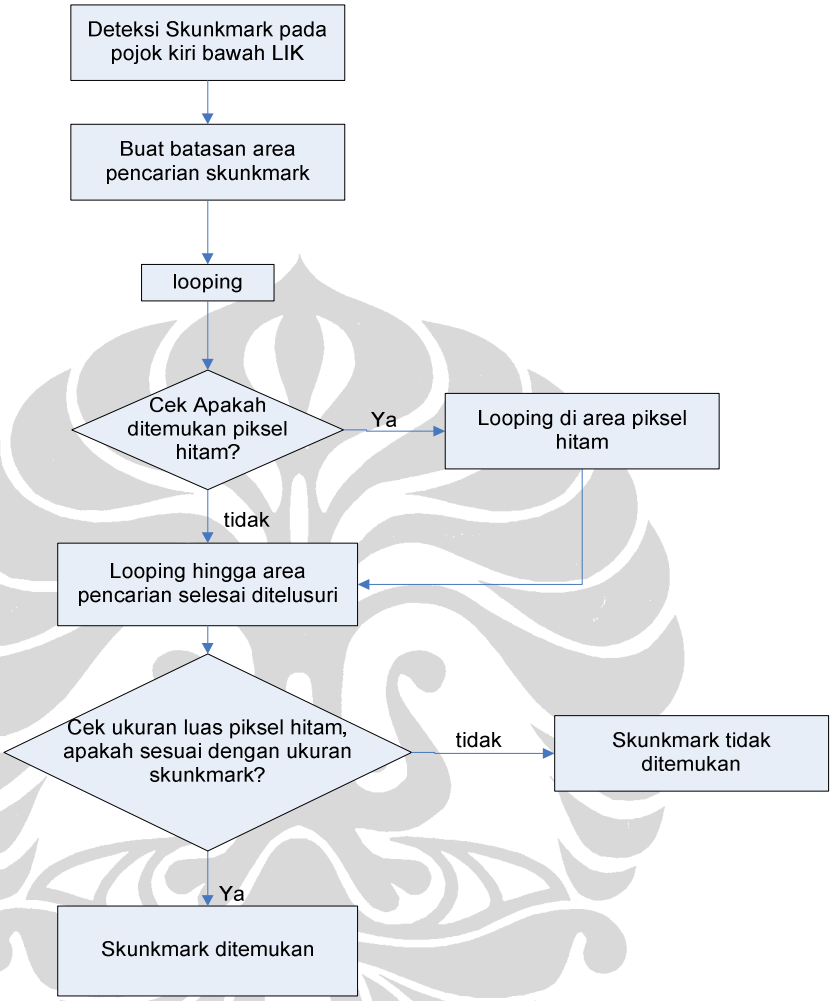


Gambar 3. 13 Diagram Proses Ekstrak Data

Masing-masing proses memiliki alurnya sendiri yang akan diperlihatkan pada gambar-gambar diagram alir proses di bawah ini.

3.6.2 Alur Proses Deteksi *Skunkmark*

Diagram ini menggambarkan proses deteksi *skunk mark*.

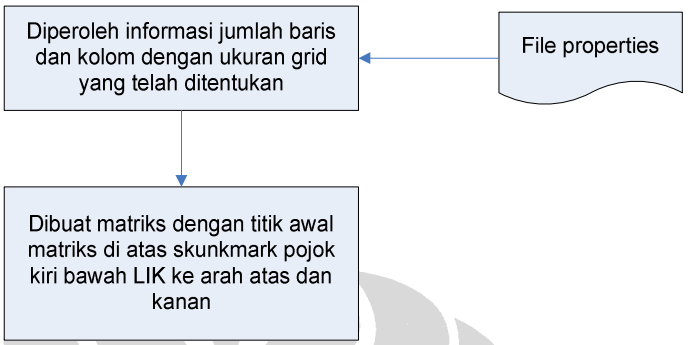


Gambar 3. 14 Diagram Alir Proses Deteksi *Skunk mark*

Tahapan-tahapan di atas berlaku untuk deteksi *skunk mark* pada pojok kiri bawah LIK (untuk BCS, JWB, dan PDF) dan deteksi *skunk mark* pada pojok kanan bawah LIK (untuk JWB dan PDF).

3.6.3 Alur Proses Pembuatan Matriks *Boolean* 2 Dimensi

Setelah *skunk mark* ditemukan, dibuat matriks *Boolean* 2 dimensi. Diagram ini menggambarkan proses pembuatan matriks 2 dimensi tersebut.

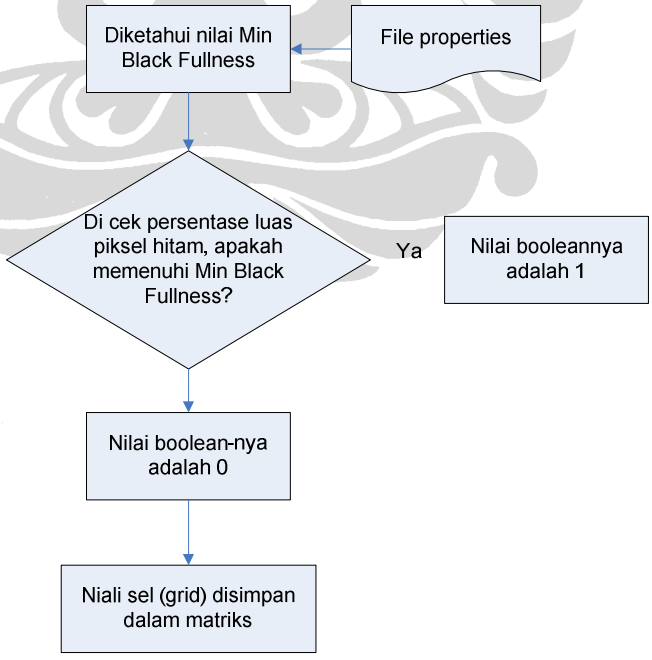


Gambar 3. 15 Alur Proses Pembuatan Matriks *Boolean* 2 Dimensi

Informasi jumlah baris dan kolom diketahui dari berkas *properties*.

