PENGENALAN WARNA BERBASIS ANDROID MENGGUNAKAN METODE *TEMPLATE MATCHING*

Indra Hading Kurniawan[1], Nurcahyani Dewi Retnowati[2]
Departemen Informatika
Sekolah Tinggi Teknologi Adisutjipto Yogyakarta
Jl. Janti, Blok-R, Lanud Adisucipto Yogyakarta
Informatika@stta.ac.id

ABSTRACT

Template matching method is a simple and widely used method to recognize patterns. The weakness of this algorithm is the limited model that will be used as a template as a comparison in the database such as shape, size, and orientation. The Extraction Feature algorithm addresses the problem of template models such as the shape, size, and orientation that exist in the matching template algorithm by mapping the characteristics of the image object to be recognized. Optical character recognition is used to translate characters into digital images into text formats. Its simple implementation makes the template matching method widely used. In this final project discusses the introduction of color in an image to be detected color, this color recognition is not fully successful because of the influence of lightness. The workings of this application take picture is by taking a picture and then the application identifies the color of any existing and will issue results in the form of text percent, with a success rate of 15% and 85% failure when detecting a color.

Keywords*: Template Matching, Smartphone, Color, Picture*

1. Pendahuluan

Perkembangan teknologi pada era saat ini memang tidak dapat dibendung lagi. Banyak aspek kehidupan yang mulai bergeser ke era teknologi informasi. Tidak dapat dipungkiri lagi kebutuhan akan kemudahan dan akses yang tak terbatas mulai dirasakan oleh kalangan masyarakat diseluruh dunia salah satu ilmu yang mendalami teknik pengolahan citra merupakan salah satu pencapaian besar umat manusia saat ini. Hal ini dibuktikan dengan mulai banyaknya *device* maupun *software* yang dapat mengenali banyak hal hanya dengan menggunakan sebuah gambar atau foto.

Pada penelitian ini proses pengenalan warna menggunakan teknologi yang sama pada penelitian sebelumnya tetapi metode yang digunakan berbeda yaitu metode yang digunakan *template matching* metode ini juga memiliki algoritma yang sama serta digunakan untuk mendeteksi bentuk sebuah objek meskipun hanya digunakan untuk itu pada tugas akhir ini metode tersebut dicoba untuk mendeteksi warna dari sebuah objek, metode *template matching* merupakan metode yang sederhana dan banyak digunakan untuk mengenali pola. Metode ini bekerja dengan cara mengevaluasi pola citra yang akan dibandingkan dengan pola citra *template* pada basis data. kelemahan metode ini adalah terbatasnya model yang akan dijadikan *template* sebagai pembanding pada basis data seperti bentuk, ukuran, dan orientasi. Oleh karena itu peneliti membangun media untuk pengenalan warna berbasis android dengan metode *template matching.*

2. Tinjauan Pustaka

Pemanfaatan *Image To Speech Berbasis Android* untuk Pengenalan Warna Bagi Anak Bawah Tiga Tahun (BATITA), peningkatan ini juga dipicu karena perkembangan perangkat lunak pada perangkat *smartphone*. Salah satu teknologi informasi yang banyak diguanakan dalam

perangkat lunak komputer dan *smartphone* yaitu konversi bahasa tulisan menjadi bahasa percakapan dikenal sebagai *text to speech* [1].

3. Landasan Teori

3.1. *Template Matching*

Sebuah teknik dalam pengolahan citra digital untuk menemukan bagian-bagian kecil dari gambar yang cocok dengan template gambar. *Template matching* merupakan salah satu ide yang digunakan untuk menjelaskan bagaimana otak kita mengenali kembali bentuk-bentuk atau pola-pola. *Template* dalam konteks rekognisi pola menunjuk pada konstruk internal yang jika cocok (*match*) dengan stimulus penginderaan mengantar pada rekognisi suatu objek dengan *template* pada basis data. *Template* ditempatkan pada pusat bagian citra yang akan dibandingkan dan dihitung seberapa banyak titik yang paling sesuai dengan *template.*

Tingkat kesesuaian antara citra masukan dan citra *template* bisa dihitung berdasarkan nilai *error* terkecil dengan menggunakan persamaan

$$\min e = \sum_{(xy)ew}(Ixy - Txy)^2 \quad \text{.....(1)}$$

I adalah pola pixel citra masukan yang akan dibandingkan. T adalah pola pixel citra *template*. *Template* dengan nilai error paling kecil adalah *template* yang paling sesuai dengan citra masukan yang akan dibandingkan.

