PENGEMBANGAN METODE BLOCK MATCHING UNTUK DETEKSI COPY-MOVE PADA PEMALSUAN CITRA

Arya Yudhi Wijaya[1)], Said Al Musayyab[2)] , dan Hudan Studiawan[3)]
[1, 2,3)] Departemen Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: arya@if.its.ac.id[1)], saidmusayyab@gmail.com [2)], hudan@if.its.ac.id[3)]

ABSTRAK

Pemalsuan citra dengan maksud menutupi sebagian objek pada citra dengan blok lain pada citra yang sama disebut dengan copy-move. Deteksi copy-move pada citra dapat dilakukan pada domain spasial melalui pengolahan pada tiap pikselnya maupun pada domain frekeunsi melalui beberapa fungsi transformasi. Penelitian ini mengusulkan deteksi copy-move pada domain spasial dengan mengembangkan metode block macthing. Metode yang diusulkan terbagi atas dua pendekatan yaitu excact match dan robust match. Pendekatan excact match dimulai dengan: input citra RGB, pengambilan blok, penghitungan nilai hash tiap blok, pencarian blok yang mirip dan diakhiri dengan dengan operasi morfologi untuk penghalusan hasil deteksi. Sedangkan pendekatan robust match mirip dengan exact match namun nilai hash diganti dengan Discrete Cosine Transform (DCT). Hasil uji coba menujukkan bahwa pendekatan robust match mendapatkan hasil sedikit lebih baik dibandingkan dengan excact match dimana nilai rata-rata kualitas deteksi 75% dengan kualitas deteksi terbaik sebesar 97%.

***Kata Kunci**: Copy-Move, Exact Match, forensik digital, Robust Match*

ABSTRACT

Image counterfeiting in order to cover most of the object in the image with other blocks in the same image called a copy-move. Detection of copy-move on the image can be performed in the spatial domain through the processing of each pixel and in the frequency domain through some transformation functions. This research proposes the detection of copy-move in the spatial domain by developing macthing block method. The proposed method is divided into two approaches, excact match and robust match. Excact match approach begins with input RGB image, making blocks, calculating the hash value of each block, block similar searches and ends with the morphology operations for enhancing the detection result. While the robust match approach is similar to the exact match but the hash value is replaced with a Discrete Cosine Transform (DCT). The experiment results showed robust match get slightly better results than the excact match where the average value of the quality of detection is 75% with the best quality of detection is 97%.

***Keywords**: Copy-Move, Digital Forensic, Exact Match, Robust Match*

I. PENDAHULUAN

Perkembangan teknologi pengolahan citra digital menjadikan pengguna semakin mudah untuk melakukan modifikasi citra yang berdampak pada maraknya pemalsuan citra. Pemalsuan citra dapat menimbulkan kesalahpahaman bagi seseorang yang melihat citra tersebut. Orang yang melihatnya juga merasa kesulitan untuk membedakan apakah citra tersebut otentik atau citra hasil modifikasi dari citra otentik.

Salah satu jenis pemalsuan citra paling umum karena sangat mudah dan efektif untuk diimplementasikan adalah *copy-move* [1]. *Copy-Move* adalah pemalsuan citra yang dilakukan dengan maksud untuk membuat suatu objek "menghilang" dari gambar dengan menutupinya dengan blok kecil disalin dari bagian lain dari gambar yang sama [2]. Teknik ini biasanya digunakan untuk menutupi objek yang dianggap mengganggu dengan cara menempelkan objek yang serupa di sekitarnya. Selain itu *copy-move* juga digunakan untuk menduplikasi objek pada citra, sehingga terdapat objek serupa yang lebih banyak dalam sebuah citra.

Contoh pemalsuan citra dengan *copy-move* dapat diamati pada Gambar 1. Gambar 1 sebelah kanan merupakan hasil modifikasi Gambar 1 sebelah kiri. Objek pohon pada Gambar 1 sebelah kiri diduplikasi dan menutup objek lain sehingga menjadi Gambar 2 sebelah kanan.

Dikarenakan pemalsuan citra dilakukan dengan menduplikasi bagian lain dari suatu citra ke citra yang sama, maka metode pencarian objek yang paling efektif adalah *exhaustive search*. Akan tetapi, *exhaustive search* memerlukan kompleksitas waktu yang cukup tinggi [3]. Agar kecepatan meningkat, banyak penelitian dilakukan dengan membagi citra dalam blok-blok untuk mendeteksi *copy-move* [4]. J. Fridrich mengusulkan pembagian citra dalam blok dan menggunakan metode *exact-match* piksel demi piksel dan *robust-match* dengan Discrete Cosine

Gambar. 1. Contoh copy-move. Gambar sebelah kiri merupakan gambar otentik, sedangkan gambar sebelah kanan merupakan hasil dari duplikasi sebagian objek gambar sebelah kiri

Transform (DCT) untuk mendeteksi bagain citra yang diduplikasi [5]. Metode ini sudah mengatasi masalah namun *exact-match* masih kurang efisien dikarenakan blok yang dihasilkan cukup besar. A. C. Popescu dan H. Farid mengusulkan metode deteksi yang sama [6], di mana blok gambar dikurangi dimensinya dengan menggunakan Principal Component Analysis (PCA) namun masih kurang efisien karena blok langsung diambil dari citra asli sehingga jumlah blok masih cukup besar. G. Li, Q.Wu, D.Tu mengembangkan metode ketetanggan berbasis Discrete wavelet Transform (DWT) dan Singular Value Decomposition (SVD) [7]. Metode ini membutuhkan banyak waktu dan komputasi yang kompleks pada perhitungan SVD.