3.6.4 Alur Proses Pemeriksaan Nilai *Boolean* Matriks

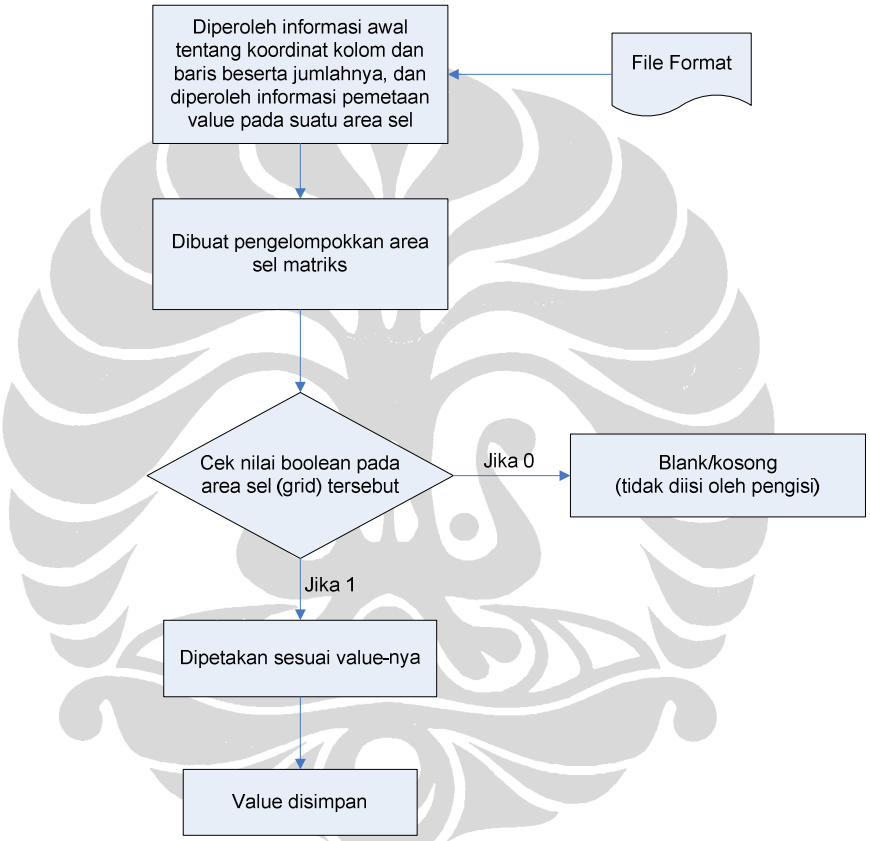
Setelah matriks terbentuk, kemudian dilakukan pemeriksaan nilai *boolean* setiap elemen dari matriks tersebut.



Gambar 3. 16 Alur Proses Pemeriksaan *Boolean* Matriks

3.6.5 Alur Proses Pengelompokkan Area Sel Matriks dan Pemetaan *Value*

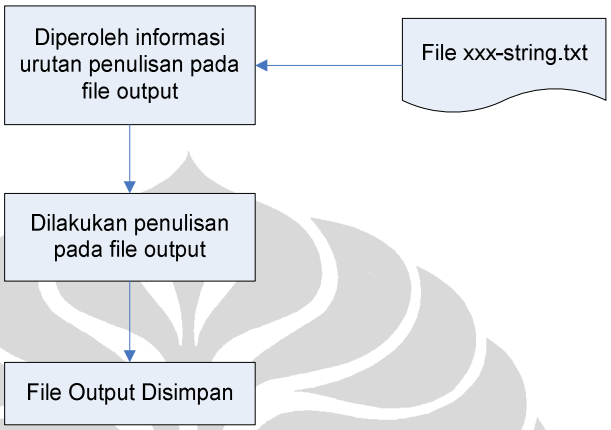
Setelah nilai *boolean* setiap elemen matriks diketahui, kemudian dibuat pengelompokan area sel matriks dan pemetaan *value* dari nilai *boolean*. Diagram ini menggambarkan proses tersebut.



Gambar 3. 17 Alur Proses Pengelompokkan Area Sel Matriks dan Pemetaan *Value* dari nilai *boolean*

3.6.6 Alur Proses Penulisan Berkas *Output*

Setelah pemetaan *value* selesai, dilanjutkan dengan penulisan hasil ekstraksi ke berkas *output*. Diagram Alir di bawah ini menggambarkan proses penulisan berkas *output*.



Gambar 3. 18 Alur Proses Penulisan Berkas *Output*

Informasi urutan penulisan pada berkas output diperoleh dari berkas string yang bersesuaian. Nama berkas string bersesuaian dengan projectname dalam pengekstaksian LIK. Sebagai contoh, jika LIK yang sedang diekstrak adalah formulir jawaban tahun 2008, maka projectname-nya adalah JWB08. Dengan demikian, berkas string yang bersesuaian adalah JWB08-sting.txt. Untuk lebih jelasnya tentang Standard Operating Procedure (SOP), dapat dilihat pada lampiran.

BAB IV

PERANCANGAN SISTEM

Pada bab ini akan dibahas tentang perancangan sistem yang dilakukan berdasarkan analisis sistem yang telah dijelaskan pada bab sebelumnya. Kegiatan perancangan sistem meliputi perancangan *Sequence Diagram* dan perancangan *Class Diagram* berdasarkan *Use-Case Diagram* pada bab Analisis.

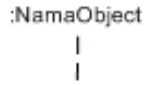
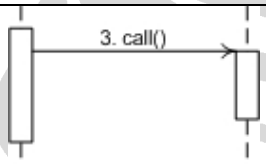
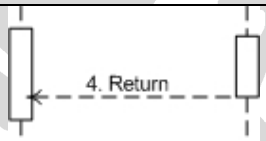
4.1 Perancangan *Sequence Diagram*

Sequence Diagram adalah bentuk model yang paling umum digunakan dalam menggambarkan *Interaction Diagram*. *Interaction Diagram* menggambarkan bagaimana sekelompok objek saling berkolaborasi dengan aturan tertentu yang berlaku. *Sequence Diagram* menggambarkan setiap objek tersebut dan interaksi pesan dan data yang terjadi di antara objek tersebut secara sekuensial terhadap waktu. *Sequence Diagram* ini dibentuk dari *Use-CaseDiagram* yang telah disusun pada bab sebelumnya.