3.2. Pengenalan dengan Algoritma *Template Matching*

Pengenalan pola dengan menggunakan metode *template matching* dilakukan dengan cara membandingkan citra masukkan dengan citra *template* , citra masukkan dihitung berdasarkan gambar yang sesuai dengan citra *template. Pixel* citra biner ditelusuri mulai dari kiri atas hingga ke kanan bawah, citra biner dengan *pixel* berwarna hitam akan direpresentasikan dengan nilai 1. Sedangkan *pixel* citra yang berwarna putih akan direpresentasikan dengan nilai 0. gambar yang mengilustrasikan angka 1 dan 0 yang mewakili nilai pixel citra [2].

3.3. Proses Pencocokan Pola (*Template Matching*)

Secara umum teknik pencocokan pola bertujuan untuk mengkalsifikasi dan mendeskripsi pola atau melalui pengukuran sifat-sifat atau ciri-ciri objek yang bersangkutan. Metode ini mendeteksi kehadiran suatu objek dengan mencocokkan citra yang untuk diketahui dengan sekumpulan citra lain yang menjadi acuan (*template*) metode dasarnya menggunakan logika ExOR; yaitu mengidentifikasi antara persamaan perbedaan suatu objek, misalnya, citra masukan tulang punggul sehat dibandingkan dengan citra *template* tulang panggul sehat maka tidak akan dihasilkan sebuah nilai yang menunjukkan tidak diketemukannya penyakit *osteoporosis*. Lain halnya jika citra masukkan tulang panggul sehat dibandingkan dengan citra *template* tulang punggul sakit maka akan dihasilkan sebuah nilai yang menunjukkan adanya penyakit *osteoporosis* [3].

3.4. OpenCV

OpenCV adalah sebuah library yang berisi fungsi-fungsi pemograman untuk teknologi computer vision secara real time. OpenCV sudah menggunakan antarmuka bahasa C++ dan seluruh pengembangannya terdapat dalam format bahasa C++. Contoh aplikasi dari OpenCV yaitu interaksi manusia dan computer, identifikasi, segmentasi dan pengenalan objek, pengenalan wajah, pengenalan gerakan dan penelusuran gerakan.

4. Perancangan Sistem

4.1. Kebutuhan Perangkat Keras

Satu unit laptop dengan spesifikasi :

1. *Prosesor core i3*

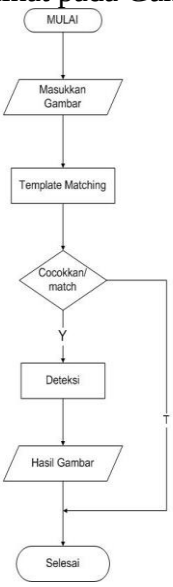
2. *RAM* 2 *GB*
3. *Harddisk* 500 *GB*

4.2. Kebutuhan Perangkat Lunak

1. Sistem Operasi Windows 10: Sistem Operasi digunakan untuk menjalankan main aplikasi, sehingga tanpa system operasi aplikasi tidak dapat dijalankan dengan sempurna
2. *Android development tools, Eclipse, Opencv*.
3. Objek: Citra digital bersumber dari *smartphone* android dengan format jpg dengan ukuran 400 x 300 piksel.

4.3. Alur Proses

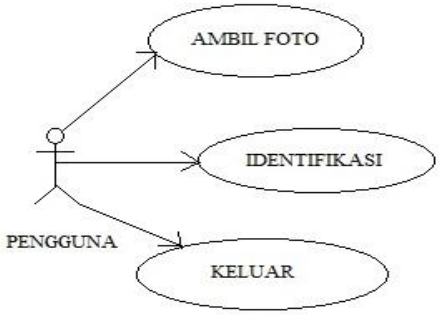
Alur proses *template matching* dapat dilihat pada Gambar 1.



