

**PENINGKATAN PENDETEKSIAN *COPY DETECTION PATTERN*  
PADA *SECURE QR CODE* MENGGUNAKAN PENANDA ARUCO**

**SKRIPSI**



**Disusun oleh:**

**Ade Firmansyah  
19/440303/TK/48630**

**PROGRAM STUDI TEKNOLOGI INFORMASI  
DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI  
FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA  
YOGYAKARTA  
2023**

## **HALAMAN PENGESAHAN**

### **PENINGKATAN PENDETEKSIAN *COPY DETECTION PATTERN* PADA *SECURE QR CODE* MENGGUNAKAN PENANDA ARUCO**

#### **SKRIPSI**

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh  
Gelar Sarjana Teknik  
pada Departemen Teknik Elektro dan Teknologi Informasi  
Fakultas Teknik  
Universitas Gadjah Mada

Disusun oleh:

**Ade Firmansyah**  
**19/440303/TK/48630**

Telah disetujui dan disahkan

Pada tanggal 5 Mei 2023

Dosen Pembimbing I

Dosen Pembimbing II

**Syukron Abu Ishaq Alfarozi, S.T., Ph.D.**  
**NIKA 1111 9920 5202 10 1 102**

**Azkario Rizky Pratama, S.T., M.Eng., Ph.D.**  
**NIKA 1111 9910 2201 60 7 102**

## **PERNYATAAN BEBAS PLAGIASI**

Saya yang bertanda tangan di bawah ini :

Nama : Ade Firmansyah  
NIM : 19/440303/TK/48630  
Tahun terdaftar : 2019  
Program Studi : Teknologi Informasi  
Fakultas : Teknik Universitas Gadjah Mada

Menyatakan bahwa dalam dokumen ilmiah Skripsi ini tidak terdapat bagian dari karya ilmiah lain yang telah diajukan untuk memperoleh gelar akademik di suatu lembaga Pendidikan Tinggi, dan juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang/lembaga lain, kecuali yang secara tertulis disitasi dalam dokumen ini dan disebutkan sumbernya secara lengkap dalam daftar pustaka.

Dengan demikian saya menyatakan bahwa dokumen ilmiah ini bebas dari unsur-unsur plagiasi dan apabila dokumen ilmiah Skripsi ini di kemudian hari terbukti merupakan plagiasi dari hasil karya penulis lain dan/atau dengan sengaja mengajukan karya atau pendapat yang merupakan hasil karya penulis lain, maka penulis bersedia menerima sanksi akademik dan/atau sanksi hukum yang berlaku.

Yogyakarta, tanggal-bulan-tahun

Materai Rp10.000

(Tanda tangan)

Ade Firmansyah  
19/440303/TK/48630

## **HALAMAN PERSEMPAHAN**

Tugas akhir ini kupersembahkan kepada kedua orang tuaku. Kupersembahkan pula kepada keluarga dan teman-teman semua, serta untuk bangsa, negara, dan agamaku.

## **KATA PENGANTAR**

Puji syukur ke hadirat Allah SWT atas limpahan rahmat, karunia, serta petunjuk-Nya sehingga tugas akhir berupa penyusunan skripsi ini telah terselesaikan dengan baik. Dalam hal penyusunan tugas akhir ini penulis telah banyak mendapatkan arahan, bantuan, serta dukungan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Bapak Ir. Hanung Adi Nugroho, S.T., M.Eng., Ph.D., IPM., selaku Kepala Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada,
2. Bapak Ir. Lesnanto Multa Putranto, S.T., M.Eng., Ph.D., IPM. selaku Sekretaris Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada,
3. Bapak Syukron Abu Ishaq Alfarizi, S.T., Ph.D. selaku dosen pembimbing pertama yang telah memberikan banyak bantuan, bimbingan, serta arahan dalam Tugas Akhir ini,
4. Bapak Azkario Rizky Pratama, S.T., M.Eng., Ph.D. selaku dosen pembimbing kedua yang juga telah memberikan banyak bantuan, bimbingan, serta arahan dalam Tugas Akhir ini,
5. Seluruh dosen dan tenaga pendidik DTETI FT UGM yang telah memberikan ilmu dan pengalaman, serta membantu kelancaran penulis dari masa awal hingga akhir perkuliahan,
6. Bapak dan ibu, Miftakhul Hudha, S.T., M.Eng dan Tri Retnowati S.E. yang telah memberi banyak dukungan moral dan percaya bahwa saya bisa menyelesaikan skripsi ini tepat pada waktunya,
7. Teman-teman TETI Padmanaba 74 yang selalu mendukung satu sama lain, baik dalam bidang akademis maupun non-akademis,
8. Teman-teman Podang yang membuat masa perkuliahan di DTETI menjadi lebih menyenangkan,
9. Teman-teman satu bimbingan skripsi dengan Pak Syukron yang selalu datang tiap minggu untuk bimbingan,
10. Semua teman-teman TETI angkatan 2019,
11. Seluruh pihak, baik yang penulis kenal maupun tidak, yang telah membuat penulis menjadi pribadi yang lebih baik hingga saat ini,
12. Seluruh pihak yang tidak dapat disebutkan satu per satu, yang telah membantu dan memberikan dukungan dalam proses penulisan skripsi ini.

Akhir kata penulis berharap semoga skripsi ini dapat memberikan manfaat bagi kita semua, aamiin.

## DAFTAR ISI

<b>HALAMAN PENGESAHAN .....</b>	ii
<b>PERNYATAAN BEBAS PLAGIASI .....</b>	iii
<b>HALAMAN PERSEMBAHAN .....</b>	iv
<b>KATA PENGANTAR .....</b>	v
<b>DAFTAR ISI .....</b>	vi
<b>DAFTAR TABEL .....</b>	ix
<b>DAFTAR GAMBAR .....</b>	x
<b>DAFTAR SINGKATAN .....</b>	xii
<b>INTISARI .....</b>	xiii
<b>ABSTRACT .....</b>	xiv
<b>BAB I Pendahuluan .....</b>	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	2
1.3 Tujuan Penelitian .....	2
1.4 Batasan Penelitian .....	2
1.5 Manfaat Penelitian .....	3
1.6 Sistematika Penulisan .....	3
<b>BAB II Tinjauan Pustaka dan Dasar Teori .....</b>	5
2.1 Tinjauan Pustaka .....	5
2.1.1 Apakah Pola Deteksi Duplikat (CDP) dapat Disalin .....	5
2.1.1.1 Prinsip Degradasi Informasi .....	5
2.1.1.2 Definisi Teoritis dari Sistem Autentikasi CDP .....	6
2.1.1.3 Komponen dari Detektor .....	7
2.1.2 Deteksi Pembajakan menggunakan SQR .....	7
2.1.2.1 Struktur dari SQR .....	8
2.1.2.2 Pembuatan dan Pencetakan SQR .....	9
2.1.2.3 Autentikasi SQR .....	9
2.1.3 Autentikasi Digital menggunakan CDP .....	10
2.2 Dasar Teori .....	12
2.2.1 Kode QR .....	12
2.2.1.1 Bagaimana Kode QR Bekerja .....	12
2.2.1.2 Versi Kode QR .....	13
2.2.1.3 Koreksi Kesalahan Kode QR .....	13
2.2.2 <i>Copy Detection Pattern (CDP)</i> .....	14
2.2.3 Lokalisasi Objek dengan Pengenalan Pola .....	15
2.2.4 ArUco Marker .....	15

2.2.5	Fitur Spasial pada Citra Gambar .....	15
2.2.6	Fitur Histogram pada Citra Gambar .....	16
2.2.7	Fitur DCT pada Citra Gambar .....	19
2.2.8	Koefisien Jarak .....	19
2.2.8.1	Koefisien Jarak Euclidean .....	19
2.2.8.2	Koefisien Jarak Korelasi .....	20
2.2.8.3	Koefisien Jarak Kosinus .....	20
2.2.8.4	Koefisien Jarak Canberra .....	21
2.2.9	Transformasi Homografi .....	21
2.2.10	Uji Statistik <i>T-test</i> .....	21
2.2.11	Pembelajaran Mesin .....	22
2.2.12	AutoML .....	22
2.2.13	AutoGluon .....	23
<b>BAB III Metode Penelitian</b>	.....	25
3.1	Alat dan Bahan Tugas Akhir .....	25
3.1.1	Alat Tugas Akhir .....	25
3.1.2	Bahan Tugas akhir .....	26
3.2	Metode yang Digunakan .....	27
3.3	Tahapan Penelitian .....	27
3.4	Pembuatan Model SQR .....	30
3.4.1	Penentuan Ukuran dan Versi Kode QR .....	30
3.4.2	Penentuan Level Toleransi Kerusakan Kode QR .....	30
3.4.3	Penentuan Ukuran <i>Watermark</i> .....	30
3.4.4	Pembuatan CDP .....	30
3.4.5	Menentukan Jumlah, Jenis, Letak, dan Ukuran Penanda ArUco ..	31
3.4.6	Menempelkan <i>Watermark</i> ke SQR .....	32
3.5	Pembuatan dan Pengolahan <i>Dataset SQR</i> Foto Pertama (Orisinal) .....	33
3.5.1	Pembuatan <i>Batch SQR</i> Foto Pertama (Orisinal) .....	33
3.5.2	Pencetakan <i>Batch SQR</i> Foto Pertama (Orisinal) .....	34
3.5.3	Pemotretan <i>Dataset SQR</i> Foto Pertama (Orisinal) .....	35
3.5.4	Lokalisasi CDP dari <i>Raw</i> Foto Pertama (Orisinal) .....	35
3.5.4.1	Mengganti Nama Fail .....	35
3.5.4.2	Lokalisasi CDP dari Foto .....	36
3.5.5	Pembuatan Fitur Jarak (Orisinal dengan <i>Template</i> ) .....	39
3.5.6	Analisis <i>Dataset CDP</i> Foto Pertama (Orisinal) .....	40
3.6	Pembuatan dan Pengolahan <i>Dataset SQR</i> Foto Kedua (Palsu) .....	41
3.6.1	Pembuatan Model SQR Foto Kedua (Palsu) .....	41
3.6.2	Pembuatan <i>Batch SQR</i> Foto Kedua (Palsu) .....	41
3.6.3	Pencetakan <i>Batch SQR</i> Foto Kedua (Palsu) .....	41

3.6.4 Pemotretan <i>Dataset SQR</i> Foto Kedua (Palsu) .....	41
3.6.5 Lokalisasi CDP dari <i>Raw Foto</i> Kedua (Palsu) .....	41
3.6.6 Pembuatan Fitur Jarak (Palsu dengan <i>Template</i> ) .....	41
<b>3.7 Pembuatan dan Pengujian Model Klasifikasi SQR Orisinal dan Palsu.....</b>	<b>42</b>
3.7.1 Pembuatan Model Klasifikasi Biner menggunakan AutoGluon .....	42
<b>BAB IV Hasil dan Pembahasan.....</b>	<b>43</b>
4.1 Hasil Akhir Desain SQR.....	43
4.2 Hasil Parameter P&S .....	43
4.2.1 Konfigurasi Kamera.....	43
4.2.2 Jenis Kertas dan Tinta.....	44
4.3 Hasil Pendekripsi Penanda ArUco.....	45
4.3.1 Parameter Akhir yang Digunakan.....	46
4.4 Deskripsi <i>Dataset</i> .....	46
4.5 Analisis Hasil Lokalisasi CDP .....	47
4.5.1 Analisis Perbandingan Hasil Lokalisasi CDP Orisinal dengan Ber- bagai Interpolasi Penskalaan .....	47
4.5.2 Analisis Perbandingan Hasil Lokalisasi CDP menggunakan Pe- nanda ArUco dan Tanpa Penanda ArUco.....	48
4.5.3 Analisis Signifikansi CDP Orisinal dan Palsu .....	50
4.6 Hasil Pemodelan Klasifikasi Biner.....	53
4.6.1 Fitur Tunggal .....	53
4.6.2 Multi Fitur .....	54
<b>BAB V Kesimpulan dan Saran.....</b>	<b>55</b>
5.1 Kesimpulan.....	55
5.2 Saran.....	55
<b>DAFTAR PUSTAKA.....</b>	<b>57</b>
<b>LAMPIRAN .....</b>	<b>L-1</b>
L.1 Kode Sumber.....	L-1

## DAFTAR TABEL

Tabel 2.1	Tabel perbandingan level koreksi kesalahan dengan persentase toleransi kesalahannya.....	14
Tabel 2.2	Perbandingan performa AutoGluon dengan AutoML kompetitor .....	24
Tabel 4.1	Parameter Konfigurasi Kamera .....	44
Tabel 4.2	Tabel hasil pendektsian penanda ArUco dengan berbagai ukuran....	45
Tabel 4.3	Tabel hasil pendektsian penanda ArUco dengan berbagai ukuran...	46
Tabel 4.4	Hasil rata-rata jarak CDP orisinal dengan <i>template</i> dari berbagai interpolasi penskalaan dan koefisien jarak .....	48
Tabel 4.5	Hasil perbandingan rata-rata jarak hasil lokalisasi 8 titik dan 4 titik pada <i>dataset</i> CDP orisinal 4 level .....	49
Tabel 4.6	Hasil perbandingan rata-rata jarak hasil lokalisasi 8 titik dan 4 titik pada <i>dataset</i> CDP palsu 4 level .....	49
Tabel 4.7	Hasil pengujian <i>T-test</i> pada data grup fitur jarak CDP orisinal dengan palsu, pada CDP yang dilokalisasi dengan 8 titik dan CDP yang dilokalisasi dengan 4 titik.....	50
Tabel 4.8	Hasil akurasi model klasifikasi biner menggunakan fitur tunggal ....	53
Tabel 4.9	Perbandingan akurasi autentikasi klasifikasi biner pada <i>dataset</i> CDP 4 level yang dilokalisasi dengan 8 titik dan <i>dataset</i> CDP 4 level yang dilokalisasi dengan 4 titik.....	54

## DAFTAR GAMBAR

Gambar 2.1	Contoh dari CDP a) CDP <i>template</i> digital (I) b) CDP yang telah terdegradasi kualitasnya akibat proses P&S (I) .....	5
Gambar 2.2	Perbandingan dari CDP dengan data-matriks: Data-matriks memiliki ukuran unit komponen yang lebih besar, sehingga degradasi informasi tidak berdampak signifikan pada struktur kode ....	6
Gambar 2.3	Perbandingan dari kode QR orisinal dan palsu, keduanya menyimpan informasi yang sama dan sama-sama dapat dipindai .....	8
Gambar 2.4	SQR diharapkan dapat mendeteksi kode QR palsu pada saat di-autentikasi oleh pengguna .....	8
Gambar 2.5	Struktur SQR secara umum .....	9
Gambar 2.6	Perangkat seluler melakukan pemindaian menggunakan aplikasi khusus .....	10
Gambar 2.7	Perbandingan 1-D dengan 2-D <i>barcode</i> .....	12
Gambar 2.8	Struktur modul pada kode QR dua dimensi .....	13
Gambar 2.9	Jumlah modul berdasarkan versi kode QR .....	13
Gambar 2.10	Salah satu contoh CDP .....	14
Gambar 2.11	Histogram pada gambar dengan pencahayaan gelap .....	16
Gambar 2.12	Histogram pada gambar dengan pencahayaan terang .....	16
Gambar 2.13	Histogram pada gambar dengan kontras tinggi .....	17
Gambar 2.14	Histogram pada gambar dengan kontras rendah .....	17
Gambar 2.15	Histogram pada gambar dengan saturasi tinggi .....	17
Gambar 2.16	Gambar yang berbeda memiliki distribusi histogram yang sama ..	18
Gambar 2.17	Gambar yang ideal untuk di- <i>thresholding</i> .....	18
Gambar 2.18	Gambar yang tidak ideal untuk di- <i>thresholding</i> .....	18
Gambar 3.1	Diagram alir tahapan penelitian .....	29
Gambar 3.2	Sampel 50 penanda ArUco dengan ukuran modul 4x4 piksel .....	32
Gambar 3.3	Sampel 50 penanda ArUco dengan ukuran modul 6x6 piksel .....	32
Gambar 3.4	<i>Batch</i> SQR orisinal (foto pertama) dalam ukuran piksel .....	33
Gambar 3.5	Konfigurasi saat melakukan pencetakan menggunakan Adobe Photoshop .....	34
Gambar 3.6	Konfigurasi menambahkan ukuran kertas A3+ dan cetak menjadi PDF .....	34
Gambar 3.7	Lingkungan pemotretan menggunakan boks untuk melakukan pemotretan <i>dataset</i> SQR .....	35
Gambar 3.8	Nama fail sebelum perubahan .....	36
Gambar 3.9	Nama fail setelah perubahan .....	36
Gambar 3.10	xy_list yang menyimpan koordinat titik tengah penanda ArUco dari id 0 s.d. 7 .....	37
Gambar 3.11	Hasil plot koordinat titik tengah penanda ArUco pada xy_list ke gambar hasil foto .....	37
Gambar 3.12	<i>List</i> koordinat tujuan transformasi homografi .....	38
Gambar 3.13	SQR awal dan setelah dilakukan transformasi homografi .....	38
Gambar 3.14	Perbandingan CDP <i>template</i> (kiri) dengan CDP hasil lokalisasi foto pertama (orisinal) (kanan) .....	39

Gambar 3.15 Hasil <i>dataset CDP</i> foto pertama (orisinal) yang sudah dilokalisasi dan diganti nama failnya.....	39
Gambar 3.16 <i>Dataframe</i> fitur jarak <i>dataset</i> foto pertama (orisinal) .....	40
Gambar 4.1 Hasil akhir desain SQR .....	43
Gambar 4.2 Hasil pemotretan SQR menggunakan perangkat Iphone XR .....	44
Gambar 4.3 Perbandingan gambar hasil pemotretan SQR yang dicetak dengan berbagai tipe kertas .....	45
Gambar 4.4 Sampel data yang dipotret.....	47
Gambar 4.5 Perbandingan CDP <i>template</i> dengan yang dilokalisasi dengan 4 titik (tengah) dan CDP yang dilokalisasi dengan 8 titik (kanan) ..	50
Gambar 4.6 Plot distribusi koefisien jarak pada CDP 4 level yang dilokalisasi dengan 8 titik.....	52
Gambar 4.7 Plot distribusi koefisien jarak pada CDP 4 level yang dilokalisasi dengan 4 titik.....	52

## **DAFTAR SINGKATAN**

$\mathcal{H}_0$	= Hipotesis Nol
$\mathcal{H}_1$	= Hipotesis Alternatif
AutoML	= Automated Machine Learning
BER	= Bit Error Rate
CDP	= Copy Detection Pattern
DCT	= Discrete Cosine Transform
FN	= False Negative
FP	= False Positive
GBM	= Gradient-Boosting Machine
ML	= Machine Learning
P&S	= Print & Scan
QR	= Quick Response
TN	= True Negative
TP	= True Positive
SQR	= Secure Quick Response Code
XGBoost	= Extreme Gradient-Boosting

## INTISARI

Pembajakan produk atau yang dikenal juga sebagai tindakan pemalsuan, merupakan suatu tindakan ilegal yang dilakukan dengan tujuan memperoleh keuntungan dengan cara meniru atau menyalin produk asli yang sudah dipatenkan atau memiliki hak cipta. Pada tahun 2016 perdagangan barang palsu dan bajakan mencapai angka \$509 miliar (3,3% dari total perdagangan dunia). Untuk mengatasi permasalahan tersebut, produsen dapat menggunakan *secure QR code* (SQR) yang ditempelkan di produk. SQR merupakan kode QR biasa yang dilengkapi *copy detection pattern* (CDP). CDP merupakan sebuah pola matriks yang tahan terhadap penyalinan. Melalui proses *print & scan* (P&S), CDP akan terdegradasi kualitasnya, sehingga dapat dibedakan antara CDP original dan palsu.

Untuk mendapatkan objek CDP yang digunakan untuk proses autentikasi, dapat digunakan 4 titik sudut kode QR yang nantinya akan ditransformasi dan dipotong tengahnya sebesar ukuran CDP yang telah ditentukan. Hasil lokalisasi menggunakan 4 titik menghasilkan kualitas lokalisasi CDP yang kurang baik apabila SQR dipotret dalam kondisi yang tidak ideal, misalnya ada lekukan pada sisi SQR karena SQR ditempelkan pada permukaan tabung. Penelitian ini mencoba untuk mengatasi permasalahan tersebut dengan menggunakan 8 penanda ArUco di sekitar objek CDP untuk meningkatkan kualitas lokalisasi CDP pada kondisi yang kurang ideal.

Kata kunci : *Copy Detection Pattern, Secure QR Code, ArUco Marker, AutoML, Auto-Gluon, Anti-Pemalsuan*

## ABSTRACT

*Product piracy or counterfeiting is an illegal act carried out with the aim of gaining profit by imitating or copying patented or copyrighted products. In 2016, the trade of counterfeit and pirated goods reached \$509 billion (3.3% of total global trade). To address this problem, manufacturers can use secure QR codes (SQR) affixed to their products. SQR are ordinary QR codes equipped with a copy detection pattern (CDP), which is a matrix pattern that is resistant to copying. Through the print and scan process, the quality of the CDP degrades, making it possible to distinguish between the original and counterfeit CDPs.*

*To obtain the CDP object used for the authentication process, four QR code corner points can be used, which will then be transformed and cropped to the predetermined CDP size. However, the quality of CDP localization using four points is not good if the SQR is captured in less-than-ideal conditions, such as when the SQR is affixed to a curved surface. This research aims to address this problem by using eight ArUco markers around the CDP object to improve the quality of CDP localization in less-than-ideal conditions.*

**Keywords :** *Copy Detection Pattern, Secure QR Code, ArUco Marker, AutoML, Auto-Gluon, Anti-Counterfeiting*

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Pembajakan produk atau yang dikenal juga sebagai tindakan pemalsuan, merupakan suatu tindakan ilegal yang dilakukan dengan tujuan memperoleh keuntungan dengan cara meniru atau menyalin produk asli yang sudah dipatenkan atau memiliki hak cipta. Pembajakan produk semakin marak di era digital dan globalisasi, seiring dengan perkembangan teknologi. Di era digital, pembajakan produk semakin mudah dilakukan dengan memanfaatkan internet dan teknologi digital. Sementara itu, globalisasi mempermudah transportasi dan distribusi produk palsu dari satu negara ke negara lain. Pembajakan produk juga menyebabkan kerugian ekonomi yang signifikan bagi produsen dan pemilik hak merek, serta dapat membahayakan keselamatan konsumen. Hal ini terjadi karena pembajakan produk dapat merusak citra perusahaan, mengurangi pendapatan, serta merugikan konsumen yang membeli produk palsu yang seringkali berkualitas rendah dan dapat membahayakan diri, baik secara langsung maupun tidak langsung [1].

Teknologi percetakan dan pemindai digital telah mengalami perkembangan yang pesat selama beberapa dekade terakhir. Namun, kemajuan ini tidak hanya digunakan untuk kegiatan positif juga dimanfaatkan oleh pelaku pembajakan untuk memproduksi produk-produk bajakan. Dengan kemampuan teknologi percetakan dan pemindai yang semakin canggih, pembajakan produk menjadi lebih mudah, lebih cepat, dan lebih murah [2].

Printer 2D beresolusi tinggi dan Printer 3D, memungkinkan pembuat produk bajakan untuk membuat produk-produk dengan kualitas hampir sama dengan produk asli. Pemindai 3D juga pembuat produk bajakan untuk menyalin produk asli hingga detail terkecil dengan cepat dan mudah. Selain itu, teknologi digital seperti desain grafis dan *software* pemodelan juga memudahkan pelaku pembajakan dalam membuat desain dan cetakan produk tanpa harus membeli hak cipta atau paten produk tersebut [3].

*Copy detection pattern* (CDP) merupakan sebuah matriks pola dengan ukuran tertentu yang peka terhadap salinan yang akan terdegradasi kualitasnya akibat proses *print & scan* (P&S). Jumlah informasi yang hilang atau terdegradasi digunakan untuk membedakan CDP asli atau palsu. CDP ini dapat diimplementasikan ke dalam kode QR. Kode QR yang dilengkapi CDP dalam penelitian ini nantinya akan disebut dengan *secure QR code* (SQR). SQR dapat diimplementasikan pada produk untuk mengautentikasi keaslian produk.

## **1.2 Rumusan Masalah**

Dari permasalahan yang didapatkan dari penelitian sebelumnya, yaitu hasil lokalisasi CDP yang kurang baik jika hanya menggunakan 4 titik acuan (4 titik sudut kode QR), penulis mencoba menggunakan 8 penanda ArUco [4] di sekitar objek CDP untuk meningkatkan kualitas lokalisasi CDP pada kondisi SQR yang kurang ideal (misalnya ada lekukan pada sisi SQR karena SQR ditempelkan pada permukaan tabung).

## **1.3 Tujuan Penelitian**

Tujuan dari penelitian ini adalah untuk mengetahui efektivitas penggunaan penanda ArUco di sekitar CDP untuk membantu melokalisasi CDP yang digunakan dalam autentikasi SQR orisinal atau palsu.

