ANALISIS PENDETEKSI KECOCOKAN OBJEK PADA CITRA DIGITAL MENGGUNAKAN MATLAB DENGAN METODE ALGORITMA SIFT

Rosidin

12917216

Tesis diajukan sebagai syarat untuk meraih gelar Magister Komputer

Konsentrasi Forensika Digital

Program Studi Magister Teknik Informatika

Program Pascasarjana Fakultas Teknologi Industri

Universitas Islam Indonesia

2018

Lembar Pengesahan Pembimbing

ANALISIS PENDETEKSI KECOCOKAN OBJEK PADA CITRA DIGITAL MENGGUNAKAN MATLAB DENGAN METODE ALGORITMA SIFT

Rosidin
12191716

Yogyakarta, Januari, 2018

Pembimbing I	Pembimbing II
(Dr. Bambang Sugiantoro, S.Si.,MT)	(Yudi Prayudi, S.Si.,M.Kom)

Lembar Pengesahan Penguji

ANALISIS PENDETEKSI KECOCOKAN OBJEK PADA CITRA DIGITAL MENGGUNAKAN MATLAB DENGAN METODE ALGORITMA SIFT

Rosidin
12917216

Yogyakarta, Januari, 2018

Tim Penguji,

Dr. Bambang Sugiantoro, S.Si.,M.T
Ketua

Yudi Prayudi, S.Si.,M.Kom
Anggota I

Dr. Imam Riadi, M.Kom
Anggota II

Mengetahui,
Ketua Program Pascasarjana Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. R. Teduh Dirgahayu, ST., M.Sc)

Abstrak

ANALISIS PENDETEKSI KECOCOKAN OBJEK PADA CITRA DIGITAL MENGGUNAKAN MATLAB DENGAN METODE ALGORITMA SIFT

Image pada citra digital dapat menjadi sumber informasi bagi siapa saja yang yang mengamatinya, dengan adanya tools untuk pengolahan image pada citra digital, seperti aplikasi gimp dan adobe photoshop. Dapat dengan begitu mudahnya merubah atau memanipulasi keaslian dari image tersebut. Kemampuan ini tentunya juga dapat disalahgunakan untuk merusak kredibilitas keaslian image dalam berbagai aspek, sehingga dapat dilakukan sebagai tindakan kejahatan. Penerapan Algoritma SIFT (Scale Invariant feature tranform) dan histogram warna RGB pada Matlab dapat mendeteksi kecocokan objek pada citra digital dan melakukan pengujian secara akurat.

Pada penelitian ini membahas tentang implementasi untuk mendapatkan kecocokan objek pada citra digital yang sudah dimanipulasi menggunakan metode Algoritma SIFT pada source Matlab, yaitu dengan membandingkan image yang asli dengan image yang sudah dimanipulasi. Kecocokan objek pada citra digital didapat dari banyaknya jumlah keypoint yang didapat, parameter tambahan lainnya yaitu membandingkan jumlah piksel pada image yang dianalisa, serta perubahan histogram pada warna RGB pada masing - masing image yang sudah dinalisa.

Forensik citra digital merupakan salah satu metode ilmiah pada bidang penelitian yang bertujuan untuk mendapatkan fakta-fakta pembuktian dalam menentukan keaslian image pada citra digital. Penggunaan algoritma SIFT dipilih sebagai metode ekstraksi karena metode ini invarian terhadap perubahan skala, rotasi, translasi, dan iluminasi. SIFT digunakan untuk memperoleh ciri dari pola keypoint yang didapatkan. Hasil pengujian menggunakan metode Algoritma SIFT (Scale Invariant feature tranform), diharapkan dapat menghasilkan analisa image yang lebih baik.

Kata kunci

citra, sift, keypoint, histogram, rgb

Abstract

AN ANALYSIS OF OBJECT FITNESS DETECTOR IN DIGITAL IMAGE USING MATLAB THROUGH SIFT ALGORITHM METHOD

Image on digital images can be a source of information for anyone who observes it, with tools for image processing on digital images, such as gimp and adobe photoshop applications. It can so easily change or manipulate the authenticity of the image. This capability can of course also be misused to undermine the credibility of the authenticity of the image in various aspects, so it can be done as a crime. The application of the SIFT Algorithm (Scale Invariant feature tranform) and RGB color histogram in Matlab can detect object matches in digital images and perform accurate testing.

In this study discuss about the implementation to get the match object on digital image that has been manipulated using SIFT Algorithm method on the Matlab source, that is by comparing the original image with the image that has been manipulated. Match objects in digital images obtained from the number of keypoint obtained, additional parameters are comparing the number of pixels in the analyzed image, as well as changes in the histogram on RGB color on each image that has been analyzed.

Digital image forensics is one of the scientific methods in the field of research that aims to obtain evidence facts in determining the authenticity of the image on digital images. The use of the SIFT algorithm is chosen as an extraction method because it is invariant to scale, rotation, translation, and illumination changes. SIFT is used to obtain characteristics of the pattern of keypoint obtained. The test results using SIFT Algorithm method (Scale Invariant feature tranform), is expected to produce better image analysis.

Keywords

image, sift, keypoint, histogram, rgb

Pernyataan Keaslian Tulisan

Dengan ini saya menyatakan bahwa tesis ini merupakan tulisan asli dari penulis, dan tidak berisi material yang telah diterbitkan sebelumnya atau tulisan dari penulis lain terkecuali referensi atas material tersebut telah disebutkan dalam tesis. Apabila ada kontribusi dari penulis lain dalam tesis ini, maka penulis lain tersebut secara eksplisit telah disebutkan dalam tesis ini.

Dengan ini saya juga menyatakan bahwa segala kontribusi dari pihak lain terhadap tesis ini, termasuk bantuan analisis statistik, desain survei, analisis data, prosedur teknis yang bersifat signifikan, dan segala bentuk aktivitas penelitian yang dipergunakan atau dilaporkan dalam tesis ini telah secara eksplisit disebutkan dalam tesis ini.

Segala bentuk hak cipta yang terdapat dalam material dokumen tesis ini berada dalam kepemilikan pemilik hak cipta masing-masing. Apabila dibutuhkan, penulis juga telah mendapatkan izin dari pemilik hak cipta untuk menggunakan ulang materialnya dalam tesis ini.

Yogyakarta, Januari, 2018

Rosidin, S.Kom

Daftar Publikasi

Publikasi selama masa studi

Publikasi yang menjadi bagian dari tesis

Rosidin., Sugiantoro, B., & Prayudi, Y. (2018). "An Analysis of Object Fitness Detector in Digital Image using Matlab Through Sift Algorithm Method". International Journal on Informatics for Development (IJID), the Faculty of Science and Technology State Islamic University (UIN) Sunan Kalijaga Yogyakarta-Indonesia.

Sitasi Publikasi 1

Kontributor	JenisKontribusi
Rosidin	Mendesain eksperimen (40%) Menulis *paper* 50%)
Yudi Prayudi, M.Kom	Mendesain eksperimen (30%) Menulis dan mengedit *paper* (20%)
Dr. Bambang Sugiantoro, S.Si., M.T	Mendesain eksperimen (30%) Menulis dan mengedit *paper* (30%)

Halaman Kontribusi

Kontribusi dari pihak terkait dalam penyelesaian penulisan tesis ini:

1. Bapak Dr. Bambang Sugiantoro, MT selaku Pembimbing I, Bapak Yudi Prayudi, S.Si., M.Kom selaku Pembimbing II dan Bapak Dr. Imam Riadi selaku dosen penguji yang telah memberikan arahan-arahannya kepada penulis, sehingga penulisan tesis ini bisa selesai dengan baik.
2. Adikku yang tercinta Eni Suhaeni dan Suaminya Sutiawan, Pembina Yayasan Cahaya Putra Bangsa Cirebon, Dr. H. Eman Suryaman dan Abil yang telah memberikan bantuan berupa pinjaman dana untuk biaya perkuliahan sehingga dapat selesai masa studi nya sampai dengan tahap pembuatan tesis ini.
3. M. Kukuh dan M. Irfan, yang telah memberikan tempat saya menginap dan memberikan layanan lain selama penyelesain tesis ini, Pa Moko, Dudung, Dedi Haryadi, Danang, Kristono yang telah memberikan masukan - masukan serta motivasi kepada penulis.

Halaman Persembahan

Kupersembahkan karya ini kepada orang-orang yang tersayang dan yang telah berjasa :

1. Kepada Kedua Orang tuaku Usman dan Ibu Niknik Sumarni, yang selama ini selalu memberikan saya dukungan dan tak henti-hentinya memanjatkan do'a kepada Allah Subhanahu wa ta'ala sehingga terselesainya tesis ini.
2. Kepada istri saya Evi Yulianti, yang telah memberikan dukungan dan do'a, sehingga selesai tesis ini, juga kepada anak - anak saya yang tercinta, Nurin Najwa Rosidin dan Ahmad Arsyad Rosidin, yang menjadi pemberi semangat dalam menyelesaikan studi sampai penyelesaian tesis ini, dan akan menjadi penerus dan pewaris orang tuanya, dengan harapan dapat menempuh studi yang lebih baik lagi serta menjadi anak-anak yang soleh.
3. Kepada Adik saya tercinta Eni Suhaeni, dan Suaminya Sutiawan, Pembina Yayasan Cahaya Putra Bangsa, Dr. H. Eman Suryaman, dan juga Abil yang telah memberikan kepercayaannya kepada saya dalam melanjutkan Studi ini, Terima kasih banyak atas pinjaman dana dan bantuan-bantuan lainnya yang diberikan, semoga saya dapat berguna dan dapat memberikan manfaat bagi perkembangan Kampus Universitas Nahdlatul Ulama Cirebon.

Kata Pengantar

Puji syukur kehadirat Allah SWT, karena atas berkat rahmat, taufik serta hidayahnya sehingga dapat diselesaikannya Laporan Tesis ini di Universitas Islam Indonesia Yogyakarta. Shalawat serta salam tak lupa saya panjatkan kepada junjungan Nabi Muhammad SAW, keluarga beliau dan sahabat-sahabat nya, dan para pengikutnya, sampai akhir zaman. Laporan Tesis yang berjudul **"ANALISIS PENDETEKSI KECOCOKAN OBJEK PADA CITRA DIGITAL MENGGUNAKAN MATLAB DENGAN METODE ALGORITMA SIFT"** diajukan untuk memenuhi salah satu syarat akademik dalam menempuh kelulusan Program Pascasarjana Magister Teknik Informatika pada Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta. Dan ucapan terima kasih yang sebesar - besarnya kepada semua pihak yang telah banyak membantu dalam menyelesaikan laporan ini utamanya kepada yang terhormat:

1. Bapak Nandang Sutrisno, S.H., LL.M., M.Hum., Ph.D sebagai Rektor Universitas Islam Indonesia Yogyakarta

2. Bapak Dr. R. Teduh Dirgahayu, ST., M.Sc sebagai Ketua Program Pascasarjana Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta.

