



**ITS**  
Institut  
Teknologi  
Sepuluh Nopember

**TUGAS AKHIR - TE 141599**

**PENGENALAN CITRA UANG KERTAS RUPIAH DENGAN  
METODA *LOCAL BINARY PATTERN***

M. Kukuh Prayogo  
NRP 07111240000142

Dosen Pembimbing  
Ir. Tasripan, MT.  
Dr. Ir. Hendra Kusuma, M.Eng., Sc.

DEPARTEMEN TEKNIK ELEKTRO  
Fakultas Teknologi Elektro  
Institut Teknologi Sepuluh Nopember  
Surabaya 2018





**ITS**  
Institut  
Teknologi  
Sepuluh Nopember

**FINAL PROJECT - TE 141599**

**RUPIAH BANKNOTES IMAGE RECOGNITION USING  
LOCAL BINARY PATTERN METHOD**

M. Kukuh Prayogo  
NRP 07111240000142

Supervisor  
Ir. Tasripan, MT.  
Dr. Ir. Hendra Kusuma, M.Eng., Sc.

DEPARTMENT OF ELECTRICAL ENGINEERING  
Faculty of Electrical Technology  
Institut Teknologi Sepuluh Nopember  
Surabaya 2018



## LEMBAR PERNYATAAN KEASLIAN

### PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Pengenalan Citra Uang Kertas Rupiah Dengan Metoda *Local Binary Pattern*" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi peraturan yang berlaku.

Surabaya, 01 Juni 2018



M. Kukuh Prayogo  
NRP. 07111240000142



**LEMBAR PENGESAHAN**

**Pengenalan Citra Uang Kertas Rupiah dengan  
Metoda Local Binary Pattern**

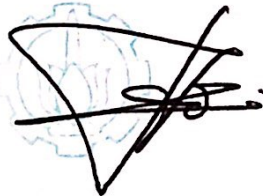
**TUGAS AKHIR**

**Diajukan Guna Memenuhi Sebagian Persyaratan Untuk  
Memperoleh Gelar Sarjana Teknik  
Pada  
Bidang Studi Elektronika  
Departemen Teknik Elektro  
Institut Teknologi Sepuluh Nopember**

**Menyetujui**

**Dosen Pembimbing I,**

**Dosen Pembimbing II,**



**Ir. Tasripan, MT.  
Nip: 196204181990031004**

**Dr. Ir. Hendra Kusuma, M.Eng.Sc.  
Nip: 196409021989031003**











## ABSTRAK

### Pengenalan Citra Uang Kertas Rupiah Dengan Metoda *Local Binary Pattern*

M. Kukuh Prayogo  
07111240000142

Dosen Pembimbing I : Ir. Tasripan, MT.

Dosen Pembimbing II : Dr. Ir. Hendra Kusuma, M.Eng., Sc.

#### **Abstrak:**

Metode *Local Binary Patterns* adalah salah satu *descriptor* citra yang digunakan untuk klasifikasi pada *Computer Vision* dengan cara menilai piksel pada area *threshold* citra dengan turunan biner. Toleransi LBP pada iluminasi monotonik dan komputasi yang sederhana menjadikan LBP salah satu metode yang banyak digunakan pada *Computer Vision*, salah satunya pada *facial recognition*. Pada tugas akhir ini, akan dilakukan percobaan pengenalan uang kertas menggunakan metoda LBP. Pengenalan uang kertas dilakukan sebagai bahan uji eksperimen karena proses tersebut bersifat umum dan aplikatif untuk perancangan sistem yang lebih lanjut, serta akan memberikan gambaran bagaimana performa metode LBP akan bekerja pada suatu benda statis dengan ukuran dan warna yang bervariasi pada bidang datar untuk setiap uang kertas yang diuji. percobaan akan terdiri dari perancangan perangkat keras dan perancangan perangkat lunak. Pada perancangan perangkat keras akan dirancang sebuah perangkat yang dapat melakukan pengambilan citra uang kertas dengan pencahayaan yang cukup. Pada perancangan perangkat lunak akan dirancang sebuah program pra-pemrosesan untuk persiapan citra sebelum dilakukan operasi LBP untuk pengenalan dan citra *database* dan operasi LBP, keseluruhan program akan menggunakan *Microsoft Visual Studio* 2015 dengan *OpenCV*. Hasil dari percobaan menggunakan 3 variasi uang kertas Rupiah bernilai 1000, 2000, 5000, 10000, 20000, 50000 dan 10000 menggunakan segmentasi 50, 128, 200 dan 800 menunjukkan persentase kebenaran sebesar 92,857% untuk 50 segmentasi, 97,61% untuk 128 segmentasi, 92,857% untuk 200 segmentasi dan 83,333% untuk 800 segmentasi.

**Kata kunci:** *Local Binary Pattern*, Pengenal Objek, *OpenCV*.

*Halaman ini sengaja dikosongkan*

## **ABSTRACT**

### ***Rupiah Banknotes Image Recognition Using Local Binary Pattern Method***

M. Kukuh Prayogo  
07111240000142

Supervisor I : Ir. Tasripan, MT.

Supervisor II : Dr. Ir. Hendra Kusuma, M.Eng., Sc.

#### ***Abstract:***

Local Binary Patterns is one of image descriptor that has been used for computer vision classification by labelling the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. Mononic tolerance and simple computation making LBP is one of the used methods in computer vision. Facial recognition is one of the process using LBP. In this final project, banknotes recognition using LBP methods will be tested. Banknotes recognition has been used widely and applicative for advanced implementation, also from this experiment we will know the performance of LBP operator processing still images with different colors on a level surface for every banknotes. the experiment will consist of hardware design and software design. For hardware design, we will design a device that can capture banknotes image with good lighting. for software design, we will make preprocessing program for LBP operation preparation that continues to recognition and image database operation, all of the program will be made within microsoft visual studio 2015 with opencv library. The result of the experiment using 3 variants of Rupiah banknotes ranging from 1000, 2000, 5000, 10000, 20000, 50000 and 100000 using 50, 128, 200 and 800 segmentations shows that the percentage of accuracy is 92,587% for 50 segmentations, 97,61% for 128 segmentations, 92,587% for 200 segmentations and 83,333% for 800 segmentations.

***Keywords:*** *Local Binary Pattern, Object Recognition, OpenCV.*

*Halaman ini sengaja dikosongkan*

## KATA PENGANTAR

Segala puji syukur kepada Tuhan YME , atas segala nikmat, berkat dan karunia-Nya yang tak terkira kepada penulis, hingga penulis mampu menyelesaikan Tugas Akhir dengan judul:

### **Pengenalan Citra Uang Kertas Rupiah Dengan Metoda *Local Binary Pattern***

Tujuan utama tugas akhir ini adalah sebagai salah satu persyaratan untuk menyelesaikan jenjang pendidikan pada Bidang Studi Elektronika Teknik Elektro Institut Teknologi Sepuluh Nopember.

Atas selesainya penyusunan tugas akhir ini, penulis ingin mengucapkan terima kasih kepada:

1. Dr. Ir. Hendra Kusuma, M.Eng., Sc. dan Ir. Tasripan, MT. selaku dosen pembimbing Tugas Akhir yang telah memberi bimbingan, penjelasan, nasehat dan kemudahan dalam penyelesaian Tugas Akhir ini.
2. -----dosen penguji----- selaku dosen penguji yang telah mengoreksi Tugas Akhir ini.
3. Dr. Eng. Ardyono Priyadi, ST., M.Eng. selaku ketua Departemen Teknik Elektro ITS.
4. Bapak Ir. Toto Sudiby, MM. dan Ibu Erma Suryani selaku orang tua penulis yang selalu mendukung dan mendoakan penulis.
5. M Bagus Pratomo, M Satrio Prabowo, M Akbar Prasetyo dan Muthia Nur Sabrina selaku saudara kandung penulis yang selalu memberi dukungan di setiap waktu penulis.
6. Kristoper Lukas, Halum Ghulami, Nitya A Fasalina, Eber Wonda dan teman-teman E52 yang tidak dapat disebutkan satu persatu.
7. Molly, Mercy, Myup, Meatball, Noodle, Peanut dan Jovita yang selalu menemani penulis.
8. Serta semua pihak yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis berharap para pembaca Tugas Akhir ini bersedia memberikan kritik, saran dan masukan yang membangun agar selanjutnya dapat menambah manfaat untuk kedepannya. Semoga laporan tugas akhir ini dapat bermanfaat dan bisa dijadikan referensi bagi Tugas Akhir selanjutnya.

*Halaman ini sengaja dikosongkan*



## DAFTAR ISI

HALAMAN JUDUL .....	1
LEMBAR PERNYATAAN KEASLIAN .....	5
LEMBAR PENGESAHAN .....	7
ABSTRAK .....	i
ABSTRACT .....	iii
KATA PENGANTAR.....	v
DAFTAR ISI .....	vii
DAFTAR GAMBAR.....	ix
DAFTAR TABEL .....	xi
BAB I .....	1
1.1 Latar Belakang.....	1
1.2 Perumusan Masalah.....	2
1.3 Tujuan dan Manfaat Penelitian.....	2
1.4 Metodologi Penelitian.....	3
1.5 Sistematika Penulisan.....	5
1.6 Relevansi .....	5
BAB II .....	7
2.1 Uang Indonesia.....	7
2.1.1 Karakteristik dan desain uang kertas Rupiah TE 2016 ..	8
2.2 <i>Local Binary Patterns</i> .....	10
2.2.1 Keunggulan Metoda <i>Local Binary Patterns</i> .....	14
2.2.2 Perbandingan Metoda LBP dengan HOG .....	15
2.3 <i>Mini PC</i> .....	16
2.4 OpenCV.....	17
2.4.1 Akses citra dari <i>file</i> dan kamera.....	18
2.4.2 <i>Grayscale conversion</i> .....	20
2.4.3 <i>Thresholding</i> .....	21
2.4.4 <i>Morphology</i> .....	22
2.4.5 <i>Contour</i> .....	23
2.4.6 <i>Region Of Interest</i> .....	24
BAB III.....	27

3.1	Perancangan perangkat keras.....	27
3.2	Perancangan perangkat lunak .....	30
3.2.1	Pengambilan citra.....	31
3.2.2	Thresholding dan morphology .....	32
3.2.3	<i>Contour</i> .....	35
3.2.4	<i>Region Of Interest</i> .....	36
3.2.5	Segmentasi .....	37
3.2.6	<i>Local binary pattern</i> .....	37
3.2.7	<i>Data learning</i> .....	38
3.2.8	Pengenalan uang kertas.....	39
BAB IV	.....	41
4.1	Pengujian algoritma preprocessing .....	41
4.1.1	Pengambilan gambar.....	41
4.1.2	<i>Thresholding</i> dan <i>morphology</i> .....	41
4.1.3	Segmentasi .....	42
4.2	Pengujian proses utama .....	43
4.2.1	Operasi <i>Local Binary Pattern</i> .....	43
4.2.2	Inisiasi <i>data learning</i> .....	44
4.2.3	Pengenalan uang kertas.....	44
4.2.4	Pengujian pengenalan .....	45
BAB V	.....	49
PENUTUP	.....	49
5.1	Kesimpulan.....	49
5.2	Saran.....	49
DAFTAR PUSTAKA	.....	51
LAMPIRAN	.....	53
BIODATA PENULIS	.....	73

## DAFTAR GAMBAR

Gambar 1.1 Blok diagram perancangan perangkat keras .....	3
Gambar 1.2 Alur kerja program .....	4
Gambar 2.1 Uang rupiah kertas.....	7
Gambar 2.2 Desain depan dan belakang uang kertas Rupiah TE 2016... 8	8
Gambar 2.3 Flowchart ekstraksi fitur LBP .....	11
Gambar 2.4 Contoh komputasi metode LBP.....	12
Gambar 2.5 Operasi <i>Local Binary Pattern</i> sederhana.....	13
Gambar 2.6 Citra proses LBP .....	14
Gambar 2.7 Contoh komputasi metode HOG .....	15
Gambar 3.1 Jarak dan posisi kamera terhadap objek. ....	28
Gambar 3.2 Dimensi kerangka alat .....	28
Gambar 3.3 Posisi dan jarak LED terhadap objek. ....	29
Gambar 3.4 Struktur <i>workspace</i> .....	29
Gambar 3.5 Dimensi dan bagian kerangka alat. ....	30
Gambar 3.6 Diagram alur program pengenalan uang kertas .....	31
Gambar 3.7 Hasil pengambilan citra menggunakan kamera .....	32
Gambar 3.8 Hasil citra <i>threshold</i> .....	33
Gambar 3.9 Hasil citra <i>morphology</i> .....	35
Gambar 3.10 Hasil citra ROI.....	37
Gambar 3.11 Ilustrasi nilai histogram data learning pada array 4 dimensi. .....	38
Gambar 4.1 Hasil pengambilan citra.....	41
Gambar 4.2 Hasil proses <i>threshold</i> dan <i>morphology</i> pada citra.....	42
Gambar 4.3 Hasil ROI citra .....	42
Gambar 4.4 Hasil segmentasi citra.....	43
Gambar 4.5 Hasil konversi LBP citra .....	43
Gambar 4.6 Citra <i>data base</i> pada <i>data learning</i> .....	44
Gambar 4.7 Tampilan hasil pengenalan uang kertas. ....	45

*Halaman ini sengaja dikosongkan*

## DAFTAR TABEL

Tabel 2.1 Tabel spesifikasi uang kertas Rupiah TE 2016 .....	10
Tabel 2.2 Tabel perbandingan nilai akurasi .....	15
Tabel 4.1 Hasil pengujian pengenalan uang kertas .....	46
Tabel 4.2 Hasil presentase kebenaran pengenalan uang kertas .....	46

*Halaman ini sengaja dikosongkan*

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Uang adalah alat tukar resmi yang digunakan dalam suatu negara, penggunaan uang sudah dilakukan sejak jaman pra-penjajahan hingga sekarang. Bentuk uang secara umum terbagi 2 jenis, yaitu uang kertas dan uang koin. Uang kertas yang sering digunakan kebanyakan terbuat dari bahan plastik dengan kombinasi bahan khusus untuk menunjukkan keaslian uang tersebut, sedangkan uang koin menggunakan logam sebagai bahan utamanya. Uang yang resmi beredar di Indonesia adalah uang Rupia yang diedarkan secara resmi melalui Bank Indonesia.