## **1.4 Batasan Penelitian**

Beberapa batasan yang penulis gunakan dalam penelitian ini antara lain:

1. Objek Penelitian: Analisis penggunaan penanda ArUco untuk melokalisasi objek CDP yang digunakan untuk autentikasi SQR orisinal atau palsu.
2. Metode Penelitian: Analisis dilakukan dengan eksperimen yang mencakup verifikasi parameter pembuatan model SQR, pembuatan *dataset* SQR orisinal dan palsu, serta pembuatan model untuk mengautentikasi SQR. Hasil dari penelitian ini adalah mengetahui efektivitas penanda ArUco dalam melokalisasi objek CDP. Hasil tersebut diperoleh dari analisis data yang didapatkan melalui eksperimen.
3. Waktu dan Tempat Penelitian:
  - Waktu penelitian: Agustus 2022 s.d. April 2023
  - Tempat penelitian: Ruang rispro, laboratorium informatika, laboratorium jaringan dan komputer, dan rumah penulis.
4. Populasi dan Sampel: Populasi adalah seluruh *dataset* SQR yang ada. Sampel penelitian adalah *dataset* SQR yang dibuat dan digunakan oleh penulis dalam penelitian.
5. Variabel: Variabel bebas meliputi *dataset* SQR yang dibuat oleh penulis, variabel kontrolnya adalah *environment* dan parameter yang digunakan dalam pemotretan *dataset*, sedangkan variabel terikatnya adalah hasil lokalisasi CDP dan hasil klasifikasi biner oleh model autentikasi (SQR orisinal atau palsu).
6. Hipotesis:
  - Hasil lokalisasi CDP menggunakan delapan penanda ArUco (8 titik) lebih akurat dibandingkan tanpa menggunakan penanda ArUco (4 titik sudut kode QR).

## 7. Keterbatasan Penelitian:

- *Printer* yang digunakan untuk mencetak *dataset SQR* seragam.
- Pemotretan *dataset SQR* dilakukan dalam kondisi pencahayaan yang baik, berasal dari *flash smartphone*.
- Kamera, konfigurasi kamera, sudut dan kondisi pemotretan *dataset* adalah tetap (menggunakan bantuan boks, sehingga jarak objek dengan kamera tetap).
- Penggunaan penanda ArUco memiliki kelemahan, yaitu model deteksi akan sulit mendeteksi penanda ArUco jika ukurannya sangat kecil.

## 1.5 Manfaat Penelitian

- Bagi peneliti dan pengembang SQR selanjunya, mereka dapat menggunakan model SQR, parameter *print & scan* (P&S), dan model autentikasi (klasifikasi CDP orisinal dan palsu) yang memiliki akurasi terbaik.
- Bagi penulis, penelitian ini dapat menambah wawasan, ilmu dan pengetahuan dalam pembuatan tulisan ilmiah, khususnya pada topik spesifik seperti keamanan digital, pengolahan citra gambar, dan pembelajaran mesin.
- Bagi pelaku bisnis, penerapan SQR dalam produk dapat mengurangi risiko dan melindungi produk dari pembajakan.

## 1.6 Sistematika Penulisan

### BAB I : PENDAHULUAN

Pada bab ini dijelaskan latar belakang, rumusan masalah, tujuan penelitian, batasan penelitian, manfaat penelitian, dan sistematika penulisan.

### BAB II : TINJAUAN PUSTAKA DAN LANDASAN TEORI

Pada bab ini dijelaskan teori-teori dan penelitian terdahulu yang digunakan sebagai acuan dan dasar dalam penelitian.

### BAB III : METODOLOGI PENELITIAN

Pada bab ini dijelaskan metode yang digunakan dalam penelitian meliputi alat dan bahan, metode, serta tahapan dan alur penelitian.

### BAB IV : HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan hasil penelitian dan pembahasannya.

## **BAB V : KESIMPULAN DAN SARAN**

Pada bab ini ditulis kesimpulan akhir dari penelitian dan saran untuk pengembangan penelitian selanjutnya.

## BAB II

### TINJAUAN PUSTAKA DAN DASAR TEORI

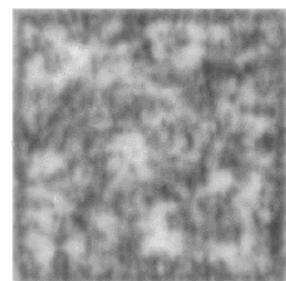
#### 2.1 Tinjauan Pustaka

##### 2.1.1 Apakah Pola Deteksi Duplikat (CDP) dapat Disalin

CDP dinilai dapat digunakan untuk mendeteksi pemalsuan, sehingga akhir-akhir ini mendapatkan banyak perhatian dari akademisi dan industri. Tingkat keamanan CDP dalam mendeteksi serangan pemalsuan yang canggih telah dipelajari secara teoritis dan praktis dalam beberapa penelitian, namun hasilnya masih belum sepenuhnya meyakinkan [5]. Kontribusi utama dari penelitian ini adalah untuk menyajikan *dataset* secara publik dan berbagai jenis contoh penyerangan terdapat CDP tersebut, sehingga kinerja CDP terhadap beberapa penyerangan dapat diketahui. *Dataset* CDP tersebut terdiri dari lebih dari 27.500 gambar CDP dan merupakan *dataset* CDP terbesar hingga saat ini [5]. Kontribusi selanjutnya adalah meneliti tentang kinerja detektor CDP dalam mendeteksi CDP orisinal ataupun salinan. Verifikasi dari CDP dapat dilakukan dengan perangkat seluler, tanpa harus menggunakan pemindai khusus [6], [7]. Dengan *dataset* dan detektor yang dibuat, peneliti sebelumnya ingin menguji validitas hipotesis bahwa CDP yang dicetak berulang kali akan terdegradasi kualitasnya dan dapat dideteksi sebagai CDP palsu [5]. Kontibusi terakhir dari penelitian ini adalah meneliti beberapa metode yang dapat dilakukan untuk meningkatkan performa klasifikasi dari model.



(a)



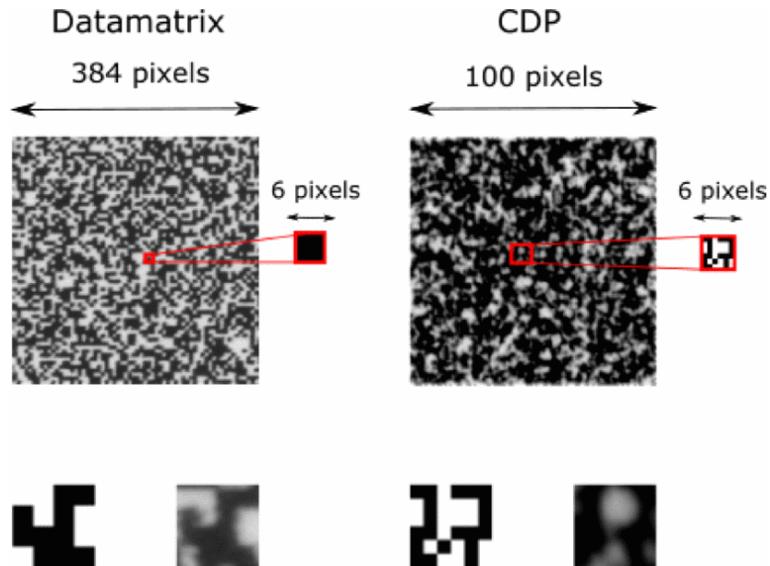
(b)

Gambar 2.1. Contoh dari CDP a) CDP *template* digital ( $I$ ) b) CDP yang telah terdegradasi kualitasnya akibat proses P&S ( $\tilde{I}$ ) [5]

###### 2.1.1.1 Prinsip Degradasi Informasi

Prinsip dari pendekslan CDP palsu dilakukan berdasarkan hilangnya informasi, yang mana muncul dari proses P&S [8]. Proses P&S merupakan proses stokastik (mempunyai unsur peluang atau kebolehjadian) [9], yang menyebabkan perubahan struktur dan kualitas gambar pada CDP, seperti yang ditampilkan pada Gambar 2.1 derau yang dihasilkan dari proses P&S CDP sulit untuk dikarakterisasi [10] karena setiap *printer* dan

pemindai memiliki karakteristiknya sendiri.



Gambar 2.2. Perbandingan dari CDP dengan data-matriks: Data-matriks memiliki ukuran unit komponen yang lebih besar, sehingga degradasi informasi tidak berdampak signifikan pada struktur kode [5]

CDP sering dibandingkan dengan kode batang dua dimensi seperti data-matriks karena kemiripan visualnya. Namun, seperti yang diilustrasikan pada Gambar 2.2, unit elemen data-matriks jauh lebih besar dibandingkan unit elemen CDP. Oleh karena itu, prinsip kehilangan atau degradasi informasi tidak berdampak pada struktur data-matriks. Sebagai contoh, korelasi antara data-matriks *template* digital hasil *generate* dengan versi rusak akibat proses P&S bisa lebih dari 0,9, sedangkan korelasi antara CDP *template* digital hasil *generate* dengan versi rusak akibat proses P&S-nya hanya sekitar 0,45 s.d. 0,55 tergantung pada perangkat pemindai dan pencetaknya. Oleh karena itu, ukuran dari unit elemen pola CDP, yaitu  $100 \times 100$  piksel seperti yang ditampilkan pada Gambar 2.2, ukuran pola keseluruhan juga sangat mempengaruhi proses autentikasi dan kemampuan pemalsu dalam mereproduksi pola. Pada praktiknya, ukuran unit elemen pada CDP adalah  $1 \times 1$  piksel atau  $2 \times 2$  piksel agar bisa memanfaatkan prinsip degradasi informasi secara maksimal.

### 2.1.1.2 Definisi Teoritis dari Sistem Autentikasi CDP

Autentikasi dari CDP terdiri dari dua langkah utama. Langkah pertama adalah langkah registrasi di mana pola dihasilkan kemudian dicetak dengan pencetak untuk menghasilkan CDP orisinal. Langkah kedua adalah verifikasi CDP, menggunakan sebuah perangkat pemindai yang telah terautentikasi (perangkat seluler dengan kamera), CDP dipindai dan dilewatkan melalui tes autentikasi (dengan parameter-parameter pemindaian tertentu). Jika tes tersebut positif, item dianggap autentik.

Upaya penyerangan yang paling sering dilakukan oleh pembajak adalah sebagai

berikut: Pembajak melakukan pemindaian CDP pada sebuah item menggunakan pemin-dai beresolusi tinggi, mengestimasi pola asli dari CDP, kemudian mencetak pola yang telah diestimasi menggunakan pencetak beresolusi tinggi. Pada skenario ini  $I$  merupakan CDP *template* digital hasil *generate*, kemudian  $\Pi(I)$  merupakan CDP *template* digital yang dicetak, dengan  $\Pi(\cdot)$  *noise* yang dihasilkan dalam proses pencetakan menggunakan pencetak yang telah terautentikasi. Selanjutnya, proses autentikasi dapat dirumuskan sebagai uji hipotesis berikut ini:

$$\begin{aligned}\mathcal{H}_0 : \tilde{I} &\sim \Sigma(\Pi(I)), \\ \mathcal{H}_1 : \tilde{I} &\not\sim \Sigma(\Pi(I)),\end{aligned}\tag{2-1}$$

di mana  $\tilde{I}$  adalah gambar CDP *grayscale* yang diterima oleh pusat autentikasi.  $\tilde{I}$  dapat berupa CDP orisinal (i.e.  $\Sigma(\Pi(I))$ ) atau CDP palsu (i.e.  $\Sigma(\Pi'(\hat{I}))$ ). Metriks yang digunakan untuk membandingkan CDP orisinal dengan palsu adalah koefisien jarak ataupun korelasi [11].

### 2.1.1.3 Komponen dari Detektor

Hasil menunjukkan bahwa autentikasi menggunakan CDP *raw grayscale* lebih efisien dibandingkan dengan CDP yang telah di-*thresholding* [9]. Kemudian, secara umum, ada beberapa langkah yang dilakukan untuk melakukan autentikasi CDP, antara lain:

- Melakukan *resizing* pada *template* CDP menggunakan faktor skala tertentu.
- Menggunakan teknik pencocokan *template* dengan mengambil sub-bagian dari CDP.
- Menggunakan *high pass filtering* (seperti *unsharp masking*) sebelum mela-kukan penyekoran korelasi.

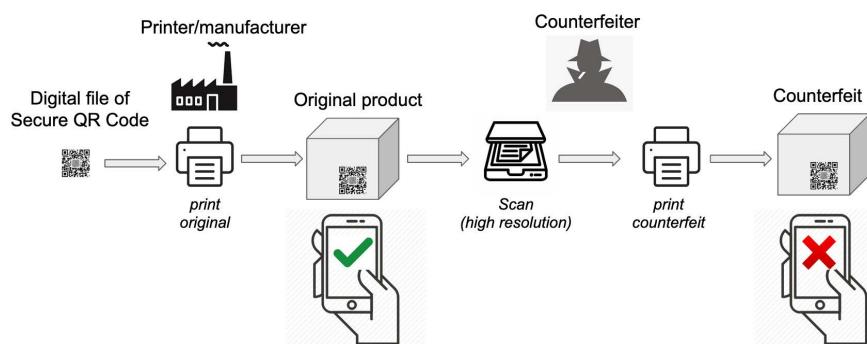
### 2.1.2 Deteksi Pembajakan menggunakan SQR

Pendekatan keamanan tradisional pada produk sebelumnya sudah diimplementasikan melalui *taggant*, hologram, dan tinta keamanan. Beberapa metode tersebut memang mudah diimplementasikan, mudah diverifikasi, dan murah. Namun, dalam sebuah kode QR metode-metode tersebut belum dapat diimplementasikan. Produk yang ditempel oleh kode QR palsu biasanya masih dapat dipindai dan mengembalikan keluaran sama dengan produk orisinal. Penelitian yang dilakukan oleh Justin Picard, Paul Landry, dan Michael Bolay ini membahas tentang bagaimana mengimplementasikan CDP ke dalam SQR untuk mendekripsi kode QR palsu [12].



Gambar 2.3. Perbandingan dari kode QR orisinal dan palsu, keduanya menyimpan informasi yang sama dan sama-sama dapat dipindai [12]

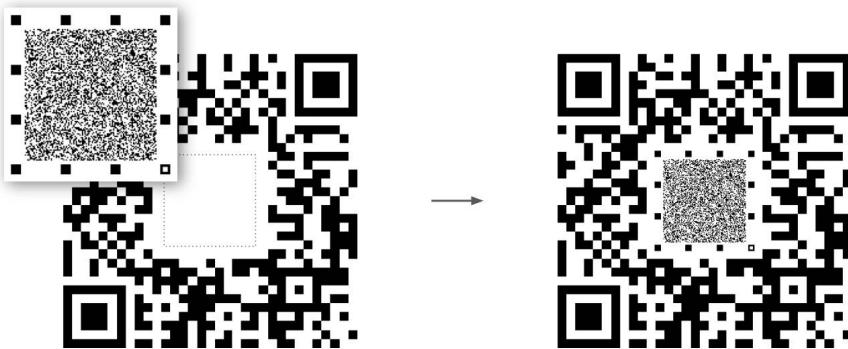
Pada Gambar 2.3 terlihat bahwa sangat sulit untuk membedakan antara kode QR orisinal dan palsu hasil replikanya, apalagi jika tidak ada pembandingnya. Skenario yang dapat dilakukan oleh pembajak dalam menerbitkan kode QR palsu dapat dilihat pada Gambar 2.4. Kode QR palsu dipindai dengan perangkat beresolusi tinggi, kemudian dicetak ulang. Harapannya dengan CDP yang diletakkan pada kode QR, kode QR palsu yang dipindai untuk autentifikasi dapat terdeteksi sebagai kode QR palsu, lihat pada Gambar 2.4.



Gambar 2.4. SQR diharapkan dapat mendeteksi kode QR palsu pada saat diautentikasi oleh pengguna [12]

### 2.1.2.1 Struktur dari SQR

Secara umum, CDP akan diletakkan di tengah-tengah dari kode QR. CDP nantinya akan digunakan untuk proses autentifikasi. Di sekitar CDP diletakkan beberapa penanda untuk memudahkan program dalam mendeteksi dan mendapatkan objek CDP (lokalisasi CDP). CDP mungkin ditempelkan di tengah-tengah kode QR karena adanya fitur koreksi kesalahan dalam kode QR. Ada empat jenis koreksi kesalahan pada kode QR: L, M, Q, dan H, yang secara teoritis memiliki kemampuan untuk memulihkan informasi yang hilang sebesar 7%, 15%, 25%, dan 30% kerusakan pada kode QR. Pada penelitian ini, area yang dirusak untuk meletakkan CDP adalah sebesar 1/9 dari ukuran



Gambar 2.5. Struktur SQR secara umum [12]

awal kode QR. Aplikasi yang digunakan untuk melakukan autentikasi pada perangkat pemindai (perangkat seluler) hanya mengambil objek CDP saja, hal tersebut memiliki beberapa keuntungan, area yang digunakan untuk autentikasi menjadi lebih spesifik, sehingga kecepatan deteksi akan lebih cepat, fokus dari kamera juga relatif akan lebih baik karena mengambil objek yang lebih kecil dan terfokus, selain itu karena sistem autentikasi berada di *remote server* yang mana data akan dikirim dari perangkat seluler pemindai, semakin kecil area yang dikirimkan, semakin hemat *bandwidth* yang digunakan.

### 2.1.2.2 Pembuatan dan Pencetakan SQR

CDP yang digunakan dalam penelitian ini menggunakan dua level *grayscale* atau biner. Resolusi mesin *printer* yang digunakan adalah 812,8 ppi dengan merek HP Indigo. CDP di-*generate* menggunakan *pseudo-random number generator*, menggunakan *seed* tertentu. CDP dapat dibuat unik untuk setiap kode QR ataupun sama pada sekelompok kode QR tertentu. Dalam melakukan pencetakan dan pemindaian SQR, perangkat pencetak dan pemindai akan diverifikasi terlebih dahulu dengan konfigurasi dan parameter tertentu untuk menjamin kualitas dari pencetakan.

### 2.1.2.3 Autentikasi SQR

Perangkat pemindai QR biasa sudah pasti dapat melakukan *decode* informasi dari kode QR, namun untuk melakukan autentikasi terhadap CDP, tentunya diperlukan aplikasi khusus. Aplikasi khusus ini berjalan di perangkat seluler, secara umum proses pemindaian yang dilakukan oleh aplikasi khusus tersebut adalah sebagai berikut:

- Aplikasi akan melakukan pemindaian per-*frame* dari kamera hingga mendapatkan kode QR.
- Kode QR akan di-*decode* untuk mengekstrak informasi dalam kode QR.
- Indeks kualitas pemindaian akan dikalkulasi berdasarkan *frame* yang didapatkan.

- Jika indeks kualitas pemindaian dinilai cukup tinggi, area CDP akan dideteksi, dipotong, dan dikirim ke *remote server* bersamaan dengan informasi dari kode QR yang telah ter-*decode* untuk autentikasi.



Gambar 2.6. Perangkat seluler melakukan pemindaian menggunakan aplikasi khusus [12]

Setelah *server* mendapatkan CDP yang telah dipotong beserta informasi data kode QR, autentikasi yang dilakukan di *server* adalah sebagai berikut:

- *Identifier* unik akan diekstrak dari informasi kode QR yang didapatkan yang mana dibutuhkan sebagai parameter autentikasi.
- *Template* CDP digital akan di-*generate*.
- Matriks similaritas akan dikalkulasi dari perbandingan antara CDP yang dikirimkan dari hasil pemindaian dengan CDP *template* yang di-*generate*.
- Penyesuaian lain seperti, ketajaman gambar, filter akan dilakukan untuk memperoleh hasil terbaik.
- Normalisasi nilai fitur akan dilakukan.
- Keluaran berupa CDP "orisinal", "palsu", atau "pemindaian buruk" akan dikeluarkan oleh *server* (pemindaian buruk bisa disebabkan oleh hasil gambar yang kabur).

### 2.1.3 Autentikasi Digital menggunakan CDP

Penelitian ini memberikan solusi atas permasalahan pembajakan produk dengan membuat sebuah gambar digital dengan properti tertentu, yang disebut dengan *Copy detection patterns* (CDP), yang akan diletakkan pada produk ataupun dokumen. Penelitian ini merupakan salah satu penelitian awal dari penelitian-penelitian selanjutnya tentang pengembangan CDP untuk deteksi pembajakan. Permasalahan yang coba dibahas pada

penelitian ini adalah apakah mungkin untuk membuat dokumen salinan yang sempurna? Untuk menjawab pertanyaan tersebut peneliti melakukan eksperimen untuk mendemonstrasikan kelayakan penerapan CDP untuk mengamankan dokumen dari pemalsuan [8].

Spesifikasi CDP dan perangkat yang digunakan dalam penelitian adalah sebagai berikut:

- Gambar CDP berukuran 100x100 piksel, dicetak dengan ukuran 0,72cm<sup>2</sup>.
- *Printer*: Printer laser kantor standar dengan 1200 dpi.
- *Pemindai*: Canon 1240 pada 300 dpi *grayscale* dan 200 dpi *grayscale*.

Kemudian, ada tiga jenis data kopian yang diproduksi, yaitu:

- Salinan pemindai: Dibuat menggunakan Epson 1650 pada 2400 dpi dan dicetak ulang dengan *printer* dan kertas yang sama dengan CDP orisinal.
- Fotokopi berwarna
- Pola yang dibuat ulang

Hasil dari percobaan tersebut adalah:

- Indeks kualitas maksimum 100 hanya diperoleh dengan CDP digital. Proses P&S menyebabkan penurunan kualitas, pembuatan salinan dari cetakan pertama juga menyebabkan penurunan kualitas lagi.
- Resolusi pemindaian yang lebih rendah memengaruhi hasil secara signifikan pada cetakan orisinal dibandingkan pada cetakan salinan. Namun, perbedaan antara dokumen asli dan salinan tetap dapat dibedakan secara signifikan.
- Salinan yang dibuat ulang dengan mengestimasi pola memiliki indeks kualitas mendekati nol, hal tersebut disebabkan nilai pikselnya hampir tidak berkorelasi sama sekali dengan *template* digital.
- Salinan berkualitas lebih baik dapat dibuat dengan mencetak ulang hasil pindaiannya beresolusi tinggi pada kertas yang memiliki noda tinta lebih sedikit (misal kertas foto) dan dengan *printer* yang lebih baik. Namun, membuat salinan seperti itu akan sangat menambah beban pemalsu. Selain itu, hasil salinannya akan terlihat berbeda dengan mata telanjang dan tetap memiliki indeks kualitas yang jauh lebih rendah daripada aslinya.

Kesimpulan yang didapatkan dari penelitian ini [8] antara lain:

- Setiap kali gambar dicetak dan dipindai (proses P&S), kualitasnya akan menurun dan terdegradasi. Hal ini disebabkan oleh sifat dari *printer*, kertas, dan pemindai.
- Sebagian besar gambar jenis dokumen belum menerapkan prinsip degradasi informasi untuk melindungi produk dari pembajakan.

- Berlawanan dengan gambar dokumen, gambar yang terdiri dari derau mur-ni dapat terdegradasi secara maksimal, sehingga akan mudah membedakan gambar orisinal dan palsunya.
- CDP adalah gambar dengan entropi maksimum yang dihasilkan dengan menggunakan sebuah kunci rahasia. CDP umumnya disisipkan pada gambar digital yang akan dicetak atau langsung dicetak pada dokumen.
- CDP tidak cocok untuk dideteksi dengan mata telanjang, namun akan sangat efektif dideteksi menggunakan komputer.

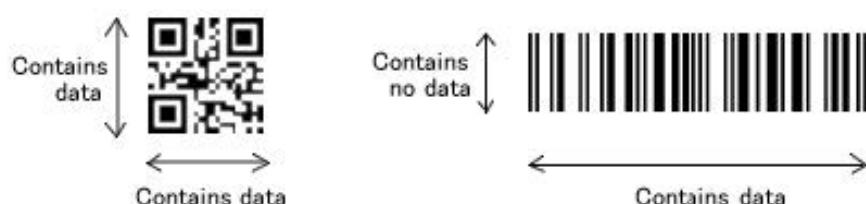
## 2.2 Dasar Teori

### 2.2.1 Kode QR

Kode QR (*Quick Response*) adalah sebuah kode matriks dua dimensi (2-D) yang dapat dibaca oleh komputer. Kode QR dua dimensi dapat menyimpan data yang lebih banyak dibandingkan dengan kode satu dimensi (*barcode*) dengan ruang yang lebih kecil. Selain itu, kode QR memiliki fitur koreksi kesalahan pembacaan dan beberapa fitur unik lainnya [13].

Seperti bahasa tertulis lainnya, kode batang atau *barcode* merupakan representasi visual dari informasi. Namun, berbeda dengan bahasa yang dapat dibaca oleh manusia, kode batang dirancang untuk dibaca dan dipahami oleh komputer atau mesin. Menggunakan sistem penglihatan dari mesin berupa pemindai laser optik ataupun kamera dan perangkat lunak yang dapat menginterpretasikan kode batang. Aturan bagaimana *barcode* dikonstruksikan disebut sebagai *grammar*, sedangkan set karakter yang digunakan (alfabet) disebut *symbology* [13].