3. Bapak Yudi Prayudi, S.Si., M.Kom selaku Ketua PUSFID Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta sekaligus Pembimbing II yang telah menyempatkan waktunya untuk membimbing dan membantu penulis selama penulisan Tesis ini.

4. Bapak Dr. Bambang Sugiantoro, MT selaku Pembimbing I yang telah memberikan arahan-arahan dalam membimbing dan mebantu penulis selama penulisan Tesis ini.

5. Bapak Dr. Imam Riadi, M.Kom selaku Penguji yang telah memberikan masukan dan saran kepada penulis dalam tahap perbaikan-perbaikan penyusunan laporan tesis ini.

6. Segenap Dosen dan Staf Universitas Islam Indonesia Yogyakarta, khususnya Fakutas Teknologi Industri, yang telah membantu dalam proses perkuliahan.

7. Mahasiswa Universitas Nahdlatul Ulama Cirebon, Insan, Galuh dan Verdian yang telah memberikan kontribusinya untuk memberikan sampel data berupa gambar asli dan gambar hasil manipulasi.

8. Teman-teman Forensika Digital khususnya angkatan VII dan VIII yang sudah banyak membantu dalam penyelesaian tesis ini.

Penulis menyadari sepenuhnya bahwa penyusunan laporan Tesis ini masih banyak kekurangan dan kelemahan. Untuk itu saran, kritik dan tambahan yang sifatnya membangun penulis harapkan dari pembaca untuk selanjutnya. Dengan terselesainya tesis ini penulis banyak berterima kasih, dan besar harapan penulis bisa bermanfaat dalam bidang ilmu pengetahuan kedepannya.

Yogyakarta, Januari 2018

Rosidin, S.Kom

Daftar Isi

Daftar Tabel

Daftar Gambar

BAB 1
Pendahuluan

1.1 Latar Belakang

Pengolahan image pada citra digital dapat dengan mudah dibuat, diedit dan dimanipulasi tanpa meninggalkan petunjuk visual oleh penggunanya, seperti aplikasi gimp dan adobe photoshop contoh nya, yang dengan begitu mudah nya keaslian image dapat diubah / dimanipulasi. Kemampuan ini tentunya juga dapat disalahgunakan untuk merusak kredibilitas keaslian image dalam berbagai aspek, sehingga dapat dilakukan sebagai tindakan kejahatan, perubahan image pada citra digital dapat menyampaikan informasi yang berbeda dengan image pada citra digital aslinya. Pelaku kejahatan berupaya untuk memanipulasi image untuk keuntungan mereka sendiri.Terbentuknya suatu image pada citra digital didapat dari beberapa kombinasi pikselnya. Untuk mendapatkan image yang sudah diubah, diperlukan beberapa tahap dalam melakukan proses forensik citra digital. Didalam nya terdapat beberapa perubahan dari aslinya pada setiap nilai pikselnya (Ahmad Raf'ie Pratama, MIT, 2014).

Teknologi Digital khususnya image, telah menjadi teknologi utama untuk menciptakan, memproses, mentransmisikan dan menyimpan informasi berupa pengetahuan dan aset intelektual. Pengetahuan multidimensional dan aset intelektual diproduksi dan diwakili dalam berbagai bentuk seperti audio, video, teks, gambar, jika dikelompokkan, kita dapat menyebutnya sebagai bentuk multimedia. Akhirnya semua bentuk disimpan sebagai bentuk digital dan bentuk byte yaitu konten digital. Image pada Citra digital banyak digunakan di masyarakat kita. Dari surat kabar ke majalah, jurnal ilmiah, dokter di bidang medis, industri mode, ruang pengadilan dan sebagainya sangat bergantung pada image digital. Integritas informasi sangat mendasar di berbagai bidang. Teknologi digital saat ini mulai mengikis kepercayaan. Meskipun kasus manipulasi merusak foto\image bukanlah hal yang baru, selama beberapa tahun terakhir, image yang dirusak muncul dengan frekuensi dan kecanggihan, pengembangan perangkat lunak pengolah image digital yang semakin canggih ini, telah menjadi mudah untuk membuat pemalsuan image dari satu atau beberapa image tanpa meninggalkan petunjuk yang jelas. Kejahatan digital tumbuh pada tingkat sangat pesat. Kejahatan ini telah menyebabkan banyak masalah, termasuk masalah hukum dan etika.

Pemalsuan image pada citra digital banyak dilakukan melalui pendekatan secara pasif. Salah satu metode pendekatan secara pasif yang popular, yaitu melakukan manipulasi dengan cara teknik copy-move. Pemalsuan image pada citra digital dengan cara suatu image disalin kemudian disisipkan ke bagian citra yang lain. Pemalsuan image Copy-Move dilakukan untuk menyembunyikan rincian tertentu atau untuk menduplikasi objek dalam suatu citra. Karena pemalsuan tersebut dilakukan dalam satu image, maka wilayah yang rusak hampir sama, sifat citra yang asli akan sulit diidentifikasi oleh manusia. (Salma Amtullah, Dr. Ajay Koul, 2014).

Gambar 1.1 adalah contoh image yang sudah dimanipulasi dari image aslinya

		
a	b	c

Gambar 1.1 Teknik Pemalsuan Citra Copy-Move

Gambar (a) merupakan image pada citra digital yang asli

Gambar (b) merupakan image pada citra digital yang sudah dimanipulasi.

Gambar (c) merupakan wilayah image yang telah ditandai

Dengan adanya bidang ilmu Forensik citra digital, akan membantu para penegak hukum, intelijen, investigasi swasta, dan media. Semakin majunya teknologi image pada saat ini mengangkat isu-isu baru dan tantangan dalam menentukan keaslian image pada citra digital. Forensik citra digital merupakan salah satu metode ilmiah pada bidang penelitian yang bertujuan untuk mendapatkan fakta-fakta pembuktian dalam menentukan keaslian image pada citra digital. Pada tulisan ini dijelaskan dalam menganalisa suatu image pada citra digital pada aplikasi matlab menggunakan Algoritma SIFT (Scale Invariant feature tranform), untuk menggunakan aplikasi tersebut, diharapkan dapat menghasilkan analisa image yang lebih baik.

Algoritma SIFT (Scale Invariant feature tranform) adalah algoritma dalam visi komputer untuk mendeteksi dan menggambarkan fitur lokal pada image. Algoritma ini dipatenkan di Kanada oleh University of British Columbia dan diterbitkan oleh David Lowe pada tahun 1999. Algoritma ini dapat melakukan pengenalan objek, pemetaan robot

dan navigasi, image stitching, pemodelan 3D, pengenalan isyarat, pelacakan video, mengidentifikasi satwa liar (Lowe, 1999). Pada penelitian yang lain, Algoritma SIFT adalah (Scale Invariant Feature Transform) yang digunakan untuk mencocokkan gambar berdasarkan fitur keypoin utama (invarian skala dan rotasi). Algoritma SIFT adalah salah satu metode ekstraksi fitur yang paling banyak digunakan. Algoritma Sift digunakan untuk menemukan titik-titik kunci pada image, dalam metode ini termasuk deskripsi sift dan sift deskriptor (Anantharaj, 2014). Dengan banyaknya kasus kejahatan terhadap manipulasi image, diharapkan metode yang peneliti gunakan, dapat menganalisa image pada citra digital sehingga dapat mendeteksi image yang sudah dirubah dari citra digital aslinya.

1.2 Permasalahan

Berdasarkan latar belakang yang sudah dijelaskan, maka terdapat permasalahan-permasalahan yang terjadi, diantaranya : meningkatnya kebutuhan tools atau aplikasi yang dapat digunakan untuk memanipulasi, mengedit ataupun memperbaiki kualitas image, berpotensi juga terhadap terjadinya kejahatan dengan maksud dan tujuan tertentu, yang tentunya merugikan banyak pihak. Sehingga diperlukannya suatu tools / aplikasi, dalam hal ini menggunakan code matlab, untuk melakukan forensik atau analisa image untuk dapat mendeteksi keaslian image. Fokus penelitian ini adalah bagaimana dua sumber image ini, yang asli dan sudah di manipulasi setelah dianalisis dapat menghasilkan output yang berbeda, sehingga mendapatkan letak keypoint skala dan rotasi dan di posisi mana saja image telah dimanipulasi pada titik - titik piksel tertentu.

1.3 Rumusan Masalah

Salah satu masalah mendasar teknik forensik citra digital mencoba untuk memecahkan atau mengidentifikasi image pada citra digital, yang dihasilkan dari kamera digital, scanner, cam recorder, dll. Untuk mengetahui lebih lanjut tentang pengaruh analisis image pada citra digital berdasarkan Algoritma SIFT, maka hal yang akan dilakukan dalam penelitian ini adalah :

a. Bagaimana menerapkan Algoritma SIFT (Scale Invariant feature tranform) dan histogram warna RGB dengan menggunakan kode matlab untuk mendapatkan kecocokan objek pada citra digital.
b. Bagaimana menguji keakuratan dalam mengidentifikasi image pada citra digital menggunakan Algoritma SIFT (Scale Invariant feature tranform).

1.4 Batasan Masalah

Batasan masalah pada penelitian ini mengacu kepada :

a. Hanya melakukan analisis image pada citra digital berekstensi .JPG.
b. Tidak menggunakan informasi exif (*Exchangeable Image File*) file pada citra yang di analisis, istilah exif sering digunakan pada fotografi untuk mengetahui informasi penting image pada pada citra digital.
c. Pengujian dalam melakukan analisa pendeteksian pencocokan objek pada citra digital dilakukan hanya pada satu algoritma saja.

1.5 Tujuan Penelitian

Penelitian tentang metode ganda yang digunakan pada image pada citra digital memiliki tujuan, diantaranya :

a. Menerapkan Algoritma SIFT (Scale Invariant feature tranform) dan histogram warna RGB dengan menggunakan aplikasi Matlab untuk dapat mendeteksi kecocokan objek pada citra digital.
b. Melakukan pengujian keakuratan yang lebih baik dalam menganalisa image pada citra digital yang sudah diubah dari aslinya.