Gambar 1. Alur Proses *Template Matching*

4.4. *Use Case Diagram*

Use case diagram aplikasi dapat dilihat pada Gambar 2. Pengguna tidak dibatasi aksesnya terhadap aplikasi ini. Pengguna dapat menggunakan menu pengambilan citra baik dengan kamera maupun dari galeri. Kemudian pengguna dapat memproses citra tersebut dan aplikasi akan menampilkan hasil dari pengidentifikasian warna.



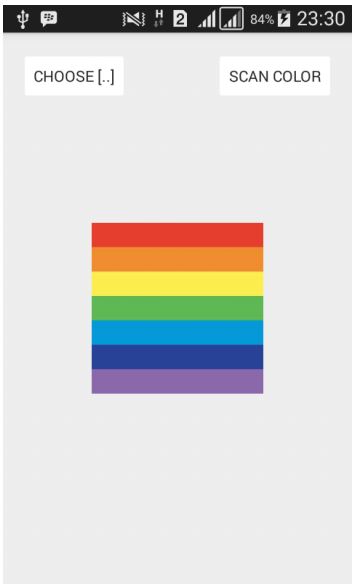
Gambar 2. *Use Case* Aplikasi

5. Implementasi dan Pembahasan

5.1. Akusisi Citra

Citra yang digunakan dalam pendeteksian warna merupakan citra yang bersumber dari *smartphone* android, baik foto secara langsung maupun dengan mengambil dari galeri. Sebelum

citra diproses pada tahap selanjutnya terlebih dahulu citra diubah ukurannya menjadi 400 x 300 piksel guna mempercepat proses pendeteksian warna, lihat pada gambar 3 Sedangkan citra yang digunakan sebagai *template* warna merupakan citra dengan ukuran 30 x 30 piksel dan masing-masing citra mewakili 9 warna yaitu merah, kuning, hijau, biru, ungu, abu-abu, jingga, putih dan hitam (Gambar 4).

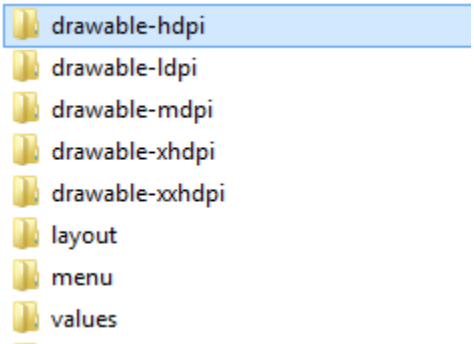


Gambar 3. Akuisisi Citra

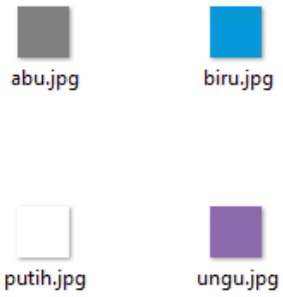
Gambar 4. *Template* 9 Warna

5.2. Implementasi *Template Matching* Pada *Smartphone Android*

Untuk mengimplementasikan metode *template matching* pada *smartphone* android terlebih dahulu memilih *template* dari warna dan dimasukkan kedalam *resource* yaitu pada folder *res/drawable-hdpi* dari aplikasi android agar nantinya penggunaan *template* ini menjadi lebih mudah dan mempercepat proses pengolahan (Gambar 5).