Penelitian ini akan mengusulkan metode pendeteksian *copy-move* dengan menggunakan metode *block matching*. *Block matching* berjalan dengan cara membagi citra menjadi beberapa blok lalu dilakukan operasi perhitungan dan perbandingan antar blok. Metode *block matching* akan diimplementasikan melalui dua pendekatan, yaitu *exact match* berbasis fungsi *hash* dan *robust match* berbasis DCT dengan perluasan blok. Kedua pendekatan ini akan dibandingkan untuk diketahui mana yang lebih baik dalam mendeteksi *copy-move*. Diharapkan dengan menggunakan fungsi *hash* dan perluasan blok pada DCT akan dapat mendeteksi *copy-move* dengan berbagai karakteristik citra.

II. TINJAUAN PUSTAKA

A. Forensik Digital

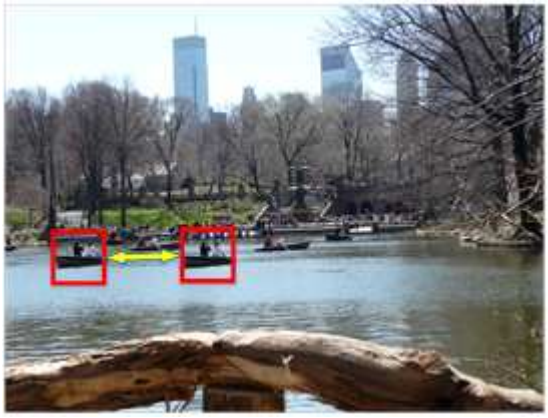
Forensik digital adalah aktivitas yang berhubungan dengan pemeliharaan, identifikasi, ekstraksi dan dokumentasi bukti digital dalam kejahatan [8]. Tujuan dari forensik digital adalah menemukan, menganalisis, menjaga barang bukti pada artefak terkait. Bukti-bukti dalam forensik digital sangatlah banyak, seperti semua bukti fisik seperti komputer atau hardisk, dokumen-dokumen yang tersimpan dalam komputer seperti (citra, PDF, video, lalu lintas data dalam jaringan, dan lain sebagainya). Dengan identifikasi barang bukti dapat memperkirakan efek yang terjadi, dan menemukan identitas pelaku yang mungkin. Karena luasnya cangkupan, forensik digital dibagi menjadi beberapa cabang, seperti: forensik citra digital, forensik komputer, forensik analisis data, forensik jaringan, forensik perangkat bergerak, dan forensik basis data.

Forensik citra digital bertujuan untuk memvalidasi keaslian suatu citra dengan mengetahui informasi riwayatnya. Forensik citra digital kegiatan memfokuskan pendeteksian pada aktivitas pemalsuan citra. Contoh pemalsuan citra yang sering terjadi adalah menambahkan atau mengurangi informasi yang ada pada citra [9].

Untuk menambah informasi pada citra, dibutuhkan informasi tambahan yang akan diletakkan pada citra. Untuk mengurangi informasi, biasanya pemalsu citra membutuhkan konten lain untuk menutupi informasi yang ingin dihilangkan. Pemalsuan seperti ini dapat terjadi dengan melibatkan satu atau lebih citra. Pemalsuan dengan melibatkan satu citra dapat disebut *copy-move*. Sedangkan pemalsuan dengan melibatkan lebih dari satu citra dapat disebut *copy-splice*. Gambar 1 sebelah kiri merupakan citra otentik di mana citra tersebut dilakukan *copy-move*. Tantangan forensik digital adalah melakukan deteksi terhadap pemalsuan-pemalsuan yang terjadi.

B. Copy-move

Copy-Move adalah pemalsuan citra yang dilakukan dengan maksud untuk membuat suatu objek "menghilang" dari gambar dengan menutupinya dengan blok kecil disalin dari bagian lain dari gambar yang sama [10]. Teknik ini biasanya digunakan untuk menutupi objek dari suatu citra dengan cara menempelkan objek yang serupa di sekitarnya atau menempelkan objek pada posisi lain sehingga objek terlihat lebih banyak. Untuk menutupi objek,



Gambar 2. Visualisasi block matching. Dua blok dengan ukuran yang sama akan dicocokkan dan disimpulkan apakah dua blok tersebut mirip antara satu dengan yang lainnya atau tidak.

umumnya citra yang disalin adalah tekstur seperti rumput, dedaunan, kerikil, ataupun objek lain yang memiliki pola yang tidak beraturan tujuannya agar objek yang menutupi serupa dengan latar atau objek di sekitarnya sehingga sulit dibedakan dengan mata [5].

Contoh pemalsuan citra dengan *copy-move* dapat diamati pada Gambar 1. Gambar 1 sebelah kanan merupakan hasil modifikasi Gambar 1 sebelah kiri. Objek pohon pada Gambar 1 sebelah kiri diduplikasi dan menutup objek lain sehingga menjadi Gambar 2 sebelah kanan.