1.6 Manfaat Penelitian

Manfaat yang diharapkan dari hasil analisa Algoritma SIFT (Scale Invariant feature tranform) adalah :

a. Untuk Pengembangan Ilmu.
    - Memberikan kemudahan dalam melakukan pendeteksian pencocokan objek pada citra digital, sehingga dapat menghasilkan informasi yang akurat dari image tersebut.
    - Mengetahui kelebihan dan kekurangan fungsi dari metode Algoritma SIFT (Scale Invariant feature tranform) dalam melakukan analisis image pada citra digital.
b. Untuk Penelitian lain
    - Dapat dijadikan regerensi untuk penelitian lain dalam mengkaji penelitian yang sama atau bisa menjadikan wawasan untuk pengembangan penelitian selanjutnya.
    - Diharapkan dapat mengembangkan pengajaran ilmu di bidang Forensik Citra Digital.

c. Untuk Penulis

- Diharapkan dapat menambah wawasan khususnya keilmuan di bidang forensik image pada citra digital.

d. Untuk Penyidik Forensik

Diharapkan dapat membantu dalam melakukan penyidikan tentang informasi pendeteksian perubahan image pada citra digital yang didapat setelah dilakukan penelitian.

1.7 Review Penelitian

Tabel 1.1 Review Penelitian Sebelumnya

No.	Peneliti / Tahun	Domain Penelitian	Pendekatan / Metode Penelitian	Hasil Penelitian
1	(Lin et al., 2013)	Deteksi kecocokan objek pada citra digital berdasarkan copy-move, splicing dan retouching	Metode deteksi kecocokan objek pada citra digital menggunakan Algoritma a colour image specific, berdasarkan pada pola sensor noise yang diekstraksi, metode ini diklaim kuat terhadap transformasi geometrik, seperti rotasi dan penskalaan, noise dan kompresi JPEG.	Mendeteksi kecocokan objek pada citra digital di daerah yang di splicing, retouching dan kompresi JPEG.
2	(Hassaballah, Abdelmgeid, & Alshazly, 2016)	Deteksi kecocokan objek pada citra digital berdasarkan fitur deskriptor dan fitur ekstrasi.	Metode deteksi kecocokan objek pada citra digital menggunakan gabungan Algoritma SIFT fitur deskriptor dan Algoritma SURF fitur ekstrasi, mendeteksi daerah keypoint yang sifatnya kovarian ke kelas transformasi. Robustness, dapat mendeteksi lokasi lokasi yang sama. Repeatability, mendeteksi fitur objek yang sama secara berulang.	Mendeteksi kecocokan objek pada citra digital di daerah yang di skala, noise, ekstrasi

No.	Peneliti / Tahun	Domain Penelitian	Pendekatan / Metode Penelitian	Hasil Penelitian
			Accuracy, mendeteksi fitur secara akurat melokalkan fitur gambar (lokasi piksel yang sama), terutama untuk pencocokan gambar, di mana tepat korespondensi diperlukan untuk memperkirakan geometri epipolar.	
3	(Bhullar, Budhiraja, & Dhindsa, 2014)	Deteksi kecocokan objek pada citra digital berdasarkan pada perubahan Copy-Move	Metode deteksi kecocokan objek pada citra digital menggunakan Algoritma DiscreteWavelet Transform (DWT) dan Scale Invariant Feature transform (SIFT) yang digabungkan, diekstrak dan dikelompokkan menjadi cluster menggunakan salah satu metode linkage (median, centroid atau lingkungan).	Mendeteksi kecocokan objek pada citra digital pada daerah yang di refleksi, rotasi, skala, kompresi, noise
4	(Susan Oommen & M, 2015)	Deteksi kecocokan objek pada citra digital berdasarkan pada hybrid Copy-Move Regional Similarity Indices	Metode deteksi kecocokan objek pada citra digital, melibatkan berbagai aktivitas seperti pemalsuan copy-move, image slicing, retouching, morphing (perubahan) menggunakan	Mendeteksi kecocokan objek pada citra digital menggunakan dimensi fraktal lokal untuk segmentasi tekstur blok yang hilang.

No.	Peneliti / Tahun	Domain Penelitian	Pendekatan / Metode Penelitian	Hasil Penelitian
			Regional Similarity Indices.	
5	(Ushma, Scholar, & Shanavas, 2014)	Deteksi kecocokan objek pada citra digital berdasarkan pada peningkatan citra digital	Metode deteksi kecocokan objek pada citra digital menggunakan peningkatan gambar dan deteksi tepi.	Mendeteksi kecocokan objek pada citra digital dengan memanipulasi piksel dan mengurangi kualitas noise pada sebuah gambar
6	(Inoue, Hara, & Urahama, 2017)	Deteksi kecocokan objek pada citra digital berdasarkan pada Histogram Warna RGB	Metode deteksi kecocokan objek pada citra digital dengan menghitung histogram target berdasarkan geometri ruang warna RGB (red, hijau dan biru)	Mendeteksi kecocokan objek pada citra digital pada peningkatan warna dan memperbaiki saturasi warna.
7	(Salahat & Qasaimeh, 2017)	Deteksi kecocokan objek pada citra digital berdasarkan fitur yang diekstraksi pada transformasi gambar	Metode deteksi kecocokan objek pada citra digital menggunakan gabungan Algoritma MSER dan SIFT.	Mendeteksi kecocokan objek pada citra digital pada kualitas fitur yang diekstraksi, rotasi, penskalaan dan affine.
8	(Korman, Reichman, Tsur, & Avidan, 2017)	Deteksi kecocokan objek pada citra digital berdasarkan transformasi affin 2D	Metode deteksi kecocokan objek pada citra digital menggunakan Algoritma Fast-Match, yang dilakukan dengan cara ekstraksi dari gambar dan	Mendeteksi kecocokan objek pada citra digital pada rotasi, skala atau transformasi affin.

No.	Peneliti / Tahun	Domain Penelitian	Pendekatan / Metode Penelitian	Hasil Penelitian
			dicocokkan kembali dan dicocokkan dengan yang lain.	
9	(Lindeberg, 2015)	Deteksi kecocokan objek pada citra digital berdasarkan deteksi fitur, seleksi skala, skala penghubung, pengenalan objek, relevansi skala dan afinitas, skala ruang	Metode deteksi kecocokan objek pada citra digital menggunakan gabungan Algoritma SIFT dan Algoritma SURF.	Mendeteksi kecocokan objek pada citra digital pada skala, pengenalan objek, deteksi fitur, rotasi, Gaus-SIFT Deskriptor, Gaus-SURF Deskriptor
10	(Guan, You, & Angeles, 2013)	Deteksi kecocokan objek pada citra digital berdasarkan	Metode deteksi kecocokan objek pada citra digital menggunakan variasi iluminasi yang besar, dengan memanfaatkan piksel tepi yang memiliki nilai gradien cukup besar untuk memastikan piksel ini mungkin muncul di semua gambar.	Mendeteksi kecocokan objek pada citra digital pada besarnya variasi iluminasi, kecocokan keypoint, tepi piksel dengan nilai gradien.
11	(Ghosh, Pandey, & Pati, 2015)	Deteksi kecocokan objek pada citra digital berdasarkan	Metode deteksi kecocokan objek pada citra digital menggunakan Algoritma Harris, SURF (Speeded-Up Robust Features), FAST (Features from	Mendeteksi kecocokan objek pada citra digital pada deteksi tepi piksel, rotasi, sudut, gumpalan, persimpangan

No.	Peneliti / Tahun	Domain Penelitian	Pendekatan / Metode Penelitian	Hasil Penelitian
			accelerated segment), dan FREAK (Fast Retina Key Point), setelah dilakukan perbandingan pada empat algoritma tersebut, dalam hal akurasi deteksi fitur dan waktu, Algoritma FREAK lebih unggul.	
12	Yang diusulkan peneliti	Bagaimana mengimplementasikan metode Algoritma SIFT dengan penambahan parameter lain pada sejumlah piksel berdasarkan histogram warna RGB yang diharapkan mendapatkan akurasi pendeteksi kecocokan objek pada citra digital yang lebih baik. Menggunakan metode Algoritma SIFT dan histogram warna RGB. Metode Algoritma SIFT ini yaitu bekerja dengan mencari sejumlah titik keypoint yang sama yang diduga ada kesamaan objek pada citra digital, serta menghitung sejumlah piksel pada citra digital menggunakan metode histogram warna RGB. Metode ini dapat mendeteksi perubahan skala, rotasi dan refleksi, serta perubahan peningkatan warna pada citra digital.		

1.8 Metodologi Penelitian

Dalam menyelesaikan penelitian, dilakukan secara sistematis, dengan tahapan - tahapan metodologi sebagai berikut :

1. Studi Pustaka

   Penelitian ini dilakukan dengan melakukan studi kepustakaan, dengan mengumpulkan beberapa bahan referensi yang terkait dengan penelitian, baik melalui buku, artikel, paper, jurnal, makalah, dan mengunjungi beberapa situs yang terdapat pada internet terkait dengan *image forensik, citra digital, deteksi keaslian image, algoritma processing image, forgery image, copy move, tampering image, citra digital, rotation, scala, keypoint* khususnya algoritma yang dapat mendeteksi kecocokan objek pada citra digital.

2. Persiapan Alat dan Bahan Penelitian

   Tahapan ini melakukan persiapan *tools* yang digunakan dalam melakukan analisis pendeteksi kecocokan image menggunakan MATLAB. Image yang didapat dari hasil kamera dan internet di olah menggunakan Photoshop.

3. Pengembangan Sistem

   Membangun sistem untuk mendeteksi kecocokan objek pada citra digital, menggunakan Algoritma SIFT, dengan mencari dan mencocokan keypoint yang didapat pada image yang akan dianalisis.

4. Implementasi Sistem

   Implementasi adalah proses untuk menggunakan metode algoritma SIFT sebagai dasar untuk mendeteksi kecocokan objek pada citra digital, yang diharapkan digunakan untuk mempermudah user / pengguna, dalam hal ini adalah investigator.

5. Hasil dan Pembahasan

   Pada tahapan ini dilakukan dengan tujuan untuk mepresentasikan hasil dan pembahasan dari hasil yang sudah dicapai dari metode yang sudah diterapkan.