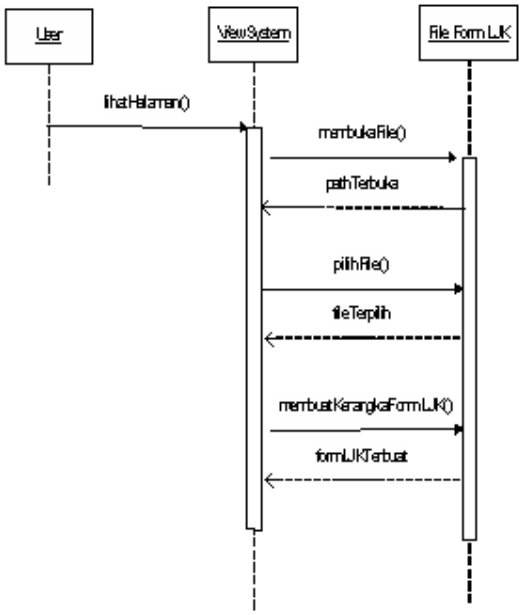
Bentuk *Sequence Diagram* akan menyerupai *timeline* proses antar objek yang ada disertai transfer data antar objek tersebut. Semua kemungkinan alir kejadian sistem ini digambarkan secara berurutan dari atas ke bawah secara sekuensial. Alir kemungkinan kejadian utama pada sebuah sub *Use-Case* digambarkan di bagian paling atas dari diagram tersebut. Sedangkan alir-alir kemungkinan lainnya yang dapat terjadi pada sub *Use-Case* tersebut digambarkan di bawahnya. Untuk menggambarkan *Sequence Diagram*, tim menggunakan aplikasi Microsoft Office Visio 2007 dengan penjelasan singkat dari notasi-notasi yang digunakan.

Tabel berikut ini berisi notasi *Sequence Diagram*.

Tabel 4. 1 Notasi *Sequence Diagram* [WHI04]

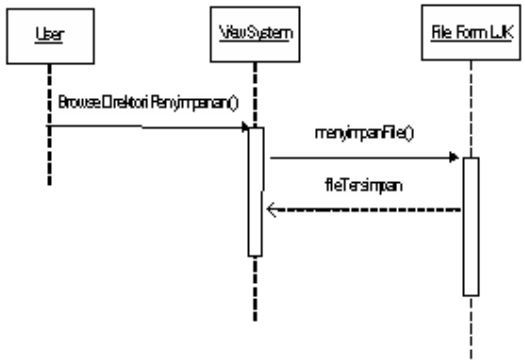
Notasi	Keterangan
	*Object* adalah sebuah enkapsulasi dari data yang memiliki prilaku tertentu yang mampu mengubah data yang dimilikinya.
	*Activation Bar* menggambarkan masa aktif dari sebuah *object* untuk melakukan suatu operasi sejak proses instansiasi hingga *object* tersebut dihapus dari *system memory*.
	*Call Message* adalah interaksi antar objek berupa pemanggilan operasi (*method*) dari suatu objek oleh objek lainnya.
	*Return Message* adalah pesan yang dikirimkan sebagai respon atas *call message* yang diterima.

Gambar-gambar berikut menunjukkan *Sequence Diagram* dari sistem "*Scanner Project"* pada Lembar SPMB. Berikut ini adalah *Sequence Diagram* dari *Use-Case* Membuat Kerangka LIK. *Actor* dari proses ini adalah Desainer LIK.



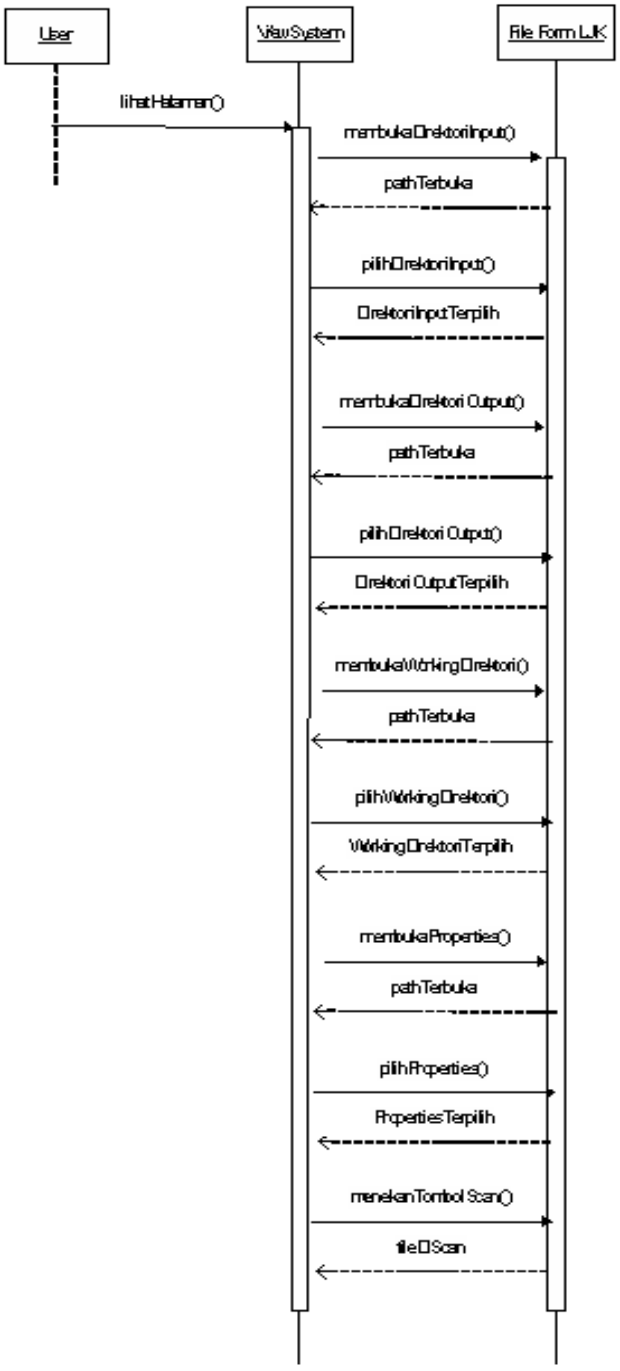
Gambar 4. 1 *Sequence Diagram* Membuat Kerangka LIK

Berikut ini adalah *Sequence Diagram* dari *Use-Case* Menyimpan Hasil Kerangka LIK. *Actor* dari proses ini adalah Desainer LIK.



Gambar 4. 2 *Sequence Diagram* Menyimpan Hasil Desain Format Kerangka LIK

Berikut ini adalah *Sequence Diagram* dari *Use-Case* Mengekstrak Data Citra LIK. *Actor* dari proses ini adalah Operator Ekstrak Data.