C. *Metode Block-Matching*

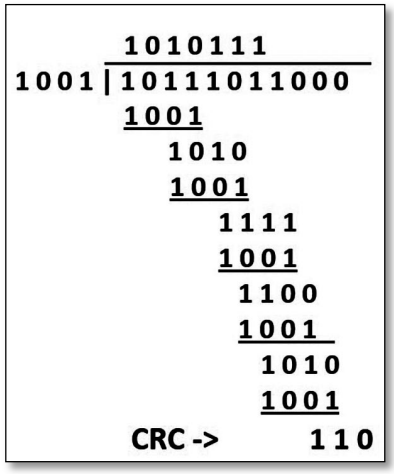
Block matching adalah proses membagi citra menjadi beberapa bagian persegi atau blok. Kemudian fitur pada setiap blok diekstrak untuk dapat dicocokkan dengan blok lain [10]. Metode *block matching* memiliki banyak variasi dalam proses penerapannya. Variasi metode dapat berupa cara mendapatkan blok, fitur yang diambil dalam satu blok, maupun cara melakukan perbandingan antar fitur atau blok.

Visualisasi *block matching* dapat diamati pada Gambar 2. Dua blok dengan ukuran yang sama akan dicocokkan dan disimpulkan apakah dua blok tersebut mirip antara satu dengan yang lainnya atau tidak.

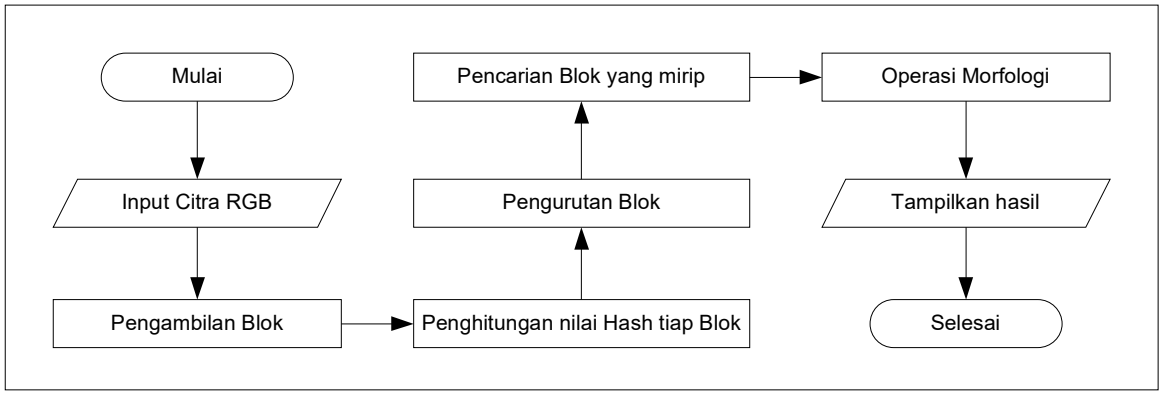
D. *Fungsi Hash*

Hash adalah sebuah nilai yang digunakan untuk mengecek integritas sebuah data. *Hash* juga bisa dikatakan enkripsi satu arah. Nilai hash didapatkan melalui fungsi *hash*. Fungsi *hash* adalah sebuah fungsi yang memiliki input bit string dengan panjang dinamis dan mengeluarkan bit string dengan panjang yang statis [11]. *Hash* biasanya digunakan dalam konteks keamanan informasi dan jaringan.

Contoh penggunaan hash dalam pengecekan integritas data adalah nilai *hash* pada situs yang menyediakan dokumen yang dapat diunduh oleh pengguna. Untuk mengetahui dokumen telah diunduh dengan sempurna, situs menyediakan nilai *hash* pada setiap dokumen. Apabila pengguna ingin mengecek apakah dokumen yang diunduh telah sesuai dengan yang ada pada situs tersebut, pengunduh dapat menghitung nilai hash pada dokumen yang diunduh,



Gambar 3. Ilustasi penghitungan CRC 3 bit.



Gambar 4. Diagram alir metode *exact match* yang diusulkan

kemudian disamakan dengan nilai hash yang ada pada situs. Apabila nilai hash sama, maka dokumen yang diunduh sesuai dengan dokumen pada situs.

Salah satu conth fungsi *hash* adalah *Cyclic Redudancy Check* (CRC). Prinsip kerja CRC adalah menganggap suatu file yang diproses sebagai suatu string yang besar dan terdiri dari bit-bit dan dioperasikan terhadap suatu bilangan polinomial yang sangat besar [12]. Dalam mendeteksi kerusakan file, metode ini menggunakan suatu nilai yang disebut dengan *checksum*. *Checksum* atau nilai CRC merupakan sisa hasil bagi polinomial CRC atau disebut poly oleh polinomial yang merepresentasikan suatu file. Perhitungan dalam metode CRC menggunakan bilangan biner karena beroperasi dalam level bit. Selain itu, operasi pengurangan dan penjumlahan dilakukan dengan mengabaikan *carry* yang didapat.

Contoh perhitungan CRC dapat dilihat pada Gambar 3. Bit sumber 10111011000 akan dibagi oleh 1001 menghasilkan 1010111 dan sisa 110. Sisa 110 ini yang disebut sebagai *checksum* atau nilai CRC. Panjang bit CRC ditentukan sebelumnya dan biasa mengikuti nama dibelakang CRC. Misal CRC32 adalah metode CRC dengan panjang CRC 32 bit.

III. METODE PENELITIAN

Penelitian ini mengusulkan pengenalan *copy-move* berbasis *block matching*. Metode *block matching* akan diimplementasikan melalui dua pendekatan, yaitu *exact match* berbasis fungsi *hash* dan *robust match* berbasis DCT dengan perluasan blok.

A. Exact Match

Tahapan metode *exact match* dapat diamati pada Gambar 4 dengan penjelasan sebagai berikut:

1. Input Citra RGB

File citra berupa citra warna RGB berukuran *M*x*N* yang diduga mengandung *copy-move* dimasukkan sebagai inputan.