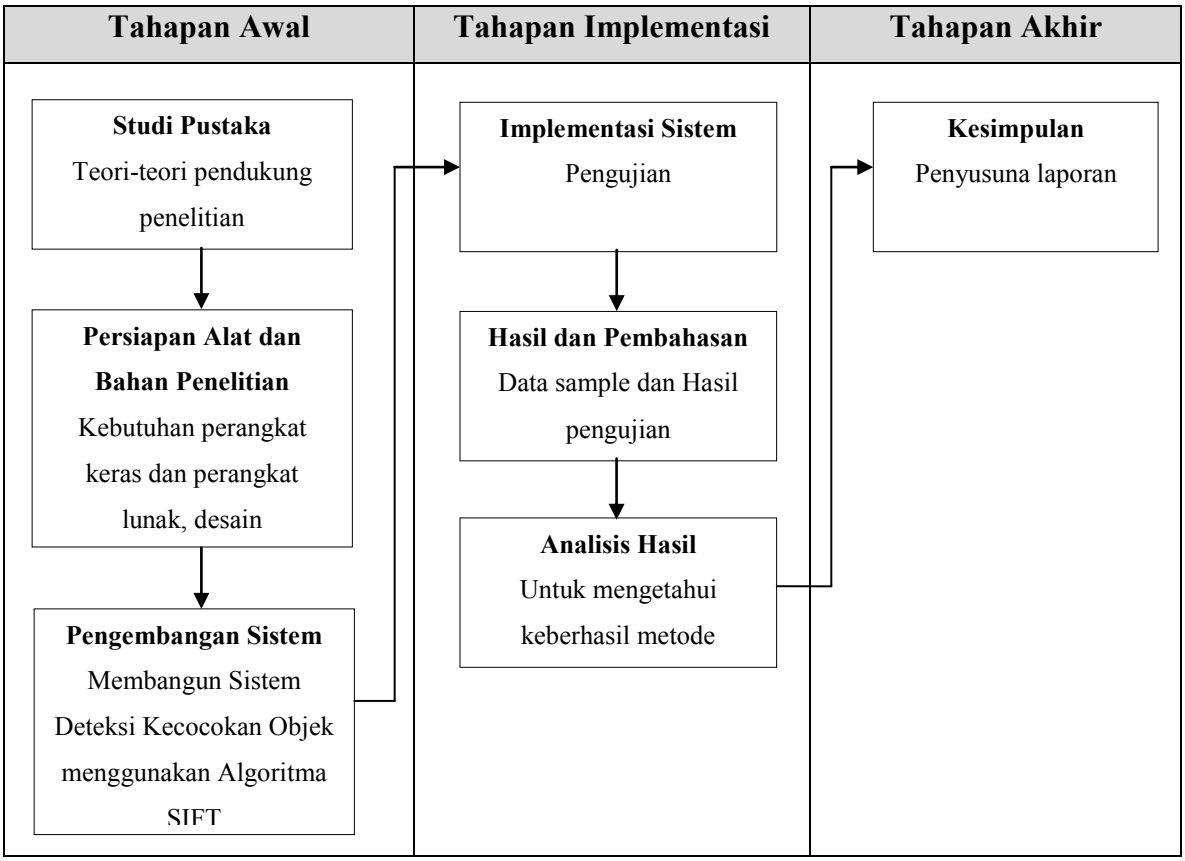
6. Analisis Hasil

   Untuk mengetahui keberhasilan dalam menerapkan metode Algoritma SIFT yang digunakan.

7. Kesimpulan

   Tahapan akhir yaitu penyampaian kesimpulan atas hasil dari penelitian ini.

Pada Gambar 1.2 berikut ini ditampilkan bagan tahapan - tahapan metode penelitian menggunakan Algoritma SIFT.



Gambar 1.2 Tahapan Metode Penelitian Deteksi Kecocokan Objek

1.9 Sistematika Penulisan

Dalam penyusunan penelitian ini, sistematika penulisan terbagi dalam beberapa bab yaitu :

BAB I PENDAHULUAN

Pendahuluan, merupakan pengantar terhadap permasalahan yang akan dibahas. Di dalamnya menguraikan tentang gambaran suatu penelitian yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada Bab ini menjelaskan teori-teori yang digunakan untuk memecahkan masalah dalam penelitian ini.

BAB III METODOLOGI PENELITIAN

Bab ini membahas tentang langlah-langkah penelitian, kebutuhan perangkat keras dan perangkat lunak yang akan digunakan serta perancangan antarmuka aplikasi yang akan dibuat.

BAB IV IMPLEMENTASI DAN ANALISA

Hasil dan Pembahasan, berisi tentang implementasi dari algoritma yang digunakan ke dalam aplikasi yang dibangun serta pengujian terhadap hasil enkripsi yang berupa chiper image terhadap berbagai percobaan serangan.

BAB V KESIMPULAN DAN SARAN

Simpulan dan Saran, memuat kesimpulan-kesimpulan dari hasil penelitian dan saran-saran yang perlu diperhatikan berdasar keterbatasan yang ditemukan dan asumsi-asumsi yang dibuat selama melakukan penelitian dan juga rekomendasi yang dibuat untuk pengembangan penelitian selanjutnya.

BAB 2
Tinjauan Pustaka

2.1 Pengertian Citra Digital

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra terbagi 2 yaitu ada citra yang bersifat analog dan ada citra yang bersifat digital. Citra analog adalah citra yang bersifat kontinu seperti gambar pada monitor televisi, foto sinar X, hasil CT Scan dll. Sedangkan pada citra digital adalah citra yang dapat diolah oleh komputer (T,Sutoyo : 2009).

Sebuah citra digital dapat mewakili oleh sebuah matriks yang terdiri dari M kolom N baris, dimana perpotongan antara kolom dan baris disebut piksel (piksel = picture element), yaitu elemen terkecil dari sebuah citra. Piksel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah f(x,y), yaitu besar intensitas atau warna dari piksel di titik itu. Oleh sebab itu, sebuah citra digital dapat ditulis dalam bentuk matriks berikut.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & \cdots & \cdots & f(1,M-1) \\ \cdots & \cdots & \cdots & \cdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \quad (2.1)$$

Berdasarkan gambaran tersebut, secara matematis citra digital dapat dituliskan sebagai fungsi intensitas f (x,y), dimana harga x (baris) dan y (kolom) merupakan koordinat posisi dan f(x,y) adalah nilai fungsi pada setiap titik (x,y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari piksel di titik tersebut. Pada proses digitalisasi (sampling dan kuantitas) diperoleh besar baris M dan kolom N hingga citra membentuk matriks ***M x N*** dan jumlah tingkat keabuan piksel G (T, Sutoyo *et al*.2009: 20).

Pengolahan citra digital adalah sebuah disiplin ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar (peningkatan kontras, transformasi warna, restorasi citra), transformasi gambar (rotasi, translasi, skala, transformasigeometrik), melakukan pemilihan citra ciri (*feature images)* yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan

data, transmisi data, dan waktu proses data. Input dari pengoalahan citra adalah citra, sedangkan outputnya adalah citra hasil pengolahan (T. Sutoyo : 2009).

2.2 Resolusi Citra

Resolusi citra merupakan tingkat detailnya suatu citra. Semakin tinggi resolusinya semakin tinggi pula tingkat detail dari citra tersebut (D.Putra : 2010). Menurut T. Sutoyo :2009) ada dua jenis resolusi yang perlu diketahui, yaitu :

1. Resolusi Spasial Resolusi spasial ini merupakan ukuran halus atau kasarnya pembagian kisi-kisi baris dan kolom pada saat sampling.Resolusi ini dipakai untuk menentukan jumlah pixel per satuan panjang. Biasanya satuan resolusi ini adalah dpi (*dot per inchi*). Resolusi ini sangat berpengaruh pada detail dan perhitungan gambar.
2. Resolusi kecemerlangan

   Resolusi kecemerlangan (intensitas/ brightness) atau biasanya disebut dengan kedalaman bit/ kedalaman warna (*Bit Depth*) adalah ukuran halus kasarnya pembagian tingkat gradasi warna saat dilakukan kuantisasi. *Bit Depth* menentukan berapa banyak informasi warna yang tersedia untuk ditampilkan dalam setiap piksel. Semakin besar nilanya, semakin bagus kualitas gambar yang dihasilkan dan tentu ukuran juga semakin besar. Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar (RGB = *Red Green Blue*).

   Setiap warna dasar menggunakan penyimpanan 8 bit = 1 byte, yang berarti mempunyai gradasi sebanyak 255 warna berarti setiap piksel mempunyai kombinasi warna sebanyak .Penyimpanan citra true color didalam memori berbeda dengan citra grayscale.Setiap piksel dari citra grayscale 256 gradasi warna diwakili oleh 1 byte. Sedangkan 1 piksel citra true color diwakili oleh 3 byte yang masing- masing byte merepresentasikan warna merah (*Red*), hijau (*Green*), biru (*Blue*) (T. Sutoyo: 2009).

2.3 Citra warna (24 bit)

Setiap pixel dari citra warna 24 bit diwakili dengan 24 bit sehingga total 16.777.216 variasi warna. Variasi ini sudah lebih dari cukup untuk memvisualisasikan seluruh warna yang dapat dilihat penglihatan manusia.Penglihatan manusia dipercaya hanya dapat membedakan hingga 10 juta warna saja.Setiap poin informasi pixel (RGB) disimpan kedalam 1 byte data. 8 bit pertama menyimpan nilai biru, kemudian diikuti dengan nilai hijau pada 8 bit kedua dan 8 bit terakhir merupakan warna merah.

2.4 Format File Citra

Sebuah format file citra harus dapat menyatukan kualitas citra, ukuran file dan kompabilitas dengan berbagai aplikasi. Format file citra standar yang digunakan saatni terdiri dari beberapa jenis. Format- format ini digunakan untuk menyimpan citra dalam sebuah file. Setiap format memiliki karakteristik masing- masing. Ini adalah contoh format umum, yaitu :*Bitmap (.bmp), tagged image format (.tif, .tiff), PortablNetwork Graphics (.png), JPEG (.jpg),* dll (D, Putra. 2010 : 58).

Bahkan menurut Sutoyo,T.Mulyanto,E.*et al* (2009 : 25), ada dua jenis format file citra yang sering digunakan dalam pengolahan citra, yaitu citra *bitmap* dan citra*vektor*. Pada citra *bitmap* ini sering disebut juga citra *raster*. Citra *bitmap* ini menyimpan data kode citra secara digital dan lengkap (cara penyimpanannya adalah per piksel). Citra *bitmap* ini dipresentasikan dalam bentuk matriks atau dipetakan dengan menggunakan bilangan biner atau sistem bilangan yang lain. Citra ini memiliki kelebihan untuk memanipulasi warna, tetapi untuk mengubah objek lebih sulit.Tampilan *bitmap* mampu menunjukkan kehalusan gradasi bayangan dan warna dari sebuah gambar. Tetapi bila tampilan diperbesar maka tampilan di monitor akan tampak pecah-pecah (kualitas citra menurun). Contoh format file citra antara lain adalah BMP, GIFF, TIF, JPG, dll. Sedangkan pada format file citra *vektor* merupakan citra vektor yang dihasilkan dari perhitungan matematis dan tidak terdapat piksel, yaitu data yang tersimpan dalam bentuk vektor posisi, dimana yang tersimpan hanya informasi vektor posisi dengan bentuk sebuah fungsi. Pada citra *vektor*, mengubah warna lebih sulit dilakukan, tetapi membentuk objek dengan cara mengubah nilai lebih mudah. Oleh karena itu, bila citra diperbesar atau diperkecil, kualitas citra relatif tetap baik dan tidak berubah.Citra *vektor* biasanya dibuat menggunakan aplikasi-aplikasi citra vektor seperti CorelDRAW, Adobe Illustrator, Macromedia Freehand, Autocad, dll.