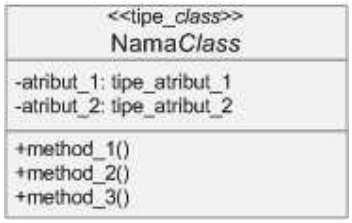
Gambar 4. 3 *Sequence Diagram* Mengekstrak Citra LIK

4.2 Perancangan *Class Diagram*

Class Diagram adalah gambar grafis mengenai struktur objek statis dari suatu sistem, menunjukkan kelas-kelas objek yang tersusun dalam sebuah sistem dan juga hubungan antara kelas objek tersebut [WHI06].

Berikut adalah tabel yang berisi daftar notasi yang digunakan dalam merancang *Class Diagram.*

Tabel 4. 2 Notasi *Class Diagram* [WHI04]

Notasi	Nama	Deskripsi
	*Class*	Kumpulan dari *instance* sebuah objek yang memiliki *properties* dan perilaku yang sama. *Properties* yang dimiliki dilambangkan dengan *attribute* dan perilaku atau operasi yang dapat dijalankan dilambangkan dengan *method.*
	Message	Menunjukkan adanya komunikasi antar *class* berupa pemanggilan *method* dari satu *class* ke *class* lain.
	Generalization	Titik akhir dari proses
	Dependency	*Dependency* terdapat di antara dua elemen jika perubahan pada definisi sebuah elemen mengakibatkan perubahan pada elemen lainnya.

Berdasarkan *Sequence Diagram* yang telah dibuat, tim pengembang mengetahui objek-objek yang dibutuhkan beserta kapabilitas dari setiap objek. Kemudian objek-objek tersebut direpresentasikan ke dalam *Class Diagram. Class Diagram* menggambarkan kelas-kelas yang terdapat dalam suatu sistem, beserta hubungan

antarkelas dan atribut yang dimiliki oleh masing-masing kelas. Diagram ini juga dapat diinterpretasikan sebagai model yang menggambarkan rancangan detil sistem berorientasikan objek.

Gambar *Class Diagram* untuk menggambarkan objek-objek apa saja yang dibutuhkan dalam mengkonstruksi sistem dapat dilihat pada lampiran.

Setelah semua *class* terdefinisi melalui *Class Diagram* yang telah dibuat, proses implementasi dapat mulai dengan memanfaatkan kerangka yang telah terbentuk dari *Class Diagram* dan *Sequence Diagram*.

4.3 Perancangan *User Interface*

Perancangan *User Interface* sistem dilakukan dalam beberapa tahapan. Dasar perancangan *User Interface* merujuk kepada *Use-Case Specification*. Dari *Use-Case Specification* dibuat perancangan *User Interface* yang direpresentasikan melalui sketsa gambar. Gambar dibuat dengan perangkat lunak pengolah gambar pada umumnya. Gambar yang sudah dibuat dijadikan dasar implementasi *User Interface*.

Perancangan *User Interface* memberikan gambaran dasar bagi implementasi sistem. Alur halaman sistem diimplementasikan berdasarkan perancangan *User Interface* yang masih berupa gambar. Dengan cara ini proses implementasi menjadi sangat terbantu. Perancangan *User Interface* dilakukan untuk semua *Use-Cases* yang ada. Untuk melihat lebih jelas mengenai rincian *User Interface* sistem dapat dilihat pada Lampiran.

BAB V

IMPLEMENTASI

Dalam bab ini akan dijelaskan mengenai implementasi dari sistem yang dikembangkan. Pembahasan bab ini memaparkan tentang proses implementasi dan solusi teknis yang digunakan tim pengembang dalam membangun sistem "*Scanner Project*". Solusi teknis berkaitan dengan *development environment* dan *supporting tools.* Selain itu, juga akan dijelaskan mengenai implementasi prosedur yang dikembangkan dalam membangun sistem ini.

5.1 Proses Implementasi

Proses implementasi sistem "*Scanner Project*", dilakukan dengan mengacu kepada desain logikal yang sudah dibuat sebelumnya. Seluruh sistem dibuat dengan menggunakan bahasa pemrograman Java dan menggunakan IDE NetBeans 6.1 untuk melakukan *coding, running, debugging,* serta *testing* program. *Delivery* sistem ini berupa sebuah sistem berbasis desktop (*desktop-based application*).

Dalam implementasi sistem "*Scanner Project*" ini, kami menggunakan mesin dengan spesifikasi sebagai berikut:

- Processor : Intel(R) Pentium(R) 4 CPU 2GHz
- Memory : 512MB RAM
- Harddisk : 40GB
- Sistem Operasi : Microsoft Windows XP Professional (5.1, Build 2600)

Bahasa pemrograman Java berikut IDE NetBeans 6.1 dipakai dengan beberapa pertimbangan sebagai berikut:

- Memudahkan kami dalam membuat tampilan keseluruhan antarmuka sistem
- Memudahkan kami dalam melakukan *debugging* dan *testing* program.
- Memudahkan kami dalam mencari metode yang terdapat dalam API JAVA.

5.2 Solusi Teknis

Solusi teknis yang digunakan dan diimplementasikan pada sistem akan dijelaskan melalui dua hal, yaitu *development environment* dan *supporting tools*. *Development environment* adalah perangkat-perangkat lunak yang digunakan untuk menunjang berjalannya sistem. Sedangkan *supporting tools* adalah perangkat-perangkat lunak yang digunakan untuk menunjang proses analisis, perancangan, dan implementasi sistem.

5.2.1 *Development Environment*

Berikut ini adalah tabel yang merupakan rangkuman komponen-komponen yang kami gunakan sebagai *development environment*.

Tabel 5. 1 Tabel *development environment*

Development Environment	Deskripsi
JDK 6.0	Java Development Kit versi 6.0 kami gunakan dalam membuat *"Scanner Project"*.
Windows XP Professional 2002	Dalam pengembangan sistem sistem operasi yang dipakai adalah Windows XP.