Kegunaan uang yang fungsinya luas dan beragam membuat uang menjadi bagian dari kehidupan sehari-hari masyarakat Indonesia yang mencakup semua umur dan golongan, tidak terlepas bagi masyarakat penyandang tuna netra yang akan menemui kesulitan dalam penggunaan uang. Pada uang Rupiah terdapat motif timbul yang dapat dirasakan oleh pembaca menggunakan indra peraba, namun kualitas uang yang akan semakin buruk menyebabkan motif tersebut pudar dan tidak akan terbaca lagi, hal ini akan menjadi masalah jika kualitas uang yang digunakan pengguna benar-benar buruk dan tidak dapat dibaca kembali sedangkan pengguna membutuhkan pembacaan nilai Rupiah yang cepat dan akurat. Sehingga adanya alat yang dapat mengidentifikasi nilai uang Rupiah tanpa harus membaca manual akan sangat membantu kegiatan penggunaan uang pada penyandang tunanetra. Didasari hal inilah yang mendorong munculnya penelitian perancangan alat pengenalan citra uang kertas Rupiah dengan metode *local binary pattern*, selain itu program ITS Smart City juga menjadi dasar munculnya ide tugas akhir ini.

Perancangan alat pengenalan uang kertas akan meliputi kamera *webcam* beresolusi 720p pada bagian atas untuk mendapatkan citra uang kertas pada bagian bawah, untuk membantu iluminasi akan digunakan 2 buah *LED Strip* pada bagian atas, proses akan dimulai dengan proses pra-pengolahan citra yaitu mempersiapkan citra untuk siap dilakukan proses *local binary pattern*, proses tersebut mencakup pengambilan citra, *morphology*, *edge detection* dan *threshold* untuk mendapatkan *Region of Interest* dari citra uang kertas, ROI dari uang kertas tersebut kemudian akan dilakukan segmentasi yang selanjutnya akan didapatkan nilai *local binary pattern* dari segmentasi citra, nilai *local binary pattern* tersebut

akan digunakan untuk proses *data learning* dan pengenalan citra uang kertas. Seluruh proses yang dilakukan akan dikendalikan dengan *Mini PC*.

*Local Binary Pattern* adalah metode yang dipilih dalam tugas akhir ini karena keunggulan metode ini, yaitu pengaruh iluminasi yang minimal dalam pengambilan citra tekstur digital. Sehingga ciri khas masing-masing uang akan dikenali dari tekstur yang didapatkan.

Hasil penelitian pengenalan uang kertas dengan metode *local binary pattern* diharapkan dapat menjadi pengembangan atau pertimbangan pemilihan metode dalam aplikasi *visual recognition*.

## **1.2 Perumusan Masalah**

Pada penelitian tugas akhir ini terdapat permasalahan yang harus diselesaikan, yaitu:

1. Bagaimana menentukan teknik pengambilan citra uang kertas dengan kamera agar dihasilkan citra uang kertas yang baik.
2. Bagaimana melakukan proses pengolahan citra, agar hasil pengambilan citra dari kamera dapat disempurnakan
3. Bagaimana menerapkan metoda *Local Binary Pattern* agar dapat mengenali setiap uang kertas dengan benar.
4. Pada penelitian ini uang kertas yang dideteksi adalah uang kertas asli.

## **1.3 Tujuan dan Manfaat Penelitian**

Tujuan dari penelitian ini adalah:

1. Mengetahui teknik pengambilan gambar uang kertas dengan kamera agar didapatkan citra uang kertas yang jelas.
2. Mengetahui proses pengolahan citra, agar citra yang dihasilkan kamera mempunyai kualitas yang lebih jelas dan baik.
3. Mengetahui cara membaca dan mengartikan uang kertas dengan metoda LBP.

Manfaat dari penelitian ini adalah dapat menghasilkan suatu sistem pengenalan uang kertas yang teruji dan efisien sehingga nantinya dapat diimplementasikan untuk fungsi yang lain.



## 1.4 Metodologi Penelitian

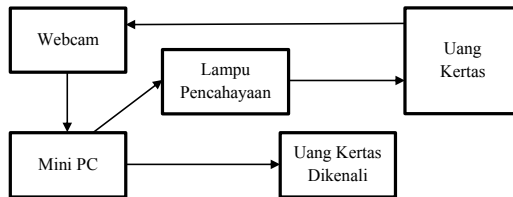
Dalam penyelesaian tugas akhir ini digunakan metodologi sebagai berikut:

### 1. Studi Literatur

Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan Tugas Akhir. Dasar teori ini dapat diambil dari buku-buku, jurnal, dan artikel-artikel di internet dan forum-forum diskusi internet yang relevan.

### 2. Perancangan Perangkat Keras

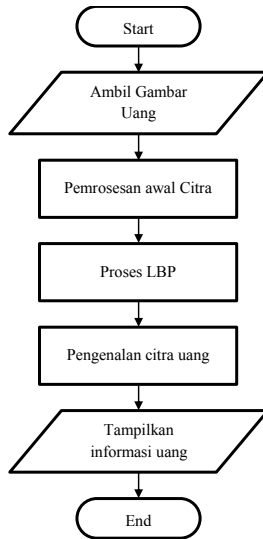
Perancangan perangkat keras pada tahap ini meliputi struktur tempat pengambilan citra uang kertas, *webcam mounting* dan struktur pemcahayaan. Semua proses akan menggunakan *Mini PC*



**Gambar 1.1** Blok diagram perancangan perangkat keras

### 3. Perancangan Perangkat Lunak

Perancangan perangkat lunak meliputi program *preprocessing* citra yang digunakan untuk mengkondisikan citra agar siap untuk tahap pengenalan tekstur menggunakan LBP, program pengenalan tekstur dengan pendekatan LBP, *data mining* setiap uang kertas yang digunakan untuk *data learning*, dan program untuk analisa klasifikasi tekstur citra menurut *data learning* yang ada.



**Gambar 1.2** Alur kerja program

4. Pengujian Alat  
 Pengujian alat dilakukan untuk menentukan akurasi dan keandalan dari sistem yang telah dirancang. Pengujian dilakukan untuk melihat apakah program dan algoritma yang digunakan dapat bekerja secara baik. Pengujian dilakukan dengan metode uji terkendali.
5. Analisis  
 Analisis dilakukan terhadap hasil dari pengujian sehingga dapat ditentukan karakteristik dari program dan algoritma yang telah digunakan. Karakteristik yang perlu diuji adalah keakuratan hasil pengenalan uang kertas dan kecepatan respon program terhadap masukan. Apabila hasil pengenalan uang kertas belum sesuai maka perlu dilakukan perancangan ulang pada sistem.
6. Penyusunan Laporan  
 Proses terakhir adalah membuat dokumentasi pelaksanaan tugas akhir yang meliputi dasar teori, proses perancangan, pembuatan, dan pengujian aplikasi.

## **1.5 Sistematika Penulisan**

Laporan tugas akhir ini terdiri dari lima bab dengan sistematika penulisan sebagai berikut:

### **Bab 1: PENDAHULUAN**

Pada bab ini meliputi latar belakang masalah yang mendasari pembuatan penelitian, perumusan masalah yang berkaitan dengan penelitian pada tugas akhir, tujuan, manfaat penelitian, metodologi penelitian yang di gunakan dalam penelitian tugas akhir, sistematika penulisan dan relevansi penelitian tugas akhir.

### **Bab 2: DASAR TEORI**

Pada bab ini meliputi dasar – dasar teori penunjang yang digunakan dalam pengerjaan tugas akhir. Dasar teori terdiri dari sample uang kertas, *mini PC*, Microsoft Visual Studio, OpenCV, pengenalan objek dan dasar teori mengenai *Local Binary Pattern*.

### **Bab 3: PERANCANGAN SISTEM**

Pada bab ini meliputi penjabran mengenai perancangan sistem yang akan dibangun seperti perancangan perangkat keras dengan menggunakan kamera dan *mini PC*. Perancangan perangkat lunak dengan menggunakan OpenCV dan Microsoft Visual Studio 2015.

### **Bab 4: PENGUKURAN DAN ANALISIS SISTEM**

Pada bab ini meliputi hasil dari rancang bangun sistem dan rancangan program yang telah di realisasikan.

### **Bab 5: PENUTUP**

Bab ini menjelaskan tentang kesimpulan dari hasil penelitian meliputi kekurangan-kekurangan pada kerja alat, sistem, algoritma dan metoda yang digunakan serta dijelaskan pula saran untuk pengembangan penelitian ke depan.

## **1.6 Relevansi**

Hasil dari penelitian tugas akhir ini diharapkan akan dapat memberi manfaat dan dampak sebagai berikut:

1. Dapat digunakan sebagai sistem untuk mengenal nominal uang kertas yang baik.

2. Hasil dari penelitian pada tugas akhir diharapkan dapat menjadi acuan pengembangan metode LBP untuk pengaplikasian dalam program lain.
3. Sebagai dasar atau acuan penelitian lebih lanjut, agar dapat dikembang menjadi lebih baik lagi.

## BAB II

### TINJAUAN PUSTAKA

#### 2.1 Uang Indonesia

Sebelum adanya uang, masyarakat Indonesia menggunakan proses barter atau *innature* untuk memenuhi kebutuhan hidup. Proses ini menyangkut dua orang yang saling membutuhkan barang tertentu dan memiliki kebutuhan yang bersifat timbul balik. Namun proses ini memiliki kekurangan karena tidak di setiap tempat dan setiap waktu terdapat dua orang yang mempunyai barang yang dibutuhkan dan mau menukarkan barang tersebut dan sulit untuk menentukan nilai barang yang akan saling ditukarkan.

Kesulitan-kesulitan inilah yang mendorong masyarakat untuk mencari solusinya, yaitu menetapkan suatu benda yang ditetapkan sebagai perantara, memiliki nilai yang pasti dan mudah untuk dibawa kemana saja. Benda tersebut yang akhirnya disebut sebagai uang.

Di Indonesia sendiri, mata uang yang digunakan adalah Rupiah. Uang Rupiah secara fisik terbagi 2 yaitu Rupiah logam dan Rupiah kertas. Pada Rupiah logam, pecahan uang terdiri dari nilai 100, 200, 500 dan 1000 Rupiah. Sedangkan pada uang kertas pecahan uang terdiri dari nilai 1000, 2000, 5000, 10.000, 20.000 dan 100.000 Rupiah seperti yang ditunjukkan pada gambar 2.1.



**Gambar 2.1** Uang Rupiah kertas

### 2.1.1 Karakteristik dan desain uang kertas Rupiah TE 2016

Pada tanggal 19 Desember 2017, Bank Indonesia meresmikan pecahan uang kertas dan uang logam baru yang diberi nama Uang NKRI Baru. Perbedaan tidak hanya berubah pada bentuk dan ukuran saja, namun tokoh Pahlawan Nasional pun berubah dari uang sebelumnya. 11 desain terbaru mata uang Rupiah yang terdiri dari tujuh pecahan uang kertas dan empat pecahan uang logam. Pada uang kertas, penggunaan tokoh nasional pada bagian muka dan pemandangan alam atau tari nusantara dari berbagai daerah pada bagian belakang bertujuan untuk menunjukkan penghormatan pada tokoh nasional dan menunjukkan kekayaan alam dan budaya di Indonesia, sedangkan pada uang logam lebih fokus pada tokoh tokoh nasional.



**Gambar 2.2** Desain depan dan belakang uang kertas Rupiah TE 2016

### 2.1.1.1 Karakteristik uang kertas Rupiah

Rupiah sebagai mata uang resmi di Indonesia memiliki ciri khas tersendiri untuk membedakan kepaluan setiap uang kertas yang beredar. Secara umum, ciri-ciri keaslian uang Rupiah dapat dikenali dari unsur pengaman yang tertanam pada bahan uang dan teknik cetak yang digunakan, yaitu :

1. **Tanda Air (Watermark) dan Electrotipe** Pada kertas uang terdapat tanda air berupa gambar yang akan terlihat apabila diterawangkan ke arah cahaya.
2. **Benang Pengaman (Security Thread)** Ditanam di tengah ketebalan kertas atau terlihat seperti dianyam sehingga tampak sebagai garis melintang dari atas ke bawah, dapat dibuat tidak memendar maupun memendar di bawah sinar ultraviolet dengan satu warna atau beberapa warna.
3. **Cetak Intaglio** Cetakan yang terasa kasar apabila diraba.
4. **Gambar Saling Isi (Rectoverso)** Pencetakan suatu ragam bentuk yang menghasilkan cetakan pada bagian muka dan belakang beradu tepat dan saling mengisi jika diterawangkan ke arah cahaya.
5. **Tinta Berubah Warna (Optical Variable Ink)** Hasil cetak mengkilap (glittering) yang berubah-ubah warnanya bila dilihat dari sudut pandang yang berbeda.
6. **Tulisan Mikro (Micro Text)** Tulisan berukuran sangat kecil yang hanya dapat dibaca dengan menggunakan kaca pembesar.
7. **Tinta Tidak Tampak (Invisible Ink)** Hasil cetak tidak kasat mata yang akan memendar di bawah sinar ultraviolet.
8. **Gambar Tersembunyi (Latent Image)** Teknik cetak dimana terdapat tulisan tersembunyi yang dapat dilihat dari sudut pandang tertentu.