2. Pengambilan Blok

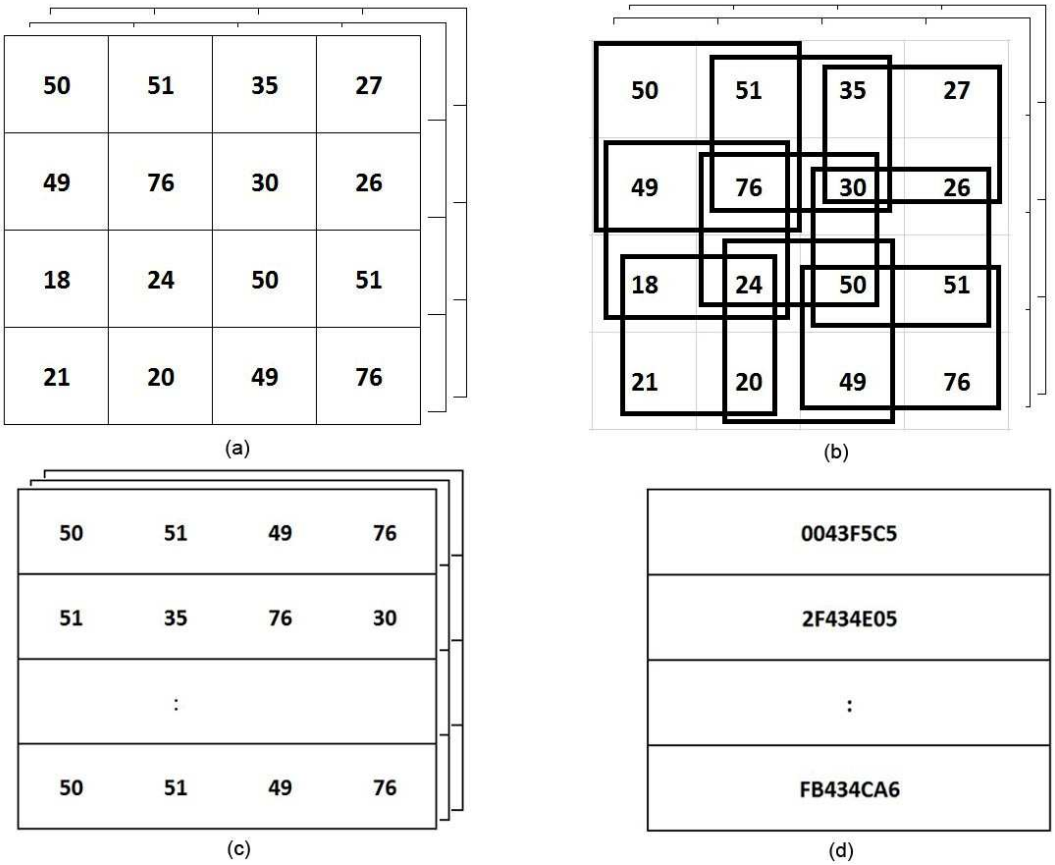
Blok adalah sebuah persegi berukuran *B*x*B* piksel pada setiap layer RGB. Blok akan digeser setiap satu piksel pada semua bagian citra mulai dari pojok kiri atas sampai dengan pojok kanan bawah dengan urutan kiri ke kanan lalu ke bawah. Di setiap pergeseran posisi blok, nilai piksel pada blok tersebut diekstrak, kemudian disimpan pada array A_R, A_G, A_B untuk masing-masing layer RGB. Di akhir iterasi masing-masing array A_i akan memiliki baris sebanyak $(M-B+1)(N-B+1)$ di mana M dan N adalah ukuran dari citra input. Setiap baris mencerminkan satu posisi dari pergeseran blok

Ilustrasi pengambilan blok dapat diamati pada Gambar 5. Gambar 5(a) adalah nilai intensitas RGB citra input. Gambar 5(b) adalah contoh pengambilan blok jika 1 blok berukuran 4x4. Gambar 5(c) adalah array A_R, A_G, A_B yang setiap barisnya menyimpan infomasi 1 blok pada masing-masing layer citra RGB.

3. Penghitungan nilai hash tiap Blok

Setiap baris yang bersesuaian dari array A_R, A_G, A_B dijajar bitnya dengan urutan dari kiri ke kanan dan berikutnya dicari nilai hash-nya menggunakan fungsi CRC32. Sehingga *array* A_R, A_G, A_B menjadi vektor A dengan baris sebanyak $(M-B+1)(N-B+1)$ yang tiap baris berisi nilai hash dari setiap blok. Ilustrasi vektor A dapat diamatai pada Gambar 5(d).

4. Pengurutan Blok



Gambar 5. Ilustasi (a) citra input RGB (b) pengambilan blok (c) representasi blok dalam piksel (d) representase blok dengan fungsi hash CRC32

Vektor A yang berisisi nilai hash diurutkan menaik dari baris pertama hingga baris ke $(M-B+1)(N-B+1)$. Hal ini bertujuan untuk mepermudah proses pencarian blok yang mirip pada tahap berikutnya.

5. Pencarian Blok yang mirip

Pencarian blok yang mirip dilakukan dari atas ke bawah pada vektor *A*. Baris yang berisi nilai *hash* yang sama dengan baris dibawahnya ditandai dan dikembalikan milik piksel pada koordinat berapa di citra input. Pencarian blok yang mirip pada array terurut lebih efisien dengan kompleksitas $O(MN\log_2(MN))$ dibandingkan dengan melakukan perbandingan pada semua blok yang memiliki kompleksitas $O((\mathrm{MN})^2)$.