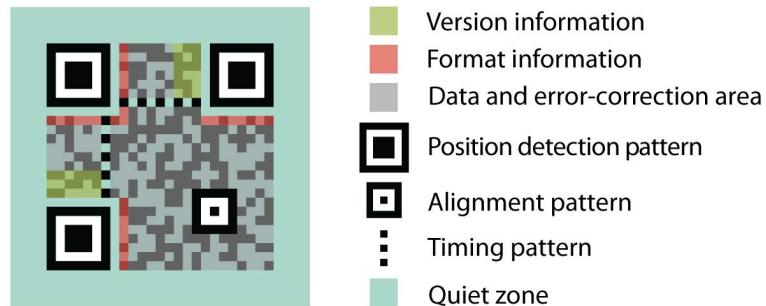
#### 2.2.1.1 Bagaimana Kode QR Bekerja



Gambar 2.7. Perbandingan 1-D dengan 2-D *barcode* [13]

Tidak seperti kode batang satu dimensi, kode QR adalah matriks 2-D yang menyimpan informasi dalam tiap modul di baris dan kolomnya yang memiliki gelap dan terang tertentu.

Setiap modul dalam kode QR memiliki fungsi-fungsi tertentu. Beberapa modul berisi tentang informasi yang tersimpan dalam kode QR itu sendiri, dengan lainnya dibagi menjadi beberapa grup berdasarkan fungsinya. Untuk memastikan kode QR dapat

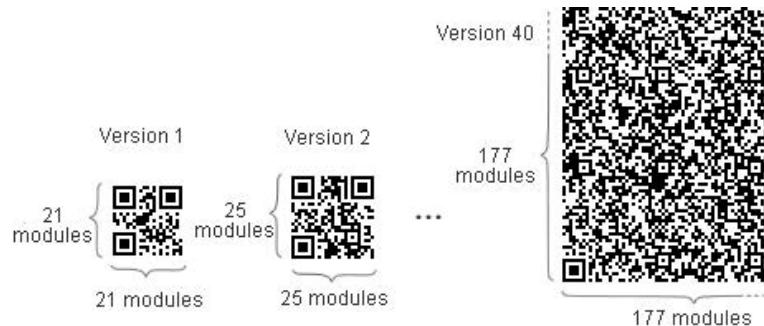


Gambar 2.8. Struktur modul pada kode QR dua dimensi [13]

dipindai dari berbagai sisi, ada tiga modul *position detection pattern* yang terletak di ketiga sudut yang memungkinkan kode QR untuk dipindai dari 360°.

### 2.2.1.2 Versi Kode QR

Kode QR dapat di-*generate* dari 40 versi yang berbeda, dari yang berukuran 21x21 modul (versi 1) hingga 177x177 modul (versi 40).



Gambar 2.9. Jumlah modul berdasarkan versi kode QR

Setiap kenaikan satu versi, maka akan ada penambahan 4 modul, sehingga dapat memuat data atau informasi yang lebih banyak. Jumlah maksimum data yang dapat disimpan tergantung pada versi kode QR, tipe karakter, dan besar toleransi kesalahan.

### 2.2.1.3 Koreksi Kesalahan Kode QR

Koreksi kesalahan kode QR mengimplementasikan *Reed-Solomon codes*, yang mana merupakan salah satu metode koreksi kesalahan matematis yang banyak digunakan. Hal ini memungkinkan kode QR tetap dapat dibaca walaupun dalam kondisi kotor ataupun rusak dengan batasan tertentu. Ada empat tipe koreksi kesalahan standar yang ada dalam kode QR. Semakin tinggi level koreksi kesalahan, semakin besar toleransi terhadap kerusakan kode QR, namun semakin besar juga versi kode QR-nya.

Dalam memilih level koreksi kesalahan, sebaiknya disesuaikan dengan kondisi lingkungan di mana kode QR tersebut digunakan. Misalnya dalam kondisi lingkung-

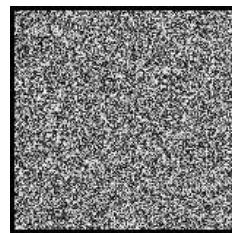
Tabel 2.1. Tabel perbandingan level koreksi kesalahan dengan persentase toleransi kesalahannya

Level Koreksi Kesalahan	Besar Toleransi Kesalahan
L	7%
M	15%
Q	25%
H	30%

an yang bersih, level L (7%) dapat digunakan. Secara umum, level yang paling sering digunakan adalah level M (15%).

### 2.2.2 *Copy Detection Pattern (CDP)*

Pola deteksi duplikat (CDP) adalah sebuah gambar berpola dengan entropi tinggi yang dibuat menggunakan kode rahasia (*secret key*). CDP memanfaatkan konsep dari "*information loss principle*" dari proses P&S pada dokumen. Biasanya, CDP digunakan di dalam gambar digital yang dicetak ataupun langsung ke dalam dokumen digitalnya. CDP tidak didesain untuk dideteksi menggunakan mata telanjang. Namun, CDP dapat bekerja secara maksimal pada pendekripsi otomatis pada gambar yang dipindai. CDP akan sangat bermanfaat saat digunakan dalam memverifikasi dokumen dalam jumlah besar [8], [14].



Gambar 2.10. Salah satu contoh CDP

CDP telah dicoba untuk dicetak menggunakan berbagai variasi *printers*: *Printer* kantor seperti *inkjet* dan *laser*, *printer offset* dan *digital offset*, dan juga *printer* termal. Hasil percobaan dari pencetakan menggunakan beberapa jenis *printer* tadi, semua CDP salinan dapat dibedakan dengan CDP orisinal dengan margin yang cukup nyaman [8], [14], [15].

CDP sebaiknya tidak dianggap sebagai pesaing untuk perangkat keamanan optik lainnya, namun dijadikan sebagai alternatif yang lebih murah pada kasus-kasus tertentu. Dengan memanfaatkan konsep degradasi gambar dan informasi yang tidak terancam oleh perkembangan perangkat pemindai dan pencetak digital, CDP menjadi alternatif yang paling murah dalam memroteksi dokumen [8], [14], [15].

### **2.2.3 Lokalisasi Objek dengan Pengenalan Pola**

Lokalisasi objek adalah proses mengidentifikasi posisi dan orientasi objek atau pola tertentu pada sebuah gambar menggunakan teknik pengolahan citra dan visi komputer. Proses ini melibatkan deteksi objek atau pola dalam gambar serta mengestimasi lokasi dan orientasi yang tepat relatif terhadap kamera [16], [17].

Berbagai teknik telah diusulkan untuk melakukan lokalisasi objek atau gambar dengan pengenalan pola, termasuk deteksi fitur, pencocokan *template*, dan metode berbasis pembelajaran mesin. Teknik-teknik ini telah digunakan dalam berbagai aplikasi, seperti *augmented reality*, robotika, dan pelacakan objek.

### **2.2.4 ArUco Marker**

ArUco *marker* adalah jenis *marker* fidusial yang biasa digunakan untuk estimasi pose kamera dan pelacakan dalam pengolahan citra dan visi komputer. ArUco *marker* terdiri dari kisi-kisi kotak hitam dan putih dengan pola unik yang mudah dideteksi dan dikenali oleh algoritma visi komputer. ArUco *marker* banyak digunakan dalam aplikasi robotika, realitas tambahan, dan pelacakan objek karena kesimpelannya, akurasi deteksi yang tinggi, dan biaya komputasi yang rendah [4].

*Marker* fidusial adalah objek berpolai yang digunakan ke dalam sebuah gambar atau tempat tertentu untuk memudahkan sistem visi komputer menentukan posisi dan orientasi objek atau *scene* dengan akurasi yang tinggi. *Marker* fidusial biasanya didesain dengan pola atau bentuk yang unik dan mudah dikenali, sehingga dapat dideteksi dan dilacak oleh algoritme pendekripsi objek visi komputer, yang memungkinkan estimasi pose dan pelacakan yang akurat. *Marker* fidusial banyak digunakan dalam berbagai aplikasi seperti *augmented reality*, robotika, dan pendekripsi objek [4].

### **2.2.5 Fitur Spasial pada Citra Gambar**

Fitur spasial pada citra gambar adalah fitur yang berkaitan dengan letak, bentuk, dan struktur dari objek dalam citra. Fitur ini melibatkan ekstraksi informasi spasial dari citra, seperti tingkat kecerahan, kejelasan, sudut, tepi, dan tekstur [18].

Jenis fitur spasial yang umum digunakan dalam pemrosesan citra antara lain adalah [18]:

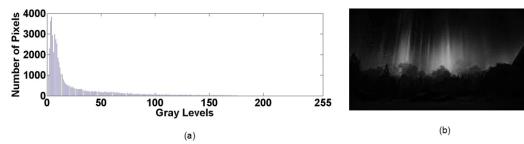
1. Tepi: fitur yang mengidentifikasi perubahan tajam dalam intensitas piksel dalam citra dan digunakan untuk menghasilkan kontur atau garis tepi dari objek dalam citra.
2. Tekstur: fitur yang menggambarkan pola dan struktur permukaan dari objek dalam citra, seperti kekasaran, kepadatan, dan konsistensi.

3. Ruang warna: fitur yang menggambarkan kualitas warna dan perbedaan warna dalam citra, seperti hue, saturasi, dan nilai.
4. Bentuk: fitur yang mengidentifikasi bentuk dan ukuran objek dalam citra, seperti lingkaran, segitiga, atau persegi panjang.

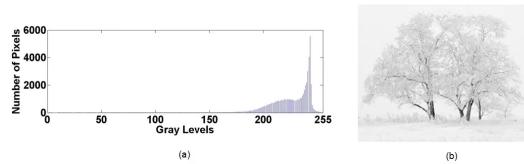
Fitur spasial ini dapat diekstraksi dan digunakan untuk melatih model deep learning untuk melakukan tugas tertentu pada citra, seperti klasifikasi objek, deteksi objek, atau segmentasi citra. Dalam pemrosesan citra, fitur spasial merupakan komponen penting dalam analisis dan pengolahan citra [18].

### 2.2.6 Fitur Histogram pada Citra Gambar

Histogram citra adalah grafik yang menunjukkan frekuensi kemunculan intensitas piksel dalam citra. Histogram digunakan untuk mengidentifikasi dan memahami distribusi intensitas cahaya pada gambar. Histogram dapat digunakan dalam banyak aplikasi pemrosesan citra, seperti koreksi warna, segmentasi, ekstraksi fitur, dan pengenalan pola [19]. Gambar berwarna akan memiliki tiga histogram, satu untuk setiap saluran warna (merah, hijau, dan biru), sedangkan gambar hitam-putih hanya akan memiliki satu histogram.



Gambar 2.11. Histogram pada gambar dengan pencahayaan gelap [19]

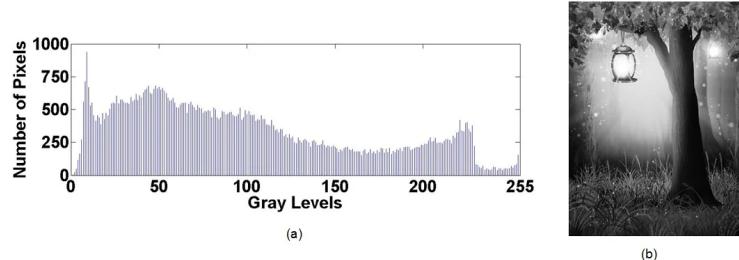


Gambar 2.12. Histogram pada gambar dengan pencahayaan terang [19]

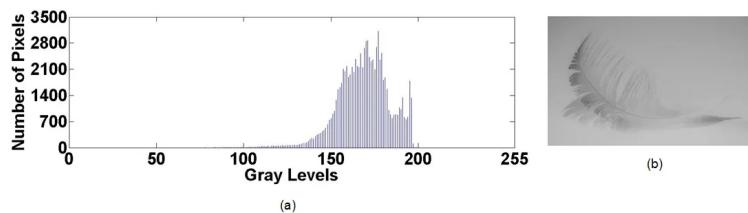
Histogram citra memiliki beberapa karakteristik penting yang dapat digunakan untuk analisis citra. Pertama, histogram dapat digunakan untuk melihat distribusi level dari piksel gambar. Kedua, histogram dapat digunakan untuk mengidentifikasi kecerahan pada gambar, jika nilainya terkonsentrasi ke kiri, maka citra gambar tersebut relatif gelap. Jika konsentrasi condong ke kanan, maka citra gambar tersebut terang. Perbandingan distribusi histogram pada citra gambar gelap dan terang dapat dilihat pada Gambar 2.11 dan Gambar 2.12.

Ketiga, histogram dapat digunakan untuk mengidentifikasi kontras pada citra gambar. Histogram yang terdistribusi secara merata menunjukkan bahwa gambar memiliki kon-

tras yang baik, sedangkan histogram yang terdistribusi terfokus pada bagian tertentu menunjukkan kontras gambar yang rendah. Perbandingan distribusi histogram pada citra gambar kontras tinggi dan kontras rendah dapat dilihat pada Gambar 2.13 dan Gambar 2.14.

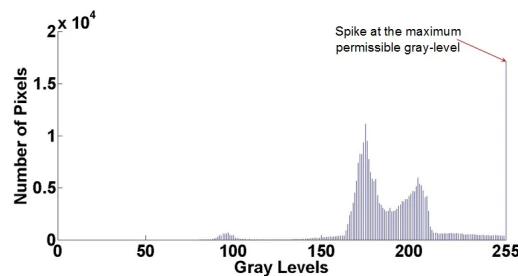


Gambar 2.13. Histogram pada gambar dengan kontras tinggi [19]



Gambar 2.14. Histogram pada gambar dengan kontras rendah [19]

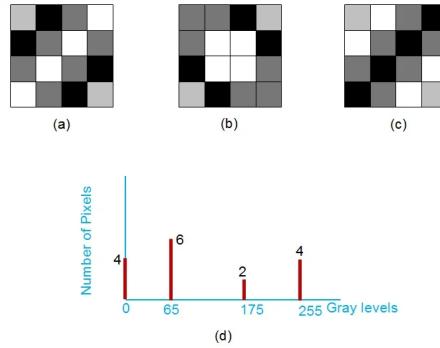
Keempat, histogram juga dapat digunakan untuk melihat efek saturasi pada gambar. Jika pada histogram terdapat nilai dengan *spike* yang tinggi, hal tersebut berarti ada piksel gambar yang mengalami kerusakan akibat saturasi tinggi. Contoh distribusi histogram gambar yang memiliki saturasi tinggi dapat dilihat pada Gambar 2.15



Gambar 2.15. Histogram pada gambar dengan saturasi tinggi [19]

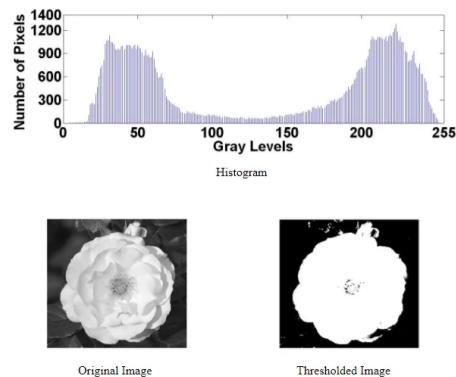
Namun, fitur histogram dari sebuah citra gambar memiliki kelemahan, yaitu tidak memberikan informasi mengenai distribusi spasial dari nilai piksel citra gambar. Dengan demikian, dapat ditemui gambar berbeda yang memiliki distribusi histogram yang sama, seperti yang dapat dilihat pada Gambar 2.16.

Fitur histogram pada gambar juga dapat digunakan untuk melakukan *thresholding*, yaitu mengubah citra *grayscale* yang bergradasi menjadi beberapa level saja dengan ambang batas tertentu untuk tiap-tiap levelnya. Histogram merupakan salah satu

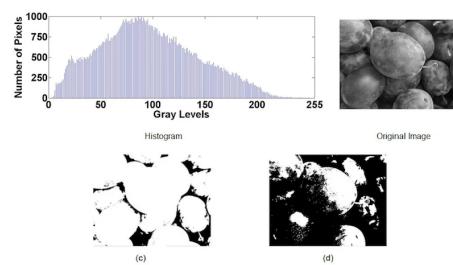


Gambar 2.16. Gambar yang berbeda memiliki distribusi histogram yang sama [19]

cara mudah untuk mengidentifikasi ambang batas yang sesuai. pada Gambar 2.17, nilai piksel terkonsentrasi dalam dua kelompok, sehingga ambang batas dapat ditentukan yaitu nilai tengah kedua kelompok tersebut. Pada Gambar 2.18, sifat histogram lebih kontinyu, menunjukkan bahwa gambar tersebut bukan kandidat yang baik untuk *thresholding* karena untuk menentukan nilai pembatas yang ideal akan sulit.



Gambar 2.17. Gambar yang ideal untuk di-*thresholding* [19]



Gambar 2.18. Gambar yang tidak ideal untuk di-*thresholding* [19]

### 2.2.7 Fitur DCT pada Citra Gambar

DCT (*Discrete Cosine Transform*) adalah transformasi matematis yang digunakan untuk mengubah sinyal digital atau data spasial menjadi domain frekuensi. DCT menghasilkan serangkaian koefisien frekuensi yang merepresentasikan sinyal atau data spasial dalam bentuk domain frekuensi. Dalam domain frekuensi, sinyal atau data citra direpresentasikan oleh koefisien-koefisien frekuensi. DCT memanfaatkan sifat alami dari citra dan memungkinkan informasi citra yang relevan dikonsentrasi dalam beberapa koefisien frekuensi teratas, sedangkan koefisien frekuensi yang lain diabaikan atau dikuantiasi dengan rinci. Transformasi ini banyak digunakan dalam pengolahan sinyal, pengolahan citra, dan kompresi data, seperti pada format file JPEG dan MP3 [20].

Dalam aplikasi pengolahan citra, DCT digunakan untuk menghasilkan representasi domain frekuensi dari citra digital. Hal ini memungkinkan penghapusan informasi yang tidak penting pada citra. Dalam kompresi audio, DCT digunakan untuk memisahkan sinyal audio menjadi beberapa subband frekuensi [20].

DCT juga banyak digunakan pada pembandingan citra dan gambar. Teknik ini dapat digunakan untuk mendeteksi keaslian citra dan *watermarking*. Dalam pembandingan citra dan gambar, DCT digunakan untuk memperoleh fitur-fitur citra yang berbeda. Fitur-fitur ini kemudian digunakan untuk menghitung kesamaan antara dua citra atau gambar. Dalam aplikasi deteksi keaslian citra, DCT dapat digunakan untuk menghasilkan *signature* unik dari setiap citra asli. Sementara dalam *watermarking*, DCT digunakan untuk memasukkan *watermark* atau pesan rahasia ke dalam citra [21].

Secara umum, persamaan DCT pada citra 2 dimensi (gambar berukuran N x M) didefinisikan dengan [20]:

$$F(u, v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i). \Lambda(j). \cos\left[\frac{\pi \cdot u}{2 \cdot N}(2i+1)\right] \cos\left[\frac{\pi \cdot v}{2 \cdot M}(2j+1)\right] \cdot f(i, j) \quad (2-2)$$

### 2.2.8 Koefisien Jarak

Koefisien jarak merupakan ukuran atau besaran yang menggambarkan seberapa dekat atau jauh dua objek dalam ruang atau dimensi tertentu. Koefisien jarak dapat digunakan untuk mengukur kesamaan atau perbedaan antara dua objek yang diamati. Ada beberapa koefisien jarak yang sering digunakan, antara lain:

#### 2.2.8.1 Koefisien Jarak Euclidean

Jarak *euclidean* adalah ukuran jarak yang paling umum digunakan dalam matematika dan ilmu komputer untuk mengukur jarak antara dua titik dalam ruang *euclidean* n-dimensi. Jarak *euclidean* dihitung sebagai akar kuadrat dari jumlah kuadrat perbedaan

an koordinat antara dua titik. Secara formal, jarak *euclidean* antara dua vektor  $u$  dan  $v$  dalam ruang *euclidean* n-dimensi didefinisikan sebagai:

$$d(u, v) = \sqrt{\sum_{i=1}^n (u_i - v_i)^2} \quad (2-3)$$

di mana  $n$  adalah jumlah dimensi dalam *vector space*, dan  $u_i$  dan  $v_i$  merepresentasikan komponen ke- $i$  pada vektor.

### 2.2.8.2 Koefisien Jarak Korelasi

Koefisien jarak korelasi adalah suatu ukuran kemiripan atau perbedaan antara dua vektor, berdasarkan korelasi antara komponen-komponennya. Nilai dari jarak korelasi di-normalisasi antara 0 dan 1, di mana 0 menunjukkan korelasi positif sempurna sedangkan 1 menunjukkan korelasi negatif sempurna.

Untuk menghitung jarak korelasi antara dua vektor  $u$  dan  $v$ , dapat dituliskan dengan:

$$d(u, v) = 1 - \frac{(u - \bar{u}) \cdot (v - \bar{v})}{\|(u - \bar{u})\|_2 \|(v - \bar{v})\|_2} \quad (2-4)$$

di mana  $d(u, v)$  adalah jarak korelasi antara  $u$  dan  $v$ ,  $\bar{v}$  adalah rata-rata elemen dari vektor  $v$ , dan  $x \cdot y$  adalah *dot product* dari  $x$  dan  $y$ . Jarak korelasi sering digunakan dalam algoritma pengelompokan dan klasifikasi untuk mengukur perbedaan antara sampel, terutama ketika vektor memiliki banyak komponen dan data sangat berkorelasi.

### 2.2.8.3 Koefisien Jarak Kosinus

Koefisien jarak kosinus adalah sebuah metode untuk mengukur kemiripan antara dua vektor dalam ruang n-dimensi [22], [23]. Metode ini mengukur sudut antara dua vektor dan menghasilkan nilai berkisar antara 0 dan 1, di mana 0 menunjukkan bahwa vektor tersebut saling tegak lurus, sedangkan 1 menunjukkan bahwa vektor tersebut saling sejajar. Semakin kecil nilai koefisien jarak kosinus, semakin mirip kedua vektor tersebut. Jarak kosinus antara dua vektor  $u$  dan  $v$  dapat dituliskan sebagai berikut:

$$d(u, v) = 1 - \frac{u \cdot v}{\|u\|_2 \|v\|_2} \quad (2-5)$$

di mana  $u \cdot v$  adalah hasil *dot product* antara vektor  $u$  dan  $v$ , sedangkan  $\|u\|_2$  dan  $\|v\|_2$  adalah panjang dari vektor  $u$  dan  $v$ .

#### 2.2.8.4 Koefisien Jarak Canberra

Jarak Canberra adalah salah satu metode mengukur jarak antara dua vektor dalam ruang n-dimensi. Metode ini mengevaluasi perbedaan proporsional antara nilai-nilai pada setiap elemen vektor. Metode ini sering digunakan dalam analisis data untuk mengukur kemiripan antara dua set data numerik [24], [25]. Secara formal, perhitungan jarak canberra antara dua vektor  $u$  dan  $v$  adalah:

$$d(u, v) = \sum_i \frac{|u_i - v_i|}{|u_i| + |v_i|} \quad (2-6)$$

#### 2.2.9 Transformasi Homografi

Transformasi homografi adalah sebuah transformasi geometri pada ruang n-dimensi yang memetakan setiap titik pada bidang ke titik yang sesuai pada bidang lainnya, dengan menerapkan konsep persamaan linier homogen. Transformasi homografi dapat mengubah ukuran, rotasi, dan persepektif dari gambar atau objek pada bidang ruang n-dimensi.

Transformasi homografi biasanya dilakukan dalam ruang koordinat *homogeneous*, yang merupakan ruang  $n+1$  dimensi dengan koordinat homogen yang memungkinkan dilakukannya transformasi perspektif. Dalam ruang koordinat *homogeneous*, sebuah titik pada bidang n-dimensi dinyatakan dalam bentuk vektor homogen  $(x, y, z, w)$ , di mana  $x$ ,  $y$ , dan  $z$  adalah koordinat euclidean, dan  $w$  adalah koordinat homogen.

#### 2.2.10 Uji Statistik *T-test*

Uji statistik *T-test* adalah sebuah teknik statistik yang digunakan untuk membandingkan rata-rata dari dua sampel independen. Uji ini menghasilkan nilai *t-statistics* dan *p-value* yang dapat digunakan untuk menentukan apakah perbedaan antara kedua sampel signifikan secara statistik [26].

*T-statistics* adalah ukuran dari perbedaan rata-rata antara dua sampel, yang di-normalisasi dengan standar deviasi dari kedua sampel. Semakin besar nilai *t-statistics*, semakin signifikan perbedaan antara kedua sampel. *P-value* adalah nilai probabilitas yang diperoleh dari perhitungan statistik, dan menunjukkan seberapa signifikan perbedaan tersebut. Semakin kecil nilai *P-value*, semakin signifikan perbedaan antara kedua sampel [26].

