Analisis Penerapan *Brute Force Algorithm* untuk *descriptor matcher* pada *Fingerprint Recognition System*

Kevin Katsura Dani Sitanggang 13519216
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519216@std.stei.itb.ac.id

***Abstract*—Algoritma menjadi dasar dari pengembangan suatu sistem perangkat lunak. Algoritma menggambarkan suatu pemrosesan sehingga dihasilkan data hasil proses yang diinginkan. Beberapa algoritma terdefinisi menjadi suatu pemrosesan umum yang dapat digunakan secara berulang-ulang. Salah satu algoritma tersebut adalah *brute force. Brute force* telah menjadi suatu metode yang sangat terkenal dalam dunia algoritma. *Brute Force* secara umum menggambarkan cara pemikiran manusia secara umum dalam menyelesaikan suatu komputasi. Dengan begitu, algoritma ini banyak digunakan dalam berbagai hal seperti dalam sistem pengenalan sidik jari. Algoritma *brute force* pada sistem pengenalan sidik jari digunakan ketika pencocokan *interest point* dua sidik jari. Hal tersebut tentunya mengartikan bahwa *brute force* merupakan algoritma yang penting. Untuk mengetahui secara jelas, makalah ini akan menjelaskan peran algoritma *brute force* pada sistem pengenalan sidik jari.**

***Keywords—brute force,* algoritma, *biometrics,* sidik jari, ORB, pencocokan, *keypoint, descriptors,* FAST, BRIEF**

I. PENDAHULUAN

Keamanan sistem merupakan hal mutlak yang tidak dapat ditawar. Keamanan pada sistem merupakan salah satu faktor utama dalam menilai kualitas dari sistem. Sistem yang baik adalah sistem yang memiliki keamanan yang tinggi dan menjamin keamanan data pengguna. Menjamin keamanan dari sistem dapat meningkatkan nilai dari produk tersebut. Begitu juga sebaliknya, jika keamanan data tidak dijamin, maka data dari pengguna bisa menjadi bahan uji dan sumber kejahatan. Hal tersebut telah sering terjadi dalam dunia digital. Sistem keamanan dalam aplikasi dapat dijebol oleh *black hacker* yang tidak bertanggung jawab. Salah satu contohnya adalah 91 juta data pengguna *online shop* Tokopedia bocor yang diduga sebagai ulah dari *hacker* dengan nama samaran ShinyHunter. Hal tersebut tentunya menimbulkan dampak yang sangat buruk dan berkurangnya keyakinan pengguna terhadap sistem keamanan aplikasi tersebut. Dalam kasus Tokopedia, akibat yang ditimbulkan dapat berupa pengaksesan tidak bertanggung jawab terhadap akun pengguna dengan memanfaatkan saldo pengguna bahkan mencuri data kartu kredit pengguna. Data pengguna hasil retasan tersebut juga biasanya digunakan oleh para *hacker* untuk diperjual belikan. Hal tersebut tentunya sangat fatal dan perlu dilakukan peningkatan keamanan sistem. Kasus tersebut juga telah menunjukkan bahwa sangat pentingnya jaminan keamanan data terlebih pada saat ini, segala sesuatu saat ini dilakukan secara online baik itu pengiriman uang, pembelian, pembayaran yang meninggalkan informasi-informasi penting dalam dunia maya. Dengan begitu, keamanan merupakan hal penting yang bahkan setiap orang akan rela mengorbankan banyak materi demi jaminan keamanan tersebut.

Salah satu bentuk dari sistem keamanan digital adalah sidik jari. Sidik jari merupakan bagian dari *biometrics* yang menjadikan bagian dari fisik tubuh manusia sebagai identitas unik. Hal tersebut merupakan salah satu keamanan terbaik yang pernah ada. Sistem keamanan *biometrics* menjamin bahwa setiap orang memiliki identitas unik masing-masing yang tidak dapat dibagi-bagi atau ditawarkan kepada orang lain. Penggunaan sidik jari ini tentunya telah populer dan telah banyak teraplikasikan dalam berbagai sistem keamanan. Beberapa sektor yang memanfaatkan sidik jari sebagai keamanan antara lain sistem absensi kelas di berbagai sekolah dan kantor Indonesia, sistem keamanan pada *smartphone*, sistem keamanan pada komputer, dan sebagainya. Dengan adanya keamanan berbasis *biometrics*, kejahatan yang mengatasnamakan orang lain dapat dihindari. Dengan begitu, penggunaan sidik jari dalam mendukung keamanan sistem perlu diketahui dan kemudian dikembangkan.

Gambar 1. Sidik Jari (sumber : https://metanteibayoo.com/fingerprint-xiaomi-tidak-berfungsi-begini-cara-mengatasinya/)

Pada sistem pengenalan sidik jari, terdapat dua proses umum yaitu pembacaan sidik jari dari pengguna dan pencocokan sidik jari terhadap data sidik jari referensi. Pada proses pembacaan sidik jari, terdapat dua teknik yang sering digunakan yaitu dengan teknik *optical* dan *ultrasonic*. Teknik *optical* bekerja dengan menerapkan pencahayaan pada jari dan memanfaatkan pattern yang terlihat dari perbedaan intensitas cahaya pada objek. Berbeda halnya dengan teknik *ultrasonic* yang bekerja dengan memancarkan gelombang *ultrasonic* pada jari dan kemudian memantulkannya kembali sehingga dapat dibaca oleh sistem. Teknik *ultrasonic* ini lebih terandalkan dibandingkan *optical* karena dengan pemanfaatan citra 3d tersebut, teknik ini dapat bekerja dengan baik walaupun jari dalam keadaan basah, penggunaan *lotion,* dsb. Setelah dilakukan pembacaan sidik jari, gambar hasil interpretasi sidik jari akan dibandingkan dengan data sistem.