6. Operasi Morfologi

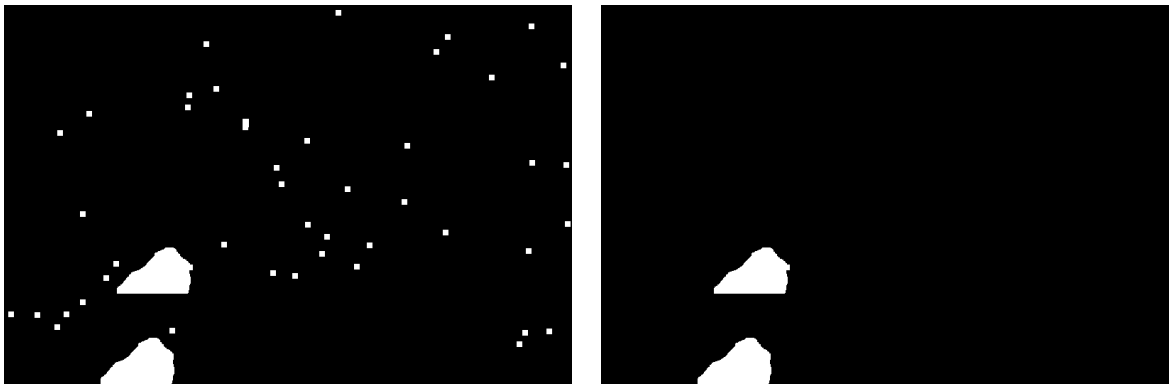
Hasil pada metode tahap sebelumnya masih menimbulkan beberapa kesalahan deteksi berupa blok-blok kecil yang dideteksi sebagai copy-move seperti pada Gambar 6(a). Padahal blok tersebut bukanlah copy-move. Oleh karena itu, objek-objek tersebut dapat dihapus dengan operasi morfologi *opening*. *Opening* dari himpunan A dengan "structuring element" B didefinisikan

$$A \circ B = (A \Theta B) \oplus B \tag{1}$$

Opening memberikan efek menghilangkan objek-objek kecil sebagimana terlihat pada Gambar 6. Gambar 6(a) adalah citra biner dimana warna putih adalah blok yang dideteksi terjadi copy-move dari tahap sebelumnya. Morfologi opening menjadikan area blok-blok kecil yang seharusnya bukan copy-move terhapus sebagaimana Gambar 6 (b)

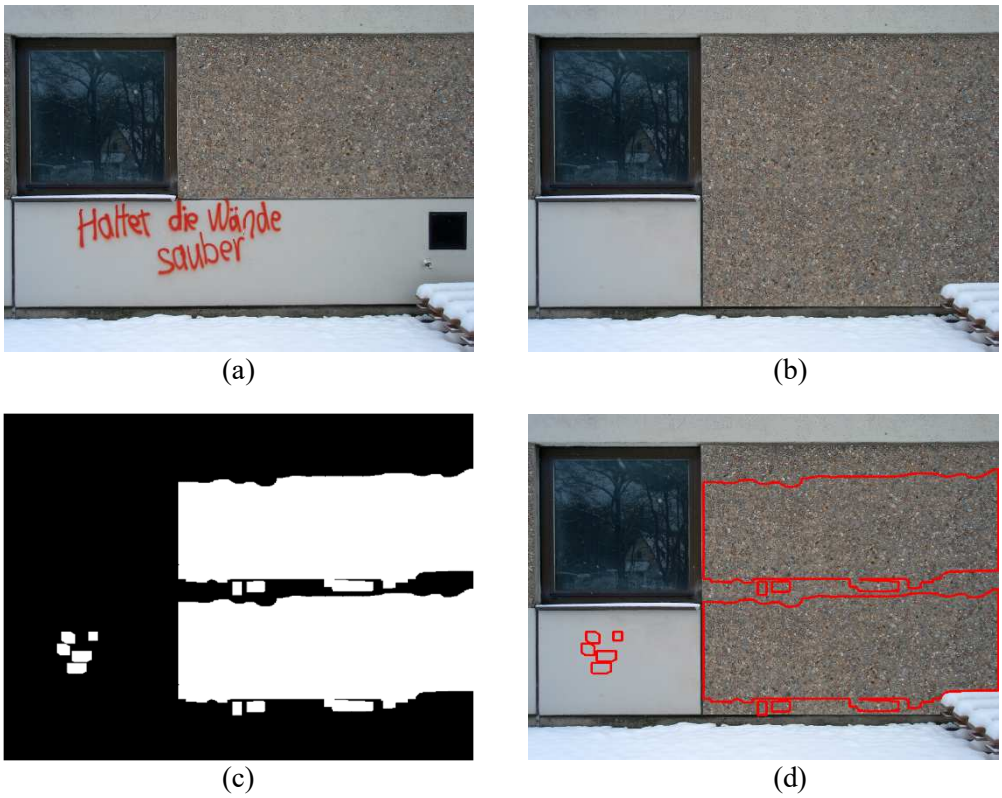
7. Tampilkan hasil

Citra ditampilkan dan ditandai bagian yang mengandung *copy-move*. Ilustrasi lengkap tahap awal hingga akhir dapat dilihat pada Gambar 7. Gambar 7(a) citra otentik, Gambar 7(b) citra *copy-move*, Gambar 7(c) deteksi *copy-move*, dan Gambar 7(d) adalah penampilan hasil citra yang dideteksi sebagai objek *copy-move*.