### 2.1.1.2 Desain uang kertas Rupiah

Secara umum desain uang kertas rupiah TE (Tahun emisi) 2016 tidak memiliki perbedaan yang signifikan terhadap desain mata uang tahun sebelumnya. Spesifikasi uang kertas rupiah secara ukuran, warna dan gambar ditampilkan pada tabel 2.1:

PECAHAN	UKURAN (mm)	WARNA DOMINAN	GAMBAR UTAMA (depan)
100.000	151 x 65	Merah	Dr. (H.C.) Ir. Soekarno Hatta dan Dr. (H.C.) Drs. Mohammad Hatta.
50.000	149 x 65	Biru	Ir. H.Djuanda Kartawidjaja.
20.000	147 x 65	Hijau	Dr. G.S.S.J. Ratulangi
10.000	145 x 65	Ungu	Frans Kaisiepo
5000	143 x 65	Cokelat	Dr. K.H. Idham Chalid
2000	141 x 65	Abu-abu	Muhammad Husni Thamrin
1000	141 x 65	Hijau	Tjut Meutia

**'Tabel 2.1** Tabel spesifikasi uang kertas Rupiah TE 2016

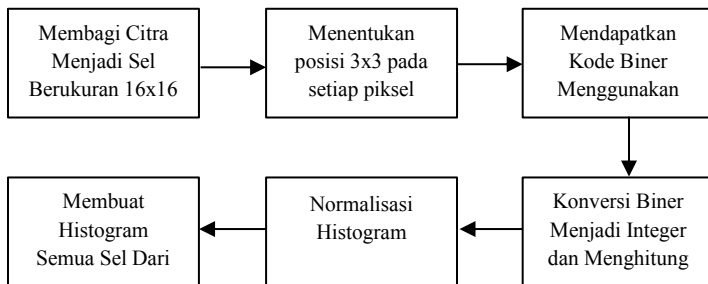
## 2.2 *Local Binary Patterns*

Local Binary Pattern adalah descriptor citra yang digunakan klasifikasi pada computer vision. Metode ini terbukti menjadi fitur yang kuat untuk klasifikasi tekstur. Penggunaan metoda *Local Binary Pattern* awalnya ditujukan untuk menganalisa tekstur citra dalam skala yang lebih kecil. Selanjutnya LBP berkembang untuk bisa diaplikasikan dalam banyak hal. Dengan menggunakan metoda LBP struktur citra didapatkan dengan cara membandingkan nilai piksel citra dengan beberapa nilai piksel disekitarnya. Keuntungan yang didapat dari metoda ini adalah toleransi terhadap iluminasi monotonik dan komputasi yang sederhana.

Dasar dari pengembangan operator LBP adalah tekstur permukaan dua-dimensi yang dibagi menjadi *local special patterns* dan *gray scale contrast*. Operator LBP dikembangkan untuk menggunakan ukuran yang berbeda-beda.



Pada LBP operasi yang dikerjakan adalah operasi yang memberikan label pada suatu piksel citra dengan suatu integer yang disebut dengan *local binary pattern code*, dimana label tersebut adalah hasil dari *encode* struktur lokal piksel-piksel yang berada di sekitar piksel citra tersebut. Operasi LBP paling sederhana dapat dilakukan dengan cara komparasi setiap delapan piksel tetangga disekitarnya dalam jangkauan 3x3 piksel dengan mengurangkan setiap nilai piksel tetangga dengan nilai piksel yang berada di tengah (menjadi acuan nilai *threshold* untuk piksel disekitarnya), piksel bernilai positif akan diberi nilai 1 sedangkan piksel yang bernilai negatif akan diberi nilai 0. Proses tersebut akan menghasilkan 8 buah angka biner dengan menyatukan hasil pengkodean pada 8 piksel tetangga mengikuti arah jarum jam dimulai dari piksel kiri atas dan nilai desimal dari delapan digit biner. Proses LBP ini yang digunakan untuk label LBP pada piksel tengah dalam jangkauan 3x3 piksel citra.

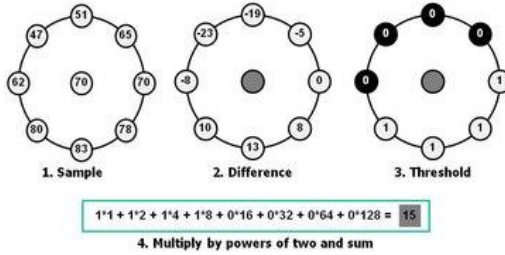


**Gambar 2.3** Flowchart ekstraksi fitur LBP

Operasi LBP dikembangkan untuk digunakan area yang bervariasi. Menggunakan area berbentuk lingkaran dan nilai interpolasi bilinear pada koordinat piksel *non-integer* memungkinkan semua radius dan jumlah piksel didalam area. Varian *grayscale* pada area lokal dapat digunakan sebagai ukuran kontras pelengkap. Pada gambar dibawah ini, notasi (P,R) akan digunakan sebagai piksel area dimana nilai sampling rata-rata P pada lingkaran dengan radius R. lihat pada gambar 2.3 sebagai contoh dari komputasi LBP

The value of the LBP code of a pixel  $(x_c, y_c)$  is given by:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p \quad s(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$



**Gambar 2.4** Contoh komputasi metode LBP

Notasi dibawah ini menunjukkan penggunaan dari LBP operator  $LBP_{P,R}^{U2}$ . Notasi ini merepresentasikan penggunaan menggunakan operator pada daerah  $(P,R)$ .  $U2$  menunjukkan hanya penggunaan pola yang seragam dan *labelling* pada sisa pola dengan *label* tunggal. Setelah citra LBP yang diberi label image  $f_1(x,y)$  didapatkan, histogram daripada LBP dapat didefinisikan sebagai berikut

$$H_i = \sum_{x,y} I\{f_1(x,y) = i\}, i = 0, \dots, n - 1, \quad (2.1)$$

Dimana  $n$  adalah jumlah label yang berbeda yang dihasilkan oleh operator LBP, dan  $I\{A\}$  adalah 1 apabila  $A$  benar, dan 0 apabila  $A$  salah.

Ketika histogram bagian citra dibandingkan dengan ukuran yang berbeda, maka histogram tersebut mesti diubah untuk mendapatkan deskripsi koheren.

$$N_i = \frac{H_i}{\sum_{j=0}^{n-1} H_j} \quad (2.2)$$

Pengembangan LBP memungkinkan kita untuk menentukan piksel sekitar lebih dari 8 dan memiliki radius yang bervariasi sesuai kebutuhan

dan hasil yang diinginkan. Secara umum nilai LBP piksel ( $X_c, Y_c$ ) pada citra dapat dirumuskan sebagai berikut:

$$LBP_{p,r}(X_c, Y_c) = \sum_{p=0}^{p-1} s(ip - ic) 2^p \quad (2.3)$$

Dimana,

$p$  = jumlah *pixel* tetangga

$r$  = jarak radius *pixel* tetangga

$X_c$  = koordinat ( $x$ ) *pixel* tengah

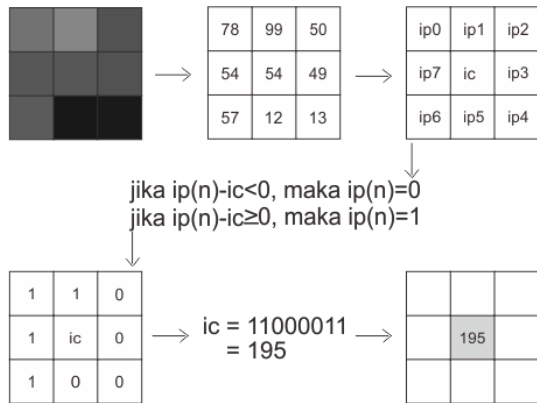
$Y_c$  = koordinat ( $y$ ) *pixel* tengah

$ip$  = nilai *grayscale pixel* tetangga atau *neighbour*

$ic$  = nilai *grayscale pixel* tengah, dan fungsi  $s(x)$  didefinisikan sebagai:

$$s(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \quad (2.4)$$

Yang menunjukkan bahwa jika hasil pengurangan bernilai negatif maka nilai *pixel* tetangga dikodekan dengan nilai 0 dan jika hasil pengurangan bernilai 0 atau positif maka nilai *pixel* tetangga dikodekan dengan nilai 1.

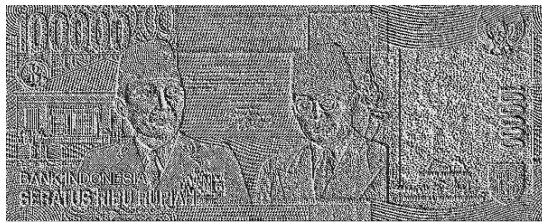


**Gambar 2.5** Operasi *Local Binary Pattern* sederhana.

Citra yang telah melalui proses LBP, memiliki channel sebanyak 1 channel. Hasil perubahan citra LBP dapat dilihat pada gambar 2.7. Pada citra tersebut terdapat warna colorspace grayscale, dan pada citra yang dihasilkan terdapat beberapa feature yang nanti akan dapat diambil nilai histogramnya. Untuk mendapatkan hasil histogram yang terbaik, maka citra tersebut akan melalui proses segmentasi untuk menambah histogram dari citra dan menambah tekstur baru pada citra.



(a)



(b)

**Gambar 2.6** Citra proses LBP.

(a) citra sebelum proses LBP. (b) citra setelah proses LBP.

Limitasi yang dimiliki operasi LBP sederhana adalah ukuran  $3 \times 3$  *pixel* tetangga tidak dapat menghasilkan sifat dominan pada struktur berskala besar. Untuk itu pengaturan ukuran radius tetangga dapat diatur sehingga dapat dihasilkan sifat dominan pada struktur berskala besar.

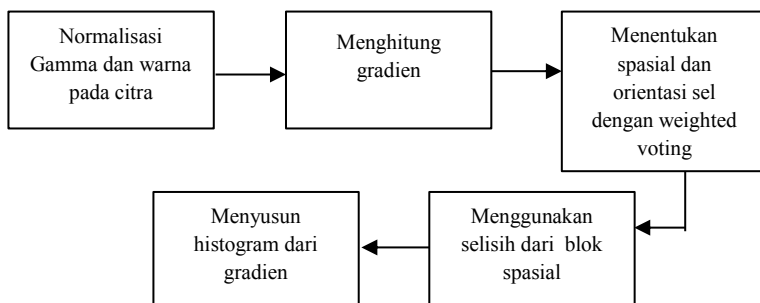
### 2.2.1 Keunggulan Metoda *Local Binary Patterns*

Keunggulan utama dari LBP adalah kemampuan untuk mendapatkan detil yang optimal pada citra. Metode LBP terbukti sangat

berpengaruh pada proses *facial recognition* baik dari segi waktu komputasi serta akurasi dari proses tersebut. Kemampuan komputasi yang singkat berpengaruh dengan jumlah segmentasi yang digunakan pada operasi LBP, semakin banyak segmentasi maka akan mempengaruhi waktu komputasi semakin lama, sebaliknya semakin sedikit segmentasi maka waktu komputasi akan semakin singkat. Keunggulan LBP dapat dikombinasikan dengan metode lain seperti gabor dan SIFT.

### 2.2.2 Perbandingan Metoda LBP dengan HOG

*Histogram of Oriented Gradient* (HOG) mendapatkan fitur citra dengan menghitung kemungkinan dari orientasi gradasi. HOG tradisional membagi citra menjadi beberapa sel yang kemudian akan menghitung orientasi gradasi dalam bentuk histogram. HOG sudah diaplikasikan secara luas dalam pengenalan objek seperti *facial recognition*.



**Gambar 2.7** Contoh komputasi metode HOG

Untuk mengklasifikasi *dataset*, 3 model yang dibentuk menggunakan ekstraksi fitur terhadap algoritma yang dipilih. Ukuran fitur ditentukan menyesuaikan dengan jumlah *bins* untuk setiap operator. Hasil yang didapat adalah berupa persentasi akurasi citra.

Model Ekstraksi Fitur	Ukuran Fitur	Akurasi
LBP	1182	90.52%
HOG	1224	34.37%

**Tabel 2.2** Perbandingan nilai akurasi

Akurasi yang didapatkan fitur ekstraksi HOG dapat ditingkatkan dengan mengganti parameter pada algoritma, namun akan membutuhkan banyak *resource* untuk melakukan eksperimen tersebut. Fitur vektor LBP yang dibuat secara manual dapat menghasilkan akurasi lebih bagus daripada operasi HOG. Dengan implementasi sederhana, metode LBP dapat melampaui *deep features* dengan hasil yang diluar dugaan, mengingat sedikitnya dibutuhkan *resources* dan perancangan program yang dirancang.

## 2.3 Mini PC

*Mini PC* merupakan sebuah perangkat keras yang terdiri dari *processor*, *motherboard*, *Video Graphic Array*, dan *RAM*. *Mini PC* memiliki cara kerja dan proses yang sama dengan computer pada umumnya, terdiri dari 3 (tiga) komponen utama, komponen pertama adalah *Hardware*, komponen kedua adalah *Software*, komponen ketiga adalah *Brainware*. Komponen ini merupakan bagian terpenting untuk dapat mengoperasikan *mini PC*.

Komponen pertama pada *mini PC* adalah *hardware*, dimana terdiri dari *motherboard* untuk meletakkan semua perangkat keras seperti *processor*, *VGA Card*, dan *RAM*. Pada *mini PC* jenis *motherboard* yang digunakan adalah jenis *Mini Motherboard*, hal ini dikarenakan ukuran sebuah *mini PC* yang tidak terlalu besar dan pada *mini PC* hanya dapat melakukan beberapa proses dengan menggunakan sistem yang kecil sehingga *motherboard* pada *mini PC* tidak memiliki *clock* yang tinggi dan *socket* yang banyak. *Processor* yang digunakan dapat menggunakan *processor* pada umumnya seperti Intel core i3 ataupun Intel core i7. *Mini PC* dengan dimensi dan ukuran yang lebih kecil dari *PC* memudahkan untuk dibawa kemana saja.

*Software* yang digunakan pada *mini PC* dapat menggunakan *Windows-based operation* atau dengan menggunakan *Linux-based operation*. Pada *mini PC* digunakan *Windows-based operation*, dikarenakan program yang digunakan untuk menjalankan program adalah Microsoft Visual Studio 2015. *Minimum system requirement* pada Visual Studio 2015 adalah sebagai berikut :

1. 1.6 GHZ *Processor*.
2. 1GB *RAM*.
3. Kapasitas minimal *hard disk* adalah 10GB.

4. 600MB *space* pada *hard disk*.
5. DirectX 9 yang dapat memproses citra dengan ukuran minimal 1024 x 768*pixel*.

Dengan menggunakan Microsoft Visual Studio 2015 maka akan menggunakan bahasa pemrograman dengan C / C++ API. Pada Microsoft Visual Studio diperlukan sebuah *library* untuk dapat memproses sebuah citra secara digital. *Library* yang digunakan adalah OpenCV 3.3. Untuk dapat menggunakan *library* dari Opencv diperlukan proses *compailing* pada Visual Studio.