Pada proses pencocokan sidik jari, sidik jari yang merupakan data *testing* akan dicocokan pada data referensi yang tersimpan dalam sistem. Jika terdapat kecocokan terhadap sidik jari, maka dapat dipastikan bahwa pengguna yang bersangkutan memiliki hak akses terhadap akses yang ditawarkan. Dalam proses pencocokan, titik-titik yang menjadi titik unik sidik jari akan dibandingkan melalui algoritma-algoritma yang memungkinkan. Salah satu algoritma tersebut adalah *brute force*. Algoritma ini adalah algoritma pencocokan yang mudah dipahami dan diimplementasikan. Selain itu, algoritma ini selalu dapat memberikan solusi dalam banyak kasus pencocokan.

Mengingat bahwa proses pencocokan merupakan bagian penting dalam pemrosesan sidik jari, maka makalah ini akan menjelaskan bagaimana proses pencocokan sidik jari data *testing* dengan data *training* dilakukan dengan algoritma *brute force.*

II. LANDASAN TEORI

A. Algoritma *Brute Force*

Algoritma *Brute Force* adalah salah satu algoritma pencarian solusi persoalan yang biasa disebut sebagai *naive algorithm*. Algoritma ini merupakan algoritma pencarian solusi yang mudah dipahami dan juga mudah diterapkan. Selain itu, algoritma ini dapat memecahkan banyak permasalahan. Jika dibandingkan dengan algoritma lainnya seperti *greedy, divide and conquer, decrease and conquer,* dan sebagainya, algoritma ini merupakan algoritma dengan pemecah permasalahan terbanyak. Hal tersebut disebabkan karena cara kerja dari algoritma ini yang bekerja lurus, hanya perlu mencocokkan setiap solusi yang umum tanpa adanya faktor lainnya yang mempengaruhi. Beberapa kasus yang dapat diselesaikan dengan *brute force* pencarian elemen terbesar, pencarian beruntun, mengalikan dua buah matriks, uji bilangan prima, pencocokan *string*, pengurutan, dan sebagainya.

Algoritma *Brute Force* dikenal sebagai pendekatan yang lempang atau *straightforward* untuk memecahkan suatu persoalan. Algoritma ini memiliki beberapa keunggulan antara lain sangat sederhana, langsung, dan jelas caranya. Akan tetapi, dibalik keunggulan tersebut, algoritma *brute force* dikenal sebagai algoritma yang kurang mangkus dan tidak cerdas karena membutuhkan waktu komputasi yang besar dan waktu yang lama dalam menyelesaikan persoalan yang ada. Alasan tersebut juga tergambar dari nama nya yaitu *force* yang mengindikasikan tenaga daripada otak. Dengan begitu, penggunaan algoritma *brute force* tidak seharusnya diterapkan dalam mengatasi permasalahan dengan masukan atau data yang kecil.

Kelebihan algoritma *brute force* :

1. Dapat memecahkan hampir seluruh permasalahan komputasi yang ada (*wide applicability*)
2. Mudah dimengerti, diimplementasikan, dan sederhana.
3. Algoritma basis dari pensolusian beberapa masalah penting antara lain pencocokan *string,* pencarian, pengurutan, dan perkalian matriks.
4. Algoritma baku untuk beberapa komputasi seperti penentuan elemen maksimum dan minimum pada senarai, penjumlahan dan perkalian bilangan pada senarai, dsb.

Kekurangan algoritma *brute force* :

1. Jarang menghasilkan algoritma yang mangkus.
2. Lambat

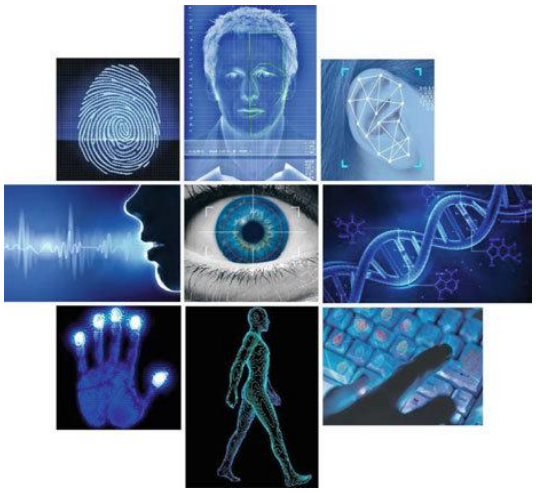
B. *Biometrics*

Biometrik adalah pengukuran atau perhitungan yang dilakukan berdasarkan fisik manusia. Istilah ini biasanya digunakan dalam keamanan sistem. Beberapa bagian dari fisik manusia memiliki keunikan jika dilakukan pengukuran. Dengan begitu, beberapa bagian dari fisik manusia dapat menjadi sistem keamanan yang baik. Hal tersebut tentu sangat menguntungkan dan menjadi penemuan yang baik.