Dalam uji statistik *T-test* untuk dua sampel independen, hipotesis nol menyatakan bahwa tidak ada perbedaan yang signifikan antara kedua sampel. Jika *P-value* kurang dari alfa (tingkat signifikansi yang telah ditentukan sebelumnya), maka hipotesis nol ditolak dan dapat disimpulkan bahwa terdapat perbedaan signifikan antara kedua sampel [26].

Definisi formal dari uji statistik *T-test* dapat dituliskan dengan [26]:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \quad (2-7)$$

di mana  $\bar{x}_1$  and  $\bar{x}_2$  adalah rata-rata dari sampel,  $s_1$  dan  $s_2$  adalah standar deviasi sampel, dan  $n_1$  dan  $n_2$  adalah ukuran sampel.

### 2.2.11 Pembelajaran Mesin

Pembelajaran mesin adalah sebuah cabang dari kecerdasan buatan yang memungkinkan sistem komputer untuk belajar dari data, mengidentifikasi pola, dan melakukan prediksi dalam menyelesaikan permasalahan. Secara umum, pembelajaran mesin mencoba untuk menemukan pola tersembunyi dalam data dan mempergunakan informasi tersebut untuk menghasilkan hasil yang lebih baik dalam setiap iterasinya.

Sistem pembelajaran mesin dilatih dengan menggunakan data masukan dan berbagai teknik pemrosesan data untuk menghasilkan output yang diharapkan. Proses pembelajaran ini melibatkan pembuatan model matematis dan analisis data. Dalam pembelajaran mesin, data digunakan untuk melatih model atau algoritma, sehingga sistem dapat belajar dan meningkatkan kinerjanya dalam memecahkan masalah atau menyelesaikan tugas tertentu. Secara umum, ada tiga jenis pembelajaran mesin, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. *Supervised learning* melibatkan penggunaan data yang telah dilabeli atau dikategorikan sebelumnya untuk melatih model atau algoritma. *Unsupervised learning* melibatkan penggunaan data yang belum dilabeli atau dikategorikan sebelumnya untuk menemukan pola atau struktur yang terdapat dalam data. Sedangkan *reinforcement learning* melibatkan penggunaan sistem *reward* dan *punishment* untuk melatih model atau algoritma.

Dalam praktiknya, pembelajaran mesin sering digunakan dalam berbagai aplikasi bisnis, seperti analisis data, personalisasi produk, dan deteksi kecurangan. Dalam kehidupan sehari-hari, pembelajaran mesin juga diterapkan dalam pengenalan suara, identifikasi gambar, dan pengembangan sistem kendali otomatis [27], [28], [29].

### 2.2.12 AutoML

AutoML (*Automated Machine Learning*) adalah teknologi yang dirancang untuk memudahkan proses pembuatan model pembelajaran mesin secara otomatis. AutoML semakin populer karena memungkinkan organisasi dan perusahaan untuk menghasilkan model pembelajaran mesin secara cepat dan efisien, sehingga mempercepat inovasi dan pengambilan keputusan dalam bisnis. Namun, AutoML bukanlah solusi yang sempurna dan masih memiliki beberapa tantangan seperti kurangnya interpretasi model dan pengurangan fleksibilitas dalam menghasilkan model yang dapat disesuaikan dengan kebutuhan khusus [30], [31].

AutoML memungkinkan otomatisasi dari beberapa tahapan dalam proses pengembangan model, seperti *pre-processing*, *feature selection*, *hyperparameter tuning*, dan *model selection*. Tujuan dari AutoML adalah untuk mengurangi waktu dan biaya dalam pengembangan model pembelajaran mesin, dan memungkinkan pengguna yang tidak berpengalaman dalam machine learning untuk menghasilkan model yang dapat digunakan dalam berbagai aplikasi [30], [31].

AutoML dapat menangani berbagai jenis tugas, termasuk klasifikasi, regresi, klasering, dan segmentasi. Selain itu, AutoML juga dapat mengolah berbagai jenis data, seperti data tabular, gambar, dan teks. Dalam pengolahan data tabular, AutoML dapat mengidentifikasi tipe data yang berbeda, seperti numerik dan kategorikal, dan secara otomatis melakukan pra-pemrosesan data, seperti normalisasi, pengisian nilai yang hilang, dan pemilihan fitur. Sampai saat ini, AutoML optimal digunakan pada data berjenis tabular [30], [31].

Adapun *framework* AutoML populer yang dikembangkan, baik bersifat *open* ataupun *closed source*, seperti Auto-WEKA, auto-sklearn, GCP-Tables, TPOT, H2O, AutoGluon, dan masih banyak lagi. *Framework* AutoML tersebut sudah banyak digunakan di industri dan terkadang memiliki akurasi yang jauh lebih baik dibandingkan model yang dibuat secara manual [32].

### 2.2.13 AutoGluon

AutoGluon merupakan *framework machine learning* (ML) dan *deep learning* (DL) sumber terbuka yang memungkinkan pengguna untuk membuat model ML ataupun DL yang akurat dan optimal dengan mudah. *Framework* ini dikembangkan oleh AutoGluon Inc., dan dirilis pertama pada tahun 2019 dengan dukungan penuh pada Python versi 3. Berbeda dengan model lain yang fokus pada pemilihan model dan *hyperparameter*, AutoGluon sukses membuat model yang lebih baik dengan teknik *ensembling* beberapa model dan melakukan *stacking* model-model tersebut pada *multi-layer*. Dari hasil eksperimen oleh tim, pada rangkaian 50 tes permasalahan klasifikasi dan regresi dari Kaggle, AutoGluon lebih cepat, kokoh, dan jauh lebih akurat dari beberapa model AutoML lain yang terkenal seperti TPOT, H2O, AutoWEKA, auto-sklearn, dan Google AutoML Tables. Melalui kompetisi Kaggle, AutoGluon bahkan sering mengungguli kompetitornya yang merupakan ekspert data *scientist*. Bahkan, pada 2 kompetisi Kaggle populer, AutoGluon mengalahkan 99% partisipan yang merupakan *data scientist* dan juga pengembang model AutoML lain [32].

AutoGluon menyediakan antarmuka yang mudah digunakan, namun juga menyediakan fleksibilitas yang cukup bagi pengguna. Beberapa fitur utama dari AutoGluon adalah sebagai berikut:

- AutoML: Autogluon memiliki kemampuan untuk membuat model *automated machine learning* yang memungkinkan pengguna untuk secara otomatis menyelesaikan tugas seperti klasifikasi, regresi, dan deteksi objek. AutoGluon dapat bekerja pada data tabular, multimodal, maupun *time-series*.
- Pemilihan model otomatis.
- Model *ensembling*.
- Penyetelan *hyperparameter*.
- Pemrosesan fitur (*feature engineering*).
- Data *preprocessing*.
- Pemisahan data (data *splitting*).

AutoGluon memiliki performa yang sangat baik apabila dibandingkan AutoML kompetitor seperti H2O, TPOT, GCP-Tables, auto-sklearn, dan Auto-WEKA. Pada Tabel 2.2, terlihat bahwa AutoGluon jauh mengungguli kompetitornya. Dari pengujian menggunakan 39 *dataset*, AutoGluon berhasil menjadi juara dengan akurasi terbaik pada 23 *dataset*. Rata-rata waktu *training* AutoGluon juga sangat baik dibandingkan kompetitor-nya, hanya kalah dari GCP-Tables [32].

Tabel 2.2. Perbandingan performa AutoGluon dengan AutoML kompetitor [32]

Framework	Wins	Losses	Failures	Champion	Avg. Rank	Avg. Rescaled Loss	Avg. Time (min)
<b>AutoGluon</b>	-	-	<b>1</b>	<b>23</b>	<b>1.8438</b>	<b>0.1385</b>	201
<b>H2O AutoML</b>	4	26	8	2	3.125	0.2447	220
<b>TPOT</b>	6	27	5	5	3.375	0.2034	235
<b>GCP-Tables</b>	5	20	14	4	3.75	0.3336	<b>195</b>
<b>auto-sklearn</b>	6	27	6	3	3.8125	0.3197	240
<b>Auto-WEKA</b>	4	28	6	1	5.0938	0.8001	244

## **BAB III**

### **METODE PENELITIAN**

Secara umum, tujuan dari penelitian ini adalah menguji efektivitas penggunaan penanda ArUco dalam melokalisasi objek CDP. Penelitian ini juga mencarai parameter P&S yang tepat dalam pembuatan *dataset* SQR seperti konfigurasi kamera, serta jenis kertas dan tinta yang digunakan untuk mencetak SQR, sehingga mendapatkan hasil akurasi autentikasi yang tinggi. Dari hasil yang didapatkan, peneliti selanjutnya dapat menggunakan model SQR, parameter P&S, dan model autentikasi yang digunakan penulis dalam penelitian ini. Dalam prosesnya, metode-metode penelitian yang dirancang dan dilakukan penulis akan dijelaskan pada bab ini.

#### **3.1 Alat dan Bahan Tugas Akhir**

##### **3.1.1 Alat Tugas Akhir**

Alat yang digunakan pada penelitian ini terbagi atas perangkat keras dan perangkat lunak yang dengan rincian sebagai berikut:

1. Laptop Lenovo Legion Y530

Merupakan perangkat keras utama yang digunakan dalam penelitian. Laptop ini memiliki spesifikasi sebagai berikut: Prosesor Intel I5-8300H, RAM 8 GB DDR4, dan kartu grafis Nvidia Geforce GTX 1050TI (4 GB).

2. Python 3.9

Perangkat lunak yang merupakan bahasa pemrograman utama yang digunakan dalam mengembangkan model SQR, pengolahan data, dan membuat model untuk mengautentikasi SQR.

3. AutoGluon

Merupakan pustaka AutoML sumber terbuka yang mengotomasi pembuatan model *deep learning* (DL) dan *machine learning* (ML) untuk menyelesaikan permasalahan di dunia nyata yang melibatkan *dataset* gambar, teks, dan tabular. AutoGluon dapat menyederhanakan alur kerja pembuatan model yang biasanya kita lakukan. Dengan AutoGluon, kita dapat mengembangkan dan memperbaiki model hanya dengan mengoptimasi parameter dengan beberapa baris kode saja.

4. Microsoft Visual Studio Code

Merupakan salah satu perangkat lunak editor kode sumber terpopuler dan gratis yang dikembangkan oleh Microsoft. Editor ini memiliki berbagai fitur yang memungkinkan pengguna untuk menulis, mengedit, dan mengelola kode sumber dengan mudah dan efisien. Selain itu, editor ini juga mendukung berbagai bahasa

pemrograman, termasuk JavaScript, Python, Java, C++, dan masih banyak lagi. Tampilan antarmuka pengguna VS Code sangat bersih dan dapat disesuaikan, sehingga pengguna dapat mengatur tata letak dan tema yang sesuai dengan preferensi mereka. Selain itu, editor ini juga mendukung berbagai bahasa pemrograman, termasuk JavaScript, Python, Java, C++, dan masih banyak lagi. Fitur-fitur kunci dari VS Code termasuk kemampuan untuk mengeksekusi kode secara langsung dari editor, penyelesaian otomatis kode, refactoring kode, debugging, pengelolaan paket, dan integrasi dengan sistem kontrol versi seperti Git.

5. Ekstensi Jupyter Notebook untuk VS Code

Merupakan ekstensi yang dimiliki oleh Microsoft Visual Studio Code untuk menjalankan *notebook* di dalam *environment* Microsoft Visual Studio Code.

6. Adobe Acrobat DC

Merupakan perangkat lunak yang digunakan sebagai pembaca dan pengedit fail PDF. Perangkat lunak ini banyak penulis gunakan untuk mengubah ukuran *batch QR* menjadi ukuran standar percetakan.

7. Adobe Photoshop 2022

Merupakan perangkat lunak yang digunakan sebagai editor gambar. Perangkat lunak ini penulis gunakan dalam penelitian untuk mengubah fail PNG hasil *generate batch QR* dari kode menjadi PDF.

8. *Smartphone* Realme GT Neo 3T

Perangkat keras yang digunakan sebagai pemindai SQR dalam pembuatan *dataset* SQR orisinal dan palsu. *Smartphone* ini memiliki spesifikasi sebagai berikut: *Chipset* Qualcomm SM8250-AC Snapdragon 870 5G, sistem operasi Android 12, kamera utama dengan resolusi 64 MP, f/1.8, 25mm.

9. *Printer* Xerox Versant

Perangkat keras yang digunakan sebagai pencetak utama *dataset* SQR orisinal dan palsu. *Printer* ini merupakan *printer* berspesifikasi tinggi yang banyak digunakan dalam industri percetakan.

10. Boks

Boks digunakan sebagai *environment* yang digunakan untuk menstandarisasi hasil pemotretan atau pemindaian *dataset* SQR menggunakan kamera *smartphone*.

11. Gunting

Gunting digunakan untuk memotong *dataset* SQR yang nantinya akan dipotret.

### 3.1.2 Bahan Tugas akhir

Bahan yang digunakan pada penelitian secara garis besar merupakan *dataset* SQR dengan rincian sebagai berikut:

### 1. Dataset SQR 4 Level *Grayscale*

*Dataset SQR 4 level grayscale* terdiri dari total 400 data (200 data SQR orisinal dan 200 data SQR palsu).

### 2. Dataset SQR untuk Pengujian Parameter

*Dataset* ini digunakan untuk melakukan verifikasi parameter pembuatan model SQR, pemotretan, dan pencetakan SQR, misalnya ukuran penanda ArUco, ukuran SQR, ukuran CDP, konfigurasi kamera, serta jenis kertas dan tinta yang digunakan untuk mencetak SQR.

## 3.2 Metode yang Digunakan

Penelitian ini termasuk ke dalam penelitian kuantitatif yang dilakukan dengan eksperimen. Eksperimen yang dilakukan adalah dengan melakukan verifikasi parameter yang akan digunakan dalam pembuatan SQR, pembuatan *dataset* SQR, dan pembuatan model autentifikasi SQR orisinal dan palsu menggunakan AutoML.

Eksperimen yang dilakukan secara berulang bertujuan untuk mencari parameter terbaik dalam pengembangan model SQR, baik parameter model SQR itu sendiri ataupun parameter yang digunakan dalam proses P&S. Parameter yang digunakan dalam pengembangan model SQR antara lain: Ukuran dan versi kode QR, level toleransi kesalahan, ukuran *watermark*, ukuran CDP, ukuran penanda ArUco, dan juga metode dalam pembuatan CDP.

Untuk parameter dalam proses P&S antara lain, jenis kertas dan tinta yang digunakan untuk mencetak SQR, serta konfigurasi kamera yang digunakan dalam pembuatan *dataset*. Selain itu, dengan penelitian ini, dapat diketahui fitur mana saja yang memiliki pengaruh signifikansi tinggi dalam pembuatan model klasifikasi biner SQR orisinal dan palsu. Dalam melakukan perubahan dan kombinasi pada parameter-parameter tersebut, setiap eksperimen akan menunjukkan hasil yang berbeda dan unik. Hal tersebut menunjukkan bahwa setiap parameter memiliki peran dalam keberhasilan pembuatan SQR. Eksperimen juga dilakukan untuk menguji efektivitas penanda ArUco dalam melokalisasi objek CDP.

Dalam melakukan eksperimen, penulis menggunakan Python sebagai bahasa pemrograman utama, pustaka OpenCV dan Pillow untuk pengolahan gambar, pustaka Pandas dan NumPy untuk pengolahan data, pustaka Matplotlib dan Seaborn untuk visualisasi, serta AutoGluon sebagai *framework* AutoML untuk membuat model klasifikasi biner.

## 3.3 Tahapan Penelitian

Penelitian ini diawali dengan melakukan studi literatur, khususnya tentang SQR dan CDP. Selain itu, penulis juga banyak melakukan studi mengenai penanda ArUco

untuk membantu lokalisasi objek, transformasi homografi, *template matching*, koefisien jarak, uji statistik *T-test* untuk menguji signifikansi pada data grup, dan juga AutoML untuk membuat model autentikasi berjenis klasifikasi biner SQR orisinal atau palsu.

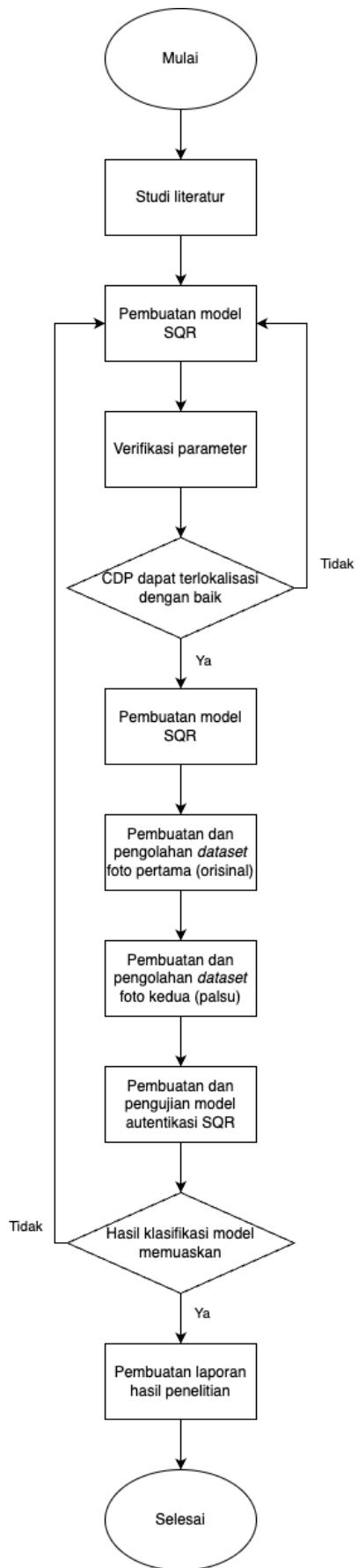
Selanjutnya, penulis melakukan desain dan pembuatan model SQR. Percobaan pembuatan desain SQR terdiri dari penentuan ukuran SQR dan toleransi kerusakannya, ukuran *watermark*, pembuatan CDP, dan juga penentuan ukuran penanda ArUco. Penentuan desain dan parameter-parameter tersebut dilakukan dengan cara studi literatur dari penelitian terdahulu dan juga pembuatan serta pengolahan *dataset* pengujian parameter.

Setelah itu penulis melakukan verifikasi dari parameter-parameter yang digunakan dalam pembuatan model SQR dan proses P&S. Parameter tersebut meliputi ukuran penanda ArUco, konfigurasi kamera, serta jenis kertas dan tinta yang digunakan untuk mencetak SQR. Verifikasi dilakukan dengan menggunakan *dataset* pengujian parameter berjumlah 40 data. Verifikasi dikatakan sukses apabila program dapat mendeteksi seluruh penanda ArUco yang ada di dalam SQR. Apabila semua penanda ArUco terdeteksi, berarti CDP dapat dilokalisasi dengan baik.

Setelah semua parameter didapatkan, selanjutnya adalah pembuatan *dataset* orisinal dan palsu dengan jumlah besar. *Dataset* yang digunakan untuk pembuatan model klasifikasi SQR orisinal dan palsu adalah sebanyak 200 data SQR dengan CDP 4 level orisinal serta 200 data SQR dengan CDP 4 level palsu.

Setelah *dataset* dibuat, selanjutnya adalah pembuatan model autentikasi SQR orisinal dan palsu. Fitur yang digunakan dalam pembuatan model adalah *distance* atau jarak dari data orisinal dan data palsu dengan *template* yang telah di-*generate* pada proses pengolahan data sebelumnya. Pustaka pembelajaran mesin yang penulis gunakan dalam pemodelan klasifikasi biner adalah AutoGluon.

Setelah model selesai dibuat, selanjutnya adalah melakukan pengujian akurasi model dalam mengklasifikasikan SQR orisinal dan palsu. Hasil yang diharapkan dari penelitian ini adalah model SQR yang dapat diautentikasi dengan baik, serta mengetahui pengaruh penggunaan penanda ArUco terhadap kualitas hasil lokalisasi CDP dan akurasi model autentikasi (klasifikasi SQR orisinal dan palsu). Setelah seluruh hasil penelitian dinilai memuaskan oleh penulis, tahapan terakhir adalah pembuatan laporan hasil penelitian. Secara umum, alur penelitian yang dilakukan oleh penulis dapat dilihat pada Gambar 3.1.



Gambar 3.1. Diagram alir tahapan penelitian

### **3.4 Pembuatan Model SQR**

#### **3.4.1 Penentuan Ukuran dan Versi Kode QR**

Dalam penelitian ini, versi kode QR yang digunakan oleh penulis adalah versi 3. Hal tersebut berdasarkan kebutuhan data yang akan digunakan dalam pembuatan dataset tidak lebih dari 35 karakter alfanumerik, sehingga kode QR versi 3 dengan ukuran modul  $29 \times 29$  dapat memenuhi kebutuhan tersebut. Selain itu, adapun parameter *box-size* dan *border*, yaitu jumlah piksel dari tiap-tiap modul pada kode QR dan lapisan pembatas di luar kode QR. Pada pembuatan model SQR ini, penulis menetapkan ukuran *box-size* 20 piksel, dan ukuran *border* 3. Setelah itu penulis menambahkan *padding* tambahan berukuran 10 piksel di sisi terluar. Dengan demikian, ukuran total dari SQR adalah  $(29 \times 20) + (2 \times (3 \times 20)) + (2 \times 10) = 720 \times 720$  piksel.

#### **3.4.2 Penentuan Level Toleransi Kerusakan Kode QR**

Dalam pembuatan SQR, nantinya CDP akan ditempelkan atau diletakkan di tengah kode QR. Oleh karena itu, dibutuhkan level toleransi kerusakan yang besar. Diketahui bahwa level toleransi kerusakan tertinggi yang ada pada kode QR adalah level H dengan toleransi kerusakan kurang lebih 30%. Oleh karena itu, level toleransi kerusakan yang dipilih dalam model SQR ini adalah level H.

#### **3.4.3 Penentuan Ukuran Watermark**

Dengan level toleransi kode QR yang ditentukan adalah level H yang mana menerima kerusakan kurang lebih 30% kerusakan kode QR, maka ukuran piksel yang dapat dirusak adalah  $0,3 \times 29 = 8,7$ . Karena hasilnya tidak bulat, penulis melakukan pembulatan ke atas, sehingga ukuran piksel yang dapat dirusak adalah 9 piksel. Setelah dikalikan dengan *box-size*, maka toleransi kerusakan untuk meletakkan *watermark* adalah  $9 \times 20 = 180 \times 180$  piksel.

#### **3.4.4 Pembuatan CDP**

Setelah menentukan ukuran *watermark*, selanjutnya adalah mendesain CDP yang digunakan sebagai komponen autentikasi SQR. Dari hasil studi literatur penulis pada penelitian sebelumnya, CDP yang dinilai efektif adalah berukuran  $100 \times 100$  piksel [5]. Semakin besar ukuran CDP, pola unit terkecilnya akan menjadi semakin besar. Hal ini menyebabkan informasi yang ada pada CDP tidak terdegradasi maksimal akibat proses P&S. Ukuran CDP  $100 \times 100$  piksel dinilai optimal karena tetap sulit untuk dibuat ulang, namun ukuran unit terkecilnya masih kecil.

Penulis di sini akan menggunakan CDP dengan 4 level yang nantinya akan dianalisis hasilnya. CDP di-*generate* berdasarkan DATA + SECRET. Hal ini merupakan salah

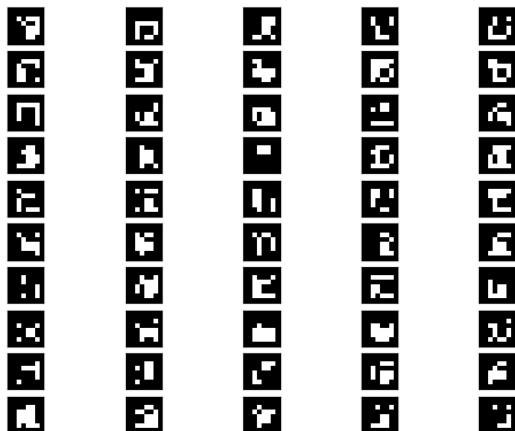
satu standar keamanan yang banyak digunakan saat ini, menggunakan kombinasi DATA + SECRET yang di-*hash* dan menghasilkan sebuah *unique identifier*. DATA merupakan nilai data yang tersimpan dalam kode QR, sedangkan SECRET merupakan *string* rahasia yang disimpan oleh pengembang SQR. DATA + SECRET akan di-*hash* menggunakan algoritma SHA1, kemudian keluarannya diset menjadi heksadesimal. Kemudian batasan nilainya diset menjadi UINT32 atau memiliki rentang nilai 0 s.d. 4294967296.