(a) (b)

Gambar 6. Area yang terdeksi sebagai copy-merge (a) sebelum morfologi opening (b) setelah morfologi opening



(a) (b)

(c) (d)

Gambar 7. Ilustrasi lengkap pengenalan objek *copy-move* (a) citra otentik (b) citra yang telah megalami *copy-move* (c) blok yang terdeteksi sebagai lokasi objek *copy-move* (d) tampilan hasil citra yang dideteksi sebagai objek *copy-move*

Gambar 8. Diagram alir metode *robust match* yang diusulkan

B. Robust Match

Pada umumnya metode *robust match* tidak jauh berbeda dengan *exact match*. Perbedaan utamanya yaitu pada tahap pengambilan blok dan beberapa tahapan tambahan untuk mengurangi kesalahan deteksi. *Robust match* juga menggunakan konsep blok yang sama dengan metode *exact match*. Urutan metode yang diusulkan sebagaimana ditampilkan pada Gambar 8 juga mirip dengan metode *exact match*. Bagian yang berbeda adalah pengambilan bloknya tidak langsung pada piksel namun dengan DCT. Selain itu, peningkatan akurasi tidak dilakukan dengan morfologi, namun digunakan sejumlah nilai batas (*threshold*) yang ditentukan di awal.

Blok yang digunakan hanya dapat berukuran minimal 8x8, karena harus sesuai dengan matriks kuantisasi yang berukuran 8x8. Citra dikonversi menjadi bentuk *grayscale* agar citra dapat dilakukan perhitungan DCT dan kuantisasi. Selanjutnya tahap pengambilan blok, seperti pada metode *exact match* yaitu dilakukan pergeseran blok setiap satu piksel. Setiap posisi diambil piksel pada blok, kemudian dilakukan perhitungan DCT dengan rumus

$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y} \cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right], \quad (2)$$

dimana:

- u adalah arah horizontal , untuk integer $0 \le u < 8$
- v adalah arah vertikal, untuk integer $0 \le v < 8$
- $\alpha(u) = \begin{cases} \frac{1}{2}, u = 0 \\ 1, u \neq 0 \end{cases}$ adalah faktor skala normalisasi untuk membuat transformasi *orthonormal.*
- $g_{x,y}$ adalah nilai piksel pada koordinat (x,y)
- $G_{u,v}$ adalah koefisien DCT pada koordinat

Kemudian dihitung nilai kuantisasi dengan rumus

$$C_{i,j} = round\left(\frac{D_{i,j}}{Q_{i,j}}\right), \quad (3)$$

dimana :

- Q = matriks kuantisasi
- D = matriks hasil DCT
- C = hasil kuantisasi

Proses perhitungan DCT dan kuantisasi bertujuan untuk mendapatkan representasi yang *robust* dari blok. Apabila blok berukuran lebih dari 8x8, sebuah blok dapat direpresentasikan gabungan dari beberapa blok 8x8. Dimana setiap blok 8x8 dilakukan perhitungan DCT dan kuantisasi. Setelah perhitungan kuantisasi, didapatkan nilai *hash* untuk disimpan pada *array A*.

Kemudian blok diurutkan secara leksikografi untuk memperkecil kompleksitas pencarian. Tahap pencarian dilakukan dengan cara melakukan perulangan sepanjang *array A*, apabila menemui 2 nilai berurutan yang sama, algoritma akan menghitung jarak dari 2 blok yang ditemui, jika jarak kurang dari batas yang ditentukan (*threshold J*), maka kedua blok tersebut tidak dilakukan proses berikutnya. Kemudian dilakukan penambahan pada *shift vector counter C*. Perhitungan *shift vector counter* dilakukan dengan formula

$$s = (s_1, s_2) = (i_1 - j_1, i_2 - j_2) \text{ dan} \quad (4)$$

$$C(s_1, s_2) = C(s_1, s_2) + (i_1, i_2) + (j_1, j_2), \quad (5)$$

dimana:

- (i_1,i_2) dan (j_1,j_2) adalah posisi dari dua blok yang sama
- s adalah *shift-vector*
- C adalah *shift-vector counter*


(a) giraffe.png, size 800 x 533


(b) tree.png, size 1024 x 683


(c) clean walls.png, size 1024 x 754

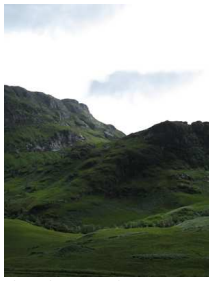

(d) cattle.png, size 1280 x 854


(e) knight moves.png, size 1440 x 1440


(f) threehundred.png, size 2362 x 1581


(g) statue.png, size 2660 x 2104


(h) malawi.png, size 3072 x 2034


(i) scotland.png, size 2304 x 3072

Gambar 9. Dataset citra yang didalamnya terdapat objek *copy-move*

Shift-vector counter C selalu ditambahkan setiap menemui dua blok berurutan yang sama pada *array A*. Diakhir pencarian, algoritma akan dengan menemukan *shift-vector* yang memiliki jumlah blok yang melebihi batas *threshold T* yang telah ditentukan pengguna.

Blok yang telah sesuai dengan *threshold T* akan dilakukan penyaringan selanjutnya yaitu banyaknya objek pada setiap *shift-vector*. Untuk menghitung banyak objek pada setiap *shift-vector*, posisi pada setiap *shift-vector* dipisah menjadi dua *list* berdasarkan koordinat *x* dan *y*. Setiap list diurutkan, kemudian diiterasi sambil menghitung selisih dari dua bilangan yang berurutan. Jika terdapat selisih lebih dari satu, maka sebuah *counter* objek ditambahkan. *Counter* ini yang menunjukkan banyaknya persebaran objek. *Counter* ini akan dibandingkan dengan nilai *threshold* banyak objek *K*. Nilai *threshold J, T, K* dan ukuran blok *B* merupakan faktor yang mempengaruhi sensitifitas algoritma. Nilai-nilai ini yang akan digunakan divariasikan untuk dilakukan uji coba.