Biometrik menjadi keamanan yang baik dibandingkan penggunaan *password.* Penggunaan biometrik memungkinkan setiap orang tidak akan memiliki akses terhadap keamanan orang lain karena identitas unik dari masing-masing orang tidak dapat dibagi dengan orang lain. Beda hal nya dengan penggunaan password yang memungkinkan orang lain memiliki hak akses asal memiliki sandinya.

Terdapat enam *biometrics* yang dapat digunakan pada manusia antara lain :

1. Sidik jari.
2. Foto dan video (pengenalan wajah dan *retina*).
3. Pengenalan fisiologis (wajah, iris, retina, urat telapak tangan, telinga)
4. Suara
5. Tanda tangan
6. DNA



Gambar 2. *Biometrics* (sumber : https://www.researchgate.net/figure/Set-of-popular-physiological-and-behavioral-IoT-biometrics_fig2_328516514)

Biometrics juga digunakan dalam membedakan manusia dengan robot. Beberapa biometrik tersebut yang umum digunakan antara lain :

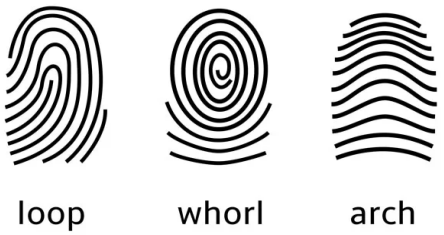
1. Pola mengetik
2. Pergerakan fisik
3. Pola navigasi (penggunaan mouse, *trackpad, touch screen*)
4. Pola sifat umum

C. *Fingerprints*

Fingerprints atau sidik jari merupakan bagian fisik dari tubuh manusia yang dijadikan sebagai *biometrics.* Sidik jari merupakan sistem keamanan yang populer digunakan saat ini. Sidik jari merupakan salah satu yang terbaik untuk dijadikan sebagai *biometrics.* Hal tersebut disebabkan karena tidak pernah adanya perubahan pada sidik jari. Berbeda halnya dengan penggunaan suara atau wajah sebagai biometrik. Suara dapat berubah ketika kondisi kesehatan tidak dalam keadaan baik. Dengan begitu, penggunaan *fingerprint* sebagai pola keamanan sistem sangatlah menguntungkan.

Terdapat tiga pola umum yang terdapat pada sidik jari manusia. Pola tersebut digunakan sebagai analisis dasar *biometrics* sidik jari. Pola tersebut dikelompokkan menjadi tiga :

1. *Loops*
   Pola mengalir yang kemudian membentuk lingkaran dan kemudian kembali ke asal aliran. Pola ini merupakan pola yang paling sering ditemukan yaitu sekitar 60% dari mayoritas sidik jari.
2. *Whorls*
   Pola yang menyerupai pusaran air. Pola ini membentuk sekitar 35% dari sidik jari.
3. *Arches*
   Pola melengkung menyerupai gelombang. Lengkungan yang dimaksud memiliki perbedaan dari lengkungan biasa yaitu titik puncak yang lebih tinggi dan tajam. Pola ini membentuk hanya sekitar 5% dari sidik jari.



Gambar 3. Pola pengelompokkan sidik jari (sumber : https://www.azolifesciences.com/article/Fingerprint-Analysis-in-Forensic-Science.aspx#:~:text=Analyzing%20fingerprints%20left%20at%20the,%2C%20parole%2C%20and%20other%20details.)

D. *Digital Image processing*

Pemrosesan gambar digital merupakan pengolahan gambar yang dilakukan oleh komputer digital. Pemrosesan gambar digital merupakan salah satu keilmuan yang sangat populer saat ini. Hal tersebut dilakukan untuk mengekstrak suatu informasi penting dari gambar digital. Pemrosesan ini dilakukan dengan memanfaatkan algoritma-algoritma seperti SIFT (*Scale Invariant Feature Transform*), SURF (*Speeded-Up Robust Features*), CenSurE (*Center Surround Extremas detector*), MSER (*Maximally Stable Extremal Regions*), *Harris corner detector, Shi and Tomasi corner detector,* FAST (*Features from accelerated segment test*), *dense interest point,* BRIEF (*Binary Robust Independent Elementary Features*), ORB (*Oriented FAST and Rotated BRIEF*), BRISK (*Binary Robust Invariant Scalable Keypoints*), dan FREAK (*Fast Retina Keypoints Descriptor*).