Setelah *seed* dibuat yang merupakan keluaran dari fungsi sebelumnya, selanjutnya adalah membuat CDP berukuran 100 x 100 piksel dengan 4 level *grayscale*. Untuk CDP 4 level, nilai *grayscale*-nya adalah [0, 85, 170, 255]. Nilai-nilai tersebut didefinisikan secara otomatis hanya dengan memasukkan nilai parameter *quant* saja. Kemudian, untuk memastikan tiap-tiap nilai memiliki distribusi yang seimbang dalam CDP, penulis membatasi perbandingan distribusi maksimal antar nilai adalah 1:3, misal untuk 4 level perbandingan distribusi yang dapat dipilih adalah [[1, 1, 1, 1], [3, 1, 2, 1], ..., [1, 1, 3, 1]]. Program nantinya akan memilih secara acak dari 10 variasi distribusi yang ada. Kemudian barulah CDP berukuran 100 x 100 piksel di-*generate* secara acak berdasarkan nilai *seed*, sehingga nilai *seed* akan berpengaruh pada distribusi elemen dan letak tiap-tiap elemen dalam CDP.

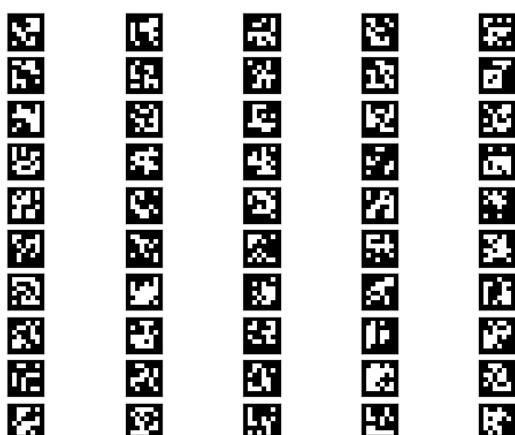
#### 3.4.5 Menentukan Jumlah, Jenis, Letak, dan Ukuran Penanda ArUco

Setelah CDP dibuat, ruang kosong yang tersisa di dalam *watermark* adalah  $180 - 100 = 80$  piksel. Sisa ruang kosong tersebut digunakan untuk meletakkan penanda ArUco sebagai pembantu dalam melokalisasi objek CDP yang akan digunakan untuk autentikasi SQR. Jumlah penanda ArUco yang digunakan adalah 8 buah. Kedelapan penanda ArUco diletakkan di area kosong di sekitar CDP, pojok kiri atas, tengah atas, pojok kanan atas, kiri tengah, kanan tengah, pojok kiri bawah, tengah bawah, dan pojok kanan bawah. Kedelapan penanda tersebut dinilai penulis cukup untuk merepresentasikan dan mewakili titik-titik yang nantinya akan digunakan untuk transformasi homografi. Penelitian sebelumnya menggunakan empat titik, yaitu keempat titik sudut kode QR. Hal tersebut memiliki kelemahan, yaitu jika ada lekukan pada foto pada bagian sisi-sisi kode QR (misal kode QR ditempelkan pada permukaan tabung), maka hasil transformasi homografinya akan kurang maksimal. Selanjutnya, untuk memudahkan program dalam mendeteksi kedelapan penanda ArUco, maka tentunya pola ArUco yang paling sederhana dapat memudahkan program dalam pendekripsi. Oleh karena itu, penulis memilih ArUco dengan ukuran modul 4 piksel. Pada Gambar 3.2 dan Gambar 3.3 dapat dilihat bahwa ArUco dengan ukuran modul 4 piksel memiliki pola yang relatif sederhana, sehingga toleransi degradasi gambar akibat proses P&S dapat diminimalisir.

Terakhir, adalah menentukan ukuran penanda ArUco yang diletakkan di sekitar CDP. Metode yang digunakan untuk mendapatkan ukuran adalah dengan membuat *da-*



Gambar 3.2. Sampel 50 penanda ArUco dengan ukuran modul 4x4 piksel



Gambar 3.3. Sampel 50 penanda ArUco dengan ukuran modul 6x6 piksel

taset sebanyak 40 data yang tiap 5 datanya memiliki ukuran ArUco 20 s.d. 34 piksel. Kemudian nantinya akan diuji, penanda ArUco dengan ukuran berapakah yang memiliki akurasi lokalisasi paling tinggi yang berarti kedelapan penanda ArUco dapat terdeteksi oleh program. Tiap-tiap penanda ArUco memiliki id unik yang membedakan penanda satu dengan lainnya dalam sebuah pustaka penanda ArUco. Oleh karena itu, penulis menggunakan id penanda ArUco tetap, yaitu 0 s.d. 7 untuk memudahkan pemetaan titik asal ke titik tujuan saat melakukan transformasi homografi, misalnya titik tujuan pojok kiri atas harus ditempati oleh penanda ArUco dengan id = 0, titik pojok kanan bawah harus ditempati oleh marker dengan id = 7 dan seterusnya.

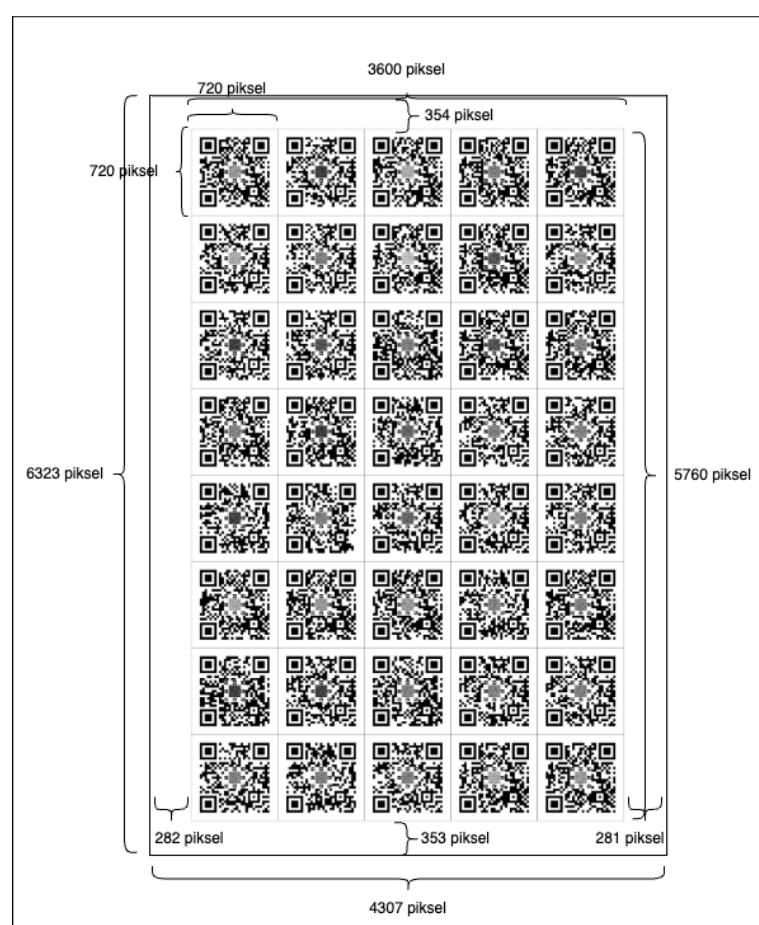
#### 3.4.6 Menempelkan Watermark ke SQR

Setelah seluruh komponen watermark berukuran 180 x 180 piksel selesai dibuat, selanjutnya adalah menempelkannya atau meletakkannya ke tengah-tengah kode QR yang telah dibuat di awal. Hal ini dapat dilakukan dengan menggunakan fitur *slicing* pada NumPy *array*.

### 3.5 Pembuatan dan Pengolahan Dataset SQR Foto Pertama (Orisinal)

#### 3.5.1 Pembuatan Batch SQR Foto Pertama (Orisinal)

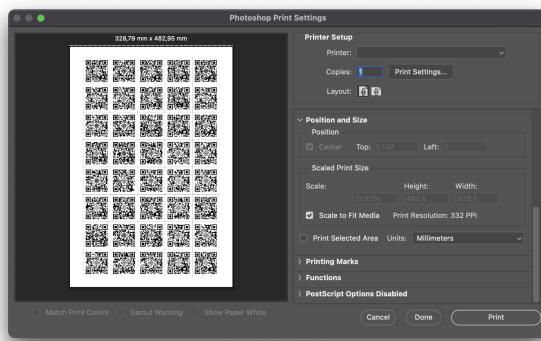
SQR berjumlah 40 akan disusun menjadi 1 *batch*, dengan konfigurasi 8 baris 5 kolom. Diketahui juga bahwa ukuran satu SQR yang telah dirancang adalah 720 x 720 piksel. Nantinya SQR akan dirancang untuk dicetak dengan ukuran 55 x 55 mm, sehingga jumlah piksel per milimeter pada batch QR adalah  $720 / 55 = 13$  ppmm (piksel per milimeter). Kemudian, untuk menyesuaikan ukuran piksel terhadap milimeter, perlu dilakukan penskalaan terhadap ppmm serta pemberian *padding*. Diketahui ukuran kertas A3+ adalah 329 x 483 mm, jika dijadikan ukuran piksel dengan 13 ppmm, maka ukurannya menjadi 4307 x 6323 piksel. Kemudian *padding* horizontal diset sebesar  $(6323 - (720 \times 55)) / 2 = 281,5$  sedangkan *padding* vertikal diset sebesar  $(4307 - (720 \times 8)) / 2 = 353,5$ . Karena hasilnya tidak bulat, maka *padding* horizontal dan vertikalnya diset ke + bilangan bulat terdekat. Hasil akhir *batch* dalam ukuran piksel dapat dilihat pada Gambar 3.4.



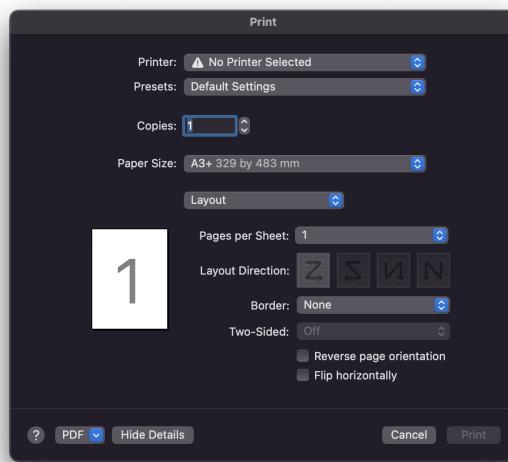
Gambar 3.4. *Batch* SQR orisinal (foto pertama) dalam ukuran piksel

### 3.5.2 Pencetakan Batch SQR Foto Pertama (Orisinal)

Dari hasil *generate batch* oleh program yang berukuran piksel dan berformat PNG, hasil tersebut masih perlu di-*scaling* menjadi ukuran kertas A3+ dengan format PDF, sesuai dengan standar percetakan. Untuk melakukannya, penulis menggunakan perangkat lunak tambahan, yaitu Adobe Photoshop. Caranya adalah meng-*import batch* PNG ke dalam Photoshop, kemudian tekan CTRL + P untuk melakukan pencetakan. Kemudian di panel konfigurasi pencetakan, centang bagian *Scale to Fit Media*, hal ini bertujuan untuk melakukan penskalaan menuju ukuran kertas yang dituju. Selanjutnya, jika belum ada, masukkan konfigurasi untuk ukuran kertas A3+, yaitu dengan ukuran 329 x 483 mm. Terakhir adalah ubah target cetak menjadi PDF. Langkah-langkah lengkap konfigurasi pencetakan dan ukuran kertas dapat dilihat pada Gambar 3.5 dan Gambar 3.6.



Gambar 3.5. Konfigurasi saat melakukan pencetakan menggunakan Adobe Photoshop



Gambar 3.6. Konfigurasi menambahkan ukuran kertas A3+ dan cetak menjadi PDF

Terakhir, untuk jenis kertas dan tinta yang digunakan haruslah bersifat *doff*. Hal ini supaya hasil pemotretan tidak memantulkan cahaya ketika dipotret menggunakan flash dari

kamera.

### 3.5.3 Pemotretan *Dataset SQR* Foto Pertama (Orisinal)

Pemotretan *dataset* foto pertama (orisinal) dilakukan menggunakan boks. Boks dibuat dengan kardus yang dilubangi bagian depan dan atasnya, kemudian sisi dalamnya ditempel kertas putih supaya dapat menghasilkan hasil pemotretan yang terang. Penggunaan boks dalam pemotretan *dataset* dapat menjaga kualitas *dataset* dan juga meningkatkan kecepatan pengambilan *dataset*. Pemotret hanya perlu mengganti SQR tanpa harus memindah-mindahkan *smartphone* yang digunakan untuk memotret. Lingkungan pemotretan menggunakan boks yang penulis gunakan dalam mendapatkan *dataset* orisinal dapat dilihat pada Gambar 3.7.



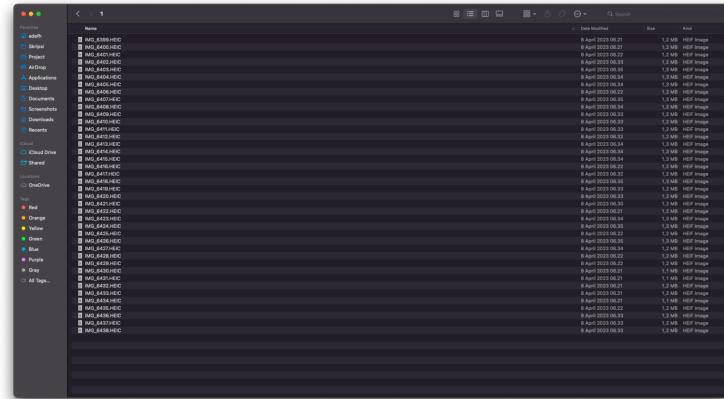
Gambar 3.7. Lingkungan pemotretan menggunakan boks untuk melakukan pemotretan *dataset* SQR

### 3.5.4 Lokalisasi CDP dari *Raw* Foto Pertama (Orisinal)

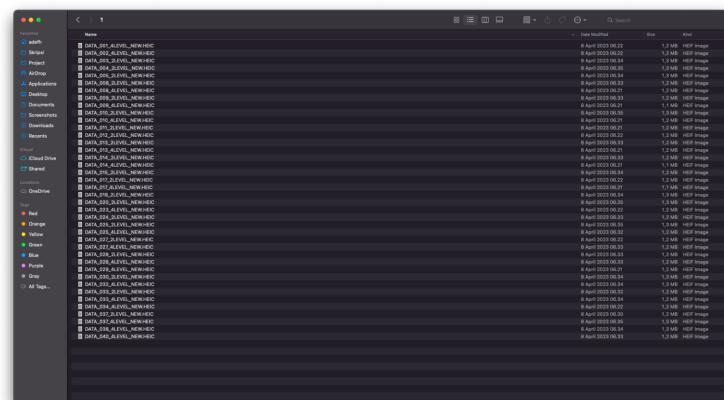
#### 3.5.4.1 Mengganti Nama Fail

Untuk melakukan lokalisasi CDP dari *dataset* foto, langkah pertama yang dilakukan adalah mengganti nama fail foto sesuai dengan data yang ada dalam kode QR. Hal ini bertujuan untuk memudahkan pengelolaan *dataset*. Nama fail sebelum dan sesudah perubahan nama dapat dilihat pada Gambar 3.8 dan Gambar 3.9.

Untuk mengganti nama fail, penulis memanfaatkan pustaka OpenCV yang digunakan untuk mendeteksi data dalam kode QR. Pustaka tersebut juga dapat digunakan untuk mendeteksi koordinat keempat titik sudut kode QR yang digunakan dalam penelitian sebelumnya untuk melakukan transformasi homografi dari titik asal ke titik tujuan. Sebelum dideteksi dan di-*decode* gambar foto *raw* akan diproses terlebih dahulu, untuk meningkatkan performa deteksi oleh pustaka QRDecoder. Pemrosesan yang dilakukan yaitu, mengubah gambar menjadi *grayscale* dan menambahkan filter GaussianBlur pada gambar yang telah diubah menjadi *grayscale* dengan ukuran kernel tertentu. Hal ini



Gambar 3.8. Nama fail sebelum perubahan



Gambar 3.9. Nama fail setelah perubahan

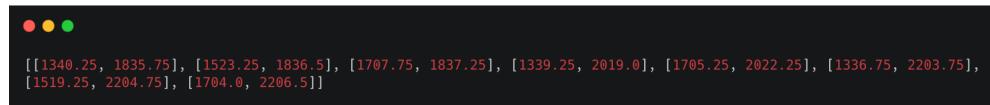
bertujuan untuk menghilangkan derau dari gambar. Setelah itu, akan dilakukan perulangan *scaling* pada gambar dengan faktor skala 1 s.d. 0,1. Pada tiap-tiap faktor skala, fungsi QRDecoder akan mencoba mendeteksi keempat titik sudut kode QR dan data di dalamnya. Jika titik sudut yang dideteksi sudah berjumlah 4 dan data di dalamnya sudah terdeteksi, maka proses penskalaan akan dihentikan pada faktor skala tertinggi, lalu kemudian data yang tersimpan pada kode QR akan dikeluarkan. Misal, saat perulangan dengan faktor skala 1, empat titik sudut pada kode QR sudah terdeteksi dan datanya juga, jika hasilnya demikian, artinya gambar tidak perlu di-*scaling* lagi. Namun, apabila perulangan berhenti di faktor skala 0,5, artinya gambar perlu di-*scaling* setengahnya untuk dapat dideteksi dan di-*decode* datanya.

### 3.5.4.2 Lokalisasi CDP dari Foto

Untuk melakukan lokalisasi CDP dari foto yang telah diganti nama fail-nya, penulis memanfaatkan penanda ArUco yang diletakkan di sekitar CDP. Telah dijelaskan sebelumnya, bahwa delapan penanda ArUco dapat melakukan lokalisasi CDP dengan lebih baik karena titik yang digunakan untuk transformasi homografi dari titik asal ke titik

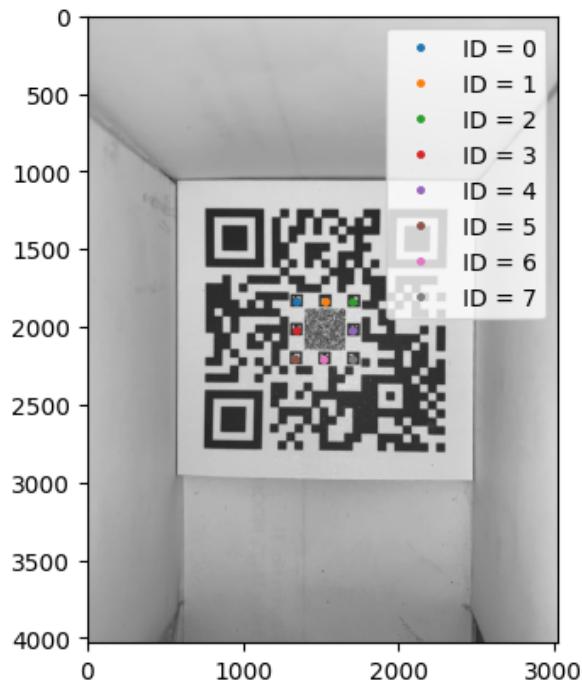
tujuan lebih banyak dari empat titik yang didapatkan dari fungsi QRDecoder. Hal ini dapat mengatasi masalah lekukan gambar pada kondisi tertentu. Sama seperti pemrosesan sebelumnya, gambar akan diubah menjadi *grayscale* dan ditambahkan filter GaussianBlur untuk meningkatkan performa deteksi penanda ArUco. Kemudian, gambar akan dilakukan perulangan penskalaan dengan faktor skala 1 s.d. 0,1. Pada tiap-tiap faktor skala, fungsi aruco.detectMarkers akan mendeteksi titik-titik sudut dari kedelapan penanda ArUco beserta id-nya. Sama seperti pendekripsi data pada kode QR, perulangan akan dihentikan apabila delapan penanda ArUco sudah berhasil dideteksi.

Dari keluaran empat titik sudut di tiap-tiap penanda, penulis memprosesnya untuk menjadikannya satu titik di tengah untuk memudahkan pendefinisian titik asal transformasi. Kemudian hasilnya akan diolah menjadi xy\_list, yaitu koordinat titik tengah dari masing-masing penanda mulai dari id = 0 s.d. id = 7. Contoh xy\_list yang telah diolah dapat dilihat pada Gambar 3.10.



Gambar 3.10. xy\_list yang menyimpan koordinat titik tengah penanda ArUco dari id 0 s.d. 7

Untuk contoh plot hasil deteksi koordinat ke dalam xy\_list pada gambar hasil foto dapat dilihat pada Gambar 3.11.



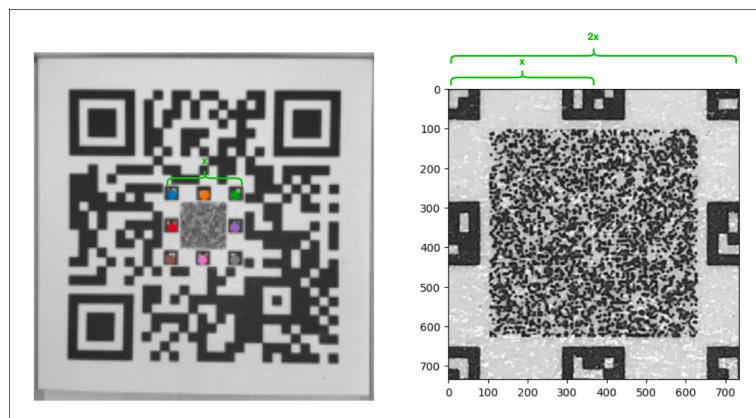
Gambar 3.11. Hasil plot koordinat titik tengah penanda ArUco pada xy\_list ke gambar hasil foto

Setelah semua titik telah didapatkan, selanjutnya adalah melakukan transformasi homografi dari kedelapan titik tersebut ke titik tujuan. Untuk menentukan titik tujuan, penulis menggunakan acuan ukuran dari jarak terdekat dari empat titik penanda ArUco yang terletak di pojok (pojok kiri atas, pojok kanan atas, pojok kiri bawah, dan pojok kanan bawah) atau ( $id = 0$ ,  $id = 2$ ,  $id = 5$ , dan  $id = 7$ ) menggunakan *euclidean distance*. Hal ini digunakan untuk menjaga kualitas hasil dari transformasi homografi. Dari eksperimen, jika ukuran tujuan transformasi homografi terlalu jauh dari ukuran awal, misalnya langsung didefinisikan berukuran  $100 \times 100$  piksel sesuai *template* hasilnya akan rusak. Setelah jarak terdekatnya didapatkan sebagai  $x$ , selanjutnya adalah menentukan titik tujuan. Titik tujuan transformasi didefinisikan dalam bentuk sebuah *list* seperti yang terlihat pada Gambar 3.12, di mana  $x$  merupakan jarak terdekat dari penanda ArUco yang terletak di pojok.



Gambar 3.12. *List* koordinat tujuan transformasi homografi

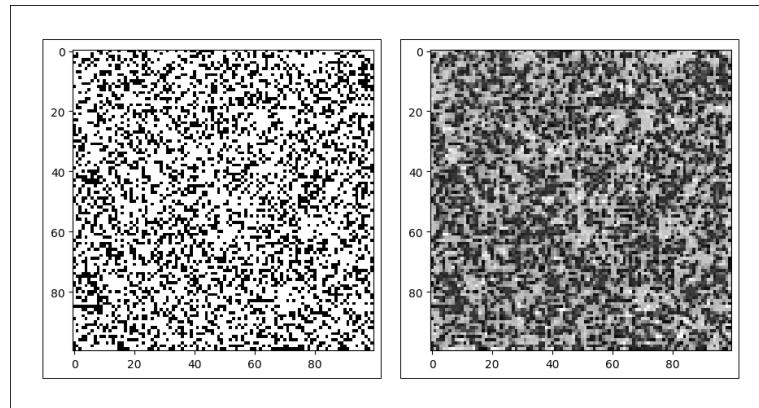
Dalam melakukan transformasi homografi, koordinat tujuan nilainya dikalikan dengan 2, sehingga ukuran dari CDP terlokalisasi adalah 2 kali dari CDP awal sebelum dilokalisasi. Hal ini bertujuan untuk menjaga kualitas CDP yang nantinya masih akan diproses lagi. Untuk melihat perbedaan SQR awal dan SQR setelah ditransformasi homografi berdasarkan koordinat dari penanda ArUco dapat dilihat pada Gambar 3.13.