2.5 Image Forensic

Image telah menjadi pembawa informasi utama di era digital. Memiliki potensi yang ekspresif pada media visual dan kemudahan dalam akuisisi, penyebaran dan penyimpanan image sangat mudah, baik melalui media penyimpanan internal atau perangkat storage penyimpanan offline, maupun penyimpanan melalui cloud system atau internet, sangat mudah untuk diakses kembali, sehingga image yang merupakan media informasi dapat dieksploitasi untuk menyampaikan informasi yang positif maupun negatif. image mewakili sumber bukti digital yang sama, baik dalam kontroversi kehidupan sehari-hari maupun

dalam persidangan. Video paling sederhana dalam berita TV biasanya diterima sebagai sertifikasi kebenaran berita yang dilaporkan.

Ahli disain pengolah image/designer image dapat dengan mudah mengakses dan memodifikasi konten image, tanpa meninggalkan jejak yang bisa dideteksi secara visual. Apalagi dengan penyebarannya alat pengeditan/tools editing image yang murah dan mudah digunakan, seni merusak dan memalsukan visual konten tidak lagi terbatas pada para ahli. Sebagai konsekuensinya, modifikasi gambar untuk tujuan jahat saat ini lebih sering digunakan. Digital Image Forensics adalah salah satu cabang keamanan multimedia yang bertujuan untuk membedakan dan mengekspos manipulasi image oleh pelaku kejahatan.

Digital image forensics (DIF) bertujuan menyediakan alat untuk mendukung penyelidikan. Watermarking and Steganography dapat memanfaatkan alat pengolahan dan analisis gambar untuk memulihkan informasi tentang riwayat gambar. Dua jalur penelitian utama berkembang dengan nama Digital Image Forensics. Dengan melakukan analisis untuk mengidentifikasi perangkat untuk mengambil foto/image, atau setidaknya untuk menentukan perangkat mana yang tidak mengcapture foto/image. Digital Image Forensics merupakan topik yang menarik bagi para peneliti.

2.5.1 Peran forensik citra digital dalam keamanan multimedia

Digital Image Forensics adalah disiplin ilmu yang cukup baru, meskipun demikian, disiplin ilmu tersebut terkait erat dengan sejumlah bidang penelitian yang berbeda. DIF mewarisi tujuan dan sikapnya dari sains forensik klasik (analog) dan dari bidang forensik komputer yang lebih baru. Disiplin forensik secara umum bertujuan untuk mengungkapkan bukti kejahatan. Untuk melakukannya, mereka harus menghadapi kemampuan pelaku kejahatan baik dalam menyembunyikan atau mungkin memalsukan jejak mereka.

Dalam citra digital baik proses akuisisi maupun teknik manipulasi image cenderung meninggalkan jejak yang sulit. Tugas ahli forensik adalah untuk mengekspos jejak ini dengan memanfaatkan pengetahuan yang ada mengenai mekanisme pencitraan digital, dibantu oleh hasil penelitian keamanan multimedia yang terkonsolidasi. Untuk mendapatkan hasil yang lebih baik oleh penyidik image forensik, dapat mengeksplorasi hubungan antara DIF dan disiplin berorientasi multimedia lainnya. Forensik pengolahan citra mendapatkan beberapa tantangan teknis terhadap watermarking dan steganography.

Digital watermarking dapat menyembunyikan tanda atau pesan dalam gambar terhadap hak ciptanya. Ada tiga kompleks trade-off antara tiga parameter yang saling bertentangan (payload, robustness, dan invisibility). Singkatnya, algoritma yang baik

dalam watermarking digital harus menyembunyikan cukup banyak bit tanpa memodifikasi secara signifikan dan harus dapat memulihkan pesan meskipun gambar mengalami beberapa perubahan. Aplikasi khusus untuk watermarking adalah perlindungan integritas gambar. Dalam hal ini, watermark yang rapuh diaplikasikan pada gambar sampul sehingga bisa hancur saat terjadi gangguan. Ini memastikan beberapa kontrol pada manipulasi konten gambar. Dalam konteks ini, watermark digital dapat dilihat sebagai proteksi aktif, sedangkan alat bantu forensik gambar digital dapat dilihat sebagai yang pasif.

Namun, dalam banyak skenario tidaklah realistis untuk membayangkan bahwa gambar dan video telah dilindungi oleh watermark yang rapuh sebelum diseminasi. Ini memberi tanda plus alat forensik citra untuk mendeteksi gangguan. Bagaimanapun, dalam kedua kasus tersebut, alat pengolah gambar menganalisis frekuensi tinggi untuk memulihkan watermark atau untuk mendeteksi ketidak konsistenan pada pola noise dan perkiraan, misalnya terkait dengan proses akuisisi. Steganografi terdiri dari komunikasi tersembunyi melalui beberapa media (khususnya gambar dan video). Pilihan sampulnya tidak terlalu penting disini. Juga, kita dapat mengasumsikan bahwa gambar stego tidak akan mengalami serangan fotometrik atau geometrik di antara transmisi. Poin utama untuk dua orang yang mengkomunikasikan beberapa informasi menggunakan teknologi ini adalah tidak terdeteksi oleh pihak ketiga. Agar pesan tidak terdeteksi, algoritma mencampur informasi rahasia dalam frekuensi tinggi dengan noise lain yang ada, misalnya terkait dengan sensor. Dapat dilakukan pengamatan sampai batas tertentu, semua teknik ini bekerja dengan frekuensi tinggi untuk menambahkan, memulihkan, mendeteksi.

2.5.2 Mengidentifikasi penggunaan perangkat dari image

Dalam menelusuri sumber image, mengidentifikasi perangkat yang digunakan untuk akuisisi merupakan kepentingan utama. Di pengadilan, asal gambar tertentu dapat mewakili bukti penting, keabsahan bukti ini mungkin dapat menghilangkan oleh keraguan bahwa image tersebut belum diambil dari perangkat yang diklaim / seharusnya diakuisisi, seperti dalam kasus materi video-surveillance atau video terselubung. Petunjuk yang dapat ditemukan perangkat pada sumber citra digital, ada pada header file (EXIF), atau dengan memeriksa konsistensi watermark. Namun, karena informasi ini dapat dengan mudah dimodifikasi atau dihapus, namun tidak selalu digunakan untuk keperluan forensik. Sebagai konsekuensinya, blind teknik lebih disukai untuk identifikasi perangkat akuisisi.

Teknik forensik image blind memanfaatkan jejak yang ditinggalkan, berbeda dalam langkah pengolahan image dalam fase akuisisi dan penyimpanan image. Jejak ini menandai

image dengan beberapa jenis sidik jari kamera, yang bisa digunakan untuk otentikasi. Teknik yang dilakukan mengambil informasi pada perangkat sumber image pada dua tahap yang berbeda. Pada tahap pertama, mencoba membedakan antara model kamera yang berbeda. Pada tahap kedua, lebih informatif meski lebih menantang, tujuannya adalah membedakan perangkat tunggal, bahkan contoh yang berbeda dari model kamera yang sama. Untuk memberi pemahaman yang lebih baik tentang teknik ini, dimulai dengan ilustrasi langkah yang paling umum dalam proses akuisisi dan penyimpanan image, sehingga menghasilkan asal usul artefak image dan cara memanfaatkannya dalam autentikasi image.

2.6 Algoritma SIFT (Scale Invariant feature tranform)

Scale-invariant feature transform (SIFT) adalah algoritma dalam visi komputer untuk mendeteksi dan menggambarkan fitur lokal pada gambar. Algoritma ini dipatenkan di Kanada oleh University of British Columbia dan dibuat oleh David Lowe pada tahun 1999. Aplikasi pada algoritma tersebut meliputi pengenalan objek, pemetaan robot dan navigasi, pemodelan 3D, pengenalan isyarat, pelacakan video, identifikasi satwa liar dan pemindahan gambar yang sama.

2.6.1 Deskriptor gambar lokal

Deskriptor keypoint dibuat dengan menghitung besarnya gradien - gradien dan orientasi pada setiap titik sampel gambar di suatu wilayah di sekitar titik fokus, seperti yang ditunjukkan gambar 2.2 sebelah kiri. Ini ditimbang oleh jendela Gaussian, ditunjukkan oleh lingkaran yang dilapisi. Sampel ini kemudian diakumulasikan menjadi histogram orientasi yang merangkum isinya di atas subregional 4x4, seperti yang ditunjukkan gambar 2.2 di sebelah kanan, dengan panjang setiap panah yang sesuai dengan jumlah magnitudo gradien di dekat arah tersebut di wilayah ini. Angka ini menunjukkan susunan deskriptor 2x2 yang dihitung dari kumpulan sampel 8x8, sedangkan percobaan dalam penelitian ini menggunakan deskriptor 4x4 yang dihitung dari sampel array 16x16.