E. Algoritma ORB

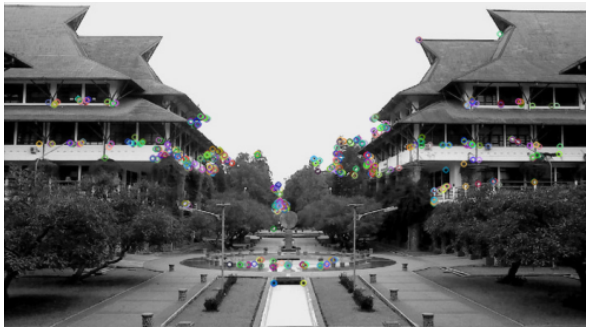
Algoritma ORB (*Oriented FAST and Rotated BRIEF)* merupakan algoritma hasil penggabungan antara *keypoints* dari FAST dan *descriptor* dari BRIEF dengan beberapa modifikasi di dalamnya untuk meningkatkan performansi algoritma. Algoritma ini adalah algoritma pemrosesan gambar digital. Pada proses penentuan *keypoints* pada algoritma ini, semua *keypoints* yang memungkinkan akan dihasilkan dengan menggunakan algoritma FAST. Setelah itu, untuk menentukan N *keypoints* teratas, algoritma ORB ini akan memanfaatkan *Harris corner measurement*. Selanjutnya, untuk mendapatkan *descriptor* untuk setiap *keypoints*, algoritma ini memanfaatkan *descriptor* BRIEF.

Dalam melakukan pemrosesan, terdapat dua informasi penting yang harus diambil pada gambar :

1. *Keypoints*
   Keypoints merupakan suatu titik penting (*interest point*) yang dapat menunjukkan suatu identitas yang unik pada gambar. *Keypoints* akan menunjukkan posisi yang sama walaupun gambar mengalami *affine transformation.*

```
<KeyPoint 0x10c6b3db0>
<KeyPoint 0x10c6b3d80>
<KeyPoint 0x10c6b3d50>
<KeyPoint 0x10c6b3d20>
<KeyPoint 0x10c6b3cf0>
```

Gambar 4. Contoh Keypoints (sumber : dokumen pribadi)



Gambar 5. *Keypoints* pada gambar kampus ITB (sumber : dokumen pribadi)

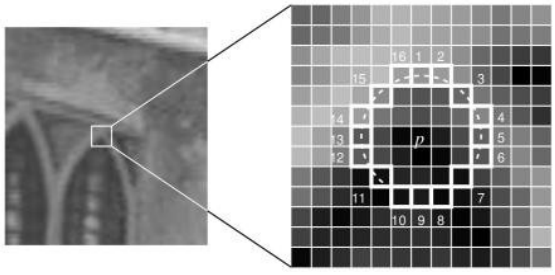
2. *Descriptor*

   Descriptor merupakan suatu nilai yang dikandung oleh setiap *keypoints.* Nilai ini menjadi pembeda antara titik yang ada. Nilai yang dikandung oleh descriptor berupa *array of number* yang berisi suatu nilai dari setiap aspek penilaian dari setiap titik.

Algoritma ORB dibentuk berdasarkan dua algoritma lainnya yaitu FAST dan BRIEF.

1. FAST (*Features from accelerated segment test*) *keypoints detector*

   Penentuan *keypoints* pada algoritma FAST dimulai dengan penentuan *interest point* yang akan diuji. *Interest point* yang diuji dapat dilakukan dengan iterasi setiap pixel yang ada pada gambar. Setelah itu, akan digambarkan *Bresenham circle* (lingkaran dengan radius 3 dari titik pusat *interest point*).



Gambar 6. Penerapan *Bresenham circle* pada suatu *interest point*. (sumber : https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf)

Setelah itu, untuk menetapkan apakah titik yang diuji dapat digunakan sebagai *keypoint,* dilakukan pengecekan terhadap setiap pixel (terdapat 16 pixel) yang dilalui oleh lingkaran yang terdefinisi. Pixel yang dimaksud adalah pixel yang ditandai dengan border putih pada gambar 6. Kemudian, cek apakah terdapat 8 atau lebih pixel yang berwarna lebih gelap atau lebih terang dari pixel pusat. Jika ada, maka *interest point* yang diuji dapat digunakan sebagai *keypoint.*

2. BRIEF (*Binary Robust Independent Elementary Features) descriptor.*

   Penentuan *descriptor* dari suatu *keypoint* pada algoritma BRIEF diawali dengan mengubah nilai dari *keypoint* menjadi bilangan biner. Setelah itu, Brief akan melakukan menghaluskan gambar dengan menerapkan prinsip Gaussian kernel untuk menghindari *descriptor* dari gangguan frekuensi tinggi. Kemudian, akan dicari dua titik random dari patch yang sama dengan patch yang mengandung *keypoint* terkait. Titik satu memiliki nilai ketersebaran sigma dari *keypoint* dan titik dua memiliki nilai ketersebaran sigma dari titik satu. Jika ternyata terbukti bahwa titik satu lebih terang dibandingkan titik dua, maka menghasilkan bit 1 dan jika tidak maka 0.