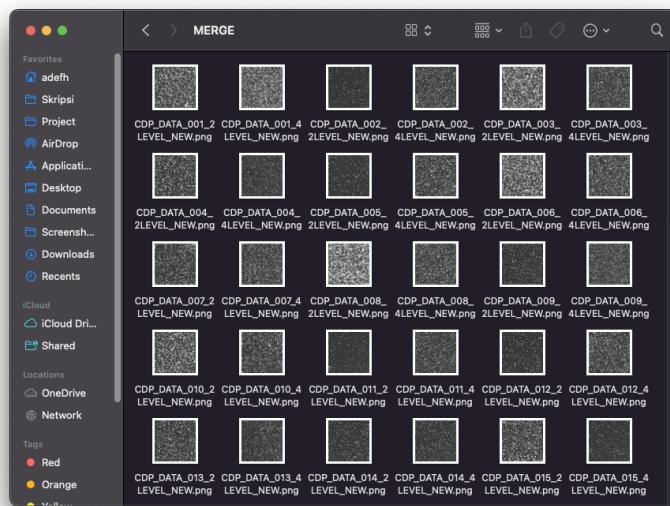
Gambar 3.13. SQR awal dan setelah dilakukan transformasi homografi

Setelah melakukan transformasi homografi, langkah selanjutnya adalah mendapatkan objek CDP itu sendiri. Diketahui bahwa desain pada desain SQR awal, CDP berukuran  $100 \times 100$  piksel, sedangkan ukuran gambar *watermark* hingga titik tengah penanda ArUco adalah  $140 \times 140$  piksel. Oleh karena itu, untuk mendapatkan objek CDP dari objek hasil transformasi adalah dengan melakukan pemotongan di tengah sebesar  $100/140$  bagian dari objek hasil transformasi. Setelah objek CDP didapatkan, lang-

kah terakhir adalah melakukan penskalaan CDP menjadi  $100 \times 100$  piksel sesuai dengan ukuran *template*. CDP hasil lokalisasi berukuran  $100 \times 100$  piksel inilah yang akan diolah datanya untuk dibandingkan dengan *template*. Perbandingan CDP *template* dengan CDP hasil akhir dari lokalisasi dapat dilihat pada Gambar 3.14. Kemudian untuk memudahkan manajemen fail CDP sebagai *dataset*, hasil lokalisasi disimpan menjadi sebuah fail gambar berformat PNG dengan format penamaan fail CDP\_{data}.png. Hasil akhir dataset CDP foto pertama (orisinal) dikumpulkan ke dalam sebuah folder seperti pada Gambar 3.15.



Gambar 3.14. Perbandingan CDP *template* (kiri) dengan CDP hasil lokalisasi foto pertama (orisinal) (kanan)



Gambar 3.15. Hasil *dataset* CDP foto pertama (orisinal) yang sudah dilokalisasi dan diganti nama failnya

### 3.5.5 Pembuatan Fitur Jarak (Orisinal dengan *Template*)

Untuk mendapatkan fitur dari *dataset* foto pertama (orisinal) yang nantinya digunakan dalam pembuatan model klasifikasi menggunakan AutoML, penulis menggunakan

jarak spasial, histogram, dan DCT dengan beberapa koefisien jarak dari CDP foto pertama (orisinal) dengan *template*.

Langkah pertama adalah membaca seluruh fail CDP dalam folder yang sudah ditentukan, kemudian dilakukan *parsing* pengambilan jenis interpolasi dan data sesuai dengan format penamaan *dataset* CDP. Selanjutnya adalah men-generate *template* berdasarkan data yang sudah di-*parsing*, simpan sebagai *template*. Setelah itu, baca CDP hasil scan, ubah menjadi *grayscale*, simpan sebagai *scanned*. Kemudian, tiap fail CDP *template* dan *scanned* akan diperbesar 4 kali pembesaran, dari awalnya 100 x 100 piksel di-*scaling* menjadi 400 x 400 piksel. Selanjutnya, matriks tersebut akan diolah menjadi tiga fitur, yaitu DCT, histogram, dan spasial. Kemudian akan dihitung jaraknya menggunakan koefisien jarak korelasi, kosinus, *euclidean*, dan *canberra* untuk tiap-tiap fitur. Untuk tiap perulangan (tiap pembacaan fail) hasil fitur jarak, data, jenis interpolasi, dan level akan di-*append* ke dalam sebuah *list*. Setelah perulangan selesai, *list* tersebut akan diubah menjadi sebuah *dataframe*, di-*sorting* berdasarkan jenis interpolasi dan datanya, lalu ditambahkan kolom label (orisinal atau palsu). Setelah *dataframe* siap, selanjutnya adalah mengekspor *dataframe* tersebut menjadi sebuah fail csv. Untuk hasil akhir *dataframe* sebelum diekspor menjadi fail csv dapat dilihat pada Gambar 3.16.

	interpolation	data	sp_corr	sp_cosine	sp_euclidean	sp_canberra	his_corr	his_cosine	his_euclidean	his_canberra	dct_corr	dct_cosine	dct_euclidean	dct_canberra	level	label
0	INTER_AREA	DATA_001_2LEVEL_NEW	0.276514	0.105260	43818.564468	97199.766805	1.088972	1.000000	0.725821	211.000000	0.065241	1.373189e+06	98948.011457	2LEVEL	ORI	
1	INTER_AREA	DATA_001_4LEVEL_NEW	0.347229	0.065885	32882.214524	51246.687912	1.039011	0.932368	0.539133	2219.165152	0.037189	0.037109	1.096647e+06	92848.128126	4LEVEL	ORI
2	INTER_AREA	DATA_002_2LEVEL_NEW	0.492337	0.376830	40294.680145	141408.297314	1.030277	1.000000	0.810349	146.000000	0.289255	0.289446	1.144804e+06	12479.386159	2LEVEL	ORI
3	INTER_AREA	DATA_002_4LEVEL_NEW	0.331102	0.122413	36173.780781	77762.781353	1.037287	0.957346	0.521934	193.925351	0.075712	0.075590	1.122782e+06	102568.295199	4LEVEL	ORI
4	INTER_AREA	DATA_003_2LEVEL_NEW	0.251024	0.075142	41157.396225	83716.713034	1.122129	1.000000	0.752067	226.000000	0.044829	0.044738	1.328214e+06	90007.984576	2LEVEL	ORI
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	INTER_NEAREST	DATA_038_4LEVEL_NEW	0.848117	0.247521	35628.271246	85653.094117	1.029187	0.961729	0.539581	192.951196	0.149387	0.149172	1.077218e+06	119270.095176	4LEVEL	ORI
396	INTER_NEAREST	DATA_039_2LEVEL_NEW	0.879638	0.249665	56019.305387	95901.322562	1.136961	0.997535	0.749396	255.999402	0.147533	0.147237	1.708828e+06	115732.938582	2LEVEL	ORI
397	INTER_NEAREST	DATA_039_4LEVEL_NEW	0.826614	0.182059	36858.806032	64068.041681	1.033648	0.938115	0.553876	240.948065	0.104194	0.104041	1.152029e+06	112321.488712	4LEVEL	ORI
398	INTER_NEAREST	DATA_040_2LEVEL_NEW	0.729232	0.244817	53257.876404	103062.861930	1.127558	0.997764	0.726175	254.999339	0.149274	0.148993	1.616389e+06	110465.427187	2LEVEL	ORI
399	INTER_NEAREST	DATA_040_4LEVEL_NEW	0.691139	0.209644	43803.895900	80447.623041	1.076971	0.957107	0.534011	250.896760	0.125290	0.125047	1.335360e+06	109099.753744	4LEVEL	ORI

Gambar 3.16. *Dataframe* fitur jarak *dataset* foto pertama (orisinal)

### 3.5.6 Analisis *Dataset* CDP Foto Pertama (Orisinal)

Setelah seluruh *dataset* CDP foto pertama (orisinal) sudah siap, selanjutnya adalah melakukan analisis menggunakan *dataframe* fitur jarak. Hal pertama yang dianalisis apakah ada perbedaan signifikan antara jarak CDP orisinal dengan *template*. Analisis selanjutnya adalah mengetahui perbedaan rata-rata jarak dari berbagai interpolasi penskalaan yang digunakan untuk melokalisasi CDP. Dari hasil tersebut, nantinya akan dibuat CDP palsu menggunakan interpolasi penskalaan yang hasil jaraknya paling kecil antara foto pertama (orisinal) dengan *template*. Hal ini dimaksudkan untuk membuat CDP palsu yang semirip mungkin dengan CDP asli. Uji signifikansi dilakukan menggunakan metode *T-test*, sedangkan untuk mencari rata-rata jarak dari data grup tiap interpolasi penskalaan, penulis menggunakan fitur *describe* yang telah disediakan oleh pustaka Pandas.

### **3.6 Pembuatan dan Pengolahan Dataset SQR Foto Kedua (Palsu)**

#### **3.6.1 Pembuatan Model SQR Foto Kedua (Palsu)**

Model SQR palsu dibuat dengan men-*generate template* sesuai data pada SQR orisinal. Kemudian CDP hasil lokalisasi foto pertama akan ditempelkan di tengah-tengah *template* SQR, sehingga perbedaan antara SQR orisinal dan palsu hanya ada pada CDP-nya saja. Hal ini tentu akan sulit dibedakan oleh mata manusia.

#### **3.6.2 Pembuatan Batch SQR Foto Kedua (Palsu)**

Untuk pembuatan *batch* SQR palsu, langkahnya sama dengan pembuatan *batch* SQR foto pertama pada 3.5.1, hanya saja SQR yang digunakan adalah SQR yang telah diganti CDP-nya. Jumlah SQR dalam satu *batch* juga sama, 40 buah, dengan konfigurasi 8 baris 5 kolom. Ukuran kertas juga disesuaikan menjadi A3+ sesuai standar percetakan.

#### **3.6.3 Pencetakan Batch SQR Foto Kedua (Palsu)**

Proses persiapan pencetakan juga sama dengan persiapan pencetakan *batch* SQR foto pertama (orisinal) pada 3.5.2, hasil *generate* yang ukurannya masih piksel, diubah menjadi mm dengan ukuran kertas A3+ menggunakan perangkat lunak Adobe Photoshop. Kertas yang digunakan untuk mencetak juga sama, yaitu yang bersifat *doff* dan tidak memantulkan cahaya. *Printer* yang digunakan untuk mencetak haruslah sama, hal ini untuk memastikan tidak adanya perbedaan kualitas akibat perbedaan perangkat *printer*.

#### **3.6.4 Pemotretan Dataset SQR Foto Kedua (Palsu)**

Proses pemotretan *dataset* SQR foto kedua (palsu) juga sama seperti proses pemotretan *dataset* SQR foto pertama (orisinal), yaitu menggunakan bantuan boks dan kamera *smartphone* dengan kondisi *flash* menyala, seperti yang ditunjukkan Gambar 3.7 pada 3.5.3.

#### **3.6.5 Lokalisasi CDP dari Raw Foto Kedua (Palsu)**

Proses lokalisasi CDP foto kedua (palsu) sama persis dengan proses lokalisasi CDP foto pertama (orisinal) pada 3.5.4. Perbedaanya adalah penamaan fail keluaran, yaitu dengan format CDP\_{interpolasi}\_{data}\_FAKE.

#### **3.6.6 Pembuatan Fitur Jarak (Palsu dengan Template)**

Proses ini juga sama seperti yang dilakukan pada pembuatan fitur jarak orisinal dengan *template* pada 3.5.5, hanya saja yang diukur jaraknya adalah CDP foto kedua (palsu) dengan *template*. Selain itu, kolom label pada *dataframe* nilainya adalah FAKE.

### **3.7 Pembuatan dan Pengujian Model Klasifikasi SQR Orisinal dan Palsu**

Model autentikasi yang dibuat merupakan model klasifikasi biner, untuk mengklasifikasikan apakah SQR tersebut orisinal atau palsu. Pembuatan model menggunakan *framework* AutoML dari AutoGluon. AutoGluon dipilih karena dapat melakukan pemrosesan fitur otomatis dan memberikan peringkat *leaderboard* untuk model-model yang digunakan, sehingga penulis dapat dengan mudah memilih model dengan akurasi klasifikasi biner tertinggi. Selain itu, banyak lagi keunggulan dari AutoGluon yang dijelaskan pada [\[2.2.13\]](#)

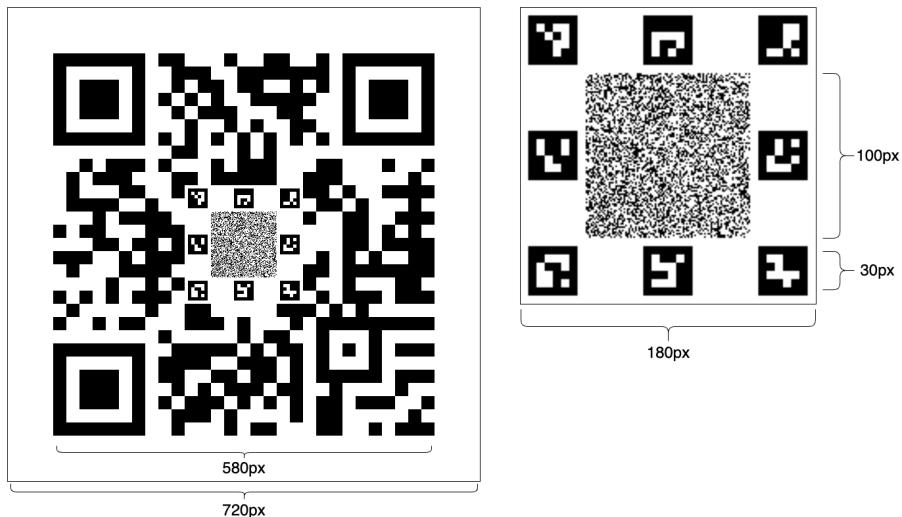
#### **3.7.1 Pembuatan Model Klasifikasi Biner menggunakan AutoGluon**

Sebelum memasukkan data ke model, perlu dilakukan pemisahan antara X dan y, X merupakan kolom fitur sedangkan y merupakan kolom label. Kolom fitur yang dipilih adalah fitur jarak yang berjumlah 12 kolom. Kolom label adalah kolom bernilai ORI atau FAKE. Setelah dilakukan pemisahan X dan y, selanjutnya adalah memisahkan X\_train, X\_test, y\_train, dan y\_test. Perbandingan data latih dan data uji adalah 7:3. Setelah itu, data yang sudah dipisah dimasukkan ke dalam model AutoGluon. AutoGluon dapat secara otomatis menampilkan *leaderboard* untuk model-model yang digunakan.

## BAB IV

# HASIL DAN PEMBAHASAN

### 4.1 Hasil Akhir Desain SQR



Gambar 4.1. Hasil akhir desain SQR

Hasil akhir dari desain SQR yang penulis buat dapat dilihat pada Gambar 4.1. Versi kode QR yang digunakan adalah kode QR versi 3, dengan 29 modul dan 20 *box-size*, sehingga jika dikalikan ukurannya menjadi  $580 \times 580$  piksel. Selain itu, ditambahkan *padding* berukuran 70 piksel, sehingga ukuran total dari SQR adalah  $720 \times 720$  piksel. *Watermark* berukuran  $180 \times 180$  piksel diletakkan di tengah-tengah kode QR. Di dalam *watermark* tersebut terdapat CDP berukuran  $100 \times 100$  piksel serta delapan penanda ArUco berukuran  $28 \times 28$  piksel. Penanda ArUco yang digunakan adalah penanda ArUco dengan tipe 4 modul mulai dari  $id = 0$  s.d.  $id = 7$ , sehingga lebih mudah didekripsi oleh program karena bentuknya yang relatif lebih sederhana dibandingkan penanda ArUco yang memiliki ukuran modul lebih tinggi.

### 4.2 Hasil Parameter P&S

Hasil parameter yang didapatkan dalam proses P&S di sini adalah parameter konfigurasi kamera dan jenis kertas maupun tinta yang digunakan untuk mencetak SQR.

#### 4.2.1 Konfigurasi Kamera

Dari eksperimen menggunakan *dataset* pengujian parameter, didapatkan hasil akhir konfigurasi kamera yang digunakan dalam pembuatan *dataset* SQR orisinal dan palsu yang dapat dilihat pada Tabel 4.1.

Tabel 4.1. Parameter Konfigurasi Kamera

ISO	Shutter Speed	Focus	White Balance
200	1/100	0,00	4000

Dengan konfigurasi kamera tersebut, 400 *dataset* CDP dapat dilokalisasi dengan baik dari gambar hasil potretan kamera. Lokalisasi yang digunakan sebagai parameter sukses atau tidaknya adalah lokalisasi menggunakan bantuan penanda ArUco.

Untuk kamera yang digunakan penulis dalam penelitian ini adalah perangkat *smartphone* lain, yaitu Iphone XR dengan konfigurasi kamera yang sama. Hasil gambar yang dipotret menggunakan Iphone XR dapat dilihat pada Gambar 4.2.



Gambar 4.2. Hasil pemotretan SQR menggunakan perangkat Iphone XR

#### 4.2.2 Jenis Kertas dan Tinta

Jenis kertas dan tinta yang digunakan dalam pencetakan *dataset* yang penulis gunakan adalah kertas bersifat *doff*, sehingga tidak memantulkan cahaya dari *flash smartphone* saat pemotretan berlangsung. Kertas bersifat *doff* yang dapat digunakan seperti *art carton* atau HVS. Opsi lain adalah dapat menggunakan kertas dan tinta yang bersifat *glossy*, namun diberikan laminasi *doff* di akhir. Pada Gambar 4.3, dapat dilihat perbedaan hasil pemotretan menggunakan kertas *doff* dan *glossy*, SQR yang dicetak menggunakan kertas dan tinta *glossy* seperti *ivory* akan memantulkan cahaya, SQR yang dicetak

menggunakan kertas dan tinta *doff* seperti *art carton* akan terjaga kualitasnya karena tidak terkena pantulan cahaya, sedangkan SQR yang dicetak menggunakan kertas dan tinta *glossy* kemudian dilaminasi *doff*, hasilnya baik, namun ada sedikit derau gelembung yang disebabkan oleh proses laminasi.



Gambar 4.3. Perbandingan gambar hasil pemotretan SQR yang dicetak dengan berbagai tipe kertas

### 4.3 Hasil Pendekripsi Penanda ArUco

Tabel 4.2. Tabel hasil pendekripsi penanda ArUco dengan berbagai ukuran

Ukuran ArUco	Jumlah Sukses Terdeteksi	Faktor Penskalaan Terkecil
20	0	-
22	1	0.3
24	4	1
26	3	0.7
28	5	1
30	5	1
32	5	1
34	5	1

Untuk mendapatkan ukuran penanda ArUco yang optimal, penulis menggunakan data uji sejumlah 40 SQR, yang ukuran penanda ArUco-nya berbeda-beda tiap lima SQR, dari ukuran 20 s.d. 34. Hasil yang didapatkan adalah penanda ArUco dengan ukuran 28 s.d. 34 sukses terdeteksi seluruhnya dengan faktor penskalaan 1, seperti dapat dilihat pada Tabel 4.2, artinya hasil gambar tidak perlu di-*scaling* untuk mendapatkan hasil deteksi kedelapan penanda ArUco. Namun, supaya ukuran penanda ArUco tidak terlalu besar, penulis memilih ukuran 28 dan 30 untuk diuji performanya lebih lanjut. Pengujian selanjutnya dilakukan dengan menambahkan penanda ArUco ke dalam gambar.

jutnya adalah menggunakan 40 data untuk masing-masing penanda ArUco berukuran 28 dan 30.

Tabel 4.3. Tabel hasil pendekripsi penanda ArUco dengan berbagai ukuran

Ukuran ArUco	Jumlah Sukses Terdeteksi	Faktor Penskalaan Terkecil
28	40	0.8
30	40	0.9

Hasilnya, seperti dapat dilihat pada Tabel 4.3, baik penanda ArUco berukuran 28 ataupun 30, semuanya dapat terdeteksi. Karena hasilnya sama baik, penulis akhirnya memutuskan untuk menggunakan ukuran penanda ArUco yang lebih kecil untuk menghemat ruang yang digunakan pada *watermark* berukuran 180x180 piksel.

#### 4.3.1 Parameter Akhir yang Digunakan

Dari hasil verifikasi parameter, berikut adalah parameter pembuatan model SQR:

- Ukuran SQR: 580 x 580 piksel
- Ukuran SQR beserta *padding*: 720 x 720 piksel
- Ukuran *watermark*: 180 x 180 piksel
- Ukuran CDP: 100 x 100 piksel
- Jumlah penanda ArUco: 8
- Ukuran penanda ArUco: 30 x 30 piksel

Untuk parameter P&S adalah sebagai berikut:

- ISO: 200
- *Shutter speed*: 1/100
- Fokus: 0,00
- *White balance*: 4000
- Jenis kertas: *Art carton*
- Jenis tinta: Tinta buku

#### 4.4 Deskripsi Dataset

*Dataset* yang digunakan dalam foto SQR sejumlah 400 data dengan rincian: 200 data SQR orisinal 4 level serta 200 data SQR palsu 4 level. *Dataset* tersebut dipotret menggunakan parameter yang telah ditetapkan melalui eksperimen. SQR orisinal merupakan SQR yang dicetak dari format digitalnya, sedangkan SQR palsu di-generate de-

ngan menempelkan CDP hasil lokalisasi ke dalam *template* digital kode QR. Contoh sampel data *raw photo* sebelum diolah dapat dilihat pada Gambar 4.4.



Gambar 4.4. Sampel data yang dipotret

## 4.5 Analisis Hasil Lokalisasi CDP

Analisis yang dilakukan adalah mencari interpolasi penskalaan yang memberikan hasil paling mirip dengan *template*, kemudian menganalisis perbandingan hasil lokalisasi CDP yang menggunakan penanda ArUco (8 titik) dan tanpa menggunakan penanda ArUco (4 titik).

### 4.5.1 Analisis Perbandingan Hasil Lokalisasi CDP Orisinal dengan Berbagai Interpolasi Penskalaan

Sebelum membuat *dataset* SQR palsu, penulis melakukan analisis terhadap *dataset* CDP orisinal hasil lokalisasi. Salah satu analisis yang dilakukan adalah mencari interpolasi penskalaan yang hasil rata-rata jaraknya dengan *template* paling minimal. Dengan jarak paling kecil, berarti hasil dari pengolahan data foto pertama dengan interpolasi penskalaan tersebut paling mirip dengan *template*. Hal tersebut menjadi salah satu langkah penyerangan paling sederhana terhadap CDP orisinal. Hasil rata-rata jarak CDP orisinal dengan template dari masing-masing interpolasi penskalaan dan koefisien jarak dapat dilihat pada Tabel 4.4.

Tabel 4.4. Hasil rata-rata jarak CDP orisinal dengan *template* dari berbagai interpolasi penskalaan dan koefisien jarak

	INTER_NEAREST	INTER_LINEAR	INTER_AREA	INTER_CUBIC	INTER_LANCZOS4
<b>sp_corr</b>	0.8733	<b>0.1558</b>	0.219	0.1818	0.1969
<b>sp_cosine</b>	0.5271	<b>0.0708</b>	0.1751	0.102	0.1117
<b>sp_euclidean</b>	0.7409	<b>0.1242</b>	0.4054	0.1998	0.2166
<b>sp_canberra</b>	0.6098	<b>0.1355</b>	0.5022	0.239	0.2464
<b>his_corr</b>	0.9097	0.7019	0.8851	0.6437	<b>0.4568</b>
<b>his_cosine</b>	0.964	0.5308	0.9642	0.5391	<b>0.461</b>
<b>his_euclidean</b>	0.7835	<b>0.1689</b>	0.7843	0.2045	0.2023
<b>his_canberra</b>	0.9329	0.6476	0.8123	0.2149	<b>0.1765</b>
<b>dct_corr</b>	0.4504	<b>0.058</b>	0.1574	0.0846	0.0922
<b>dct_cosine</b>	0.4502	<b>0.058</b>	0.1574	0.0846	0.0922
<b>dct_euclidean</b>	0.7326	<b>0.1623</b>	0.4169	0.2271	0.2408
<b>dct_canberra</b>	0.7462	<b>0.2985</b>	0.3739	0.6722	0.7312

Dari Tabel 4.4, terlihat bahwa interpolasi INTER\_LINEAR merupakan interpolasi penskalaan yang memiliki rata-rata jarak paling minimal, sedangkan INTER\_NEAREST memiliki jarak paling besar. Hal ini menunjukkan CDP hasil lokalisasi dengan interpolasi penskalaan INTER\_LINEAR paling mirip dengan *template*.

Untuk hasil perbandingan secara visual hasil lokalisasi CDP menggunakan berbagai interpolasi penskalaan dapat dilihat pada Gambar x. Terlihat secara kasat mata bahwa CDP yang dilokalisasi dengan interpolasi penskalaan INTER\_LINEAR memiliki hasil yang lebih mirip dengan *template* dibandingkan dengan yang lain, terutama CDP yang dilokalisasi dengan interpolasi penskalaan INTER\_NEAREST. Dari hasil tersebut, penulis memutuskan untuk menggunakan interpolasi penskalaan INTER\_LINEAR dalam pembuatan pemrosesan CDP dan pembuatan *dataset* selanjutnya.

#### 4.5.2 Analisis Perbandingan Hasil Lokalisasi CDP menggunakan Penanda ArUco dan Tanpa Penanda ArUco

Analisis selanjutnya adalah membandingkan hasil lokalisasi CDP menggunakan delapan titik acuan dan empat titik acuan. Delapan titik acuan di sini adalah delapan penanda ArUco, sedangkan empat titik adalah keempat titik sudut dari kode QR. Analisis yang dilakukan adalah menggunakan koefisien jarak yang mengukur jarak dari CDP hasil lokalisasi dengan *template*. Hasil lokalisasi yang lebih baik adalah yang jaraknya dengan *template* lebih kecil yang berarti lebih mirip dengan *template*. Hipotesis awal dari penulis adalah hasil lokalisasi menggunakan 8 titik acuan akan lebih baik daripada 4 titik acuan, terlebih SQR dipotret pada kondisi permukaan yang tidak rata atau ada lekukan pada sisisisi SQR. Koefisien jarak yang digunakan untuk mengukur kemiripan adalah koefisien jarak korelasi, kosinus, dan *euclidean* dari fitur spasial. Pengujian dilakukan pada *dataset* SQR orisinal dan palsu 4 level. Perbandingan yang dilakukan adalah berdasarkan rata-rata nilai jarak dari grup data.