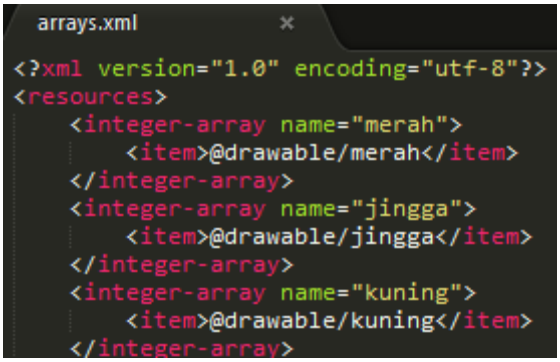


Gambar 5. Lokasi Penyimpanan *Template* Warna



Gambar 6. *Template* Warna Pada Folder *Drawable-Hdpi*

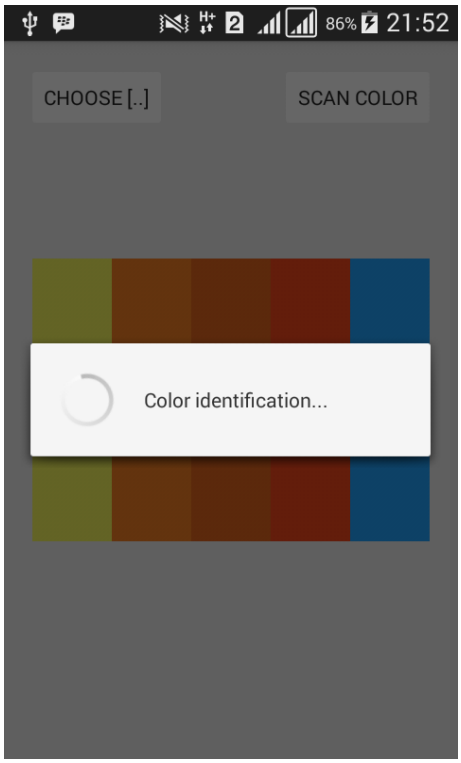
Langkah selanjutnya buat file xml yang digunakan untuk membaca mengakses setiap *template* warna kedalam array (Gambar 7).



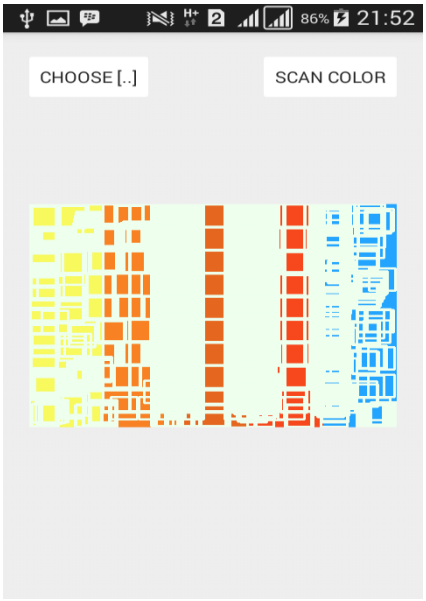
Gambar 7. File Xml Untuk Membaca *Template Di* Folder *Drawable-Hdpi*

Langkah selanjutnya aplikasi akan membaca setiap *template* dari folder *drawabel-hdpi* kemudian menggunakan metode *template matching* untuk mengidentifikasi setiap warna yang teridentifikasi pada citra yang sedang diproses.

Setelah keseluruhan *template* digunakan untuk mengidentifikasi warna pada citra, aplikasi akan membuat sebuah tanda pada citra yang teridentifikasi dengan tanda kotak/ *rectangle* pada bagian yang sesuai dengan *template* warna. Setelah proses identifikasi selesai maka dilakukan perhitungan kumulatif dimana seluruh warna yang terdeteksi dijumlah kemudian nilai pada setiap nilai dari template warna dihitung dengan persamaan $\frac{n}{t}$ x 100% dimana n = nilai dari setiap warna dan t=nilai total dari warna yang terdeteksi.



Gambar 8. Proses Identifikasi Warna



Gambar 9. Hasil Penandaan Bagian Warna Yang Teridentifikasi

Gambar 10. Hasil Identifikasi Warna

Pada Gambar 8. merupakan proses identifikasi citra menggunakan *template* warna. Pada Gambar 9. merupakan hasil dari pengidentifikasian warna berupa citra yang telah ditandai sebagai bagian yang teridentifikasi sebagai warna dari setiap *template.* Pada Gambar 10. merupakan hasil dari perhitungan secara kumulatif dari total warna yang terdeteksi kemudian dilihat warna yang paling signifikan dari citra yang diproses. *Sourcecode* yang digunakan untuk proses *template matching* adalah sebagai berikut :

1. *Imgproc.matchTemplate*
2. *Imgproc.rectangle*

5.3. Pengujian

Pada pengujian ini dilakukan percobaan pengidentifikasian 40 citra yang memiliki warna dan karakteristik yang berbeda. Kemudian citra yang teridentifikasi dengan tepat akan dikategorikan dengan nilai 1. Sedangkan pengidentifikasian yang tidak tepat akan dikategorikan dengan nilai 0 untuk melihat seberapa sering aplikasi dapat melakukan pemrosesan dengan hasil yang tepat maupun salah.