III. PEMBAHASAN DAN ANALISIS

Proses pencocokan sidik jari diawali dengan menentukan sidik jari yang akan diuji. Sidik jari ini akan diuji berdasarkan data sidik jari yang telah ada. Sidik jari yang diuji akan dicocokan pada setiap data sidik jari yang ada pada sistem. Pencocokan dilakukan sampai ditemukan data sidik jari yang sama dengan sidik jari uji. Pencocokannya dilakukan dengan membandingkan *descriptor* value dari setiap titik.

gambar 7. Sidik jari uji (sumber : https://www.codespeedy.com/fingerprint-detection-in-python/)

gambar 8. Sidik jari referensi. (sumber: https://www.codespeedy.com/fingerprint-detection-in-python/)

A. Penentuan *Keypoints* pada data *testing fingerprint*

Penentuan *keypoints* dilakukan dengan menggunakan algoritma ORB. Setiap data uji dan data referensi akan ditetapkan *keypoints* nya. Pada kasus ini, pengujian hanya dilakukan pada 500 *keypoints* untuk masing-masing sidik jari.

gambar 9. Sidik jari uji dengan *keypoints* (sumber: https://www.codespeedy.com/fingerprint-detection-in-python/ dengan modifikasi)

gambar 10. Sidik jari referensi dengan *keypoints* (sumber : https://www.codespeedy.com/fingerprint-detection-in-python/ dengan modifikasi)

B. Pengambilan *descriptor* pada setiap sidik jari.

Untuk setiap *keypoint* yang terdapat pada setiap *fingerprint*, ambil *descriptor* sebagai identitas yang akan dibandingkan pada setiap sidik jari. Pengambilan *descriptor* pada dasarnya sekaligus dengan pengambilan *keypoints*. Adapun kode program nya adalah sebagai berikut

```
# MEMBACA SIDIK JARI UJI
gambar1 = cv2.imread("test.tif")
# AKSES SELURUH PATH UNTUK SELURUH SIDIK JARI
REFERENSI
files = glob(os.getcwd()+"/images/*")

# UBAH GAMBAR SIDIK JARI UJI KE GRAYSCALE
img1 =
cv2.cvtColor(gambar1,cv2.COLOR_BGR2GRAY)
# INISIALISASI ORB UNTUK PENDETEKSIAN
KEYPOINTS DAN DESCRIPTORS SIDIK JARI UJI
orbUji = cv2.ORB_create(10)
# MEMBACA KEYPOINTS DAN DESCRIPTORS PADA
SIDIK JARI UJI
keypoints1,descriptors1 =
orbUji.detectAndCompute(img1,None)
gmbr1WKey =
cv2.drawKeypoints(img1,keypoints1,None,flag
s=None)

found  = False
# ITERASI UNTUK SETIAP SIDIK JARI REFERENSI
YANG TERSEDIA
for afile in files:
    # MEMBACA SIDIK JARI REFERENSI
    gambar2 = cv2.imread(afile)
    # UBAH GAMBAR SIDIK JARI REFERENSI KE
GRAYSCALE
    img2 =
cv2.cvtColor(gambar2,cv2.COLOR_BGR2GRAY)
    # INISIALISASI ORB
    orbRef = cv2.ORB_create(10)
    # MEMBACA KEYPOINTS DAN DESCRIPTORS PADA
SIDIK JARI REFERENSI
    keypoints2,descriptors2 =
orbRef.detectAndCompute(img2,None)
```

Pada kode program di atas, *keypoints* dan *descriptors* pada masing-masing sidik jari akan ditampung pada variabel dengan suffix *keypoints* dan *descriptors*. *Keypoints1* dan *descriptors1* untuk menampung hasil analisis pada sidik jari uji sedangkan *keypoints2* dan *descriptors2* untuk menampung hasil analisis pada sidik jari referensi. Dengan 10 *keypoints* yang didefinisikan dengan algoritma ORB, nilai dari *descriptors* untuk masing-masing sidik jari adalah sebagai berikut.

1. Data Uji

Keypoint	*Descriptor*
0x10d991090	[74 175 147 196 253 50 168 43 136 166 13 20 141 162 224 230 195 39 202 40 48 11 186 206 173 215 223 45 146 226 47 108]
0x102398bd0	[243 225 216 166 126 104 82 140 19 105 159 123 118 122 202 114 210 206 52 1 15 18 201 147 246 37 185 191 188 90 88 11]
0x102398f00	[44 188 218 193 211 14 62 83 162 191 173 151 24 135 98 210 37 157 131 174 82 151 152 80 127 165 37 101 139 162 171 146]
0x102398e10	[53 89 221 151 86 143 163 96 27 229 115 142 118 68 14 136 190 237 128 103 1 240 60 45 181 166 64 112 109 27 215 240]
0x102398ed0	[190 240 102 148 143 241 183 110 153 199 157 187 111 209 110 132 59 194 245 71 122 71 142 167 242 95 85 251 47 221 255 231]
0x114b86870	[225 111 122 177 195 172 109 23 51 46 114 147 3 135 70 93 237 159 164 4 62 112 133 147 39 252 145 39 33 216 228 100]

0x114b868a0	[140 191 241 221 27 231 3 147 58 29 229 50 138 103 164 117 39 122 140 138 34 12 186 255 107 246 33 62 18 240 133 162]
0x114b86960	[2 255 174 159 246 211 173 231 142 252 94 167 108 172 183 4 236 111163 197 208 218 45 253 60 22 222 135 140 126 189 253]
0x114b86840	[221 189 56 190 255 133 23 186 23 138 90 111 255 119 9 85 87 251 29 71 123 78 91 191 248 195 175 170 60 91 49 59]
0x114b86930	[14 222 215 94 251 149 190 249 159 136 125 96 232 247 164 244 223 115 174 202 195 139 106 253 46 255 239 202 19 183 233 173]