Tabel 4.5. Hasil perbandingan rata-rata jarak hasil lokalisasi 8 titik dan 4 titik pada *dataset* CDP orisinal 4 level

	<b>8 Titik</b>	<b>4 Titik</b>
<b>sp_corr</b>	<b>0.3994</b>	0.8153
<b>sp_cosine</b>	<b>0.1113</b>	0.1347
<b>sp_euclidean</b>	<b>27474</b>	28092

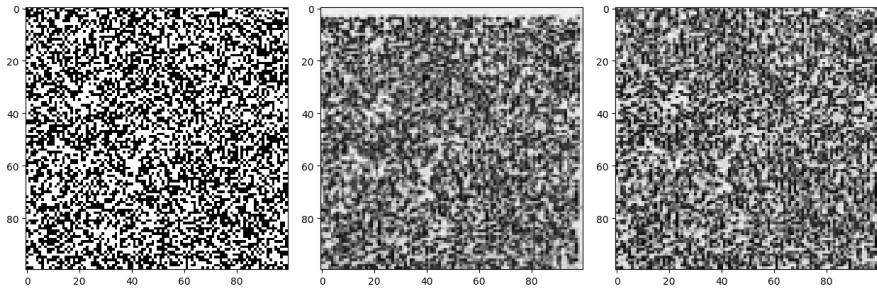
Hasil pada data CDP orisinal seperti yang terlihat pada Tabel 4.5 menunjukkan bahwa CDP yang dilokalisasi menggunakan delapan penanda ArUco atau 8 titik memiliki hasil jarak yang lebih kecil. Hal ini menunjukkan bahwa pada data CDP orisinal, lokalisasi menggunakan 8 titik lebih baik daripada 4 titik dalam hal kemiripannya dengan *template*.

Tabel 4.6. Hasil perbandingan rata-rata jarak hasil lokalisasi 8 titik dan 4 titik pada *dataset* CDP palsu 4 level

	<b>8 titik</b>	<b>4 titik</b>
<b>sp_corr</b>	<b>0.6147</b>	0.8749
<b>sp_cosine</b>	<b>0.108</b>	0.1382
<b>sp_euclidean</b>	<b>27797</b>	29880

Hasil yang sama juga didapat pada CDP palsu, CDP yang dilokalisasi menggunakan delapan penanda ArUco atau 8 titik memiliki hasil jarak yang lebih kecil. Hal ini menunjukkan bahwa pada data CDP palsu, lokalisasi menggunakan 8 titik lebih baik daripada 4 titik dalam hal kemiripannya dengan *template*. Untuk perbandingan visual salah satu hasil lokalisasi CDP menggunakan 4 titik dan 8 titik dapat dilihat pada Gambar 4.5. Terlihat bahwa hasil lokalisasi CDP menggunakan 4 titik kurang rapi, ada celah kosong di atasnya yang bukan merupakan komponen CDP itu sendiri. Selain itu, hasil lokalisasi CDP menggunakan 4 titik terlihat lebih kabur. Hal tersebut dapat disebabkan ada lekukan pada data SQR yang dipotret. Hasil lokalisasi CDP menggunakan 8 titik terlihat jauh lebih rapi pada perbandingan tersebut.

Dari kedua pengujian pada CDP orisinal dan CDP palsu, dapat disimpulkan bahwa lokalisasi CDP menggunakan delapan penanda ArUco menghasilkan hasil lokalisasi yang lebih baik dalam hal kemiripannya dengan *template*. Hal tersebut tentunya akan berpengaruh dalam akurasi model yang akan dibuat, karena data CDP yang diproses lebih berkualitas.



Gambar 4.5. Perbandingan CDP *template* dengan yang dilokalisasi dengan 4 titik (tengah) dan CDP yang dilokalisasi dengan 8 titik (kanan)

#### 4.5.3 Analisis Signifikansi CDP Orisinal dan Palsu

Analisis ini digunakan untuk mengetahui apakah ada perbedaan karakteristik yang mencolok antara CDP orisinal dan palsu, baik yang dilokalisasi menggunakan penanda ArUco (8 titik) maupun yang dilokalisasi tanpa menggunakan penanda ArUco (4 titik). Analisis dilakukan dengan melakukan uji statistik *T-test* untuk menguji signifikansi antara data kelompok CDP orisinal dengan CDP palsu. Selain itu, dilakukan visualisasi plot distribusi untuk melihat secara visual karakteristik dan perbedaan dari kedua data grup berdasarkan jaraknya dengan *template* untuk tiap-tiap koefisien jarak.

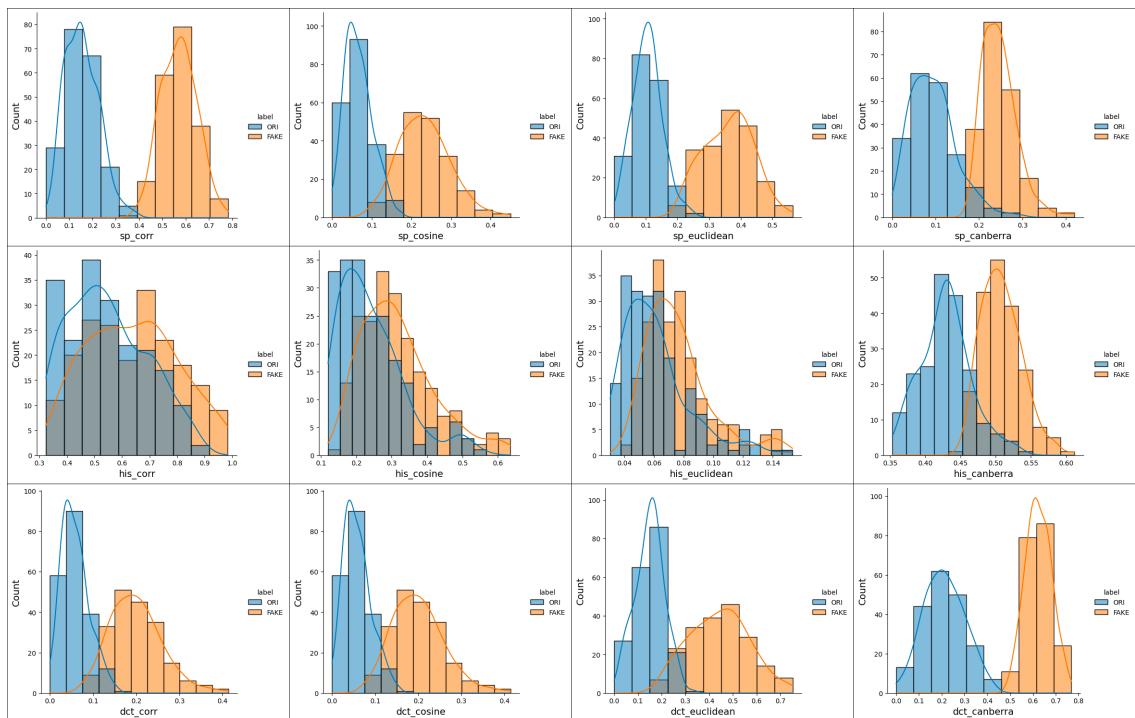
Tabel 4.7. Hasil pengujian *T-test* pada data grup fitur jarak CDP orisinal dengan palsu, pada CDP yang dilokalisasi dengan 8 titik dan CDP yang dilokalisasi dengan 4 titik

	8 Titik		4 Titik	
	T-statistic	P-value	T-statistic	P-value
<b>sp_corr</b>	-56.5046	3.32E-192	-4.3128	2.04E-05
<b>sp_cosine</b>	-32.3355	2.08E-113	-0.8523	0.3945606046
<b>sp_euclidean</b>	-39.9488	2.43E-141	-5.4538	8.67E-08
<b>sp_canberra</b>	-33.0441	3.86E-116	-5.19	3.36E-07
<b>his_corr</b>	-6.0236	3.89E-09	-6.0636	3.10E-09
<b>his_cosine</b>	-7.7801	6.30E-14	-10.7008	1.17E-23
<b>his_euclidean</b>	-6.6484	9.79E-11	-11.2637	9.84E-26
<b>his_canberra</b>	-24.5923	7.04E-82	-26.4865	7.49E-90
<b>dct_corr</b>	-29.4935	3.53E-102	-1.2032	0.2296019564
<b>dct_cosine</b>	-29.5221	2.71E-102	-1.2216	0.2225752757
<b>dct_euclidean</b>	-29.7315	3.90E-103	-8.8239	3.53E-17
<b>dct_canberra</b>	-55.4671	2.32E-189	-7.6051	2.07E-13

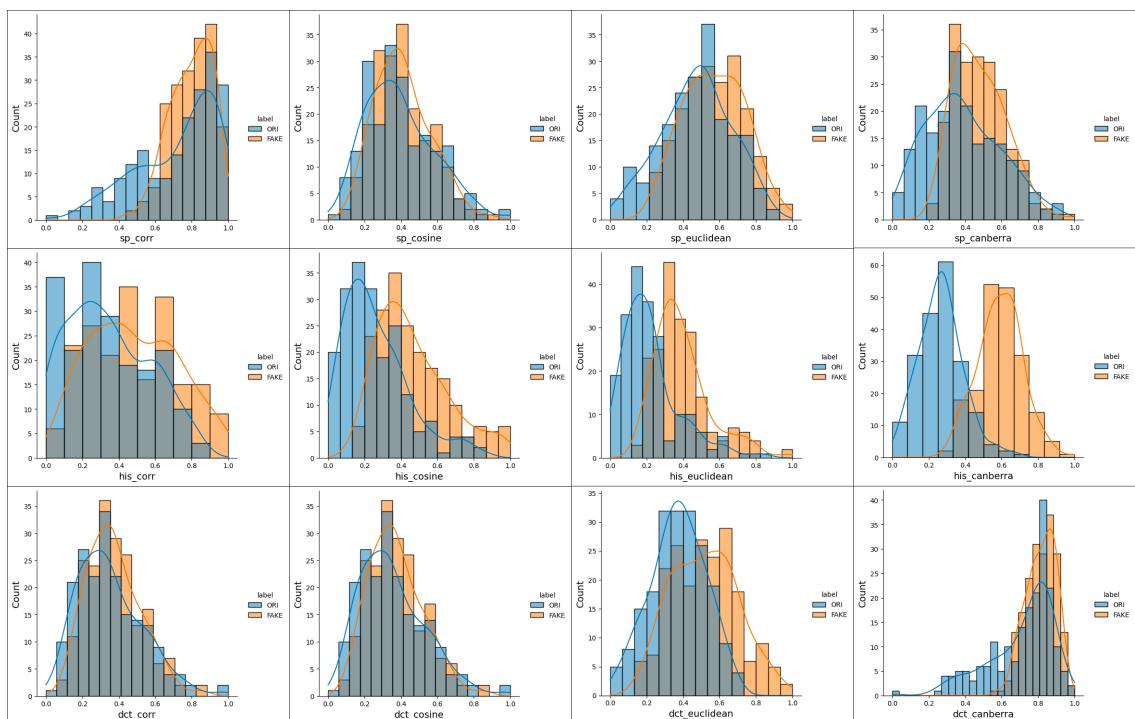
Dari Tabel 4.7 terlihat bahwa dengan menggunakan nilai *P-value* = 0,5, pada CDP yang dilokalisasi menggunakan 8 titik, seluruh fitur dapat dikatakan signifikan antara CDP orisinal dengan CDP palsunya. Namun, pada CDP yang dilokalisasi menggunakan

4 titik, ada yang nilainya tidak signifikan, yaitu pada fitur jarak *sp\_cosine*, *dct\_corr*, dan *dct\_cosine*. Selain itu, level signifikansinya juga berbeda jauh antara CDP yang dilokalisasi dengan 8 titik dan CDP yang dilokalisasi dengan 4 titik. CDP yang dilokalisasi dengan 8 titik memiliki signifikansi yang jauh lebih tinggi, terlihat dari kecilnya *P-value* yang didapat pada masing-masing koefisien jarak.

Selain itu, untuk memudahkan melihat signifikansi antara kedua grup secara visual, penulis melakukan plot distribusi untuk tiap-tiap koefisien jarak. Untuk hasil plot distribusi *dataset* CDP yang dilokalisasi dengan 8 titik dan *dataset* CDP yang dilokalisasi dengan 4 titik dapat dilihat pada Gambar 4.6 dan 4.7. Hasilnya sangat terlihat bahwa pada *dataset* CDP yang dilokalisasi dengan 8 titik, distribusi antara CDP orisinal dengan palsunya dapat terpisahkan dengan baik, sedangkan pada *dataset* CDP yang dilokalisasi dengan 4 titik, distribusi antara CDP orisinal dengan palsunya tidak terpisahkan dengan baik, terlihat dari banyaknya *overlap*. Hal tersebut menunjukkan bahwa CDP yang dilokalisasi dengan 8 titik mendapatkan fitur yang signifikansi antara CDP orisinal dengan palsunya sangat tinggi.



Gambar 4.6. Plot distribusi koefisien jarak pada CDP 4 level yang dilokalisasi dengan 8 titik



Gambar 4.7. Plot distribusi koefisien jarak pada CDP 4 level yang dilokalisasi dengan 4 titik

## 4.6 Hasil Pemodelan Klasifikasi Biner

Pemodelan dilakukan menggunakan dua metode, pertama menggunakan fitur tunggal dari tiap-tiap koefisien jarak, kedua menggunakan multifitur seluruh fitur koefisien jarak. Akurasi yang dibandingkan adalah akurasi klasifikasi biner pada *dataset* CDP yang dilokalisasi dengan 8 titik dengan *dataset* yang dilokalisasi dengan 4 titik.

### 4.6.1 Fitur Tunggal

Untuk menguji performa fitur dalam model, penulis menggunakan semua fitur koefisien jarak satu per satu, baik fitur spasial, histogram, ataupun dct. Model dibuat dengan AutoGluon dengan memilih model dengan akurasi tertinggi pada data tes. Hasil akurasi klasifikasi biner oleh model dalam mengklasifikasikan SQR orisinal atau palsu dapat dilihat pada Tabel 4.8.

Tabel 4.8. Hasil akurasi model klasifikasi biner menggunakan fitur tunggal

	Akurasi	
	8 Titik	4 Titik
<b>sp_corr</b>	<b>100.00%</b>	71.67%
<b>sp_cosine</b>	<b>94.17%</b>	60.83%
<b>sp_euclidean</b>	<b>98.33%</b>	65.00%
<b>sp_canberra</b>	<b>95.83%</b>	68.33%
<b>his_corr</b>	57.50%	<b>65.00%</b>
<b>his_cosine</b>	69.17%	<b>75.00%</b>
<b>his_euclidean</b>	69.17%	<b>83.33%</b>
<b>his_canberra</b>	90.83%	<b>92.50%</b>
<b>dct_corr</b>	<b>94.17%</b>	55.00%
<b>dct_cosine</b>	<b>94.17%</b>	57.50%
<b>dct_euclidean</b>	<b>94.17%</b>	68.33%
<b>dct_canberra</b>	<b>100.00%</b>	67.50%

Dari hasil yang terlihat pada Tabel 4.8, terlihat bahwa CDP yang dilokalisasi dengan 8 titik memiliki akurasi klasifikasi yang baik dibandingkan CDP yang dilokalisasi dengan 4 titik. Akurasi tertinggi pada CDP yang dilokalisasi dengan 8 titik didapatkan menggunakan fitur jarak *sp\_corr* dan *dct\_canberra* dengan akurasi 100%. Pada CDP yang dilokalisasi dengan 4 titik, akurasi tertinggi didapatkan menggunakan fitur jarak *his\_canberra* dengan akurasi 92,5%.

#### 4.6.2 Multi Fitur

Tabel 4.9. Perbandingan akurasi autentikasi klasifikasi biner pada *dataset* CDP 4 level yang dilokalisasi dengan 8 titik dan *dataset* CDP 4 level yang dilokalisasi dengan 4 titik

8 titik		4 titik	
Model	score_test	Model	score_test
XGBoost	100.00%	NeuralNetFastAI	98.33%

Dari hasil yang ditunjukkan pada Tabel 4.9, terlihat bahwa akurasi tertinggi yang bisa didapatkan pada *dataset* yang dilokalisasi dengan 8 titik adalah 100% menggunakan model XGBoost, sedangkan pada *dataset* yang dilokalisasi dengan 4 titik, akurasi tertinggi yang didapatkan adalah 98,33% dengan model NeuralNetFastAI.

Dari hasil-hasil tersebut, baik pembuatan model autentikasi klasifikasi biner menggunakan fitur tunggal atau multi fitur, dapat disimpulkan bahwa *dataset* yang dilokalisasi dengan 8 titik (dengan bantuan penanda ArUco) memiliki akurasi yang lebih tinggi dibandingkan *dataset* yang dilokalisasi dengan 4 titik (4 titik sudut kode QR).

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

- Model SQR yang dirancang penulis (Kode QR dengan *watermark*) dapat diautentikasi dengan akurasi sempurna 100% (pada *dataset* yang dilokalisasi dengan penanda ArUco)
- Selain model SQR, keluaran dari penelitian ini adalah parameter P&S untuk pembuatan *dataset* SQR.
- Dari uji signifikansi menggunakan metode *T-test* antara data grup orisinal dengan palsu, pada *dataset* CDP yang dilokalisasi dengan 8 titik memiliki signifikansi tinggi pada seluruh koefisien jaraknya, sedangkan pada *dataset* CDP yang dilokalisasi dengan 4 titik, hasilnya tidak terlalu signifikan, terlihat dari banyaknya *overlap* pada plot distribusinya.
- Dari pengujian pembuatan model autentikasi menggunakan fitur tunggal pada *dataset* CDP yang dilokalisasi menggunakan 8 titik, fitur *sp\_corr* dan *dct\_canberra* menjadi fitur fitur terbaik yang mendapatkan akurasi sempurna 100%, sedangkan pada *dataset* CDP yang dilokalisasi menggunakan 4 titik, akurasi tertingginya adalah 92,5% menggunakan fitur *his\_canberra*.
- Dari pengujian pembuatan model autentikasi menggunakan multi fitur, pada *dataset* CDP yang dilokalisasi menggunakan 8 titik memiliki akurasi tertinggi 100% menggunakan model XGBoost, sedangkan pada *dataset* CDP yang dilokalisasi menggunakan 4 titik, akurasi tertingginya adalah 98,33%.
- AutoML dari AutoGluon dapat membuat model autentikasi dengan baik untuk kasus klasifikasi biner CDP orisinal dan palsu, baik untuk CDP 2 maupun 4 level.

#### **5.2 Saran**

- Model SQR dan parameter P&S ini dapat digunakan untuk pembuatan *dataset* berikutnya dengan jumlah yang lebih besar.
- Dibutuhkan pembuatan *dataset* SQR dengan lingkungan pemotretan yang berbeda, semisal dipotret langsung di ruang terbuka dengan berbagai kondisi, namun hasilnya harus tetap baik (fokus dan tidak kabur) dengan menggunakan parameter konfigurasi kamera yang penulis dapatkan.
- Parameter konfigurasi kamera yang telah penulis dapatkan dapat digunakan untuk pengembangan aplikasi *mobile* untuk autentikasi SQR.

- Penelitian tentang penyerangan estimasi CDP 4 level perlu dilakukan untuk mengetahui efektivitas dan ketahanan CDP 4 level.
- Fitur-fitur yang memiliki akurasi tinggi dalam model fitur tunggal dan memiliki signifikansi distribusi tinggi dalam membedakan CDP orisinal dan palsu seperti *sp\_corr*, *sp\_cosine*, *sp\_euclidean*, *sp\_canberra*, dan *dct\_canberra* dapat dijadikan fitur utama yang digunakan dalam pembuatan model autentikasi multi fitur.
- Dibutuhkan pustaka untuk pembacaan data kode QR yang lebih cepat dari pustaka yang dimiliki Python (OpenCV), misalnya pustaka pembaca QR *native* dari Java Android.

## DAFTAR PUSTAKA

- [1] BASCAP and INTA, *The Economic Impacts of Counterfeiting and Piracy*. Business Action to Stop Counterfeiting and Piracy (BASCAP), 2016.
- [2] C. W. Hill, “Digital piracy: Causes, consequences, and strategic responses,” *Asia Pacific Journal of Management*, vol. 24, pp. 9–25, 2007.
- [3] B. Depoorter, “Intellectual property infringements & 3d printing: Decentralized piracy,” *Hastings LJ*, vol. 65, p. 1483, 2013.
- [4] 2023. [Online]. Available: [https://docs.opencv.org/4.x/d5/dae/tutorial\\_aruco\\_detection.html](https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html)
- [5] E. Khermaza, I. Tkachenko, and J. Picard, “Can copy detection patterns be copied? evaluating the performance of attacks and highlighting the role of the detector,” in *2021 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2021, pp. 1–6.
- [6] C. W. Wong and M. Wu, “Counterfeit detection based on unclonable feature of paper using mobile camera,” *IEEE Transactions on Information Forensics and Security*, vol. 12, 2017.
- [7] R. Schraml, L. Debiasi, and A. Uhl, “Real or fake: Mobile device drug packaging authentication,” in *Proceedings of the 6th ACM workshop on information hiding and multimedia security*, 2018, pp. 121–126.
- [8] J. Picard, “Digital authentication with copy-detection patterns,” in *Optical Security and Counterfeit Deterrence Techniques V*, vol. 5310. SPIE, 2004, pp. 176–183.
- [9] A. T. Phan Ho, B. A. Mai Hoang, W. Sawaya, and P. Bas, “Document authentication using graphical codes: Reliable performance analysis and channel optimization,” *EURASIP Journal on Information Security*, vol. 2014, pp. 1–17, 2014.
- [10] J. Picard, “On the security of copy detectable images,” in *NIP & Digital Fabrication Conference*, vol. 2008, no. 2. Society for Imaging Science and Technology, 2008, pp. 796–798.
- [11] A. E. Dirik and B. Haas, “Copy detection pattern-based document protection for variable media,” *IET Image Processing*, vol. 6, no. 8, pp. 1102–1113, 2012.
- [12] J. Picard, P. Landry, and M. Bolay, “Counterfeit detection with qr codes,” in *Proceedings of the 21st ACM Symposium on Document Engineering*, 2021, pp. 1–4.
- [13] D. ADC, “Qr code essentials,” 2011.
- [14] J. Picard, C. Vielhauer, and N. Thorwirth, “Towards fraud-proof id documents using multiple data hiding technologies and biometrics,” in *Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306. SPIE, 2004, pp. 416–427.
- [15] C. Harwood, “Optical document security,” *Kybernetes*, vol. 27, no. 5, pp. 586–588, 1998.

- [16] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” in *Computer Vision, IEEE International Conference on*, vol. 3. IEEE Computer Society, 2003, pp. 1470–1470.
- [17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.
- [18] R. C. Gonzalez, R. E. Woods, and B. R. Masters, “Digital image processing, third edition,” *Journal of Biomedical Optics*, vol. 14, 2009.
- [19] S. H.L, “Pixel intensity histogram characteristics: Basics of image processing and machine vision,” Dec 2017. [Online]. Available: <https://www.allaboutcircuits.com/technical-articles/image-histogram-characteristics-machine-learning-image-processing/>
- [20] 2023. [Online]. Available: <https://users.cs.cf.ac.uk/dave/Multimedia/node231.html>
- [21] M. Al Baloshi and M. E. Al-Mualla, “A dct-based watermarking technique for image authentication,” in *2007 IEEE/ACS International Conference on Computer Systems and Applications*, 2007, pp. 754–760.
- [22] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.
- [23] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for machine learning*. Cambridge University Press, 2020.
- [24] S. Al-Anazi, H. Almahmoud, and I. Al-Turaiki, “Finding similar documents using different clustering techniques,” vol. 82, 2016.
- [25] P. H. Sneath, R. R. Sokal *et al.*, *Numerical taxonomy. The principles and practice of numerical classification.*, 1973.
- [26] A. Kumar, “Two independent samples t-tests: Formula & examples,” Mar 2023. [Online]. Available: <https://vitalflux.com/two-sample-t-test-formula-examples/>
- [27] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.
- [28] E. Alpaydin, “Introduction to machine learning ethem alpaydin.” *Introduction to Machine Learning, Third Edition*, 2014.
- [29] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [30] manashgoswami, “What is automated ml? automl - azure machine learning,” Apr 2023. [Online]. Available: <https://learn.microsoft.com/en-us/azure/machine-learning/concept-automated-ml?view=azureml-api-2>
- [31] S. Singla, “What is automl and why is it important? | towards data science,” Oct 2020. [Online]. Available: <https://towardsdatascience.com/what-is-automl-6ddf27040f27>

- [32] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, “Autogluon-tabular: Robust and accurate automl for structured data,” *arXiv preprint arXiv:2003.06505*, 2020.