5.2.2 *Supporting Tools*

Selain *development environment*, tentunya diperlukan suatu *supporting tools* yang bisa mendukung proses pengembangan menjadi lebih mudah dan baik. *Supporting tools* merupakan aplikasi pembantu dalam pengembangan sistem. *Tools* tersebut digunakan antara lain untuk pengujian, pemodelan dan penggambaran sistem. *Supporting tools* yang tim pengembang gunakan antara lain sebagai berikut:

Tabel 5. 2 Tabel *supporting tools*

Development Environment	Deskripsi
GIMP	GIMP adalah program manipulasi gambar *open source*, yang kami gunakan untuk mengubah-ubah *input* "*Scanner Project*".
Microsoft Paint	Alat yang ringan ini kami gunakan untuk mengubah gambar dan melihat pikselnya dalam waktu singkat.
Microsoft Word 2007	Microsoft Word 2007 adalah aplikasi pembantu dalam membuat dokumentasi *"Scanner Project"*.
Microsoft Visio 2003	Microsoft Visio 2003 adalah aplikasi pembantu dalam membuat diagram-diagram yang dipakai pada proses perancangan sistem seperti use-case diagram, *Sequence Diagram*, dan *Class Diagram*.
Net Beans 6.1	Kami menggunakan NetBeans 6.1 untuk membangun aplikasi *"Scanner Project"*, termasuk *testing* dan *debuging*
Scite Text Editor	Scite kami gunakan untuk melihat keluaran hasil *"Scanner Project"*.

5.3 Implementasi Prosedur

Pada sub-bab ini akan dijelaskan rincian prosedur yang digunakan dalam implementasi sistem "*Scanner Project*".

5.3.1 Prosedur deteksi tanda baca LIK

Prosedur untuk mendeteksi tanda baca *skunk mark* adalah sebagai berikut:

1. Membuat batasan area pencarian *skunk mark* LIK
2. Membuat *vector* untuk menampung hasil pencarian
3. *Looping* seluas area batasan yang dibuat

{

4. Jika ditemukan piksel berwarna hitam maka mulai dilakukan pemeriksaan untuk mengidentifikasi kotak hitam *skunk mark*

{

5. Jika ukuran kotak hitam tersebut sesuai, maka kotak tersebut dianggap sebagai *skunk mark*

{

6. Hasil deteksi tanda baca ditaruh dalam *vector*

}

}

}

5.3.2 Prosedur pengambilan informasi baris dan kolom

Total baris dan kolom dibaca langsung dari berkas *properties* dan nilainya dimasukkan ke dalam suatu variabel.

5.3.3 Prosedur pembuatan area seleksi bebas

Area seleksi bebas adalah area yang dapat dipilih pada GUI untuk mendesain format kerangka LIK. Area yang dapat dipilih ini berbentuk kotak. Prosedurnya adalah sebagai berikut:

1. Mendapatkan info posisi tanda baca *skunk mark* pertama yang berada pada pojok kiri bawah LIK, dari berkas *properties*.
2. Menentukan lebar tiap elemen yang disorot
3. Mendefinisikan ukuran area seleksi bebas
4. Memvisualkan area seleksi bebas

5.3.4 Prosedur pemberian nama area pada GUI untuk mendesain format kerangka LIK

Area yang telah dipilih pada GUI untuk mendesain format kerangka LIK, akan diberi nama. Prosedurnya adalah sebagai berikut:

1. Membuat *nametag* yang menyimpan info area
2. Menambahkan info *nametag* ke dalam *array tag*, lalu digambar di layar

5.3.5 Prosedur pembuatan matriks area pindai

Matrik area pindai berisi informasi mengenai lingkaran yang dihitamkan atau tidak. Prosedurnya adalah sebagai berikut:

1. Mendapatkan jumlah baris dan kolom dari area pengisian
2. Inisiasi matriks 2 dimensi yang berisi informasi mengenai status hitam atau tidaknya suatu lingkaran
3. *Looping* sebanyak jumlah baris

{

4. *Looping* sebanyak jumlah kolom

{

5. Membuat area pindai berbentuk segi empat dari sebuah bulatan
6. Pemberian nilai elemen matriks

{

7. Jika luas area lingkaran yang dihitamkan memenuhi syarat, maka bulatan itu dianggap telah ditandai
8. Jika lingkaran tersebut dianggap ditandai maka diberi nilai 1 atau *true*, sebaliknya jika dianggap tidak ditandai diberi nilai 0 atau *false*

}

}

}

5.3.6 Prosedur pengambilan informasi dari matriks area pindai

Setelah matriks area pindai telah dibuat, maka dilakukan pengambilan informasi dari area tersebut. Prosedurnya adalah sebagai berikut:

1. Buat varibel untuk menampung hasil pemindaian
2. Jika banyaknya anggota *value* dari suatu *key* tidak sama dengan tinggi matriks yang didefinisikan, maka *return error*. Anggota *value* adalah jumlah elemen dari *value* area tersebut. Contoh, jika *value*-nya adalah angka maka jumlah anggota *value*-nya ada 10 elemen yaitu dari 0 s.d. 9
3. *Looping* sebanyak jumlah baris matriks

{

4. *Looping* sebanyak jumlah kolom matriks

{

5. Jika peserta menjawab dua bulatan sekaligus maka error.
6. Jika tidak, ambil *value* dari bulatan

}

}

7. Hasilnya berupa ejaan informasi per karakter yang ditampung dalam suatu variabel.

5.3.7 Prosedur penulisan informasi ke berkas *output*

Prosedur ini menentukan dan menampilkan urutan informasi pada berkas *output.*

1. Buat berkas untuk menyimpan *output.*
2. Baca berkas yang berisi urutan mengenai informasi apa saja yang akan dicetak ke berkas *output*
3. *Looping* sebanyak jumlah baris dari berkas nomor 2

{

4. Ambil *value* dari matriks sesuai dengan informasi pada baris

}

5. *Value* dicetak ke berkas *output*.

5.4 Hambatan yang dihadapi

Dalam mengembangkan sistem "*Scanner Project*" ini, tim pengembang mempelajari sistem yang sebelumnya telah dikembangkan oleh salah satu dosen Fakultas Ilmu Komputer Universitas Indonesia, Pak Denny, S.Kom, MIT, untuk dapat memahami ide dasar pembacaan tanda isian pada LIK. Namun pada saat mempelajari sistem tersebut kami menemui beberapa kendala diantaranya yaitu:

- sistem tersebut belum sempurna sehingga belum dapat berfungsi dengan baik
- tidak adanya dokumentasi yang memadai untuk mempelajari sistem tersebut sehingga memperlambat tim dalam memahami ide dasar pembacaan tanda isian pada LIK dan memperlambat kerja tim. Maka dari itu tim pengembang mencari referensi lain untuk membantu tim dalam mengembangkan sistem.