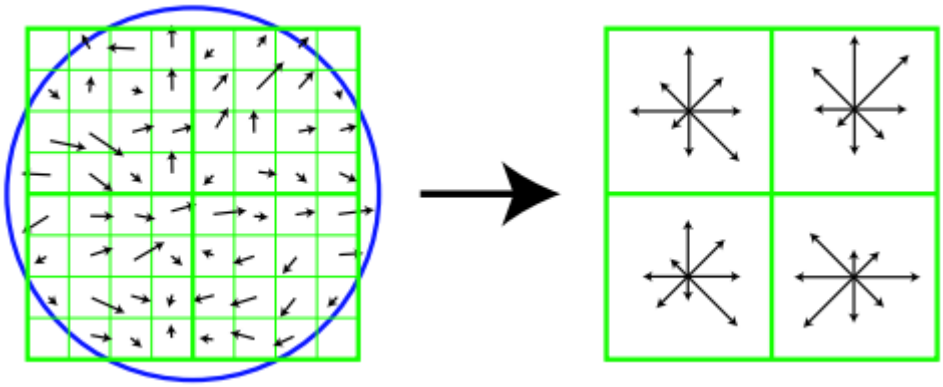
Setelah masing-masing keypoint diberikan orientasi, tahap berikutnya adalah menghitung ciri yang benar - benar membedakan suatu citra seperti perubahan intensitas cahaya atau sudut pandang 3D. Dari perhitungan besar gradient magnitude dan sudut orientasi diperoleh daerah cakupan yang dibatasi oleh jendela Gaussian, direpresentasikan dalam bentuk lingkaran. Kemudian akan dibangun histogram orientasi yang

merepresentasikan vektor ciri berukuran 16x16 yang kemudian akan dibagi ke dalam 4x4 blok. Pada tiap bloknya terdapat 8 arah gradient dengan panjang anak panah yang beragam sesuai dengan besar nilai dari histogram asal. Sebelum digunakan untuk menentukan descriptor, nilai magnitude akan dikonvolusi dengan fungsi Gaussian (σ = 1.5). Histogram yang akan dibentuk berukuran 8 bin yang mewakili 8 arah gradient, sehingga vektor ciri yang akan terbentuk nantinya berukuran 4x4x8 = 128. Rumus untuk menghitung besarnya gradien dan orientasi sekitar keypoint pada skala yang sesuai :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

$$m(x, y) = \sqrt{\big(L(x+1, y) - L(x-1, y)\big)^2 + \big(L(x, y+1) - L(x, y-1)\big)^2} \qquad (2.2)$$

$$\emptyset(x, y) = tan^{-1}\Big(\big(L(x, y+1) - L(x, y-1)\big)/\big(L(x+1, y) - (x-1, y)\big)\Big)$$



Gambar 2.1 Image Gradient dan Deskriptor Keypoint

Setelah melalui keempat tahapan di atas, akan didapatkan citra dengan keypoint yang invarian terhadap berbagai macam perubahan seperti perubahan skala, rotasi, contrast, dan intensitas cahaya. Keypoint tersebut yang akan digunakan untuk menguji kecocokan antara citra dalam database dengan citra uji.

2.6.2 Representasi deskriptor

Gambar 2.2 menggambarkan perhitungan deskriptor keypoint. Pertama, magnitude dan orientasi gradien citra diambil sampel di sekitar lokasi pengujian, dengan menggunakan skala keypoint untuk memilih tingkat kejernihan Gaussian pada image. Untuk mencapai invarian orientasi, koordinat deskriptor dan orientasi gradien diputar relatif terhadap orientasi keypoint. Untuk efisiensi, gradien didahului untuk semua tingkat piramida. Ini diilustrasikan dengan anak panah kecil di setiap lokasi sampel di sisi kiri Gambar 2.2.

Fungsi pembobotan Gaussian dengan σ sama dengan satu setengah lebar jendela deskriptor digunakan untuk menetapkan bobot pada besarnya masing-masing titik sampel. Ini diilustrasikan dengan jendela melingkar di sisi kiri Gambar 2.2, tentu saja, beratnya jatuh dengan mulus. Tujuan dari jendela Gaussian ini adalah untuk menghindari perubahan mendadak pada deskriptor dengan sedikit perubahan pada jendela posisi, dan memberi sedikit penekanan pada gradien yang jauh dari pusat deskriptor, karena ini paling terpengaruh oleh kesalahan misregistrasi.

Deskriptor keypoint ditunjukkan di sisi kanan Gambar 2.2. Hal ini memungkinkan untuk signifikan pergeseran posisi gradien dengan menciptakan histogram orientasi di atas 4x4 wilayah sampel. Angka tersebut menunjukkan delapan arah untuk setiap histogram orientasi, dengan panjang setiap panah sesuai dengan besarnya entri histogram tersebut. Sampel gradien di sebelah kiri dapat menggeser hingga 4 posisi sampel sambil tetap berkontribusi pada histogram yang sama pada sisi kanan, sehingga mencapai tujuan memungkinkan pergeseran posisi lokal yang lebih besar.

Penting untuk menghindari semua batasan yang mendasari deskriptor dengan tiba-tiba berubah saat sampel bergeser dengan lancar dari satu histogram ke posisi yang lain atau dari satu orientasi ke orientasi lainnya. Oleh karena itu, interpolasi trilinear digunakan untuk mendistribusikan nilai masing-masing sampel gradien ke tempat pembuangan histogram yang bersebelahan. Dengan kata lain, setiap entri ke bin dikalikan dengan bobot 1 - d untuk setiap dimensi, di mana d adalah jarak sampel dari nilai pusat bin yang diukur dalam satuan histogram bin spacing.

Deskriptor terbentuk dari vektor yang berisi nilai dari semua entri histogram orientasi, sesuai dengan panjang panah di sisi kanan Gambar 2.2. Gambar tersebut menunjukkan rangkaian histogram orientasi 2x2, sedangkan pada pengujian ini menunjukkan yang terbaik. Hasilnya dicapai dengan array histogram 4x4 dengan 8 tempat orientasi masing-masing. Oleh karena itu, percobaan dalam ini menggunakan vektor fitur elemen 4x4x8 = 128 pada masing-masing Inti.

Deskriptor SIFT yang invarian terhadap penskalaan, rotasi dan transformasi akan dihitung dengan menggunakan empat langkah utama berikut :

1. Scale Space Extrema Detection

   Fungsi, L (x, y, σ) didefinisikan sebagai skala ruang dari image yang dihasilkan oleh konvolusi fungsi Gaussian, G (x, y, σ) dan gambar masukan I (x, y) :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (2.3)$$

Dimana konvolusi, (x, y) adalah pixel koordinat dan σ adalah faktor ruang skala atau varian dari distribusi normal Gaussian. Untuk deteksi yang efisien terhadap keypoint yang stabil dan dapat diandalkan, fungsi DOG (Difference Of Gaussian), Dyang dihitung dengan menggabungkan perbedaan dua skala di dekatnya dipisahkan oleh faktor penskalaan konstan 'k' dengan gambar masukan.

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y)$$
$$= L(x, y, \sigma) - L(x, y, \sigma) \quad (2.4)$$

2. Keypoint Localization

Taylor expansion of scale- space function $D(x, y, \sigma)$ sehingga titik sampelnya adalah :

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \quad (2.5)$$

Untuk menentukan lokasi ekstremum, turunan w.r.t.diambil dan diset ke nol.

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}} \quad (2.6)$$

3. Orientation Assignment

Untuk mencapai invarian rotasi, masing-masing keypoint diberi orientasi. Untuk setiap sampel image Gaussian smoothed,$L(x, y)$,besarnya gradien,$m(x, y)$, dan orientasi,$0(x, y)$ dihitung dengan selisih piksel:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$
$$\theta(x, y) = \tan^{-1}\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)} \quad (2.7)$$

Arah gradien titik fitur dihitung dengan menggunakan histogram gradien berorientasi.Orientasi histogram puncak merupakan arah dominan gradien lokal.

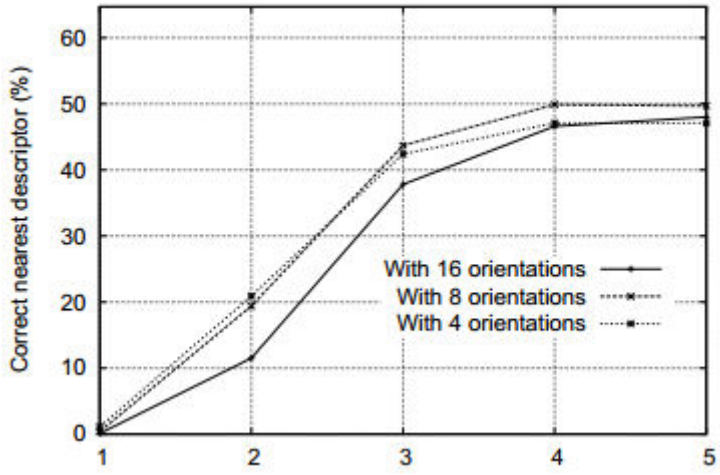
4. Keypoint Descriptor Generation

   Nilai histogram orientasi, dengan array $4x4$ dari histogram dan 8 orientasi menghasilkan $4x4x8 = 128$.

Akhirnya, vektor fitur dimodifikasi untuk mengurangi efek perubahan iluminasi. Pertama, vektor dinormalisasi menjadi satuan panjang. Perubahan dalam kontras gambar dimana setiap nilai piksel dikalikan dengan konstanta akan memperbanyak gradien dengan konstanta yang sama, jadi perubahan kontras ini akan dibatalkan oleh normalisasi vektor. Perubahan kecerahan di mana konstanta ditambahkan ke setiap piksel gambar tidak akan mempengaruhi nilai gradien, karena dihitung dari perbedaan piksel. Oleh karena itu, deskriptornya adalah invarian untuk affine perubahan iluminasi. Namun, perubahan iluminasi non linier juga bisa terjadi karena kejenuhan kamera atau karena perubahan iluminasi yang mempengaruhi permukaan 3D dengan orientasi yang berbeda dengan jumlah yang berbeda. Efek ini dapat menyebabkan perubahan besar dalam besaran relatif untuk beberapa gradien, namun cenderung mempengaruhi orientasi gradien. Oleh karena itu, mengurangi pengaruh magnitudo gradien besar dengan menetapkan nilai pada vektor fitur unit untuk masing-masing tidak lebih besar dari 0,2, kemudian renormalizing ke panjang unit. Ini berarti bahwa menyesuaikan besaran untuk gradien besar tidak lagi penting, dan bahwa distribusi orientasi memiliki penekanan lebih besar. Nilai 0,2 ditentukan secara eksperimen dengan menggunakan gambar yang mengandung iluminasi yang berbeda untuk objek 3D yang sama.

2.6.3 Pengujian Deskriptor

Ada dua parameter yang dapat digunakan untuk memvariasikan kompleksitas deskriptor: jumlah orientasi, r, histogram, dan lebar, n, dari n × n array histogram orientasi. Ukuran vektor deskriptor yang dihasilkan adalah rn2. Seiring kompleksitas deskriptor tumbuh, ia akan dapat melakukan diskriminasi lebih baik dalam database besar, namun juga akan lebih sensitif terhadap distorsi dan oklusi.