Pada komponen penyusun ketiga adalah *brainware*, merupakan pengguna dari *mini PC* atau yang disebut sebagai *user*. Jika *user* tidak memasukan atau menjalankan sebuah program pada *mini PC*, maka *mini PC* tidak dapat berjalan atau beroperasi dengan sendirinya. *Brainware* dapat melakukan proses *Input / Output*, *input* dilakukan oleh *user* melalui sebuah *hardware* seperti *keyboard*, *mouse*, ataupun kamera. Setelah *mini PC* mendapatkan sebuah *input* dari *user*, maka nilai *input* akan diubah menjadi sebuah *command* pada program *operation* dan akan di proses di *Processor*. Pada tahap selanjutnya adalah *command* akan diproses kedalam sebuah *software*, pada *software* akan dilakukan pengolahan data dari *command* sehingga didapatkan sebuah *output*. *Output* akan dikeluarkan melalui *hardware* yang memiliki fungsi pengingeraan pada *user* seperti *monitor* untuk memperlihatkan proses yang sedang berjalan atau aplikasi yang sedang bekerja, *speaker* untuk *user* dapat mendengarkan sebuah proses yang menggunakan suara.

## 2.4 OpenCV

*OpenCV* adalah sebuah pustaka perangkat lunak yang bertujuan untuk pengolahan citra dinamis secara real-time. Program ini tidak berbayar dan berlisensi BSD. OpenCV bersifat lintas *platform* yang berarti dapat dijalankan dalam sistem operasi seperti *Linux*, *Windows*, dan *Mac OS X*. *OpenCV* dikembangkan menggunakan bahasa pemrograman C dan C++, namun *OpenCV* dapat digunakan menggunakan bahasa pemrograman lain selain C atau C++ seperti Python, Ruby, Matlab dan bahasa pemrograman yang lain.

Untuk keperluan penelitian tugas akhir ini akan dijelaskan beberapa fungsi yang digunakan dan memiliki relevansi dengan penelitian tugas akhir yang dikerjakan, diantaranya adalah fungsi untuk mengambil citra

dari *file* atau kamera, *thresholding*, *morphology*, *Region of Interest*, dan deteksi *countour*. Salah satu struktur citra yang digunakan untuk seluruh operasi yang dilakukan oleh fungsi-fungsi yang dimiliki openCV adalah struktur citra `IplImage` dan `Mat`, pada struktur `IplImage` atau dapat menggunakan operasi `Mat` data citra akan dialokasikan secara otomatis pada memori dan akan me-*return* nilai pada data citra jika suatu *pointer* digunakan untuk mengakses data tersebut. Struktur `IplImage` suatu citra harus dideklarasikan pada awal program sebelum dapat digunakan untuk menampung data citra, contoh penulisan deklarasi `IplImage` seperti berikut:

```
IplImage* image;  
Mat image;
```

Di mana *image* adalah nama *pointer* yang digunakan untuk menunjuk alamat data citra pada memori. Struktur `IplImage` pada akhir program harus di-*release* agar memori yang digunakan untuk data citra pada `IplImage` struktur dapat dialokasikan untuk keperluan lainnya. Untuk mereset struktur `IplImage` atau `Mat` dapat menggunakan fungsi:

```
cvReleaseImage(&citra);
```

#### 2.4.1 Akses citra dari *file* dan kamera

Fungsi yang digunakan untuk mengakses citra pada *file* yang telah ada adalah fungsi `cvLoadImage()` pada C dan `cv::imread()` untuk menggunakan C++, yang memiliki struktur berikut:

```
Untuk C:  
IplImage* cvLoadImage(Const char* filename,  
Int iscolor = CV_LOAD_IMAGE_COLOR);
```

```
Untuk C++ :  
Mat imread(const string&filename, int pointers=1);
```

Terdapat dua bagian pada fungsi `cvLoadImage` dan `Mat imread` yaitu *filename* dan *iscolor* atau int *pointers*, bagian *filename* adalah nama *file* dari citra yang ingin dimuat, pengisiannya dengan menyertakan ekstensi format citra yang didahului dan diakhiri dengan tanda petik



ganda, contohnya seperti (“image.bmp”). Bagian kedua yaitu *iscolor*, bagian ini digunakan untuk mengalokasikan warna pada citra yang akan dimuat dengan mengisi antara CV\_LOAD\_IMAGE\_COLOR untuk mengambil warna citra dalam warna BGR dengan menggunakan 3 *channels* dengan kedalaman warna 8bit, untuk CV\_LOAD\_IMAGE\_GRAYSCALE maka citra yang dimuat akan diubah menjadi citra dengan 1 *channel* dengan kedalaman warna 1bit, sedangkan jika menggunakan CV\_LOAD\_IMAGE\_ANYCOLOR memori akan dialokasikan sesuai dengan jenis citra yang ingin dimuat dan kedalaman warna 8bit, dan jika menggunakan CV\_LOAD\_IMAGE\_UNCHANGED maka memori akan dialokasikan untuk citra dengan *channel* dan kedalaman warna sesuai dengan format awal pada citra.

Terdapat *pointer* yang bersifat opsional pada fungsi cvLoadImage() yaitu *pointer* CV\_LOAD\_IMAGE\_ANYDEPTH, jika *pointer* ini digunakan maka kedalaman warna citra yang dimuat pada struktur IplImage akan menyesuaikan dengan kedalaman warna citra awal. *Pointer* opsional ini dituliskan setelah *pointer iscolor* jika diinginkan.

Untuk fungsi menyimpan citra maka digunakan fungsi cvSaveImage() pada C dan Mat imwrite() pada C++, yaitu fungsi yang digunakan untuk menyimpan citra ke dalam sebuah *file* dari struktur citra IplImage atau Mat. Dengan struktur fungsi sebagai berikut:

Untuk C:

```
Int cvSaveImage(Cont char* filename, Conts CvArr* image );
```

Untuk C++:

```
Bool imwrite(const strin&filename, InputArray img, constvector<int>&params=vector<int>());
```

Parameter yang terdapat adalah *filename* adalah nama citra yang ingin disimpan pada *file* penulisannya seperti pada parameter *filename* fungsi cvLoadImage() atau imread(), dan CvArr\* image atau InputArray img adalah nama *pointer* struktur IplImage atau Mat yang ingin disimpan kedalam sebuah file.

Terdapat beberapa fungsi yang dapat digunakan untuk mengakses citra yang berasal dari kamera atau *input* citra dari kamera, salah satunya `cvCaptureFromCAM(int ID CAM)` untuk C dan `VideoCapture::VideoCapture(int device)`, fungsi ini digunakan untuk mengaktifkan dan mengambil citra dari kamera yang nanti akan disimpan citranya kedalam sebuah *frame*, hanya terdapat satu bagian pada fungsi tersebut yaitu bagian ID CAM atau device, untuk menentukan kamera yang akan digunakan. Berikut contoh program yang digunakan apabila akan mengambil citra dari kamera dan menyimpannya ke sebuah *file*:

```
Untuk C++:  
Mat image;  
Mat capture;  
VideoCapture capture;  
capture.open(0);  
while(1)  
{  
    capture.read(capture);  
    image = capture;  
    imwrite("Image.jpg", image);  
    return 0;  
}
```

Tugas yang dikerjakan contoh program diatas adalah menyiapkan struktur `IplImage` dan `Mat` dengan nama *pointer* `image`, baris kedua menyiapkan struktur `cvCapture` atau `capture.read` dengan nama *pointer* `capture`, baris ke tiga mengaktifkan kamera, mengambil data citra dari kamera dan mengalokasikan pada struktur `capture`, baris ke empat membaca data citra pada `capture` kemudian mengirim data tersebut ke struktur `IplImage` dan `Mat image`, baris ke lima menyimpan data citra pada `image` ke dalam *file* dan dua baris selanjutnya bertugas untuk merelease struktur `cvCapture` dan `IplImage` secara berurutan. Untuk C++ tidak membutuhkan *release* karena dibutuhkan pengambilan gambar dalam sebuah loop

### 2.4.2 *Grayscale conversion*

Citra berwarna terdiri dari 3 *channel* yaitu B(Blue), G(Green) dan R(Red), pada citra grayscale hanya terdapat satu *channel* warna dan

nilai tiap pikselnya dari *range* 0 (Putih) – 255 (Hitam) untuk 8 bit citra. Secara teori citra *grayscale* didapatkan dengan mencari nilai rata-rata setiap piksel pada ketiga *channel* dari citra berwarna.

Pada *OpenCV* terdapat beberapa fungsi untuk mengkonversikan citra berwarna ke citra *grayscale*, diantaranya menggunakan fungsi `cvLoadImage()` dengan *pointer iscolor* diisi dengan `CV_LOAD_IMAGE_GRAYSCALE`, fungsi tersebut akan mengkonversikan semua jenis citra menjadi citra *grayscale* 8 bit. Fungsi lain yang bisa digunakan adalah fungsi `CV_BGR2GRAY` dengan struktur fungsi:

```
cvtColor(src, dst, CV_BGR2GRAY);
```

Src adalah pointer Mat citra yang akan dikonversikan ke *greyscale*, dst adalah pointer Mat citra yang dibuat untuk menyimpan nilai hasil konversi yang diinginkan.

### 2.4.3 *Thresholding*

*Thresholding* adalah fungsi pada *OpenCV* yang digunakan untuk mengeliminasi piksel-piksel yang berada pada nilai dibawah atau diatas batas ambang tertentu, fungsi ini sangat berguna untuk menyingkirkan piksel-piksel citra yang tidak diinginkan atau *background*. Ada dua macam fungsi *threshold* pada *OpenCV* yaitu `cvThreshold()` dan `cvAdaptiveThreshold()`, dua fungsi tersebut memiliki perbedaan dalam penentuan nilai *threshold*, fungsi *threshold* hanya bisa diaplikasikan pada citra yang memiliki satu *channel* atau *greyscale*.

Pengaturan nilai ambang batas *threshold* pada fungsi `cvThreshold()` bisa dilakukan dengan memberikan nilai integer pada salah satu *pointer* pada fungsi. Keuntungan dari fungsi ini adalah nilai *threshold* yang tidak berubah pada seluruh nilai piksel citra, kelemahannya adalah nilai ambang batas yang tidak dapat beradaptasi pada tingkat iluminasi citra sehingga proses pembeda antara objek dan *background* pada citra tidak maksimal untuk citra yang tingkat pencahayaannya tidak merata. Struktur pada fungsi `cvThreshold` adalah sebagai berikut:

```
double cvThreshold(
    CvArr* src,
    CvArr* dst,
    double threshold,
    double max_value,
    int threshold_type
);
```

Pada fungsi `cvThreshold` terdapat lima *pointer*, yaitu *pointer* `src` yang merupakan *pointer* berisi nama *pointer* struktur `IplImage` citra yang akan dilakukan operasi `threshold`, kemudian *pointer* `dst` yang berisi nama *pointer* struktur `IplImage` citra yang akan ditampung hasil citra operasi `threshold`, kemudian *pointer* `threshold` yang merupakan *pointer* berisi nilai ambang batas yang akan diterapkan pada citra, kemudian *pointer* `max_value` yang merupakan *pointer* yang berisi nilai yang diberikan pada citra hasil selain nilai 0 yang bergantung dari tipe `threshold` yang dipilih pada *pointer* terakhir, dan yang terakhir adalah *pointer* `threshold_type` yang berisikan nilai integer atau nama tipe `threshold` yang ingin digunakan.

#### 2.4.4 Morphology

*Morphology* adalah proses transformasi citra yang terdiri dari dua operasi dasar, yaitu *dilate* dan *erode*. *Dilate* adalah operasi konvolusi antar suatu citra dengan sebuah *kernel* yang memiliki titik tengah. Operasi *dilate* dilakukan dengan cara *scanning* seluruh piksel pada citra dengan *kernel*, yang kemudian menggantikan nilai piksel yang telah di *scanning* oleh *kernel*, sehingga operasi *dilate* menghasilkan daerah terang yang lebih luas. Operasi *dilate* dapat diterapkan pada citra untuk menghilangkan atau mereduksi *noise*. Operasi *erode* adalah operasi yang berkebalikan dengan operasi *dilate*, hanya nilai piksel citra yang di *scanning* titik tengah *kernel* digantikan dengan nilai minimum piksel disekitar titik tengah *kernel*, sehingga citra yang dihasilkan lebih banyak warna gelap. Operasi *morphology* ini sangat efektif untuk menghilangkan *noise* yang tidak diinginkan pada citra. Struktur fungsi *dilate* dan *erode* adalah sebagai berikut:

```

void cvDilate(
    IplImage* src,
    IplImage* dst,
    IplConvKernel* B = NULL,
    Int iterations = 1
);

void cvErode(
    IplImage* src,
    IplImage* dst,
    IplConvKernel* B = NULL,
    Int iteration = 1
);

```

Masing-masing fungsi `cvDilate()` dan `cvErode()` memiliki 4 *pointer* yang terdiri dari *pointer* pertama yaitu nama *pointer* struktur `IplImage` citra *operand*, *pointer* kedua adalah *pointer* yang berisi *pointer* struktur menyimpan `IplImage` hasil operasi, *pointer* yang ketiga atau *pointer* B adalah *pointer* yang berisi nama *pointer* struktur *kernel* yang digunakan, jika diisi dengan `NULL` (0) maka *kernel* yang digunakan adalah konfigurasi standar yang memiliki luas 3x3 piksel. Dan yang terakhir *Pointer* keempat adalah *pointer* yang berisi nilai integer yang memberikan instruksi berapa kali iterasi yang diinginkan, jika tidak diisi maka konfigurasi standar adalah nilai 1.