2. Data referensi 1

Keypoint	*Descriptor*
0x1192eb060	[145 189 152 201 143 55 132 223 87 184 156 70 142 17 227 173 108 74 6 248 201 122 83 144 171 201 199 62 59 120 185 173]
0x10fcf2bd0	[153 249 216 193 131 21 84 223 23 185 148 66 26 151 97 172 125 152 150 216 75 62 121 144 187 201 21 62 59 152 168 189]
0x10fcf2f00	[179 185 216 215 134 23 84 55 86 166 149 22 178 50 102 33 76 170 30 202 252 216 117 248 239 136 214 50 60 72 152 15]
0x10fcf2e10	[115 181 47 164 213 95 116 97 193 125 151 31 74 93 58 123 185 132 93 45 95 179 184 205 115 63 129 183 43 10 220 214]
0x10fcf2ed0	[23 228 250 218 22 134 141 100 223 162 92 134 81 200 246 101 127 124 31 59 192 231 140 136 120 28 143 248 61 186 241 52]
0x1224de840	[74 83 23 181 32 233 111 19 173 141 95 236 236 231 169 223 32 54 174 112 1 209 2 172 140 211 238 129 138 223 149 49]
0x1224de870	[19 229 78 192 86 147 138 244 196 128 222 195 81 205 62 68 126 236 19 238 224 179 28 140 114 15 143 13 39 155 121 8]
0x1224de930	[50 159 74 189 174 64 148 111 17 70 244 167 37 174 38 67 222 127 165 38 124 138 194 139 98 247 254 155 45 108 252 219]
0x1224de810	[169 212 1 187 124 129 71 189 22 88 208 118 245 81 24 225 92 74 23 237 235 14 94 191 236 4 174 248 54 152 176 233]
0x1224de900	[43 72 167 159 44 132 235 2 132 72 218 71 176 173 181 66 200 100 63 207 137 142 67 171 68 62 250 136 13 125 112 120]

3. Data referensi 2

Keypoint	*Descriptor*
0x116d72060	[120 26 221 173 110 98 18 154 69 66 120 162 119 55 171 250 114 86 174 34 4 129 103 41 70 167 112 3 182 249 36 203]
0x105936bd0	[102 206 115 128 199 77 103 227 79 118 75 91 14 181 100 212 119 156 162 2 208 54 172 210 53 54 69 110 51 254 212 101]
0x105936f00	[201 197 6 178 21 157 108 159 221 89 31 18 238 97 58 229 176 130 174 62 111 225 31 252 250 210 5 1 170 220 153 28]
0x105936e10	[179 163 251 212 36 93 91 106 95 111 182 27 34 238 254 32 156 202 14 78 222 233 35 227 91 30 177 55 16 233 196 70]
0x105936ed0	[9 131 90 227 247 157 186 252 142 48 55 199 253 238 238 224 54 50 138 102 213 9 14 189 110 201 35 233 62 143 89 153]
0x118122840	[172 3 144 167 237 77 129 44 73 202 161 25 250 216 109 126 220 57 50 19 88 238 202 128 77 148 95 120 80 14 69 138]
0x118122870	[156 173 91 176 215 113 54 246 85 66 90 243 13 189 99 129 123 75 135 49 55 248 4 175 106 193

	181 199 167 221 188 219]
0x118122930	[246 20 106 164 22 230 88 64 21 68 195 155 71 15 59 89 118 149 189 247 45 215 85 142 147 227 202 209 45 141 60 82]
0x118122810	[40 91 114 232 239 211 181 115 205 140 124 163 28 157 167 33 94 56 151 96 208 170 102 156 118 73 101 175 127 172 252 179]
0x118122900	[184 37 113 225 247 249 80 27 87 184 124 235 15 157 193 149 58 186 133 16 94 178 194 208 254 217 33 239 119 204 206 131]

4. Data referensi 3

Keypoint	*Descriptor*
0x11f043060	[74 175 147 196 253 50 168 43 136 166 13 20 141 162 224 230 195 39 202 40 48 11 186 206 173 215 223 45 146 226 47 108]
0x10dc07bd0	[243 225 216 166 126 104 82 140 19 105 159 123 118 122 202 114 210 206 52 1 15 18 201 147 246 37 185 191 188 90 88 11]
0x10dc07f00	[44 188 218 193 211 14 62 83 162 191 173 151 24 135 98 210 37 157 131 174 82 151 152 80 127 165 37 101 139 162 171 146]
0x10dc07e10	[53 89 221 151 86 143 163 96 27 229 115 142 118 68 14 136 190 237 128 103 1 240 60 45 181 166 64 112 109 27 215 240]
0x10dc07ed0	[190 240 102 148 143 241 183 110 153 199 157 187 111 209 110 132 59 194 245 71 122 71 142 167 242 95 85 251 47 221 255 231]
0x1203f3840	[225 111 122 177 195 172 109 23 51 46 114 147 3 135 70 93 237 159 164 4 62 112 133 147 39 252 145 39 33 216 228 100]
0x1203f3870	[140 191 241 221 27 231 3 147 58 29 229 50 138 103 164 117 39 122 140 138 34 12 186 255 107 246 33 62 18 240 133 162]
0x1203f3930	[2 255 174 159 246 211 173 231 142 252 94 167 108 172 183 4 236 111 163 197 208 218 45 253 60 22 222 135 140 126 189 253]
0x1203f3810	[221 189 56 190 255 133 23 186 23 138 90 111 255 119 9 85 87 251 29 71 123 78 91 191 248 195 175 170 60 91 49 59]
0x1203f3900	[14 222 215 94 251 149 190 249 159 136 125 96 232 247 164 244 223 115 174 202 195 139 106 253 46 255 239 202 19 183 233 173]