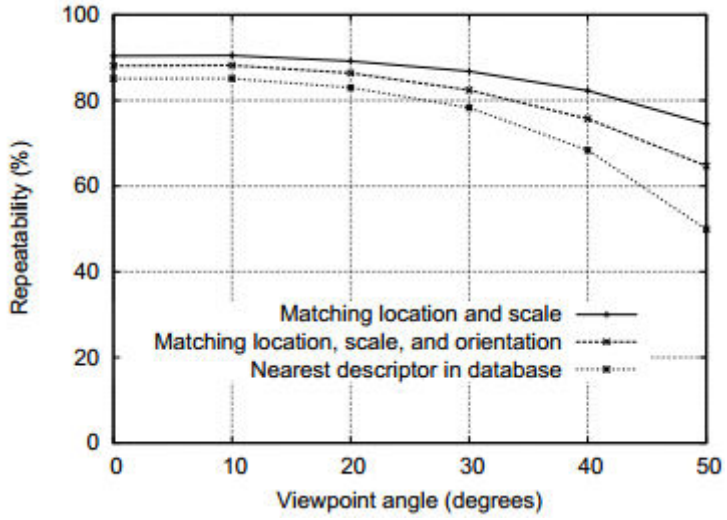
Gambar 2.2 Lebar n deskriptor (sudut 50 deg, noise 4%)

Pada gambar 2.3 menunjukkan hasil eksperimen dimana jumlah orientasi dan ukuran deskriptor bervariasi. Grafik dihasilkan untuk transformasi sudut pandang di mana permukaan planar dimiringkan 50 derajat dari penampil dan noise gambar 4% ditambahkan. Ini mendekati batas pencocokan yang andal, karena dalam kasus yang lebih sulit ini, kinerja deskriptor sangat penting. Hasilnya menunjukkan persentase titik kunci yang menemukan kecocokan yang benar dengan tetangga terdekat di antara database dengan 40.000 titik kunci. Grafik menunjukkan bahwa histogram orientasi tunggal (n = 1) sangat buruk dalam membedakan, namun hasilnya terus memperbaiki rangkaian histogram 4x4 dengan 8 orientasi. Setelah itu, menambahkan lebih banyak orientasi atau deskriptor yang lebih besar justru bisa melukai pencocokan dengan membuat deskriptor lebih sensitif terhadap distorsi. Hasil ini secara umum serupa untuk tingkat perubahan dan kebisingan sudut pandang lainnya, walaupun dalam beberapa kasus sederhana, diskriminasi terus membaik (dari tingkat yang sudah tinggi) dengan ukuran deskriptif 5x5 dan lebih tinggi. Sepanjang tulisan ini kami menggunakan deskriptor 4x4 dengan 8 orientasi, menghasilkan vektor fitur dengan 128 dimensi. Sementara dimensi deskriptor mungkin tampak tinggi, bahwa kinerjanya konsisten lebih baik daripada deskriptor dengan dimensi lebih rendah pada berbagai tugas yang sesuai dan bahwa biaya pencocokan komputasi tetap rendah bila menggunakan perkiraan metode tetangga terdekat.

2.6.4 Sensitivitas terhadap affine change

Sensitivitas deskriptor untuk perubahan affine diperiksa pada Gambar 2.4 Grafik menunjukkan reliabilitas dari pemilihan titik tunjuk dan pemilihan skala, penetapan orientasi, dan tetangga terdekat yang sesuai dengan database sebagai fungsi rotasi dalam

jarak jauh dari penampil. Dapat dilihat bahwa setiap tahap perhitungan telah mengurangi pengulangan dengan meningkatkan distorsi affine, namun akurasi pencocokan akhir tetap di atas 50% dari sudut pandang perubahan 50 derajat.



Gambar 2.3 Grafik stabilitas untuk menentukan arah dan orientas

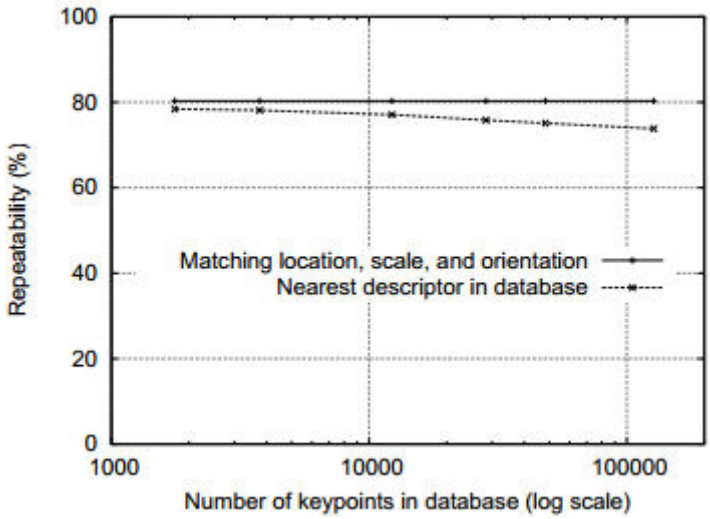
Grafik pada gambar 2.4, menunjukkan stabilitas pendeteksian untuk menentukan arah, orientasi, dengan database sebagai fungsi distorsi affine. Derajat distorsi affine dinyatakan dalam bentuk rotasi sudut pandang ekuivalen secara mendalam untuk permukaan planar. Untuk mencapai pencocokan yang andal di sudut sudut pandang yang lebih lebar, salah satu afinitas-invarian detektor dapat digunakan untuk memilih dan menentukan area gambar. Dalam metode yang paling mirip affine-invariant, Mikolajczyk (2002) telah mengusulkan dan menjalankan eksperimen terperinci dengan detektor Harris-affine. Dia menemukan bahwa pengulangan pengunciannya di bawah yang diberikan di sini sampai sekitar sudut pandang sudut pandang 50 derajat, namun kemudian mendekati pengulangan hingga 40% hingga sudut 70 derajat, yang memberikan kinerja yang lebih baik untuk perubahan affine yang ekstrem.

Kelemahannya adalah biaya komputasi yang jauh lebih tinggi, pengurangan jumlah titik kunci, dan stabilitas yang lebih buruk untuk perubahan affine kecil karena kesalahan dalam menetapkan kerangka affine yang konsisten di bawah noise (kebisingan). Dalam prakteknya, rentang rotasi yang diijinkan untuk objek 3D jauh lebih kecil daripada permukaan planar, sehingga invarian biasanya bukanlah faktor pembatas dalam kemampuan untuk mencocokkan perubahan sudut pandang. Jika berbagai macam invarian affine diinginkan, seperti untuk permukaan yang diketahui planar, maka solusi sederhana adalah mengadopsi pendekatan Pritchard dan Heidrich (2003) menjelaskan fitur SIFT

tambahan dihasilkan dari 4 versi affine transformed sesuai dengan 60 derajat sudut pandang perubahan. Hal ini memungkinkan penggunaan fitur SIFT standar tanpa biaya tambahan saat memproses gambar yang akan dikenali, namun menghasilkan peningkatan ukuran database fitur.

2.6.5 Cocok untuk database besar

Keistimewaan fitur SIFT adalah bagaimana keandalan pencocokan bervariasi sebagai fungsi dari jumlah fitur dalam database yang cocok. Pengujian dilakukan dengan menggunakan database dari 32 gambar dengan sekitar 40.000 titik kunci. Gambar 2.5 menunjukkan bagaimana reliabilitas pencocokan bervariasi sebagai fungsi dari ukuran basis data. Angka ini dihasilkan dengan menggunakan database yang lebih besar dari 112 gambar, dengan sudut pandang rotasi 30 derajat dan noise gambar 2% disamping rotasi gambar acak dan perubahan skala.



Gambar 2.4 Grafik kecocokan skala, lokasi dan orientasi descriptor database

Pada gambar 2.5 menunjukkan garis putus-putus yang menunjukkan persentase keypoint yang benar-benar sesuai dengan database, fungsi ukuran database (menggunakan skala logaritmik). Garis padat menunjukkan persentase titik kunci yang ditetapkan pada lokasi, skala, dan orientasi yang benar. Gambar memiliki skala acak dan perubahan rotasi, sebuah transformasi affine 30 derajat, dan noise gambar 2% ditambahkan sebelum pencocokan.

2.6.6 Keypoint untuk Aplikasi Pengenalan Objek

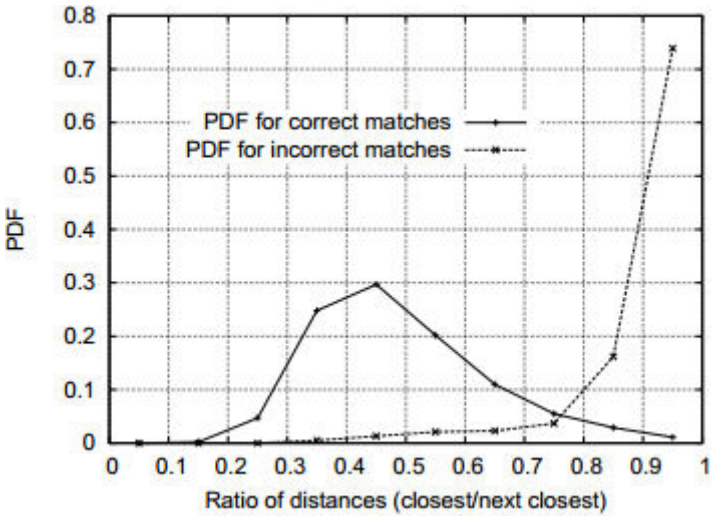
Pengenalan objek dilakukan dengan cara mencocokkan masing - masing keypoint secara independen ke database keypoint yang diambil dari sampel gambar. Banyak dari kecocokan awal ini akan salah karena fitur atau fitur ambigu yang muncul dari kekacauan

latar belakang. Oleh karena itu, kelompok dari setidaknya 3 fitur pertama kali diidentifikasi untuk menyetujui suatu objek dan posenya, karena kelompok ini memiliki probabilitas yang jauh lebih tinggi daripada yang ada pada masing-masing fitur. Kemudian, setiap cluster diperiksa dengan melakukan pemodelan geometrik yang rinci terhadap model, dan hasilnya digunakan untuk menerima atau menolak interprestasi. Berikut ini beberapa langkah yang dilakukan untuk keypoint dalam pengenalan objek :

1. Keypoint matching

   Calon kandidat terbaik untuk masing-masing keypoint ditemukan dengan mengidentifikasi tetangga terdekatnya di database titik temu dari sampel gambar. Tetangga terdekat didefinisikan sebagai titik penguraian dengan jarak Euclidean minimal untuk vektor deskriptor invarian. Namun, banyak fitur dari gambar tidak memiliki kecocokan yang benar dalam database pengujian ini, karena muncul dari kekacauan latar belakang atau tidak terdeteksi dalam sampel gambar. Oleh karena itu, akan sangat berguna untuk memiliki cara untuk membuang fitur yang tidak sesuai dengan database. Ambang global pada jarak ke fitur terdekat tidak berkinerja baik, karena beberapa deskriptor jauh lebih diskriminatif daripada yang lain. Ukuran yang lebih efektif diperoleh dengan membandingkan jarak tetangga terdekat dengan tetangga terdekat kedua. Jika ada beberapa dari sampel gambar dengaan objek yang sama, maka kita tetapkan tetangga terdekat kedua sebagai tetangga terdekat yang diketahui berasal dari objek yang berbeda dari yang pertama, seperti dengan hanya menggunakan gambar yang diketahui mengandung benda yang berbeda. Ukuran ini berkinerja baik karena cocok dan harus memiliki tetangga terdekat yang secara signifikan lebih dekat daripada kecocokan salah yang paling dekat untuk mencapai pencocokan yang andal.