#### 2.4.5 *Contour*

*Contour* adalah area yang memiliki nilai piksel sama dengan nilai piksel disekitarnya, pada *OpenCV* terdapat fungsi untuk menemukan *contour* pada citra, fungsi tersebut adalah `cvFindContours()` fungsi tersebut akan menemukan nilai integer jumlah *contour* yang ada pada citra, fungsi `cvFindContours()` memiliki struktur fungsi:

```

int cvFindContours(
    IplImage* img,
    CvMemStorage* storage,

```

```

CvSeq** firstContour,
int headerSize = sizeof(CvContour),
CvContourRetrievalMode mode = CV_RETR_LIST,
CvChainApproxMethod method = CV_CHAIN_APPROX_
SIMPLE
);

```

Fungsi `cvFindContours()` terdiri dari 6 *pointer* yang pertama adalah `img`, *pointer* `img` adalah *pointer* yang berisi *pointer* struktur `IplImage` citra yang akan ditemukan jumlah *contour*-nya, citra yang digunakan harus memiliki kedalaman warna 8 bit dan satu *channel*. *Pointer* kedua adalah *storage* yang merupakan struktur `CvMemStorage` yang digunakan untuk mengalokasikan rekaman *contour* yang ditemukan pada citra, struktur tersebut harus dialokasikan menggunakan fungsi `cvCreateMemStorage()`. *Pointer* yang ketiga adalah `firstContour` yang merupakan *pointer* yang berisi nama *pointer* struktur `CvSeq` yang merupakan struktur yang mengalokasikan memori yang digunakan menyimpan *sequence*, struktur `CvSeq` harus dideklarasikan terlebih dahulu pada bagian awal program. *Pointer* yang ke empat adalah *pointer* `headerSize` yang merupakan *pointer* yang berisi informasi ukuran objek yang akan alokasikan, *pointer* ini dapat diisi dengan `sizeof(CvContour)`. *Pointer* yang ke lima adalah `mode` yang merupakan *pointer* yang berisi pilihan susunan yang digunakan untuk menyusun hasil *contour* yang ditemukan pada citra, terdapat empat pilihan yaitu `CV_RETR_EXTERNAL`, `CV_RETR_LIST`, `CV_RETR_CCOMP` dan `CV_RETR_TREE`. *Pointer* yang terakhir adalah *pointer* `method` yang berisi pilihan metode yang ingin digunakan, terdapat 5 pilihan metode yang dapat dipilih yaitu `CV_CHAIN_CODE`, `CV_CHAIN_APPROX_NONE`, `CV_CHAIN_APPROX_SIMPLE`, `CV_CHAIN_APPROX_TC89_L1` atau `CV_CHAIN_APPROX_TC89_KCOS` dan `CV_LINK_RUNS`.

#### 2.4.6 *Region Of Interest*

*Region Of Interest* atau ROI adalah *subset* suatu citra atau *dataset* yang dikenali untuk tujuan tertentu. *Dataset* yang dimaksud bisa berbentuk *waveform* atau *dataset* satu dimensi, citra atau *dataset* dua dimensi dan *volume* atau *dataset* tiga dimensi. Pada *OpenCV* tidak terdapat fungsi khusus untuk memanggil ROI, tetapi kita bisa menggunakan beberapa fungsi dasar seperti `cv::Rect rect` untuk

menentukan bahwa ROI akan berbentuk sebuah kotak, pada fungsi ini terdapat satu parameter yaitu `rect` dimana parameter ini dapat diisi dengan `Rect(koordinat x, koordinat y, width, height)`. Fungsi pengambilan ROI memiliki struktur:

```
Untuk C++:  
Mat image;  
Mat Image ROI;  
Rect ROI = rect(x,y,Size());  
Image ROI = image(ROI);
```

Fungsi *OpenCV* membutuhkan ukuran citra atau ukuran ROI dari *pointer* sumber dan *pointer* tujuan memiliki ukuran yang sama. Namun dengan *Intel Image Processing* memproses simpangan area antara citra sumber dan citra destinasi memungkinkan penggunaan ukuran yang berbeda.

*Halaman ini sengaja dikosongkan*



## **BAB III**

### **PERANCANGAN SISTEM**

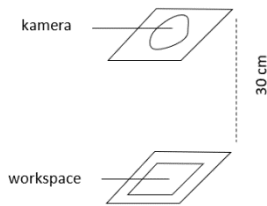
Perancangan sistem pada tugas akhir ini meliputi 2 (dua) bagian utama penyusunnya, yaitu perangkat keras dan perangkat lunak. Pada perangkat keras menggunakan 2 (dua) komponen utama yaitu *webcam* untuk mengambil citra dan *mini PC* untuk melakukan pemrosesan citra digital agar dapat dikenali. Untuk komponen pendukung akan terdiri dari kerangka pengambilan gambar dan pencahayaan.

Pada sistem perangkat lunak digunakan Microsoft Visual Studio 2015 untuk pemrogramannya dan OpenCV sebagai *library* untuk memproses citra uang kertas. Pada sistem di perangkat lunak terdapat beberapa bagian yaitu prapemrosesan yang terdiri dari *grayscale conversion*, *thresholding*, *morphology* dan *Region of Interest* untuk mendapatkan citra uang kertas. Sedangkan untuk mengenali citra uang kertas digunakan program *Local Binary Pattern*, komparasi *histogram* dengan *histogram data learning*. Program deteksi objek dan pengenalan akan ditampilkan dalam *Graphical User Interface* atau GUI. Dari keseluruhan sistem ini dibangun untuk dapat mengenali citra uang kertas.

#### **3.1 Perancangan perangkat keras**

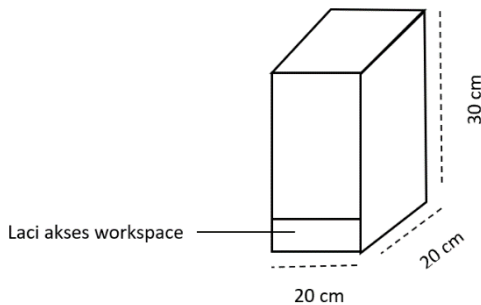
Pada perancangan perangkat keras, terdapat 2 (dua) komponen utama untuk menjalankan proses dari sistem ini. Pengambilan citra uang kertas menggunakan *webcam* yang mengambil gambar secara *one-time*, sedangkan pemrosesan citra dengan menggunakan *mini PC*.

Pengambilan citra uang kertas akan dilakukan menggunakan kamera dengan konfigurasi vertikal dengan jarak 30 cm dari uang kertas yang diuji. Agar mendapatkan hasil citra yang baik posisi kamera akan menghadap kebawah disejajarkan dengan uang kertas yang diuji yang diletakkan pada bagian bawah (*workspace*).



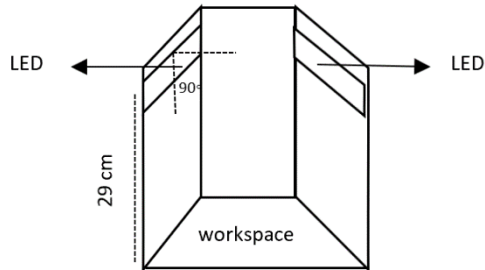
**Gambar 3.1** Jarak dan posisi kamera terhadap objek.

Kerangka yang digunakan memiliki dimensi sebesar 20x20x30 cm berbentuk kotak persegi Panjang dengan laci untuk mengakses *workspace* pada bagian depan bawah. Keseluruhan kerangka didominasi warna hitam untuk pemisahan cahaya yang bagus.



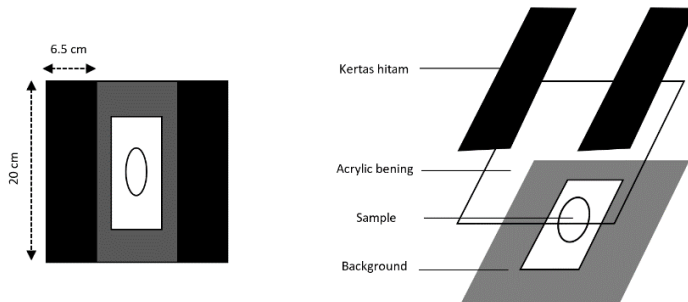
**Gambar 3.2** Dimensi kerangka alat

Teknik pencahayaan yang bagus diperlukan pada saat pengambilan citra uang kertas agar tekstur dapat teridentifikasi dengan baik, pencahayaan yang bagus akan memudahkan prapemrosesan pada pengolahan citra. Pencahayaan dilakukan menggunakan LED yang dipasangkan pada sisi kiri dan kanan dengan konfigurasi horizontal sejajar dengan Panjang uang kertas yang akan diuji dengan sudut  $90^\circ$  terhadap permukaan objek dan berjarak 29.5 cm dari *workspace*. Intensitas pencahayaan dapat diatur dengan modul LED *prebuilt* yang diberi sumber dari *mini PC*.



**Gambar 3.3** Posisi dan jarak LED terhadap objek.

Desain *workspace* yang berada di dalam alat terdiri dari lembaran *background* berwarna hitam, *acrylic* bening dan kertas hitam di sisi kiri dan kanan *workspace* sejajar dengan uang kertas yang akan diuji. Penggunaan *acrylic* bening bertujuan agar uang kertas yang akan diuji berbentuk datar yang memungkinkan proses prapemrosesan mendeteksi bentuk persegi Panjang dari uang kertas yang akan diuji. Penggunaan kertas hitam di sisi kiri dan kanan dari *workspace* bertujuan untuk mereduksi *glare acrylic* bening yang berasal dari LED pencahayaan diatas.



**Gambar 3.4** Struktur *workspace*

*Acrylic* bening yang digunakan berukuran sama dengan ukuran keseluruhan *workspace* yaitu 20x20 cm. Untuk kertas hitam digunakan ukuran 6.5x20 cm untuk masing-masing kiri dan kanan pada *workspace*.



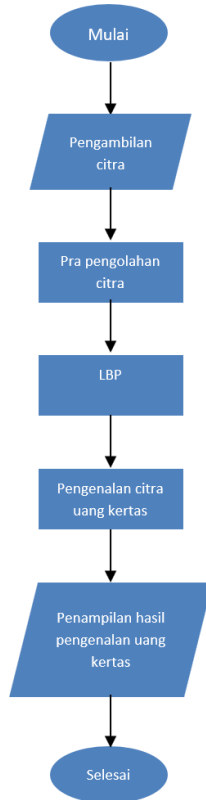
**Gambar 3.5** Dimensi dan bagian kerangka alat.

### **3.2 Perancangan perangkat lunak**

Pada *mini PC* system operasi yang digunakan adalah *Microsoft Windows* keluaran terbaru. Bahasa pemrograman yang digunakan untuk keseluruhan program adalah C++ dengan library *OpenCV*.

Perancangan sistem perangkat lunak dimulai dari penulisan program untuk pengambilan citra menggunakan kamera. Hasil pengambilan citra akan berupa *fixed image* yang akan diteruskan menuju program. Hasil citra tersebut akan dilakukan pre-pemrosesan citra agar didapatkan citra yang siap untuk proses segmentasi kemudian penerapan operasi LBP untuk mendapatkan nilai setiap segmentasi dan yang terakhir adalah pengenalan segmentasi. Proses pra-pemrosesan citra terdiri dari *greyscale conversion*, *thresholding*, *morphology* dan penentuan *Region of Interest* dari citra yang siap untuk diproses segmentasi. Selanjutnya pada proses segmentasi program akan mensegmentasi setiap bagian dari citra yang kertas agar selanjutnya setiap segmen akan dicari nilai *histogram* nya untuk menjadi nilai citra yang kertas tersebut. Proses segmentasi dilakukan dengan proses konversi *greyscale* terlebih dahulu untuk kemudian di *threshold*. Setelah proses *threshold* akan dilakukan proses *morphology* untuk mendeteksi nilai  $x$  dan  $y$  pada gambar, hasil dari proses ini akan menentukan *Region of Interest* yang kertas tersebut yang kemudian akan diproses segmentasi. Setelah proses segmentasi selanjutnya dilakukan proses LBP pada setiap segmentasi pada kertas, yang kemudian akan dicari nilai *histogram* pada setiap segmen

untuk menentukan nilai karakteristik dari uang kertas tersebut. Langkah terakhir adalah komparasi dari nilai *histogram* yang didapat dari *data learning* dan nilai *histogram* dari uang kertas yang diuji.



**Gambar 3.6** Diagram alur program pengenalan uang kertas

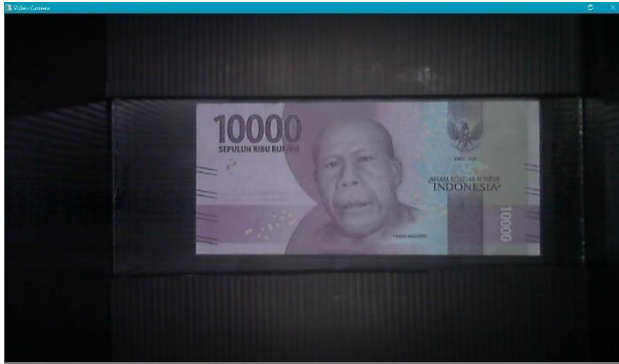
### 3.2.1 Pengambilan citra

Pengambilan citra menggunakan kamera dengan posisi vertikal menghadap kebawah yang diberi cahaya LED, kamera terhubung

langsung dengan *mini PC* dengan perintah fungsi *OpenCV* dengan struktur

```
VideoCapture capture;  
capture.open(0);
```

*pointer* (0) menunjukkan kamera mana yang digunakan pada program. Memori yang digunakan untuk pemanggilan dinamakan *frame* untuk menyimpan citra yang didapatkan dengan fungsi `::Mat frame`. Citra *frame* akan disimpan dalam memori untuk disiapkan pemanggilan menuju prapemrosesan selanjutnya



**Gambar 3.7** Hasil pengambilan citra menggunakan kamera

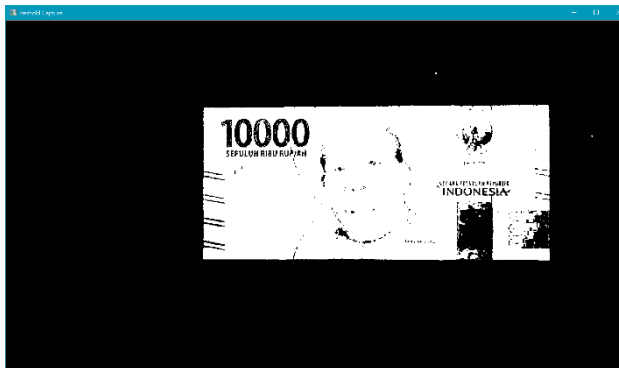
### 3.2.2 Thresholding dan morphology

Untuk membedakan antara objek dan *background* citra, digunakan operasi *thresholding*, yaitu memberikan nilai tertentu pada setiap piksel citra sesuai dengan nilai *threshold* yang diberikan. Pada citra dilakukan proses *threshold* dengan nilai yang dapat diubah-ubah sesuai hasil yang paling bagus didapat. Fungsi *threshold* yang digunakan ditunjukkan pada struktur berikut

```
threshold(cap_gray, cap_thres, 90, 255, 0);
```

Fungsi utama adalah *threshold* diikuti *pointer* sumber citra yaitu *cap\_gray*, *pointer* destinasi citra yaitu *cap\_thres* dan nilai *threshold* yang

dapat diubah-ubah, hasil pengambilan gambar *threshold* dapat dilihat pada gambar 3.7 Pada nilai *threshold*, angka pertama yaitu 90 adalah nilai *thresh*. Nilai *thresh* digunakan sebagai acuan perbandingan nilai piksel sumber citra dengan nilai *thresh* itu sendiri. Apabila nilai piksel citra sumber lebih besar daripada nilai *thresh* maka citra tujuan akan dirubah menjadi nilai *maxValue*, selain itu akan dirubah menjadi 0. Nilai *maxValue* sendiri diatur pada angka 255 yaitu nilai maksimum piksel. 0 berarti hitam, sedangkan 255 adalah putih. Nilai terakhir yang diatur adalah tipe *threshold*, tipe *threshold* yang digunakan adalah *threshold binary* yaitu apabila nilai piksel citra sumber melebihi nilai *thresh* maka nilai piksel pada citra tujuan akan bernilai *maxValue*, selain itu akan bernilai 0.