5. Data referensi 4

Keypoint	*Descriptor*
0x10ec54060	[109 247 131 160 252 6 167 214 160 189 154 101 20 162 232 210 201 204 2 177 57 251 136 0 60 165 175 108 136 122 250 145]
0x1018e3bd0	[105 17 166 172 93 86 81 60 90 237 181 79 125 23 34 117 80 164 236 128 165 203 181 221 219 103 251 221 2 5 39 122]
0x1018e3f00	[93 46 183 164 52 10 171 88 96 143 75 74 119 195 25 230 48 84 53 107 194 213 163 169 240 230 171 136 10 251 85 241]
0x1018e3e10	[9 9 177 164 60 115 80 119 84 225 140 143 231 36 178 36 154 102 74 76 21 235 53 172 218 223 83 49 26 173 197 220]
0x1018e3ed0	[145 33 29 170 219 141 162 204 142 78 112 207 111 196 43 204 54 67 130 117 119 61 27 159 60 215 20 30 217 10 9 93]
0x1140cf840	[169 237 214 184 115 158 165 119 16 205 116 212 189 199 194 181 78 42 130 124 99 254 102 142 108 193 38 201 98 156 171 233]
0x1140cf870	[241 43 111 196 84 174 75 240 193 160 78 206 115 65 156 38 120 230 32 110 153 240 180 37 115 42 192 80 39 155 216 48]
0x1140cf930	[234 154 160 166 77 80 74 32

	130 198 131 143 207 198 90 246 193 212 185 199 205 70 246 162 159 245 139 123 38 46 52 116]
0x1140cf810	[105 143 163 172 60 7 207 250 66 236 216 234 243 230 176 224 94 119 60 239 201 43 165 63 224 172 175 184 20 250 85 225]
0x1140cf900	[8 190 61 211 125 253 123 186 30 126 30 236 252 247 227 164 23 90 191 26 163 92 43 255 111 223 37 224 22 221 5 123]

C. Perbandingan *deskriptor* sidik jari uji dengan referensi

Perbandingan *deskriptor* sidik jari uji dengan referensi dilakukan dengan metode *brute force.* Metode ini bekerja dengan cara membandingkan setiap *descriptor* dari sidik jari uji dengan referensi satu per satu sampai ditemukan sidik jari referensi yang mengandung *descriptor* yang sama dengan *descriptor* sidik jari uji. Untuk meninjau lebih lanjut, dapat melihat kode program di bawah ini.

```
def BruteForce(listTrain,listTest):
    for i in listTrain:
        similar = False
        for j in listTest:
            comparison = ( i == j)
            if(comparison.all()):
                similar = True
                break
        if (not similar):
            return False
    return True
```

Pada kode program di atas, algoritma *brute force* akan bekerja dengan menerima listTrain yang merupakan kumpulan *descriptor* dari data uji dan listTest yang merupakan kumpulan *descriptor* dari data referensi. Kedua *descriptor* akan dibandingkan dengan membandingkan setiap elemen yang ada pada *descriptor* (setiap *descriptor* merupakan array of number yang berukuran 32 angka). Jika semua *descriptor* sidik jari uji terdapat pada senarai *descriptor* dari sidik jari referensi, maka algoritma *brute force* akan mengembalikan nilai *true* dan jika tidak, *false.*

Pada pengujian data sidik jari sebelumnya, proses perbandingan *brute force* antara sidik jari uji dengan sidik jari referensi mengembalikan nilai *true* yang berarti sidik jari referensi yang bersesuaian ditemukan. Sidik jari uji dinyatakan sesuai dengan sidik jari referensi 3. Untuk lebih jelas, berikut tabel persamaan antara *keypoints* kedua data dari hasil perbandingan *descriptors* nya.

Persamaan sidik jari uji dengan sidik jari referensi 3	
Keypoint **data uji**	***Keypoint*** **data referensi**
0x10d991090	0x11f043060
0x102398bd0	0x10dc07bd0
0x102398f00	0x10dc07f00
0x102398e10	0x10dc07e10
0x102398ed0	0x10dc07ed0
0x114b86870	0x1203f3840
0x114b868a0	0x1203f3870
0x114b86960	0x1203f3930
0x114b86840	0x1203f3810
0x114b86930	0x1203f3900

D. Ilustrasi persamaan sidik jari uji dengan referensi

Setiap *descriptor* dari sidik jari uji dinyatakan sama dengan sidik jari referensi. Persamaan *descriptors* dari kedua sidik jari dapat digambarkan dengan keterhubungan kedua gambar sidik jari melalui *keypoints* nya.

gambar 11. Ilustrasi keterhubungan *keypoints* kedua sidik jari dari hasil perbandingan *descriptors*
(sumber: dokumen pribadi)

Dari gambar tersebut dapat dilihat bahwa sidik jari uji merupakan transformasi rotasi sebesar 180 derajat dari sidik jari referensi yang ditemukan. Hal tersebut membuktikan bahwa walaupun antara sidik jari uji dengan sidik jari referensi terdapat perbedaan berdasarkan *affine transformation*,

kesamaan dari kedua sidik jari masih dapat ditemukan.