IV. SKENARIO DAN HASIL UJI COBA

A. Metode Evaluasi

Dibutuhkan suatu acuan untuk mengetahui performa dari metode dan uji coba yang telah dilakukan. Acuan yang digunakan yaitu menghitung *precision* dan *recall* dari hasil metode dan *groundtruth* pada dataset. *Precision* menunjukkan bahwa objek hasil deteksi adalah sesuai dengan objek serangan. Sedangkan *recall* menunjukkan kemungkinan objek serangan yang terdeteksi. Perhitungan *precision* dan recall dilakukan di tingkat piksel [10].

Nilai *precision* dan *recall* didapatkan dengan membadingkan terhadap *groundtruth*. Kemudian dicari nilai *TP, TN, FP, FN*. *True Positives* (*TP*) merupakan suatu kondisi dimana prediksi bernilai benar, dan hasil sesungguhnya juga bernilai benar. *True Negatives* (*TN*) adalah kondisi dimana prediksi bernilai salah dan yang sebenarnya bernilai salah. *False Positives* (*FP*) merupakan kondisi dimana nilai prediksi adalah benar dan sebenarnya adalah salah (*Type I error*), dan *False Negatives (FN)* adalah kondisi dimana prediksi bernilai salah dan sebenarnya bernilai benar (*Type II error*). Dari nilai-nilai tersebut, dapat diperoleh *precision* dan *recall* dengan rumus sebagai berikut

$$Precision = \frac{TP}{TP + FP} \quad \text{dan} \tag{6}$$

$$Recall = \frac{TP}{TP + FN}, \tag{7}$$

Selain moetode evaluasi di atas, dapat dihitung pula nilai *F*1 yaitu kombinasi dari *precision* dan *recall* untuk mendapatkan nilai evaluasi tunggal dengan rumus

$$F1 = 2\ Precision\ Recall\ /\ (Precision + Recall) \tag{8}$$

B. Data Uji

Data uji yang digunakan adalah 9 sampel citra pada *Image Manipulatin Dataset* [13] yang telah terdapat objek *copy-move* didalamya. Secara lengkap, 9 sampel citra yang didalamnya terdapat objek *copy-move* dapat dilihat pada Gambar 9.

C. Uji coba dengan Metode Exact Match

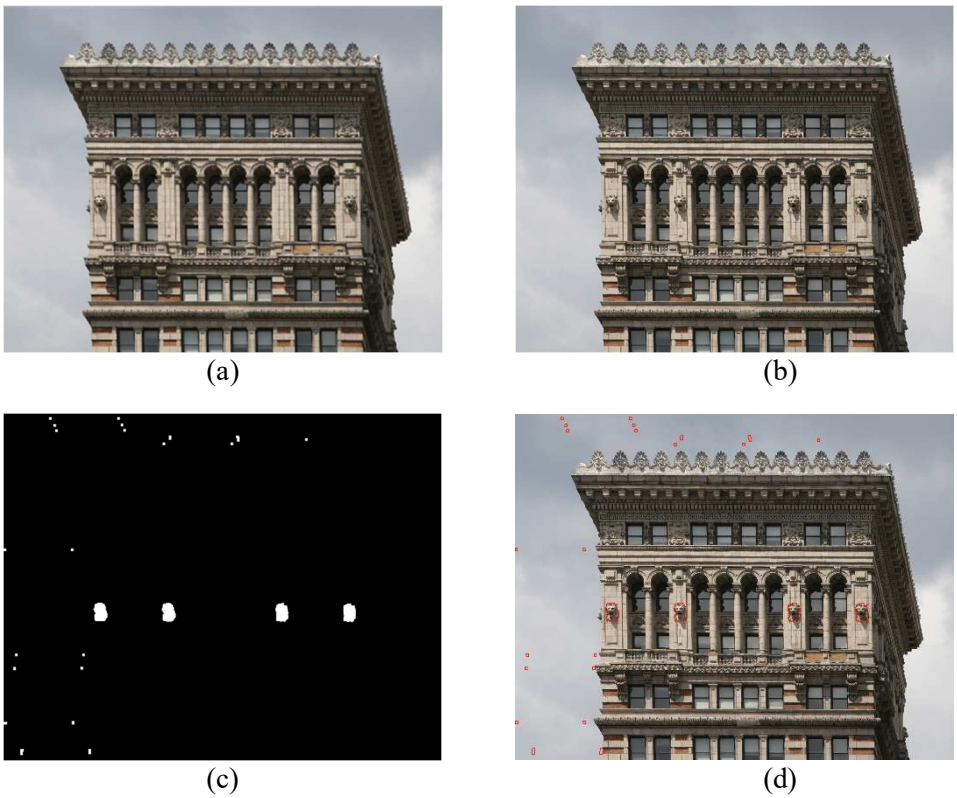
Uji coba dengan metode *exact match* dilakukan dengan menggunakan ukuran blok ukuran 8x8. Hasil uji coba pada salah satu citra sampel yaitu clean walls.png dapat dilihat pada Gambar 7. Hasil evaluasi secara kuantitatif dapat diamati pada Tabel I.