**Gambar 3.8** Hasil citra *threshold*

Setelah didapatkan citra *threshold* masih terdapat *noise* pada citra, untuk mereduksi *noise* tersebut dilakukan proses *morphology* yang terdiri dari proses *dilate* dan *erode*. Proses *morphology* diawali dengan proses *dilate* terlebih dahulu untuk menghilangkan *noise* yang memiliki area lebih kecil dari pada area objek, setelah dilakukan proses *dilate* kemudian dilakukan proses *erode* untuk memperluas citra objek agar memiliki ukuran area semula, sehingga didapatkan citra dengan *noise* yang telah tereduksi. Operasi *morphology* yang dilakukan adalah mengaplikasikan *structure element* pada citra masukan untuk menghasilkan citra keluaran

yang sudah diproses. Pada proses *dilate*, proses terdiri dari citra masukan dan *kernel B* yang bisa berbentuk persegi atau lingkaran, *kernel B* memiliki *anchor point* yang sudah terdefinisi, pada umumnya posisi *anchor point* terletak di tengah *kernel*. Pada saat *kernel B* di *scanning* pada citra. Citra akan dikomputasi nilai maksimal piksel yang melebihi *kernel B* dan mengganti piksel citra di posisi *anchor point* dengan nilai maksimal tersebut. Dari proses tersebut keluaran citra akan semakin membesar. Pada proses *erode* secara garis besar tidak jauh berbeda hanya penggantian nilai maksimal piksel menjadi nilai minimal pada piksel yang melebihi *kernel B*.

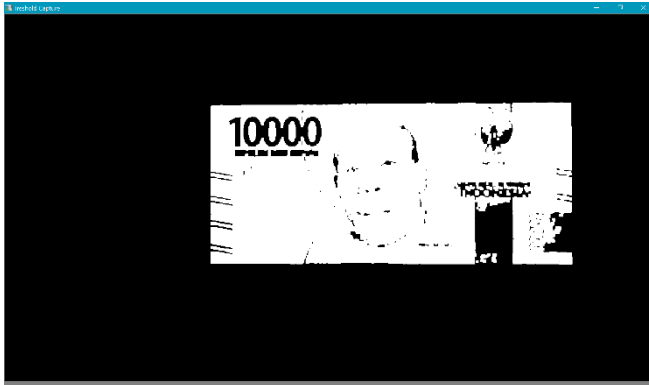
Pada operasi *dilate* dan *erode* pada program, akan digunakan fungsi dengan struktur sebagai berikut

```
Mat erodeElement = getStructuringElement(MORPH_RECT,  
Size(3, 3));  
Mat dilateElement = getStructuringElement(MORPH_RECT,  
Size(3, 3));  
  
erode(thresh, thresh, erodeElement);  
erode(thresh, thresh, erodeElement);  
dilate(thresh, thresh, dilateElement);  
dilate(thresh, thresh, dilateElement);
```

Pada awal struktur fungsi `::Mat erodeElement` dan `::Mat dilateElement` akan membuat memori citra elemen yang digunakan sebagai *kernel B*. fungsi `getStructuringElement` akan menentukan struktur dan bentuk elemen yang akan digunakan pada proses *Morphology*, pada *pointer* `MORPH_RECT` ditentukan bahwa bentuk elemen yang digunakan adalah *rectangle* atau persegi, selanjutnya pada *pointer* `Size(3,3)` adalah ukuran *kernel* yang akan digunakan. Nilai ukuran *kernel* dapat diubah-ubah sesuai dengan hasil yang diinginkan.



Selanjutnya pada fungsi erode dan dilate, masing-masing akan memanggil fungsi itu sendiri, *pointer* thresh adalah masukan dan keluaran citra setelah didapatkan citra hasil *threshold* pada proses sebelumnya. erodeElement dan dilateElement akan menjadi *kernel* bagi proses masing-masing fungsi.



**Gambar 3.9** Hasil citra *morphology*

### 3.2.3 *Contour*

Setelah didapatkan citra yang memisahkan objek dan *background* serta menghilangkan *noise* proses selanjutnya adalah mendeteksi letak uang kertas untuk mencari *Region of Interest* citra menggunakan *contour*. Fungsi yang digunakan adalah findContour dengan struktur

```
findContours(temp, contours, hierarchy, CV_RETR_EXTERNAL,  
CV_CHAIN_APPROX_SIMPLE);
```

Pada fungsi findContour terdiri dari *pointer* temp sebagai *array* masukan dan keluaran citra, contours sebagai *contour* yang terdeteksi yang akan disimpan dalam bentuk *vector*, hierarchy sebagai keluaran *vector* opsional yang menyimpan informasi *topology* citra, CV\_RETR\_EXTERNAL adalah mode pengembalian *contour*, dan CV\_CHAIN\_APPROX\_SIMPLE adalah metode pendekatan *contour* yang akan digunakan. Keluaran dari findContour berupa *vector* yang dapat menentukan posisi x dan y objek yang diuji.

### 3.2.4 *Region Of Interest*

Setelah nilai x dan y pada objek sudah didapatkan, maka selanjutnya akan dilakukan proses ROI untuk mendapatkan citra yang diinginkan. Sebelum melakukan proses *cropping* nilai x dan y awal akan ditentukan dengan fungsi `objectBoundingBox` dengan *vector* keluaran proses *contour* sebelumnya. Fungsi yang digunakan adalah `objectBoundingBox = boundingRect(largestContourVec.at(0))` fungsi tersebut membentuk persegi pada set point yang sudah ditentukan. *Pointer* `largestContourVec.at(0)` menunjukkan bahwa titik awal penghitungan. Setelah didapatkan posisi *vector* ROI, tahap selanjutnya adalah menentukan posisi x dan y untuk melakukan proses *cropping*. Struktur fungsi yang digunakan adalah

```
int xpos = objectBoundingBox.x +  
objectBoundingBox.width / 2;  
int ypos = objectBoundingBox.y +  
objectBoundingBox.height / 2;
```

Fungsi tersebut akan menghitung nilai x dan y citra dengan menambahkan posisi x dari *vector* awal dengan lebar kemudian dibagi 2, sama dengan nilai y, posisi y dari *vector* awal ditambahkan tinggi kemudian dibagi 2. Nilai x dan y yang dihasilkan akan menjadi acuan nilai x dan y proses *cropping* ROI.

Setelah didapatkan nilai acuan, kita bisa menghitung nilai x dan y untuk proses *cropping* sesuai citra yang diinginkan. Struktur yang digunakan adalah

```
ROI.x = xpos - (objectBoundingBox.width / 2);  
ROI.y = ypos - (objectBoundingBox.height / 2);  
ROI.width = objectBoundingBox.width;  
ROI.height = objectBoundingBox.height;
```

Dari fungsi diatas ROI.x dan ROI.y menjadi memori penyimpanan nilai x dan y yang akan digunakan, nilai x dan y akan diperbarui menggunakan nilai x dan y sebelumnya yang menjadi acuan. Fungsi ROI.width dan ROI.height menentukan dimensi ROI yang akan didapatkan, untuk penentuan dimensi ROI disamakan dengan dimensi `objectBoundingBox` yang sebelumnya sudah didapatkan.

Setelah didapatkan informasi ROI yang dibutuhkan, gambar ROI sudah bisa ditampilkan dan di *save* pada direktori program agar selanjutnya dapat diproses segmentasi.



**Gambar 3.10** Hasil citra ROI

### 3.2.5 Segmentasi

Hasil dari proses ROI akan menghasilkan citra yang sesuai untuk kemudian disegmentasi. Segmentasi dilakukan untuk proses LBP dan datalearning pengenalan. Pada segmentasi akan dilakukan kembali fungsi ROI dengan fungsi  $Rect ROI = Rect(x, y, small\_size.width, small\_size.height)$  hasil segmentasi akan berbentuk *rectangle* atau persegi dengan dimensi sesuai jumlah segmentasi yang diinginkan. Segmentasi akan berpengaruh dalam akurasi dan waktu proses komputasi dalam proses pengenalan.

### 3.2.6 *Local binary pattern*

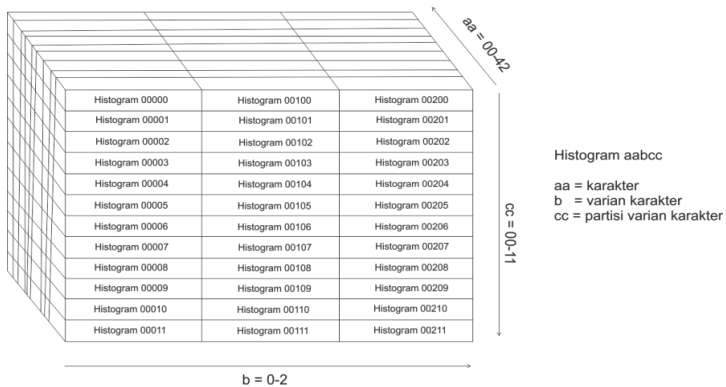
Pada proses *Local Binary Pattern*, dilakukan proses perubahan nilai citra dari citra *grayscale* menjadi citra LBP untuk didapatkan nilai *histogram*. Metode LBP yang digunakan adalah LBP original yang masih mengkomparasi dengan *array* 4 dimensi dengan menggunakan *window* kernell  $6 \times 6px$ . Untuk memperkecil nilai dan waktu *data learning* maka pada proses LBP citra akan diubah menjadi ukuran  $600 \times 240px$ . Hal ini akan memudahkan proses LBP untuk mengubah nilai karena LBP menggunakan kernel *window* sebesar  $6 \times 6px$ . Setelah citra di ubah ukuran menjadi  $600 \times 240px$  maka dilakukan proses LBP sehingga didapatkan citra LBP seperti pada gambar 3.4. Setelah didapatkan citra LBP kemudian akan didapatkan nilai *histogram* citra LBP.

### 3.2.7 Data learning

*Data learning* digunakan untuk mendapatkan nilai *data base* pengukuran *histogram* agar dapat dibandingkan dengan citra uang kertas yang didapatkan melalui kamera. Dari nilai *histogram data learning* dapat ditentukan citra uang kertas dikenali sebagai uang kertas tertentu pada proses selanjutnya.

*Data learning* yang digunakan adalah nilai *histogram* citra uang kertas yang telah melewati proses LBP dengan menggunakan sebanyak 4 jenis uang kertas dengan 10 variasi uang kertas sehingga keseluruhan terdapat nilai sebesar 40 *data learning*. Untuk mendapatkan nilai kedekatan uang kertas, setiap citra uang kertas di bandingkan dengan 40 citra *data learning* sehingga menghasilkan nilai terdekat dari uang kertas tersebut dengan nilai *histogram* pada citra *data learning*.

Pada proses pengenalan citra uang kertas akan ada proses pemanggilan *data learning* sebagai *data base* yang akan dikomparasi nilai *histogram*. Total citra *data learning* sebanyak 40 yang telah melalui prsoses LBP. Nilai *histogram data learning* akan disimpan kedalam sebuah *array* 2 dimensi dengan ukuran [4][10] untuk pelabelan dan menentukan banyak variasi *data learning*. Nilai *histogram* yang disimpan akan disimpan didalam aray sebesar [256]. Pada nilai penamaan *array* [10] digunakan untuk pengenalan citra uang kertas. Pada nilai *array* kedua [6] untuk menentukan besaran nilai variasi dari setiap penamaan yang digunakan. Jika digambarkan akan terlihat seperti gambar 3.11



**Gambar 3.11** Ilustrasi nilai histogram *data learning* pada *array* 4 dimensi.

Pada gambar3.5 nilai *data learning* di ilustrasikan sebagai nilai histogram setiap uang kertas yang disimpan dalam *array* 4 dimensi dengan ukuran [10][6], dimana 10 merupakan jenis uang kertas, sedangkan 6 merupakan varian dari satu jenis uang kertas. Setiap nilai *histogram* disimpan pada *array* dengan ukuran sebesar 256 untuk menyimpan nilai histogram dari 0-255. Digunakan pengkodean a untuk menentukan label jenis, b untuk menentukan variannya agar dapat memudahkan pencarian komparasi nilai *histogram*.