IV. KESIMPULAN

Algoritma *Brute Force* dapat digunakan dalam pencocokan *descriptor* pada *fingerprint recognition system*. Kemudahan dan kesederhanaan yang ditawarkan oleh algoritma *brute force* membuat algoritma ini menjadi algoritma yang umum digunakan untuk pencocokan *descriptor* dalam *digital image processing*. Hal tersebut berarti bahwa algoritma *brute force* juga dapat digunakan dalam pencocokan *descriptor* diluar kasus pencocokan sidik jari. Penggunaan *brute force* pada pencocokan *descriptor* secara umum disebabkan oleh sistem pencocokan *descriptor* bekerja harus dengan membandingkan setiap *descriptor* nya (tidak ada fungsi heuristik maupun bounding).

V. UCAPAN TERIMA KASIH

Penulis mengucapkan puji dan syukur kepada rahmat dari Tuhan Yang Maha Esa karena atas berkat-Nya, makalah yang berjudul "Analisis Penerapan *Brute Force Algorithm* untuk *descriptor matcher* pada *Fingerprint Recognition System*" dapat diselesaikan dengan maksimal. Penulis juga mengucapkan rasa terima kasih yang besar untuk setiap dosen mata kuliah IF2211 Strategi Algoritma khususnya Dr. Ir. Rinaldi Munir, MT yang telah memberikan ilmu nya selama saya mengikuti mata kuliah ini di K04 terlebih algoritma *brute force* yang menjadi topik pembahasan utama pada makalah ini. Penulis juga mengucapkan terima kasih kepada keluarga, teman-teman, dan orang-orang yang secara tidak langsung mendukung penulis dalam menyelesaikan makalah ini. Tidak lupa juga penulis mengucapkan terima kasih kepada setiap pihak bersangkutan yang memiliki hak atas referensi yang penulis gunakan. Akhir kata, penulis mengucapkan terima kasih kepada seluruh pihak yang telah mendukung.

LINK VIDEO YOUTUBE

https://youtu.be/JnYC7B3J-aQ

REFERENSI

[1] Munir, Rinaldi, *Algoritma Greedy* (2021). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Brute-Force-(2021)-Bag1.pdf. Diakses tanggal 10 Mei 2021, pukul 15.25 GMT+7

[2] Digital Image Processing Basics, https://www.geeksforgeeks.org/digital-image-processing-basics/ . Diakses tanggal 10 Mei 2021.

[3] Biometrics and biometric data: what is it and is it secure?, https://us.norton.com/internetsecurity-iot-biometrics-how-do-they-work-are-they-safe.html . Diakses tanggal 10 Mei 2021.

[4] What is biometrics? 10 physical and behavioral identifiers that can be used for authentication, https://www.csoonline.com/article/3339565/what-is-biometrics-and-why-collecting-biometric-data-is-risky.html. Diakses tanggal 10 Mei 2021.

[5] Fingerprint analysis in forensic science, https://www.azolifesciences.com/article/Fingerprint-Analysis-in-Forensic-Science.aspx#:~:text=Analyzing%20fingerprints%20left%20at%20the,%2C%20parole%2C%20and%20other%20details . Diakses tanggal 10 Mei 2021.

[6] Introduction to ORB (Oriented FAST and Rotated BRIEF), https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf . Diakses tanggal 10 Mei 2021.

[7] Introduction to FAST (Features from Accelerated Segment Test), https://medium.com/data-breach/introduction-to-fast-features-from-accelerated-segment-test-4ed33dde6d65 . Diakses tanggal 10 Mei 2021.

[8] Introduction to BRIEF (Binary Robust Independent Elementary Features), https://medium.com/data-breach/introduction-to-brief-binary-robust-independent-elementary-features-436f4a31a0e6 . Diakses tanggal 10 Mei 2021.

[9] ORB (Oriented FAST and Rotated BRIEF), https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html . Diakses tanggal 10 Mei 2021.

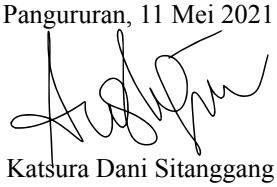
[10] Hagel, Johan dan Alexander Karlsson, 2016, *Fingerprint matching - hard cases,* https://core.ac.uk/download/pdf/289960085.pdf . Diakses tanggal 10 Mei 2021.

[11] Fingerprint detection in python, https://www.codespeedy.com/fingerprint-detection-in-python/ . Diakses tanggal 10 Mei 2021.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Pangururan, 11 Mei 2021



Kevin Katsura Dani Sitanggang
13519216