BAB VI

UJI COBA DAN ANALISIS

Bab ini terdiri atas penjelasan data, hasil uji coba beserta analisis terhadap hasil uji coba terhadap sistem "*Scanner Project*".

6.1 Data Uji Coba

Uji coba proyek ini menggunakan formulir LIK SPMB tahun 2006, formulir LIK UMB tahun 2008 dan formulir LIK SNMPTN tahun 2008. Keterangan tentang formulir-formulir tersebut dapat dilihat pada tabel berikut:

Tabel 6. 1 Data Uji Coba

Jenis formulir	Tipe Formulir	Tahun	Jumlah
SPMB	*Batch Control Sheet*	2006	2
	Jawaban		4
	Pendaftaran		2
UMB	*Batch Control Sheet*	2008	3
	Jawaban		275
	Pendaftaran		291
SNMPTN	*Batch Control Sheet*	2008	1
	Jawaban		5612
	Pendaftaran		4161

Semua gambar formulir itu didapat dari narasumber kami, Pak Dadan Hardianto. Formulir-formulir itu merupakan berkas asli ujian, sehingga tes yang kami lakukan sama dengan simulasi pemeriksaan sebenarnya.

6.3 Metode Uji Coba

Dalam pengembangan sistem tim pengembang melakukan pengujian terhadap sistem dengan metode *black box testing*. *Black box testing* adalah pengujian yang digunakan untuk memastikan apakah sistem yang dikembangkan sudah memiliki fungsi yang diinginkan, yaitu menerima *input* dengan benar, dan

menghasilkan *output* yang benar. *Black box testing* pada dasarnya dilakukan untuk mencari kesalahan pada fungsi-fungsi yang ada, pada *interface*, pada perilaku dan kinerja sistem, dan pada proses inisialisasi atau akhir proses [PRE05]..

Pengujian sistem dilakukan secara bertahap, yaitu mulai dari tingkat sub *Use-Case, Use-Case*, hingga integrasi sistem. Untuk setiap *Use-Case* dan sub *Use-Case*, dibuat sebuah *Test Case* yang berisi tahapan-tahapan pelaksanaan pengujian. Untuk lebih jelasnya *Test Case* dapat dilihat pada Lampiran. Dengan mengikuti alur yang dipaparkan pada lampiran tersebut, kegiatan pengujian dapat dilakukan dengan lebih teliti dan menyeluruh.

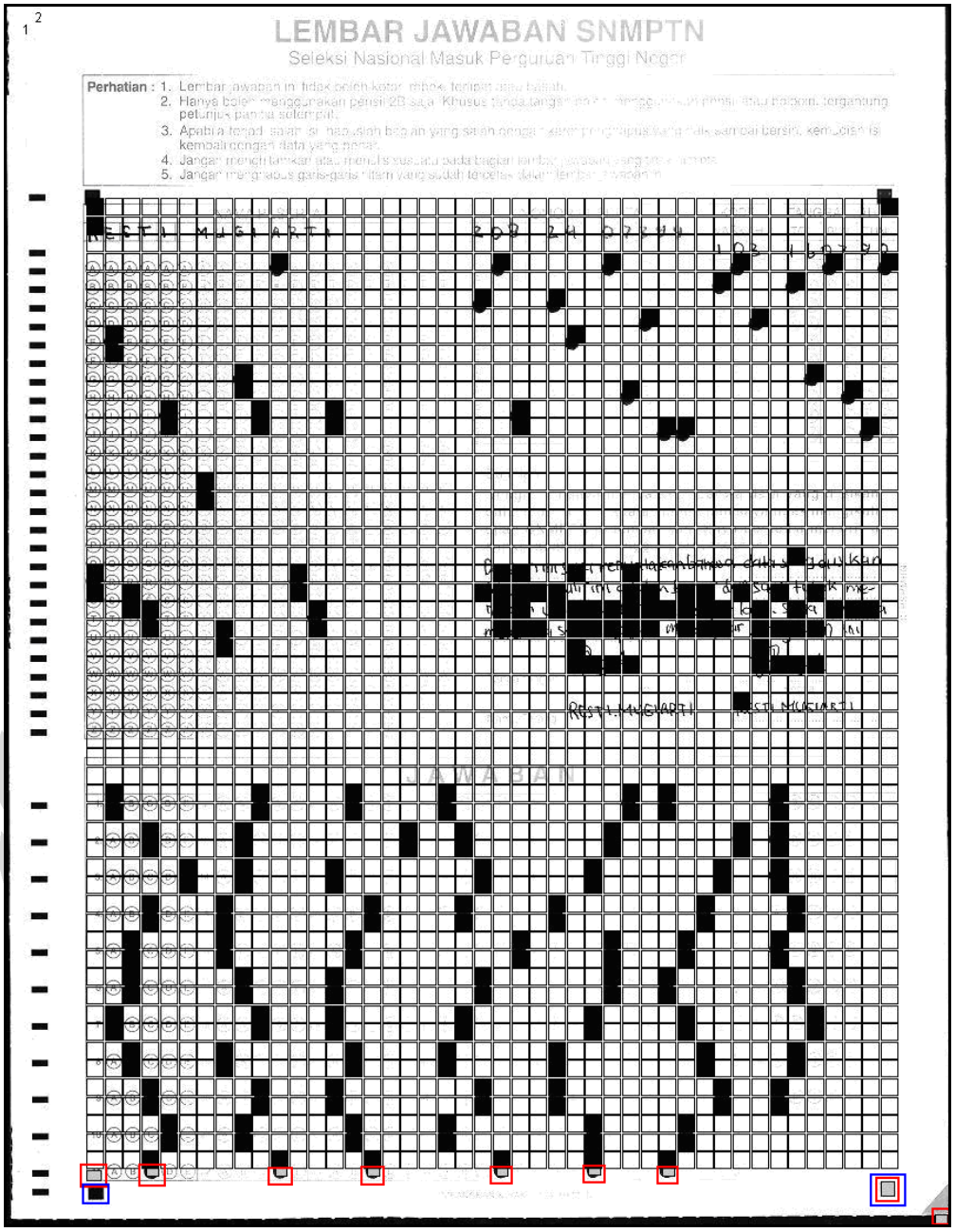
6.3 Hasil Uji Coba

Setelah kami ujicoba untuk pengekstrakan formulir LIK, hasilnya adalah sebagai berikut:

Tabel 6. 2 Hasil Uji Coba

Nama formulir	Jenis formulir	Tahun	Jumlah kesalahan
SPMB	*Batch Control Sheet*	2006	0
	Jawaban		0
	Pendaftaran		0
UMB	*Batch Control Sheet*	2008	0
	Jawaban		6
	Pendaftaran		23
SNM-PTN	*Batch Control Sheet*	2008	0
	Jawaban		23
	Pendaftaran		56

Kesalahan yang terjadi pada formulir LIK UMB 2008 dan SNMPTN 2008 karena bentuk bulatan isian pada ujung bawah formulir mirip dengan penanda *skunk mark* LIK. Hal ini menyebabkan sistem menganggap formulir LIK mempunyai penanda *skunk mark* lebih dari yang dibutuhkan, sehingga membuat system *error* dan tidak dapat membaca dan mengekstrak nilai dari formulir dengan baik. Berikut ini adalah gambar hasil *debug* yang mendeteksi bulatan sebagai *skunk mark.*



Gambar 6. 1 Bulatan-bulatan yang dideteksi sebagai *skunk mark*

Pada gambar di atas kotak merah dianggap sebagai *skunk mark*. Kotak biru adalah *skunk mark* yang sebenarnya. Dapat dilihat pada gambar terjadi banyak kesalahan karena bulatan isian yang dianggap sebagai *skunk mark*, sementara *skunk mark* yang sebenarnya hanya satu yang terdeteksi yaitu pada pojok kanan bawah LIK. *Skunk mark* pada pojok kiri bawah tidak terdeteksi. Tidak hanya

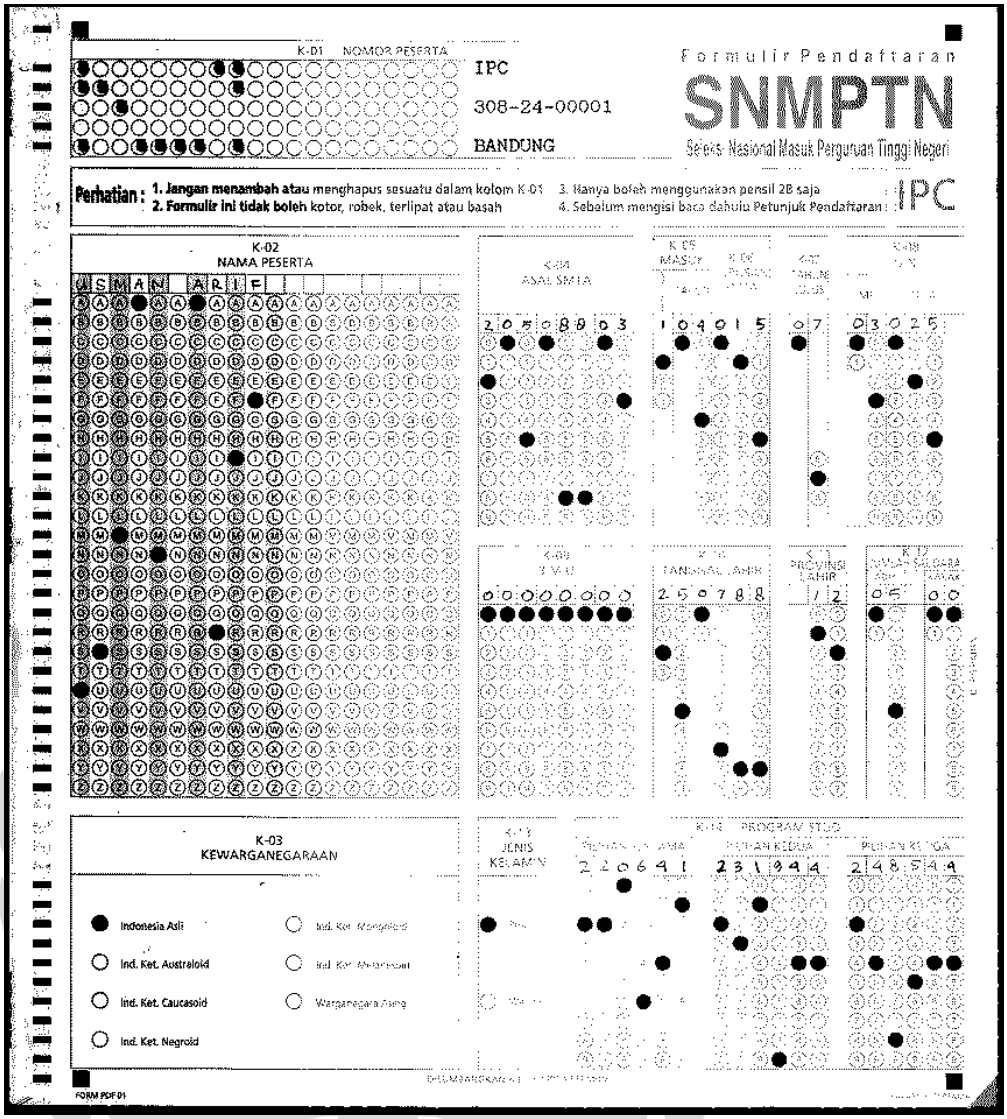
bulatan isian yang dianggap sebagai *skunk mark*, tetapi segitiga kecil yang berwarna abu-abu pada pojok kanan bawah LIK juga dianggap sebagai *skunk mark*. Lalu jika Anda perhatikan, matriks yang dibuat menjadi tidak pas dengan area isian LIK yang sesungguhnya. Matriks yang mulai dibuat dari bawah menjadi naik beberapa piksel hingga mengakibatkan kesalahan sampai bagian atas matriks.

Solusinya adalah kami mempersempit area pencarian *skunk mark* hanya pada area yang kemungkinan terdapat *skunk mark* yaitu pada area pojok kiri bawah LIK dan pojok kanan bawah LIK. Jika pada kedua area tersebut kami menemukan masing-masing lebih dari satu buah *skunk mark*, seperti pada formulir BCS (contoh formulir BCS dapat dilihat pada lampiran), sedangkan pada masing-masing area tersebut, kami hanya membutuhkan satu buah *skunk mark*, maka kami memperluas toleransi jumlah penanda *skunk mark* sehingga program tidak sensitif lagi terhadap kelebihan penanda *skunk mark* LIK. Berapapun banyaknya *skunk mark* yang ditemukan program, tidak menjadi masalah karena yang di butuhkan hanya *skunk mark* pertama dan terakhir. Kedua *skunk mark* tersebut digunakan untuk memeriksa kemiringan LIK. Sementara s*kunk mark* pada pojok kiri bawah LIK menjadi acuan titik awal pembuatan matriks.