# LAMPIRAN

## L.1 Kode Sumber

### L.1.1 Kode Sumber Utama

```
1 import pandas as pd
2 import seaborn as sns
3 import qrcode
4 import numpy as np
5 import math
6 import hashlib
7 import random
8 import cv2
9 from pillow_heif import register_heif_opener
10 import shutil
11
12 from scipy.stats import ttest_ind
13 from scipy.spatial.distance import pdist
14 from scipy.special import rel_entr
15 from scipy.spatial import distance
16 from scipy.fftpack import dct
17 from scipy.ndimage import gaussian_filter
18
19 from skimage.feature import match_template
20 from sklearn.preprocessing import MinMaxScaler
21 import matplotlib.pyplot as plt
22 from PIL import Image
23 from cv2 import aruco
24 from pathlib import Path
25 import os
26
27 SECRET = "rispro"
28
29 pd.set_option("display.max_rows", 50)
30 pd.set_option("display.max_columns", None)
31
32 qrDecoder = cv2.QRCodeDetector()
33 aruco_dict = aruco.getPredefinedDictionary(aruco.DICT_4X4_50)
34 parameters = aruco.DetectorParameters_create()
35 register_heif_opener()
```

```

36
37
38     def minimal_distance(points):
39         min_distance = float("inf")
40         for i in range(len(points)):
41             for j in range(i + 1, len(points)):
42                 min_distance = min(min_distance, math.sqrt((
43                     points[i][0] - points[j][0]) ** 2 + (points[i][1] - points[j]
44                     [1]) ** 2))
45
46     return int(min_distance)
47
48
49     def generate_watermark_with_aruco_marker(data: str, quant:
50         int, aruco_size: int, wtm_size: tuple = (100, 100), secret:
51         str = SECRET) -> np.ndarray:
52         s = data + secret
53         # String diencode jadi byte string, dicarihexdigestnya
54         # atau nilai hashnya menggunakan algoritma sha1, mengubah
55         # output hexdigest string jadi integer 16 bit, lalu mengambil
56         # nilai absolutnya (harus positif) lalu disederhanakan menjadi
57         # 32 bit integer
58         seed = abs(int(hashlib.sha1(s.encode("utf-8")).hexdigest()
59         (), 16)) % 2**32
60         random.seed(seed)
61         np.random.seed(seed)
62
63         dist = [(i * 255 // (quant - 1)) if (i * 255 // (quant -
64             1)) <= 255 else 255 for i in range(quant)] # Membuat
65         distribusi elemen [0, 255]
66         distribution = [[random.randint(1, 3) for j in range(
67             quant)] for i in range(10)] # Membuat distribusi
68         perbandingan tiap2 elemen kuantisasi sebanyak 10 macam,
69         maksimal perbandingan 1:3 -> [[1, 2], [1, 3], ..., [1, 1]]
70         # Repeat the array of values "dist" a random number "
71         random.choice(distribution) -> [1,2]" of times
72         element_distribution = np.repeat(dist, random.choice(
73             distribution))
74
75         # Membuat array sampling berukuran "size" dengan tiap
76         # elemen dapat diulang dari distribusi elemen yang sudah
77         # digenerate
78         wtm = np.random.choice(element_distribution, size=

```

```

        wtm_size , replace=True)

59
60         padding_size = 40
61         wtm_pad = np.pad(wtm, (padding_size), constant_values
62             =255)
63
64         marker_list = [aruco.drawMarker(aruco_dict , i ,
65             aruco_size) for i in range(8)]
66
67         yloc1 = wtm_pad.shape[0] // 2 - wtm.shape[0] // 2 -
68             padding_size // 2 - aruco_size // 2
69         yloc2 = wtm_pad.shape[0] // 2 - aruco_size // 2
70         yloc3 = wtm_pad.shape[0] // 2 + wtm.shape[0] // 2 +
71             padding_size // 2 - aruco_size // 2
72         xloc1 = wtm_pad.shape[0] // 2 - wtm.shape[0] // 2 -
73             padding_size // 2 - aruco_size // 2
74         xloc2 = wtm_pad.shape[0] // 2 - aruco_size // 2
75         xloc3 = wtm_pad.shape[0] // 2 + wtm.shape[0] // 2 +
76             padding_size // 2 - aruco_size // 2
77
78         wtm_pad[yloc1 : yloc1 + aruco_size , xloc1 : xloc1 +
79             aruco_size] = marker_list[0]
80         wtm_pad[yloc1 : yloc1 + aruco_size , xloc2 : xloc2 +
81             aruco_size] = marker_list[1]
82         wtm_pad[yloc1 : yloc1 + aruco_size , xloc3 : xloc3 +
83             aruco_size] = marker_list[2]
84         wtm_pad[yloc2 : yloc2 + aruco_size , xloc1 : xloc1 +
85             aruco_size] = marker_list[3]
86         wtm_pad[yloc2 : yloc2 + aruco_size , xloc3 : xloc3 +
87             aruco_size] = marker_list[4]
88         wtm_pad[yloc3 : yloc3 + aruco_size , xloc1 : xloc1 +
89             aruco_size] = marker_list[5]
90         wtm_pad[yloc3 : yloc3 + aruco_size , xloc2 : xloc2 +
91             aruco_size] = marker_list[6]
92         wtm_pad[yloc3 : yloc3 + aruco_size , xloc3 : xloc3 +
93             aruco_size] = marker_list[7]

94
95     return wtm_pad.astype("uint8") , wtm.astype("uint8")
96
97
98 def generate_sqr(data: str , quant: int , aruco_size: int ,

```

```

qr_ver: int = 3, secret: str = SECRET) -> np.ndarray:
    qr = qrcode.make(data, version=qr_ver, box_size=20,
border=3, error_correction=qrcode.constants.ERROR_CORRECT_H)
    qr = np.asarray(qr).astype("uint8") * 255
    wtm_pad, wtm = generate_watermark_with_aruco_marker(data
=data, quant=quant, aruco_size=aruco_size, secret=secret)

loc = qr.shape[0] // 2 - wtm_pad.shape[0] // 2
qr[loc : loc + wtm_pad.shape[0], loc : loc + wtm_pad.
shape[0]] = wtm_pad
qr = np.pad(qr, 9, constant_values=255)
qr = np.pad(qr, 1, constant_values=0)

return qr.astype("uint8"), wtm.astype("uint8")

```

95

96

```

def detect_qr(srcpath, interpolation, result_type: str =
"nparray", blur: bool = True, ksize: int = 7):
    interpolation = eval(f"cv2.{interpolation}")
    raw_image = np.asarray(Image.open(srcpath).convert("L
"))
    if blur:
        # blurred_raw_image = cv2.medianBlur(raw_image,
ksize=ksize)
        blurred_raw_image = cv2.GaussianBlur(raw_image, (
ksize, ksize), 0)
        yz = int(raw_image.shape[0])
        xz = int(raw_image.shape[1])
        scale_factor_list = [i / 10 for i in reversed(range(1,
11))]
        for scale_factor in scale_factor_list:
            y = int(yz * scale_factor)
            x = int(xz * scale_factor)
            if blur:
                scaled_blurred_image = cv2.resize(
blurred_raw_image, (x, y), interpolation=interpolation)
                data, points, _ = qrDecoder.detectAndDecode(
scaled_blurred_image)
            else:
                scaled_image = cv2.resize(raw_image, (x, y),
interpolation=interpolation)

```

```
114     data , points , _ = qrDecoder.detectAndDecode(
115         scaled_image)
116
117     try :
118         if len(points[0]) == 4 and data != "":
119             if blur:
120                 scaled_image = cv2.resize(raw_image , (x,
121                     y) , interpolation=interpolation)
122
123             x = [p[0] for p in points[0]]
124             y = [p[1] for p in points[0]]
125             size = minimal_distance(points[0])
126             src = np.float32([points[0]])
127             dst = np.float32([[0, 0], [size, 0], [size,
128                 size], [0, size]])
129
130             H, _ = cv2.findHomography(src , dst)
131             cropped_qr = cv2.warpPerspective(
132                 scaled_image , H, (size , size))
133
134             if result_type == "nparray":
135                 return cropped_qr
136             elif result_type == "string":
137                 return f"QR detection success , with
138 scale factor: {scale_factor}!"
139             elif result_type == "data":
140                 return data
141             elif result_type == "cdp":
142                 length = cropped_qr.shape[0]
143                 crop_size = int(length * 100 / 580)
144                 start_pos = int((length - crop_size) /
145
146                     2)
147
148                 end_pos = start_pos + crop_size
149                 cdp = cropped_qr[start_pos:end_pos ,
150                     start_pos:end_pos]
151
152                 return cdp
153
154             except:
155                 continue
156
157             if result_type == "nparray":
158                 return np.empty((0, 0))
159             elif result_type == "string":
160                 return "Failed to detect QR!"
161             elif result_type == "data":
162                 return "Data not detected!"
```

```

148
149     def detect_qr2(srcpath, interpolation, result_type: str = "nparray", blur: bool = True, ksize: int = 7):
150         interpolation = eval(f"cv2.{interpolation}")
151         raw_image = np.asarray(Image.open(srcpath).convert("L"))
152         if blur:
153             # blurred_raw_image = cv2.medianBlur(raw_image,
154             # ksize=ksize)
155             blurred_raw_image = cv2.GaussianBlur(raw_image, (ksize, ksize), 0)
156             yz = int(raw_image.shape[0])
157             xz = int(raw_image.shape[1])
158             scale_factor_list = [i / 10 for i in reversed(range(1, 11))]
159             for scale_factor in scale_factor_list:
160                 y = int(yz * scale_factor)
161                 x = int(xz * scale_factor)
162                 if blur:
163                     scaled_blurred_image = cv2.resize(
164                         blurred_raw_image, (x, y), interpolation=interpolation)
165                     data, points, _ = qrDecoder.detectAndDecode(
166                         scaled_blurred_image)
167                 else:
168                     scaled_image = cv2.resize(raw_image, (x, y),
169                     interpolation=interpolation)
170                     data, points, _ = qrDecoder.detectAndDecode(
171                         scaled_image)
172                     try:
173                         if len(points[0]) == 4 and data != "":
174                             if blur:
175                                 scaled_image = cv2.resize(raw_image, (x, y),
176                                 interpolation=interpolation)
177                                 x = [p[0] for p in points[0]]
178                                 y = [p[1] for p in points[0]]
179                                 size = minimal_distance(points[0])
180                                 src = np.float32([points[0]])
181                                 dst = np.float32([[0, 0], [580, 0], [580,
182                                     580], [0, 580]])
183                                 H, _ = cv2.findHomography(src, dst)
184                                 cropped_qr = cv2.warpPerspective(

```

```

scaled_image, H, (580, 580))
178         if result_type == "nparray":
179             return cropped_qr
180         elif result_type == "string":
181             return f"QR detection success, with
scale factor: {scale_factor}!"
182         elif result_type == "data":
183             return data
184         elif result_type == "cdp":
185             length = cropped_qr.shape[0]
186             print(length)
187             crop_size = 100
188             start_pos = int((length - crop_size) /
2)
189             end_pos = start_pos + crop_size
190             cdp = cropped_qr[start_pos:end_pos,
start_pos:end_pos]
191             return cdp
192     except:
193         continue
194     if result_type == "nparray":
195         return np.empty((0, 0))
196     elif result_type == "string":
197         return "Failed to detect QR!"
198     elif result_type == "data":
199         return "Data not detected!"

200
201
202     def detect_aruco(srcpath, interpolation: str, result_type:
str = "nparray", blur: bool = True, ksize: int = 5):
203         interpolation = eval(f"cv2.{interpolation}")
204         raw_image = np.asarray(Image.open(srcpath).convert("L
"))
205         yz = int(raw_image.shape[0])
206         xz = int(raw_image.shape[1])
207         if blur:
208             # blurred_raw_image = cv2.medianBlur(raw_image,
ksize=ksize)
209             blurred_raw_image = cv2.GaussianBlur(raw_image, (
ksize, ksize), 0)
210             scale_factor_list = [i / 10 for i in reversed(range(1,

```

```

11))]

211     for scale_factor in scale_factor_list:
212         y = int(yz * scale_factor)
213         x = int(xz * scale_factor)
214         if blur:
215             scaled_blurred_image = cv2.resize(
216                 blurred_raw_image, (x, y), interpolation=interpolation)
217                 corners, ids, _ = aruco.detectMarkers(
218                     scaled_blurred_image, aruco_dict, parameters=parameters)
219             else:
220                 scaled_image = cv2.resize(raw_image, (x, y),
221                     interpolation=interpolation)
222                 corners, ids, _ = aruco.detectMarkers(
223                     scaled_image, aruco_dict, parameters=parameters)
224             try:
225                 if len(corners) == 8:
226                     if blur:
227                         scaled_image = cv2.resize(raw_image, (x,
228                             y), interpolation=interpolation)
229                         raw_coordinates_dict = {}
230                         index = 0
231                         for id in ids:
232                             raw_coordinates_dict[int(id)] = corners[
233                                 index]
234                             index += 1
235                             xy_list = []
236                             for i in range(8):
237                                 c = raw_coordinates_dict[i][0]
238                                 xy_list.append([c[:, 0].mean(), c[:, 1].
239                                 mean()])
240 # print(xy_list)
241 points = [xy_list[i] for i in [0, 2, 5, 7]]
242 min_dist = minimal_distance(points)
243 x = min_dist
244 src = np.float32([xy_list])
245 dst = np.float32([[0, 0], [x, 0], [2 * x,
246 0], [0, x], [2 * x, x], [0, 2 * x], [x, 2 * x],
247 [2 * x, 2 * x]])
248 H, _ = cv2.findHomography(src, dst)
249 cdp_with_marker = cv2.warpPerspective(
250     scaled_image, H, (min_dist * 2, min_dist * 2))

```

```

241     length = cdp_with_marker.shape[0]
242     crop_size = int(length * 100 / 140)
243     start_pos = int((length - crop_size) / 2)
244     end_pos = start_pos + crop_size
245     cdp = cdp_with_marker[start_pos:end_pos,
246                             start_pos:end_pos]
247
248             if result_type == "nparray":
249                 # return cdp_with_marker with 2 times
250                 # scaled up
251                 # return cv2.resize(
252                 #         cdp, (200, 200), interpolation=cv2
253                 #.INTER_CUBIC)
254
255             return cv2.resize(cdp, (100, 100),
256                               interpolation=interpolation)
257
258             # return cdp_with_marker
259             elif result_type == "string":
260                 return f"CDP detection with markers
261                 success with scale factor: {scale_factor}!"
262
263             except:
264                 continue
265
266             if result_type == "nparray":
267                 return np.empty((0, 0))
268             elif result_type == "string":
269                 return f"CDP Detection with marker failed, corners
270                 detected: {len(corners)}!"
271
272
273     def generate_original_batch_a3(dstpath, first: int, last:
274         int, quant: int, aruco_size: int = 30):
275
276         col = []
277         row = []
278         quant = quant
279         first = first
280         last = last
281         aruco_size = aruco_size
282
283         for i in range(first, last + 1, 1):
284             qr, _ = generate_sqr(data=f"DATA_{i:03}_{quant}"
285             LEVEL_NEW", quant=quant, aruco_size=aruco_size)
286             col.append(qr)
287
288             if i % 5 == 0:
289                 row.append(np.hstack(col))

```

```

274         col = []
275         result = np.vstack(row)
276         result = result.astype("uint8")
277         result = np.pad(result, ((281, 282), (353, 354)), constant_values=255) # Y dulu baru X
278     try:
279         os.makedirs(dstpath)
280     except:
281         pass
282     Image.fromarray(result).save(f"{dstpath}DATA_{first:03}_{last:03}_{quant}LEVEL_NEW_ORI.png")
283
284
285     def generate_fake(srcpath, dstpath, interpolation, quant: int, ksize: int, blur: bool = False):
286         filename_list = [f for f in os.listdir(srcpath) if os.path.isfile(srcpath + "/" + f)]
287         print(filename_list)
288     try:
289         os.makedirs(dstpath)
290     except:
291         pass
292     for i, filename in enumerate(filename_list):
293         data = filename.split(".")[0]
294         qr, wtm = generate_sqr(data=data, quant=quant, aruco_size=30)
295         loc = qr.shape[0] // 2 - 100 // 2
296         print(f"{filename}")
297     try:
298         fake_wtm = detect_aruco(srcpath=f"{srcpath}/{filename}", result_type="nparray", blur=blur, ksize=ksize, interpolation=interpolation)
299         qr[loc : loc + 100, loc : loc + 100] = fake_wtm
300         Image.fromarray(qr).save(f"{dstpath}/{data}_FAKE.png")
301     except:
302         continue
303
304
305     def generate_fake_batch_a3(srcpath, dstpath, first: int, last: int, aruco_size: int = 30, quant: int = 2):

```

```

306     col = []
307     row = []
308     quant = quant
309     first = first
310     last = last
311     aruco_size = aruco_size
312     filename_list = [f for f in os.listdir(srcpath) if os.
313     path.isfile(srcpath + "/" + f)]
314     filename_list.sort()
315     print(filename_list)
316     print(len(filename_list))
317     for i in range(first, last + 1):
318         qr = np.asarray(Image.open(f"{srcpath}/{filename_list[i-1]}").convert("L"))
319         print(qr.shape[0])
320         col.append(qr)
321         if i % 5 == 0:
322             row.append(np.hstack(col))
323             col = []
324             result = np.vstack(row)
325             result = result.astype("uint8")
326             result = np.pad(result, ((259, 260), (338, 339)), constant_values=255) # Y dulu baru X
327             try:
328                 os.makedirs(dstpath)
329             except:
330                 pass
331             Image.fromarray(result).save(f"{dstpath}DATA_{first:03}_"
332             {last:03}_{quant}LEVEL_ARC{aruco_size}_FAKE.png")
333
334     def batch_detection(srcpath, interpolation, blur: bool,
335     detection_type: str, result_type: str, ksize: int = 7):
336         filename_list = [f for f in os.listdir(srcpath) if os.
337         path.isfile(srcpath + "/" + f)]
338         filename_list.sort()
339         print(filename_list)

```

```

if detection_type == "aruco":
    for i, filename in enumerate(filename_list):
        print(i + 1, detect_aruco(srcpath=f"{srcpath}/{filename}",
result_type=result_type, blur=blur, ksize=ksize,
```

```

        interpolation=interpolation))
340    elif detection_type == "qr":
341        if result_type == "rename_data":
342            for i, filename in enumerate(filename_list):
343                data = detect_qr(srcpath=f"{srcpath}/{filename}",
344                                  result_type="data", blur=blur, ksize=ksize,
345                                  interpolation=interpolation)
346                if os.path.exists(srcpath + data + "_" + str(i) +
347                                  "." + filename.split(".")[1].split("_")[0]):
348                    os.rename(srcpath + filename, srcpath +
349                              data + "_" + str(i) + "." + filename.split(".")[1].split("_")[
350                                  0])
351                else:
352                    os.rename(srcpath + filename, srcpath +
353                              data + "." + filename.split(".")[1].split("_")[0])
354                    print(i + 1, data)
355    elif result_type == "string":
356        for i, filename in enumerate(filename_list):
357            print(i + 1, detect_qr(srcpath=f"{srcpath}/{filename}",
358                                  result_type=result_type, interpolation=
359                                  interpolation))
360
361
362    # DCT Features
363    do = dct(scn_rsz).ravel()
364    dt = dct(tmp_rsz).ravel()
365
366    # Histogram Features
367    ho = np.histogram(scn_rsz, bins=range(257))[0] / sz**2
368    ht = np.histogram(tmp_rsz, bins=range(257))[0] / sz**2
369
370    # Spatial Features
371    scn_rsz = scn_rsz.ravel()
372    tmp_rsz = tmp_rsz.ravel()

```

```

369
370     features = {
371         "interpolation": interpolation,
372         "data": data,
373         "sp_corr": pdist([scn_rsz, tmp_rsz], "correlation")
374             [0],
375         "sp_cosine": pdist([scn_rsz, tmp_rsz], "cosine")[0],
376         "sp_euclidean": pdist([scn_rsz, tmp_rsz], "euclidean")
377             )[0],
378         "sp_canberra": pdist([scn_rsz, tmp_rsz], "canberra")
379             [0],
380         "his_corr": pdist([ho, ht], "correlation")[0],
381         "his_cosine": pdist([ho, ht], "cosine")[0],
382         "his_euclidean": pdist([ho, ht], "euclidean")[0],
383         "his_canberra": pdist([ho, ht], "canberra")[0],
384         "dct_corr": pdist([do, dt], "correlation")[0],
385         "dct_cosine": pdist([do, dt], "cosine")[0],
386         "dct_euclidean": pdist([do, dt], "euclidean")[0],
387         "dct_canberra": pdist([do, dt], "canberra")[0],
388         "level": level,
389     }
390
391     return features
392
393
394     def generate_features_csv(srcpath, dstpath, type: str):
395         filename_list = [f for f in os.listdir(srcpath) if os.
396             path.isfile(srcpath + "/" + f) and srcpath + "/" + f != f"{
397                 srcpath }/.DS_Store"]
398
399         result = []
400
401         for i, filename in enumerate(filename_list):
402             # filename_split = filename.split("_")
403
404             interpolation = "_" .join(filename .split("."))[0].
405             split("_")[1:3]
406             data = "_" .join(filename .split("."))[0].split("_")
407                 [3:7])
408             level = filename .split(".")[0].split("_")[5]
409             print(level)

```

```

403     if level == "2LEVEL":
404         _, template = generate_sqr(data=data, quant=2,
405         aruco_size=30)
406     elif level == "4LEVEL":
407         _, template = generate_sqr(data=data, quant=4,
408         aruco_size=30)
409         scanned = np.asarray(Image.open(f"{srcpath}/{filename}").convert("L"))
410         features = get_features(data=data, template=template,
411         , scanned=scanned, interpolation=interpolation, level=level)
412         result.append(features)
413         print(i, filename)
414
415         print(len(result))
416         df_features = pd.DataFrame(result)
417         df_features_sorted = df_features.sort_values(by=[
418             "interpolation", "data"], ascending=[True, True]).reset_index(
419             drop=True)
420         df_features_sorted["label"] = f"{type}"
421         df_features_sorted.to_csv(f"{dstpath}{type}.csv", index=
422             False)
423
424
425     def localize_cdp_from_raw_photo_batch(srcpath, dstpath):
426         RESIZING_INTERPOLATION = ["INTER_NEAREST", "INTER_LINEAR",
427             "INTER_AREA", "INTER_CUBIC", "INTER_LANCZOS4"]
428
429         filename_list = [f for f in os.listdir(srcpath) if os.
430             path.isfile(srcpath + "/" + f) and srcpath + "/" + f != f"{srcpath}/.DS_Store"]
431
432         try:
433             os.makedirs(dstpath)
434         except:
435             pass
436         for i, filename in enumerate(filename_list):
437             for interpolation in RESIZING_INTERPOLATION:
438                 result = detect_aruco(srcpath=f"{srcpath}{filename}",
439                     interpolation=interpolation, result_type="numpy")
440                 print(i, filename)

```

```

432         try :
433             Image . fromarray ( result ) . save ( f" { dstpath } CDP_
434             { interpolation }_{ filename . split ( ' .' )[ 0 ] }. png " )
435         except :
436             continue
437
438     def localize_cdp_from_raw_photo_batch_inter_linear ( srcpath ,
439         dstpath ) :
440         filename_list = [ f for f in os . listdir ( srcpath ) if os .
441             path . isfile ( srcpath + " / " + f ) ]
442
443         try :
444             os . makedirs ( dstpath )
445         except :
446             pass
447         for i , filename in enumerate ( filename_list ) :
448             result = detect_aruco ( srcpath = f" { srcpath } { filename } "
449             , interpolation = " INTER_LINEAR " , result_type = " numpy " )
450             print ( i , filename )
451             try :
452                 Image . fromarray ( result ) . save ( f" { dstpath } "
453                   INTER_LINEAR_{ filename . split ( ' .' )[ 0 ] }. png " )
454             except :
455                 continue
456
457     def localize_cdp_from_raw_photo_batch_single ( srcpath ,
458         dstpath , file_name ) :
459         RESIZING_INTERPOLATION = [ " INTER_NEAREST " , " INTER_LINEAR "
460             , " INTER_AREA " , " INTER_CUBIC " , " INTER_LANCZOS4 " ]
461         for interpolation in RESIZING_INTERPOLATION :
462             result = detect_aruco ( srcpath = f" { srcpath } " ,
463             interpolation = interpolation , result_type = " numpy " , blur = True
464             , ksize = 25 )
465             Image . fromarray ( result ) . save ( f" { dstpath } { "
466               interpolation }_{ file_name } . png " )
467
468     def separate_cdp_2level_4level ( srcpath , dstpath , level ) :
469         filename_list = [ f for f in os . listdir ( srcpath ) if os .

```

```
path.isfile(srcpath + "/" + f) and srcpath + "/" + f != f"{
srcpath }/.DS_Store" if f.split("_")[4] == "{level}LEVEL"]  
    print(filename_list)  
try:  
    os.mkdir(dstpath)  
except:  
    pass  
for filename in filename_list:  
    shutil.copy(f"{srcpath}{filename}", f"{dstpath}{filename}")
```