Dari hasil pengujian yang dilakukan pada ke-40 gambar berwarna diatas, didapat hasil keberhasilan pengidentifikasian warna sejumal 6 warna, sedangkan 34 warna yang lainnya terjadi kesalahan pengidentifikasian. Dari hal ini didapat keberhasilan pengidentifikasian yaitu 15% dan kegagalan sebesar 85% hasil yang didapat merupakan hasil uji coba yang telah dilakukan dengan histogram dan aplikasi *template matching* paparan perhitungan :

$\frac{6}{40}$x 100% = 15% Tingkat Keberhasilan identifikasi

$\frac{34}{40}$x 100% = 85% Tingkat Kegagalan identifikasi

5.4. Pengaruh Jumlah *Template* Warna Terhadap Kecepatan Pemrosesan

Akurasi dari pengidentifikasian sangat bergantung dari jumlah *template* dan karakteristik dari *template* yang beragam. Namun penambahan jumlah *template* warna akan mengakibatkan proses yang lebih lama dan akan menimbulkan kegagalan dalam proses pengidentifikasian. Hal ini dikarenakan aplikasi akan melakukan pemrosesan pada keseluruhan *template* terhadap citra yang diidentifikasi.

Penambahan jumlah *template* merupakan cara peningkatan akurasi pengidentifikasian warna. Hal ini dikarenakan metode *template matching* membutuhkan data yang bervariasi untuk meningkatkan akurasinya. Sehingga penambahan jumlah *template* bukan merupakan solusi yang tepat untuk meningkatkan akurasi pengidentifikasian warna.

6. Kesimpulan dan Saran

6.1. Kesimpulan

Berdasarkan kegiatan yang telah dilaksanakan dalam penelitian ini, maka dapat diambil beberapa kesimpulan, diantaranya:

1. Metode *template matching* tidak dapat bekerja dengan maksimal jika digunakan untuk mengidentifikasi warna. Hal ini dilihat dari jumlah keberhasilan yang hanya mencapai 15% atas pengujian yang dilakukan pada 40 citra.
2. Implementasi metode *template matching* pada aplikasi android dapat dilakukan dengan menggunakan *library opencv*. Kecepatan pemrosesan bergantung pada jumlah *template* warna dan ukuran dari citra yang diproses.

6.2. Saran

Dalam penelitian mengenai pengidentifikasian warna ini tidak terlepas dari beberapan kekurangan. Oleh karena itu, penulis menyarankan beberapa hal, antara lain:

1. Penggunaan metode lain dalam mengidentifikasi warna agar pada penelitian selanjutnya dapat dilakukan dengan lebih akurat.
2. Penggunaan metode *template matching* pada pendeteksian citra akan lebih akurat jika menggunakan *template* yang memiliki karakteristik yang detail dan tidak hanya terpaku pada perbedaan warna saja. Tetapi juga semua ciri dari citra tersebut.

Daftar Pustaka

[1] Wintolo, Hero dkk., 2015. *Pemanfaatan Image To Speech Berbasis Android Untuk Pengenalan Warna Bagi Anak Bawah Tiga Tahun (Batita)*. Jurnal Angkasa Volume 7 No.2 , November 2015, Sekolah Tinggi Teknologi adisutjipto ,Yogyakarta.

[2] Bahri, Raden Sofian dkk., 2012. *Perbandingan Algoritma Template Matching Dan Feature Extraction Pada Optical Character Recognition*. Jurusan Teknik Informatika, Universitas Komputer Indonesia, Bandung.

[3] Umam, Choirul., 2015. *Deteksi Osteoporosis Dengan Metode Template Matching Pada Citra Sinar Rontgen Tulang Panggul Manusia.* Jurusan Teknik Informatika, Universitas Dian Nuswantoro, Semarang

[4] Cahyana, Fajar MIT., 2014. *Perancangan Program Penghitung Jumlah Kendaraan Di Lintasan Jalan Raya Satu Arah Menggunakan Bahasa Pemrograman C++ Dengan Pustaka Opencv*. Universitas Brawijaya, Malang.

[5] Dennis, Alan dkk. 2005. *System Analysis Design with UML version 2.0 an Object-Oriented Approach*. Wiley. Indiana University.