TABEL I
HASIL UJI COBA METODE EXACT MATCH

Nama Citra	Precision (%)	Recall (%)	*F*1 (%)
cattle.png	100	53	69
clean walls.png	99	88	93
giraffe.png	100	82	91
knight moves.png	100	38	56
malawi.png	100	35	51
scotland.png	57	90	70
statue.png	100	89	94
threehundred.png	100	74	85
tree.png	100	44	61
Rata-rata	95	66	74

TABEL II
HASIL UJI COBA METODE ROBUST MATCH

Nama Citra	Precision (%)	Recall (%)	*F*1 (%)
cattle.png	100	68	81
clean walls.png	100	94	97
giraffe.png	100	90	94
knight moves.png	100	25	40
malawi.png	84	87	85
scotland.png	96	17	29
statue.png	81	93	87
threehundred.png	100	86	92
tree.png	100	54	70
Rata-rata	96	68	75



Gambar 10. Ilustrasi pengenalan objek *copy-move* dengan *robust match* (a) citra otentik (b) citra yang telah megalami *copy-move* (c) blok yang terdeteksi sebagai lokasi objek *copy-move* (d) tampilan hasil citra yang dideteksi sebagai objek *copy-move*

D. Uji coba dengan Metode Robust Match

Uji coba dengan metode *robust match* dilakukan dengan menggunakan ukuran blok 16x16. Pengujian ini menggunakan *threshold* jumlah blok *T* bernilai 400 dan *threshold* jarak antar blok *J* dengan nilai 50 piksel. Hasil uji coba pada salah satu citra sampel yaitu statue.png dapat dilihat pada Gambar 10. Hasil evaluasi secara kuantitatif uji coba dengan metode *robust match* dapat diamati pada Tabel II.

V. KESIMPULAN

Secara umum metode *robust match* menghasilkan kualitas yang lebih baik dibandingkan *exact match*. Akan tetapi, metode *robust match* membutuhkan waktu yang lebih lama dikarenakan kompleksitas yang lebih tinggi saat penghitugan DCT. Dalam hal kemampuan mentoleransi objek yang diduga sebagai *copy-merge*, metode *robust match* lebih akomdatif dibanding metode *exact match* dalam mengenali objek copy-move yang mengalami transformasi dikarenakan pengenalan *robust match* tidak berdasarkan kesamaan piksel, namun berdasarkan DCT. Jika ditinjau kemampuan untuk meminimalisir adanya *over estimate* objek *copy-move*, metode *exact match* memiliki kelebihan dibandingkan dengan metode *robust match* karena operasi morfologinya. Sedangkan metode *robust match* hanya dibatasi oleh *threshold* dalam menanggulangi *over estimate* objek copy-move

DAFTAR PUSTAKA

[1] Jing, Li, and Chao Shao. "Image Copy-Move Forgery Detecting Based on Local Invariant Feature." Journal of Multimedia, vol. 7, no. 1, pp.90-97, 2012.
[2] B.L.Shivakumar,Lt. Dr. S.Santhosh Baboo"Detecting Copy-Move Forgery in Digital Images: A Survey and Analysis of Current Methods "Global Journal of Computer Science and Technology Vol. 10 Issue 7 Ver. 1.0 September 2010
[3] Yadav, Preeti, and Yogesh Rathore. "Detection of Copy-Move Forgery of Images Using Discrete Wavelet Transform." International Journal on Computer Science and Engineering, vol. 4, no. 4, pp.365-370, April 2012.
[4] Sarah A. Summers, Sarah C. Wahl "Multimedia Securityand Forensics Authentication of Digital Images" http://cs.uccs.edu/~cs525/studentproj/proj52006/sasummer/doc/cs525projsummersWahl.doc
[5] J. Fridrich, D. Soukal, and J. Lukas, "Detection of copymove forgery in digital images," Proceedings of the Digital Forensic Research Workshop. Cleveland OH, USA, 2003.
[6] A.C.Popescu and H.Farid, "Exposing digital forgeries by detecting duplicated image regions," Dartmouth College, Hanover, New Hampshire, USA: TR2004-515, 2004.
[7] G.Li, Q.Wu, D.Tu, and Shaojie Sun, "A sorted neighborhood approach for detecting duplicated regions in image forgeries based on DWT and SVD," IEEE International Conference on Multimedia & Expo, 2007.

[8] P. Albert J. Marcella, Cyber Forensics - A Field Manual of Collectiong, Examining, and Preserving Evidence of Computer Crimes, CRC Press Book, 2002.
[9] J.A. Redi, W. Taktak,J.-L. Dugelay, Digital image forensics: a booklet for beginners, Multimed. Tools Appl. 51 (1) (2011) 133–162.
[10] V. Christlein, C. Riess, J. Jordan, C. Riess dan E. Angelopoulou, "An Evaluation of Popular Copy-Move Forgery Detection Approaches," IEEE Transactions on Information Forensics and Security, vol. 7, pp. 1841-1854, 2012.
[11] H. Delfs dan H. Knebl, Introduction to Cryptography: Principles and Application Second Edition, Heidelberg: Springer, 2007.
[12] Fitrokhoerani N. dan Atrika Anggraeni, "Penggunaan Metode Heuristik Dan Cyclic Redudancy Check 32 (CRC32) untuk Mendeteksi Kerusakan File," Seminar Nasional Matematika dan Pendidikan Matematika UNY, 2016.
[13] (FAU) Friedrich-Alexander-Universität Erlangen-Nürnberg, "Image Manipulation Dataset," [Online]. Available: https://www5.cs.fau.de/research/data/image-manipulation/. [Accessed 10 Nov 2016]