### 3.2.8 Pengenalan uang kertas

Proses terakhir adalah pengenalan citra uang kertas dengan cara membandingkan nilai *histogram*. Fungsi `calcHist(const Mat* images, int nimages, const int* channels, InputArray mask, OutputArray hist, int dims, const int* histSize, const float** ranges, bool uniform=true, bool accumulate=false)`. Fungsi ini untuk menghitung nilai dari citra yang akan dibuat komparasi pada proses selanjutnya. Komparasi citra uang kertas dari kamera dengan nilai *histogram data learning* menggunakan metode *Chi-square*, metode ini menggunakan komparasi dengan pendekatan nilai terkecil dari hasil perbandingan yang merupakan indikator citra yang memiliki nilai kesamaan *histogram* terdekat. Metode *Chi-square* dengan algoritme sebagai berikut:

$$chi\_square(h1, h2) = \sum_i \frac{(h1(i) - h2(i))^2}{h1(i)} \quad (3.1)$$

Di mana memiliki parameter:

- $h1(i)$  = nilai *histogram* citra 1 ke  $i$
- $h2(i)$  = nilai *histogram* citra 2 ke  $i$

*Histogram* citra uang kertas akan dibandingkan dengan semua nilai *histogram data learning*, sehingga didapatkan nilai terkecil dari hasil perbandingan, dan citra uang kertas dapat dikenali dengan melihat nilai terkecil dari nilai komparasi pada label yang ditentukan pada *data learning*. Hasil pengenalan citra uang kertas berupa sebuah nilai *integer* dengan *range* dari 0-10 yang melambangkan jenis citra uang kertas pada kamera. Setelah citra uang kertas dapat dikenali maka akan disimpan ke dalam sebuah text hasil data pengenalan citra.

*Halaman ini sengaja dikosongkan*

## BAB IV

### PENGUKURAN DAN ANALISA SISTEM

#### 4.1 Pengujian algoritma preprocessing

Pada pengujian algoritma prapemrosesan, algoritma terdiri dari pengujian hasil pengambilan gambar, *thresholding*, *morphology* dan segmentasi.

##### 4.1.1 Pengambilan gambar

Pengambilan gambar menggunakan kamera yang dipasang dengan konfigurasi vertical pada bagian atas kerangka dengan posisi menghadap ke bawah kearah objek. Pencahayaan dinyalakan dari sisi kiri dan kanan objek untuk memperjelas citra objek yang akan diambil. Selanjutnya citra akan diproses *threshold* dan *morphology*



Gambar 4.1 Hasil pengambilan citra

##### 4.1.2 *Thresholding* dan *morphology*

Pada proses *thresholding* nilai yang digunakan adalah 90 untuk nilai *thresh*, 255 untuk nilai *maxValue* dan 0 untuk tipe *threshold*. Nilai 0 pada tipe *threshold* adalah *binary threshold*, proses ini bertujuan untuk memisahkan objek dari *background*.

Setelah dilakukan proses *threshold*, selanjutnya akan dilakukan proses *morphology* untuk menghilangkan *noise* yang berada disekitar citra menggunakan *kernel B* berbentuk persegi yang bernilai 3x3 cm.



**Gambar 4.2** Hasil proses *threshold* dan *morphology* pada citra

Setelah pengambilan citra dilakukan, selanjutnya dilakukan *cropping image* untuk mengambil gambar yang diinginkan.



**Gambar 4.3** Hasil ROI citra.

#### 4.1.3 Segmentasi

Proses segmentasi mencakup algoritma yang dapat membagi uang kertas menjadi 50 segmen yang terdiri dari 5 kolom dan 10 baris, algoritma ini tidak bersifat *adaptive* sehingga dibutuhkan pemberian nilai parameter untuk mendapatkan hasil segmentasi yang baik.





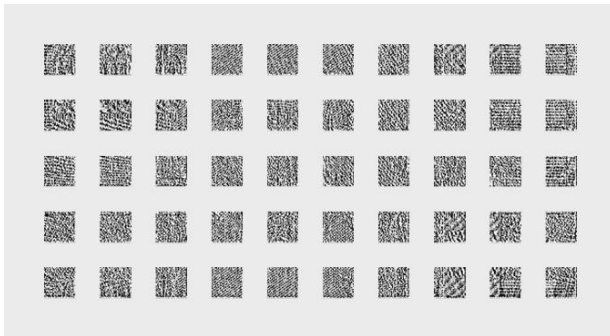
**Gambar 4.4** Hasil segmentasi citra.

## **4.2 Pengujian proses utama**

Pada pengujian proses pengenalan ini terdiri dari pengujian hasil pengoperasian LBP, inisiasi *data learning* dan pengenalan uang kertas.

### **4.2.1 Operasi *Local Binary Pattern***

Operasi *Local Binary Pattern* dari citra yang sudah tersegmentasi akan menghasilkan citra LBP yang nantinya dibutuhkan pada proses pengenalan uang kertas.



**Gambar 4.5** Hasil konversi LBP citra.

## 4.2.2 Inisiasi data learning

Hasil inisiasi data *learning* merupakan *array* 4 dimensi yang berisi nilai *histogram* dari gambar uang kertas yang digunakan untuk data *learning*. Data *base* terdiri dari 7 nominal uang kertas yang masing-masing memiliki 14 *sample* gambar untuk depan dan belakang. Berikut data *base* yang digunakan untuk inisiasi data *learning*:

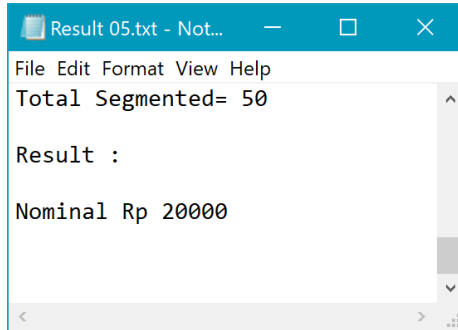


Gambar 4.6 Citra data base pada data learning.

Setiap citra yang ada pada *data learning* akan di lakukan pengambilan nilai *histogram* dan disimpan kedalam sebuah label dengan format *array* [7][14][256]. *Array* pertama berfungsi untuk menyimpan jenis mata uang yang akan di *training*, *array* kedua berfungsi untuk menyimpang variasi mata uang yang diuji, *array* ketiga adalah range [0-255]. *Array* data *learning* inilah yang digunakan untuk membandingkan nilai *histogram* uang kertas.

## 4.2.3 Pengenalan uang kertas

Setelah didapatkan nilai *histogram* pada citra yang akan diuji dan nilai *histogram* dari data *base* data *learning*. Akan dilakukan komparasi antara dua nilai *histogram* tersebut. Komparasi tersebut akan menghasilkan *Comparison Value* yang menentukan berapa nominal citra uang kertas tersebut. Hasil komparasi akan ditampilkan dengan bentuk file *.txt*



**Gambar 4.7** Tampilan hasil pengenalan uang kertas.

#### 4.2.4 Pengujian pengenalan

Pada bagian ini akan ditampilkan dan dijelaskan tentang pengujian yang telah dilakukan. Pengujian ini bertujuan untuk mengukur akurasi dari program. Pengujian dilakukan dengan mengubah segmentasi pada program, akan ada 4 segmentasi yang digunakan yaitu 50, 128, 200 dan 800. *Sample* gambar yang diambil adalah 3 uang kertas dari masing-masing nominal. Pada tabel 4.1 ditunjukkan hasil pengenalan yang berupa pembacaan nominal.

Nominal	Variasi Citra	Orientasi	Segmentasi			
			50	128	200	800
1000	Citra 1	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	<b>salah</b>
	Citra 2	Depan	benar	benar	benar	<b>salah</b>
		Belakang	<b>salah</b>	benar	<b>salah</b>	benar
	Citra 3	Depan	benar	benar	salah	benar
		Belakang	benar	benar	benar	benar
2000	Citra 1	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	benar
	Citra 2	Depan	benar	benar	<b>salah</b>	benar
		Belakang	benar	benar	benar	benar
	Citra 3	Depan	benar	benar	benar	<b>salah</b>
		Belakang	benar	benar	benar	benar
5000	Citra 1	Depan	benar	benar	benar	<b>salah</b>
		Belakang	benar	benar	benar	benar

	Citra 2	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	benar
	Citra 3	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	benar
10000	Citra 1	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	benar
	Citra 2	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	<b>salah</b>
	Citra 3	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	benar
20000	Citra 1	Depan	benar	benar	benar	<b>salah</b>
		Belakang	benar	benar	<b>salah</b>	<b>salah</b>
	Citra 2	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	benar
	Citra 3	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	<b>salah</b>
50000	Citra 1	Depan	benar	benar	benar	benar
		Belakang	<b>salah</b>	benar	benar	benar
	Citra 2	Depan	benar	benar	benar	benar
		Belakang	<b>salah</b>	benar	benar	benar
	Citra 3	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	benar
100000	Citra 1	Depan	benar	<b>salah</b>	benar	benar
		Belakang	benar	benar	benar	benar
	Citra 2	Depan	benar	benar	benar	benar
		Belakang	benar	benar	benar	benar
	Citra 3	Depan	<b>salah</b>	benar	benar	<b>salah</b>
		Belakang	benar	benar	benar	benar

**Tabel 4.1** Hasil pengujian pengenalan uang kertas

Segmentasi	Total percobaan	Benar	Salah	Presentase kebenaran
50	42	39	3	92,857%
128	42	41	1	97,61%
200	42	39	3	92,857%
800	42	35	7	83,333%

**Tabel 4.2** Presentase kebenaran pengenalan uang kertas

Dari hasil percobaan dari semua segmentasi dan semua nominal yang diuji, didapatkan hasil 92,857% untuk 50 segmentasi, 97,61% untuk 128 segmentasi, 92,857% untuk 200 segmentasi dan 83,333% untuk 800 segmentasi. Dari sample yang digunakan uang kertas nominal pecahan 1000 mendapatkan pembacaan yang salah terbanyak sebanyak 4 kali, untuk pecahan nominal 2000 mendapatkan pembacaan yang salah sebanyak 2 kali, nominal 5000 dan 10000 pembacaan yang salah sebanyak 1 kali, dan untuk pecahan 20000, 50000 dan 100000 total pembacaan yang salah adalah sebanyak 3 kali.

*Halaman ini sengaja dikosongkan*

## **BAB V**

### **PENUTUP**

#### **5.1 Kesimpulan**

Pada penelitian ini telah dibuat program pengenalan uang kertas dengan menggunakan metode *Local Binary Pattern*. Program ini terdiri dari teknik pengambilan gambar, operasi pengubahan menjadi *Local Binary Pattern*, data *learning* dan komparasi untuk pengenalan. Dari percobaan yang dilakukan menggunakan 6 nominal uang kertas rupiah masing-masing tiga lembar untuk setiap nominal menggunakan segmentasi 50, 128, 200 dan 800, didapatkan akurasi pembacaan sebesar 92,857% untuk 50 segmentasi, 97,61% untuk 128 segmentasi, 92,857% untuk 200 segmentasi dan 83,333% untuk 800 segmentasi. Dari uang kertas yang diuji didapatkan pembacaan yang salah terbanyak sebanyak 4 kali pada uang nominal 1000, untuk pecahan nominal 2000 mendapatkan pembacaan yang salah sebanyak 2 kali, nominal 5000 dan 10000 pembacaan yang salah sebanyak 1 kali, dan untuk pecahan 20000, 50000 dan 100000 total pembacaan yang salah adalah sebanyak 3 kali. Dari pengamatan percobaan penentuan parameter segmentasi berpengaruh dalam akurasi pengenalan. Selain itu citra pada *database* dan citra yang diuji juga berpengaruh.

#### **5.2 Saran**

Saran agar program ini dapat berjalan lebih baik yaitu.

1. Pencahayaan yang lebih merata akan menambah akurasi pengenalan menjadi lebih baik.
2. Penambahan *sample* pada data *base* dapat meningkatkan akurasi pengenalan dengan segmentasi yang lebih sedikit.
3. Metode *Look Up Table* dapat digunakan dengan metode komparasi *histogram* dengan jumlah data *training* yang banyak.

*Halaman ini sengaja dikosongkan*



## DAFTAR PUSTAKA

- [1] Mäenpää, Topi dan Matti Pietikäinen. 2004. *Texture Analysis With Local Binary Patterns*, University of Oulu.
- [2] Greg, Pass. Ramin Zabih. 2002. *Learning Visual Basic .NET : Introducing the Language, .NET Programming & Object Oriented Software Development*, California: O'Reilly Media, Inc.
- [3] <URL : (<https://www.bi.go.id/id/iek/mengenal-rupiah/Contents/Default.aspx>, 2018) (<https://www.bi.go.id/id/iek/mengenal-rupiah/Contents/Default.aspx>, 2018)> Diakses pada tanggal 27 Mei 2018
- [4] Bradski, Gary dan Adrian Kaehler. 2008. *Learning OpenCV*. Sebastopol: O'Reilly Media, Inc.
- [5] Smestad, Ragnar. 2008. *Introduction to C++ and OpenCV*. Forsvarets forskningsinstitutt.
- [6] <URL : <https://finance.detik.com/moneter/d-3374687/ini-11-uang-rupiah-desain-baru>> Diakses pada tanggal 1 Juni 2018

*Halaman ini sengaja dikosongkan*

## LAMPIRAN

### ➤ PROGRAM PRAPEMROSESAN

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/opencv.hpp"
#include <stdio.h>
#include <iostream>
#include <stdlib.h>

using namespace cv;
using namespace std;

string window_name1 = "Video Camera";
string window_name2 = "ROI";
string window_name3 = "Contours";

Mat frame, outputframe, scr, blurframe;

Mat src_gray;
```

```
int x = 0;

int y = 0;

const int Frame_W = 1280;

const int Frame_H = 720;

const int MAX_NUM_OBJECTS = 50;

const int MAX_OBJECT_AREA = Frame_W*Frame_H;

const int MIN_OBJECT_AREA = 10 * 10;
```

```
Rect ROI;
```

```
Rect objectBoundingBox = Rect(0, 0, 0, 0);
```

```
string intToString(int number)
```

```
{
    std::stringstream ss;
    ss << number;
    return ss.str();
}
```

```
void morphOps(Mat &thresh)
```

```
{
```

```
Mat erodeElement = getStructuringElement(MORPH_RECT,  
Size(3, 3)); //nilai kernel morphology
```