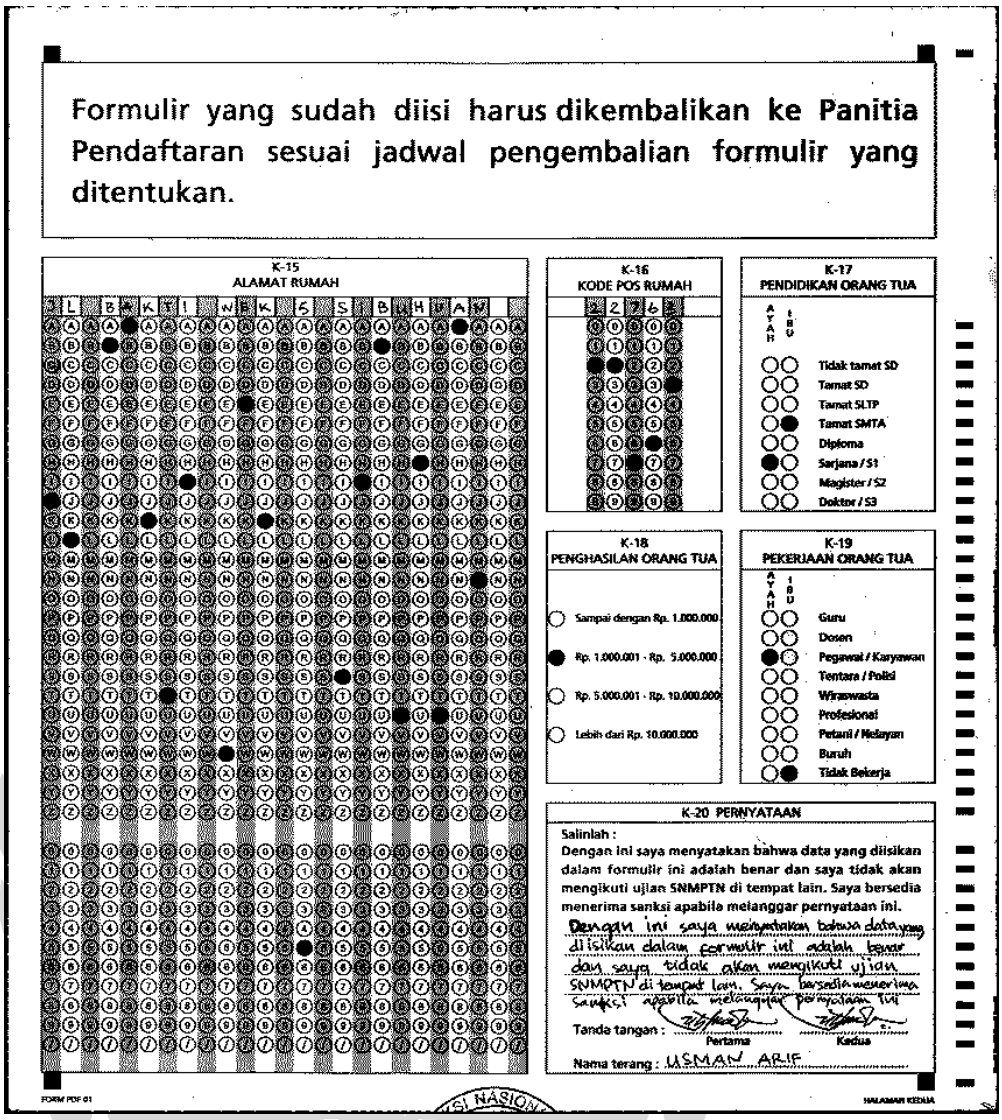
Kesalahan lainnya yang terjadi pada formulir LIK UMB 2008 dan SNMPTN 2008 adalah terdapat beberapa bagian formulir yang dihitamkan tidak dapat dikenali oleh sistem sehingga datanya tidak dapat diekstrak. Sistem tidak dapat mengenali bulatan yang dihitamkan karena tingkat kehitaman pada beberapa bagian formulir LIK yang tidak merata dan terkadang kurang hitam. Hal ini mengakibatkan persentase luas area yang dihitamkan (*piksel* yang berwarna hitam) kurang dari nilai *MinBlackFullness. MinBlackFullness* adalah nilai persentase minimum luas area piksel berwarna hitam yang telah ditentukan pada berkas *properties* agar area piksel yang dihitamkan tersebut dapat dianggap sebagai isian. Berkas *properties* merupakan salah satu berkas yang dipakai sebagai acuan untuk mengekstrak LIK, di samping itu pula terdapat berkas-berkas acuan lainnya seperti berkas format.txt, not-null.txt, dan no-space.txt, dan

lain-lain (untuk lebih jelasnya dapat dilihat pada lampiran). Tingkat kehitaman yang tidak merata ini biasanya disebabkan karena hasil pemindaian formulir kurang baik, sehingga *input* citra digital LIK yang dihasilkan kurang baik dan tidak merata kehitamannya. Karena kekurangan dalam pemindaian LIK ini, mengakibatkan beberapa bagian formulir yang tidak dapat di ekstrak datanya sehingga terjadi kesalahan pada proses pembacaan dan pengekstrakan data oleh sistem.

Berikut ini adalah contoh citra hasil pemindaian LIK yang tidak merata tingkat kehitamannya. Hal ini menyulitkan tim pengembang untuk menentukan nilai *MinBlackFullness* yang berlaku untuk setiap sel, karena tingkat kehitaman sel yang tidak merata.



Gambar 6. 2 Citra LIK hasil pemindaian yang tingkat kehitamannya tidak merata



Gambar 6. 3 Citra LIK hasil pemindaian yang tingkat kehitamannya tidak merata

Sedangkan pada gambar di atas, pada area sel Alamat Rumah dan Kode Pos Rumah, kolom yang ganjil relatif jauh lebih hitam daripada kolom genap. Hal ini bisa mengakibatkan program mendeteksi bahwa bulatan telah diisi dan atau pada satu kolom terdapat lebih dari satu buah bulatan isian sehingga menyebabkan error.