```
Mat dilateElement = getStructuringElement(MORPH_RECT,  
Size(3, 3)); //nilai kernel morphology
```

```
erode(thresh, thresh, erodeElement);
```

```
erode(thresh, thresh, erodeElement);
```

```
dilate(thresh, thresh, dilateElement);
```

```
dilate(thresh, thresh, dilateElement);
```

```
}
```

```
void Object_Detect(int &x, int &y, Mat threshold, Mat &Image) {
```

```
Mat temp;
```

```
Mat crop_1;
```

```
Mat crop_2;
```

```
threshold.copyTo(temp);
```

```
vector< vector<Point> > contours;
```

```
vector<Vec4i> hierarchy;
```

```
findContours(temp, contours, hierarchy,  
CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);
```

```

vector< vector<Point> > largestContourVec;

largestContourVec.push_back(contours.at(contours.size() - 1));

objectBoundingBox =
boundingRect(largestContourVec.at(0));

int xpos = objectBoundingBox.x +
objectBoundingBox.width / 2; // nilai x dan y acuan

int ypos = objectBoundingBox.y +
objectBoundingBox.height / 2;

ROI.x = xpos - (objectBoundingBox.width / 2);
// mencari nilai x dan y ROI

ROI.y = ypos - (objectBoundingBox.height / 2);

ROI.width = objectBoundingBox.width;

ROI.height = objectBoundingBox.height;

crop_1 = Image(ROI);

resize(crop_1, crop_2, Size(442, 200)); //
ukuran ROI

imshow(window_name2, crop_2);

imwrite("LBP_MONEY/gocapbelakang.jpg",
crop_2); //menyimpan citra ROI pada folder program

```

```

double refArea = 0;

bool objectFound = false;

if (hierarchy.size() > 0) {
    int numObjects = hierarchy.size();

    if (numObjects < MAX_NUM_OBJECTS) {
        for (int index = 0; index >= 0; index =
hierarchy[index][0]) {

            Moments moment =
moments((Mat)contours[index]);

            double area = moment.m00;

            if (area > MIN_OBJECT_AREA &&
area < MAX_OBJECT_AREA && area > refArea)
                {

                    x = moment.m10 / area;

                    y = moment.m01 / area;

                    objectFound = true;

                    refArea = area;

```

```

    }
    else objectFound = false;

}

}

else putText(Image, "TOO MUCH NOISE! ADJUST
FILTER", Point(0, 50), 1, 2, Scalar(0, 0, 255), 2);

}
}

```

```
int main(int, char**)
```

```
{
```

```
    Mat cap_img, cap_gray, cap_thres;
```

```
    VideoCapture capture; // membuka kamera
```

```
    capture.open(0);
```

```
    capture.set(CV_CAP_PROP_FRAME_HEIGHT, Frame_H);
```

```
    capture.set(CV_CAP_PROP_FRAME_WIDTH, Frame_W);
```

```
    cout << "Press 'S' to capture image" << endl;
```

```
    while (1)
```



```

{
    capture.read (frame);
    imshow(window_name1, frame);

    if (waitKey(10) % 256 == 's')
    {
        imwrite("captured.jpg", frame);
        cap_img = imread("captured.jpg",
CV_LOAD_IMAGE_ANYCOLOR);
        cvtColor(cap_img, cap_gray,
CV_BGR2GRAY);
        threshold(cap_gray, cap_thres, 90, 255, 0); //
nilai threshold

        morphOps(cap_thres);
        imshow("Treshold Capture", cap_thres);
        Object_Detect(x, y, cap_thres, cap_img);
    }
}

```

```
    if (waitKey(10) % 256 == 27)
    {
        break;
    }

}
return 0;
}
```

## ➤ PROGRAM UTAMA

```
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/core/mat.hpp>
#include <opencv2/core/types_c.h>
#include <opencv2/core/core_c.h>
#include <opencv2/highgui.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <math.h>
```

```
using namespace cv;
using namespace std;
```

```
int dd = 5;
int ff = 160 / dd;
int gg = 320 / ff;
int ee = dd*gg;
int cc = ee;
int aa = 4;
int bb = 10;
```

```
float datalearning[4][10][50][256];
float hist_char[50][256];
```

```
int main(int argc, char* argv[])
{
    Mat img_mat;
    Mat hist_mat;

    int label;
```

```

const int channels = 0;
const int numBins = 256; ///Histogram value
int char_recog[1107];
const float rangevals[2] = { 0.f,256.f };
const float* ranges = rangevals;
char lbp_imagestemp[512];
char data_learnings[1107];
char data_learning[1107];
char partitions[1107];

//=====
=====
===//

// ----- INPUT IMAGE ----- //

    Mat                input_gambar1                =
imread("LBP_MONEY/gocapbelakang.jpg",
CV_LOAD_IMAGE_COLOR);
    Mat input_gambar;
    resize(input_gambar1, input_gambar, Size(320, 160)); ///Load
input image
    Size small_size(ff, ff);
    Mat gambar_kecil = Mat(input_gambar, Rect(0, 0, ff,
ff)).clone();
    imshow("Input Image", input_gambar);

    cout << "Input Image Height = " << input_gambar.rows << endl;
    cout << "Input Image Width = " << input_gambar.cols << endl;
    cout << "\n";

//=====
=====
===//

```

```

// ----- SEGMENTATION INPUT IMAGE -----
//

cout << "----- Start Segmentation -----" << endl;
cout << "\n";

for (int c = 0; c < ee; c++)
{
    for (int y = 0; y < input_gambar.rows; y +=
small_size.height)
    {
        for (int x = 0; x < input_gambar.cols; x +=
small_size.width)
        {
            //----- ROI PROGRAM -----//
            Rect ROI = Rect(x, y,
small_size.width, small_size.height);

            //----- PENAMAAN
SEGMENTASI -----//
            sprintf(partitions, "512,
"LBP_MONEY/IN_SEG/%05d.jpg", c);
            gambar_kecil = input_gambar(ROI);
            imwrite(partitions, gambar_kecil);
            c++;
        }
    }
    cout << "Total Image = " << c << endl;
    cout << "\n";
}
cout << "----- Finish Segmentation -----" << endl;
cout << "\n";

```

```

//=====
=====
===//

// ----- LBP IMG -----//

cout << "----- Start LBP Input Image -----" << endl;
cout << "\n";

int lbp_nb[8];
char character_images[1107];
char lbp_images[1107];

for (int c = 0; c < ee; c++)
{
    snprintf(character_images, 1107,
"LBP_MONEY/IN_SEG/%05d.jpg", c);
    IplImage* img_part = cvLoadImage(character_images,
CV_LOAD_IMAGE_GRAYSCALE);
    IplImage* img_lbp = cvCreateImage(cvSize(ff, ff), 8,
1);
    cvShowImage("INPUT IMG", img_part);

    for (int y = 0; y < ff; y++)
    {
        for (int x = 0; x < ff; x++)
        {
            int center =
CV_IMAGE_ELEM(img_part, uchar, y, x);
            lbp_nb[0] =
CV_IMAGE_ELEM(img_part, uchar, y - 1, x - 1);
            lbp_nb[1] =
CV_IMAGE_ELEM(img_part, uchar, y - 1, x);

```

```

        lbp_nb[2] =
CV_IMAGE_ELEM(img_part, uchar, y - 1, x + 1);
        lbp_nb[3] =
CV_IMAGE_ELEM(img_part, uchar, y, x + 1);
        lbp_nb[4] =
CV_IMAGE_ELEM(img_part, uchar, y + 1, x + 1);
        lbp_nb[5] =
CV_IMAGE_ELEM(img_part, uchar, y + 1, x);
        lbp_nb[6] =
CV_IMAGE_ELEM(img_part, uchar, y + 1, x - 1);
        lbp_nb[7] =
CV_IMAGE_ELEM(img_part, uchar, y, x - 1);

        for (int i = 0; i < 8; i++)
        {
            if (lbp_nb[i] < center)
            {
                lbp_nb[i] = 0;
            }
            else
            {
                lbp_nb[i] = 1;
            }
        }
        center = lbp_nb[7] * 128 + lbp_nb[6]
* 64 + lbp_nb[5] * 32 + lbp_nb[4] * 16 + lbp_nb[3] * 8 + lbp_nb[2] * 4
+ lbp_nb[1] * 2 + lbp_nb[0] * 1;
        CV_IMAGE_ELEM(img_lbp,
uchar, y - 1, x - 1) = center;
    }
}
cvShowImage("LBP IMG", img_lbp);
snprintf(lbp_images,
"LBP_MONEY/LBP_IN_SEG %02d/%05d.jpg", dd, c);

```

1107,

```

        cvSaveImage(lbp_images, img_lbp);
    }
    cout << "----- Finish Input LBP Image -----" << endl;
    cout << "\n";

    //=====
    =====//

    // ----- HISTOGRAM DATA LEARNING -----
    //

    cout << "----- Data Learning Initiation Start -----" << endl;
    cout << "\n";
    ofstream segfile;
    segfile.open("LBP_MONEY/Histogram DL.txt");
    for (int a = 0; a < aa; a++)
    {
        for (int b = 0; b < bb; b++)
        {
            for (int c = 0; c < cc; c++)
            {
                snprintf(data_learning, 1107,
                    "LBP_MONEYMAKER/DL_SEG
                    %02d/LBP_DL_SEG/%01d%01d%03d.jpg", dd, a, b, c);
                img_mat = imread(data_learning,
                    CV_LOAD_IMAGE_ANYCOLOR);
                calcHist(&img_mat, 1, &channels,
                    noArray(), hist_mat, 1, &numBins, &ranges);
                snprintf(data_learnings, 1107,
                    "Histogram %01d%01d%03d.jpg =", a, b, c);
                segfile << data_learnings << endl;
                for (size_t i = 0; i < numBins; ++i)

```



```

        {
            float      val      =
hist_mat.at<float>(i);
            datalearning[a][b][c][i] =
val;
            segfile          <<
datalearning[a][b][c][i] << "|";
            if((i == 50) || (i == 100) || (i
== 150) || (i == 200) || (i == 250))
                {
                    segfile << "\n";
                }
            }
            segfile << "\n";
        }
        segfile << "\n";
    }
    segfile << "\n";
}
segfile.close();
cout << "----- Data Learning Initiation has Completed -----
-" << endl;
cout << "\n";

//=====
=====
====//

segfile.open("LBP_MONEY/Result 05.txt");
cout << "----- Recognizing Banknotes -----" << endl;
cout << "\n";

//----- HISTOGRAM INPUT IMAGE -----//

```

```

    for (int c = 0; c<cc; c++)
    {
        sprintf(lbp_imagestemp,                                1107,
"LBP_MONEY/LBP_IN_SEG %02d/%05d.jpg", dd, c);
        img_mat = imread(lbp_imagestemp,
CV_LOAD_IMAGE_ANYCOLOR);
        imshow("Input", img_mat);
        calcHist(&img_mat, 1, &channels, noArray(),
hist_mat, 1, &numBins, &ranges);
        for (size_t i = 0; i<numBins; ++i)
        {
            float val = hist_mat.at<float>(i);
            hist_char[c][i] = val;
        }
    }

//----- COMPARE HISTOGRAM DATA -----//

float comp = 3.40282347E+38;
for (int a = 0; a<aa; a++)
{
    float comp_newval = 0;
    for (int b = 0; b<bb; b++)
    {
        float comp_val = 0;
        for (int c = 0; c<cc; c++)
        {
            float chi_sqr = 0;
            for (size_t i = 0; i < numBins; ++i)
            {
                float pembilang =
datalearning[a][b][c][i] - hist_char[c][i];

```

```

datalearning[a][b][c][i];
float penyebut =
+ pow(pembilang, 2) / (penyebut);
if (penyebut != 0)
{
    chi_sqr = chi_sqr
}
}
comp_val = comp_val + chi_sqr;
}
segfile << "Comparison value = " <<
comp_val << endl;
if (comp_val < comp)
{
    comp = comp_val;
    label = a;
}
}
cout << "type = " << a << endl;
segfile << "type = " << a << endl;
segfile << "\n";
}

//----- LABEL NAMING -----//

string Char_label[5] =
{
    "5000", "10000", "20000", "50000"
};

segfile << "Total Segmented= " << ee << endl;
segfile << "\n";
segfile << "Result :" << endl;
segfile << "\n";

```

```

cout << "\n";
if (label == 0)
{
    cout << "Nominal Rp " << Char_label[0] << endl;
    segfile << "Nominal Rp " << Char_label[0] << endl;
    segfile << "\n";
}
if (label == 1)
{
    cout << "Nominal Rp " << Char_label[1] << endl;
    segfile << "Nominal Rp " << Char_label[1] << endl;
    segfile << "\n";
}
if (label == 2)
{
    cout << "Nominal Rp " << Char_label[2] << endl;
    segfile << "Nominal Rp " << Char_label[2] << endl;
    segfile << "\n";
}
if (label == 3)
{
    cout << "Nominal Rp " << Char_label[3] << endl;
    segfile << "Nominal Rp " << Char_label[3] << endl;
    segfile << "\n";
}
segfile.close();
cout << "\n";
cout << "----- Banknotes Recognizing has Completed -----
" << endl;

waitKey(0);

```

```
//=====
=====
===//
}
```

*Halaman ini sengaja dikosongkan*

## BIODATA PENULIS



**M. Kukuh Prayogo**, lahir di Padang 12 Juli 1994. Anak kedua dari lima bersaudara. Penulis memulai pendidikan jenjang dasar di sekolah dasar swasta Baiturrahmah (2000), Padang. Setelah lulus sekolah dasar tahun 2006 penulis melanjutkan ke jenjang menengah di sekolah menengah pertama negeri Sekolah Menengah Pertama Negeri 2 Padang (2006). Kemudian penulis melanjutkan jenjang pendidikan di sekolah menengah atas SMA Dwiwarna Boarding School, Bogor (2009). Setelah menyelesaikan pendidikan di jenjang sekolah menengah atas penulis melanjutkan jenjang pendidikannya di Institut Teknologi Sepuluh Nopember departemen Teknik Elektro dengan bidang studi Elektronika.

Email: [kukuh.sudiby@gmail.com](mailto:kukuh.sudiby@gmail.com)

*Halaman ini sengaja dikosongkan*