   Pada gambar 2.6 menunjukkan nilai ukuran untuk data citra nyata. Kepadatan probabilitas fungsi untuk kecocokan yang benar dan salah ditunjukkan dalam rasio yang terdekat dengan tetangga terdekat kedua dari masing-masing keypoint. Untuk penerapan pengenalan objek kami, kami menolak semua kecocokan di mana rasio jarak lebih besar dari 0,8, yang menghilangkan 90% kecocokan salah saat membuang kurang dari 5% dari kecocokan yang benar. Angka ini dihasilkan dengan mencocokkan gambar dengan skala acak dan perubahan orientasi, rotasi kedalaman 30 derajat, dan penambahan noise gambar 2%, dengan database 40.000 titik kunci.



Gambar 2.5 Grafik perbandingan rasio PDF dengan rasio jarak

Gambar 2.6 menunjukan probabilitas kecocokan yang sudah valid dapat ditentukan dengan mengambil rasio jarak dari tetangga terdekat dengan jarak terdekat kedua. Dengan menggunakan database dari 40.000 titik kunci, garis solid menunjukkan rasio PDF untuk kecocokan yang benar, sementara garis putus-putus adalah untuk kecocokan yang salah.

2. Efisiensi pengindeksan tetangga terdekat

   Tidak ada algoritma yang dapat mengidentifikasi tetangga terdekat pada titik ruang berdimensi tinggi yang lebih efisien daripada pencarian yang lengkap. Deskriptif keypoint memiliki vektor fitur 128 dimensi, dan algoritma terbaik (Friedman et al., 1977) tidak memberikan percepatan pencarian yang lebih dari sekitar 10 dimensi spasi. Oleh karena itu, digunakanlah algoritma perkiraan, yang disebut algoritma Best-Bin-First (BBF) (Beis dan Lowe, 1997). Ini adalah perkiraan dalam arti bahwa ia mengembalikan tetangga terdekat dengan probabilitas tinggi.

   Algoritma BBF menggunakan perintah pencarian yang dimodifikasi untuk algoritma pohon k-d sehingga ruang fitur dicari sesuai urutan jarak terdekatnya dari lokasi kueri. Urutan pencarian prioritas ini pertama kali diperiksa oleh Arya dan Mount (1993), dan mereka memberikan studi lebih lanjut tentang sifat komputasi di (Arya et al., 1998). Urutan pencarian ini memerlukan penggunaan antrian prioritas berdasarkan tumpukan untuk penentuan urutan pencarian yang efisien. Jawaban perkiraan dapat dikembalikan dengan biaya rendah dengan memotong pencarian lebih lanjut setelah sejumlah tempat pembuangan terdekat telah dieksplorasi. Dalam pelaksanaannya, memotong pencarian setelah memeriksa 200 calon terdekat terdekat. Untuk database

dari 100.000 keypoints, ini melayani speedup untuk pencarian tetangga yang paling dekat dengan sekitar 2 urutan magnitude.

Salah satu alasan algoritma BBF bekerja dengan sangat baik untuk masalah ini adalah bahwa hanya mempertimbangkan kecocokan di mana tetangga terdekat kurang dari 0,8 kali jarak ke tetangga terdekat kedua (seperti yang dijelaskan pada bagian sebelumnya), dan oleh karena itu tidak perlu untuk benar-benar memecahkan kasus yang paling sulit di mana banyak tetangga berada pada jarak yang sangat mirip.

3. Clustering dengan hough transform

   Untuk memaksimalkan kinerja pengenalan objek untuk objek kecil, dilakukan identifikasi objek dengan jumlah sesedikit mungkin dari fitur yang cocok. Sebuah gambar khas berisi 2.000 atau lebih fitur yang mungkin berasal dari berbagai objek dan kekacauan latar belakang. Banyak metode pemasangan kuat yang terkenal, seperti RANSAC atau Least Median of Squares, berkinerja buruk ketika persen inliers turun jauh di bawah 50%. Untungnya, kinerja yang jauh lebih baik dapat diperoleh dengan mengelompokkan fitur dalam area pose menggunakan transformasi Hough (Hough, 1962; Ballard, 1981; Grimson 1990).

   Transformasi Hough mengidentifikasi kumpulan fitur dengan interpretasi yang konsisten menggunakan setiap fitur untuk memilih semua objek pose yang sesuai dengan fitur. Bila kumpulan fitur ditemukan untuk memilih pose objek yang sama, probabilitas penafsirannya jauh lebih tinggi daripada fitur tunggal. Masing-masing titik kunci menentukan 4 parameter: lokasi, skala, dan orientasi 2D, dan setiap titik kunci yang cocok di database memiliki catatan parameter keypoint dibandingkan dengan sampel gambar yang ditemukannya. Oleh karena itu, membuat entri transformasi Hough yang memprediksi model lokasi, orientasi, dan skala dari hipotesis perbandingan. Prediksi ini memiliki batas kesalahan yang besar, karena transformasi kesamaan yang diimplikasikan oleh 4 parameter ini hanyalah perkiraan terhadap 6 derajat kebebasan berpose ruang untuk objek 3D.

2.7 Teknik Deteksi Image Copy-Move menggunakan kesamaan region

Tujuan dari teknik ini adalah untuk mengembangkan metode perbaikan deteksi pemalsuan copy-move dengan menggunakan dimensi fraktal lokal untuk segmentasi citra dan memperkirakan SSIM pasangan blok di setiap wilayah yang tersegmentasi untuk melokalisasi wilayah yang ditempa. Langkah - langkah dari pendeteksian image copy move pada teknik ini adalah sebagai berikut :

1. Identifikasi bagian image dimana pemalsuan copy-move telah dilakukan.
2. Gunakan dimensi fraktal lokal untuk segmentasi tekstur citra yang efisien.
3. Mengembangkan metode deteksi yang kuat, kontaminasi kebisingan dan kompresi.

Prosedur dari pendeteksi image copy move ini dibagi menjadi tiga bagian :

1. Image yang dicurigai dibaca sebagai nilai intensitas piksel.
2. Membagi image menjadi blok yang tumpang tindih dengan ukuran yang sama.
3. Memperkirakan fraksi fraktal lokal ,setiap blok menggunakan metode penghitungan kotak diferensial.

2.8 Matlab

Matlab adalah alat untuk komputasi teknis, perhitungan dan visualisasi yang terintegrasi, misal nya : matematika dan perhitungan, pengembangan algoritma, akuisisi data, modeling, simulasi, dan prototyping, analisis data, eksplorasi, visualisasi, grafik ilmiah dan rekayasa. pengembangan aplikasi, termasuk pengembangan antarmuka pengguna grafis.MATLAB singkatan dari MATrix LABoratory, aplikasi ini cocok untuk manipulasi matriks dan pemecahan masalah yang berhubungan dengan Aljabar Linear.MATLAB menawarkan banyak Toolbox tambahan untuk berbagai area seperti Control Design, Image Processing, Digital Signal Processing, dll.MATLAB dikembangkan oleh The MathWorks (www.mathworks.com). MATLAB digunakan di seluruh dunia oleh para periset dan universitas (“Introduction to MATLAB,” 2016).

2.8.1. Variabel pada Matlab

Variabel didefinisikan dengan operator penugasan, "=". MATLAB diketik secara dinamis, artinya variabel dapat diberikan tanpa menyatakan jenisnya, dan jenisnya dapat berubah. Nilai bisa berasal dari konstanta, dari perhitungan yang melibatkan nilai variabel lain, atau dari keluaran suatu fungsi.

2.8.2. Matrik dan Vektor

Konsep dasar penggunaan vektor dan matriks pada MATLAB, adalah sebagai berikut :

1. Matriks dan sintak - sintak pada vektor
2. Fungsi matrik

MATLAB adalah "Matrix Laboratory", dan karena itu menyediakan banyak cara mudah untuk membuat vektor, matriks, dan array multi dimensi. Dalam MATLAB, sebuah

vektor mengacu pada matriks satu dimensi (1 × N atau N × 1), yang biasa disebut sebagai array dalam bahasa pemrograman lainnya. Sebuah matriks umumnya mengacu pada array 2 dimensi, yaitu array m × n dimana m dan n lebih besar dari atau sama dengan 1. Array dengan lebih dari dua dimensi disebut sebagai array multidimensional. Sebagian besar fungsi MATLAB dapat menerima matriks dan akan menerapkan dirinya pada setiap elemen. Sebagai contoh, mod (2 * J, n) akan mengalikan setiap elemen dalam "J" dengan 2, dan kemudian mengurangi setiap elemen modulo "n". Notasi vektor MATLAB sering menghasilkan kode yang lebih mudah dibaca dan lebih cepat dijalankan. Kode ini disajikan dari fungsi magic.m, menciptakan kotak ajaib M untuk nilai n :

```
[J,I] = meshgrid(1:n);
A = mod(I+J-(n+3)/2,n);
B = mod(I+2*J-2,n);
M = n*A + B + 1;
```
(2.14)

Tabel 2.1 Perintah fungsi matriks

Command	Keterangan
eye(x), eye(x,y)	Identitas matriks x
ones(x), ones(x,y)	Mattriks dengan fungsi ones
zeros(x), zeros(x,y)	Mattriks dengan fungsi zones
diag([x y z])	Diagonal matriks
size(A)	Dimensi matriks A
A’	Inverse matriks A

2.9 Skrip dan Fungsi - File M

2.9.1. Skrip pada file M

M-file adalah file teks yang berisi kode MATLAB. Gunakan Editor MATLAB atau editor teks lain untuk membuat file yang berisi pernyataan yang sama dengan yang di ketik pada baris perintah MATLAB. Simpan file dengan nama yang berakhiran ".m".

BAB 3
Metodologi Penelitian

3.1 Studi Pustaka

Studi Pustaka merupakan kegiatan untuk mempelajari literatur-literatur dan teori yang mendukung dalam melakukan penelitian ini. Studi Pustaka dilakukan untuk mendapatkan informasi mengenai topik penelitian yang terkait dengan *image forensik, citra digital, deteksi keaslian image, algoritma processing image, forgery image, copy move, tampering image, citra digital, rotation, scala, keypoint* yang dapat bersumber dari dokumen, buku, paper, jurnal, laporan penelitian, artikel, atau bahan tertulis lainnya yang berupa teori, baik bersifat *online source* maupun *offline source*.

3.2 Persiapan Alat dan Bahan Penelitian

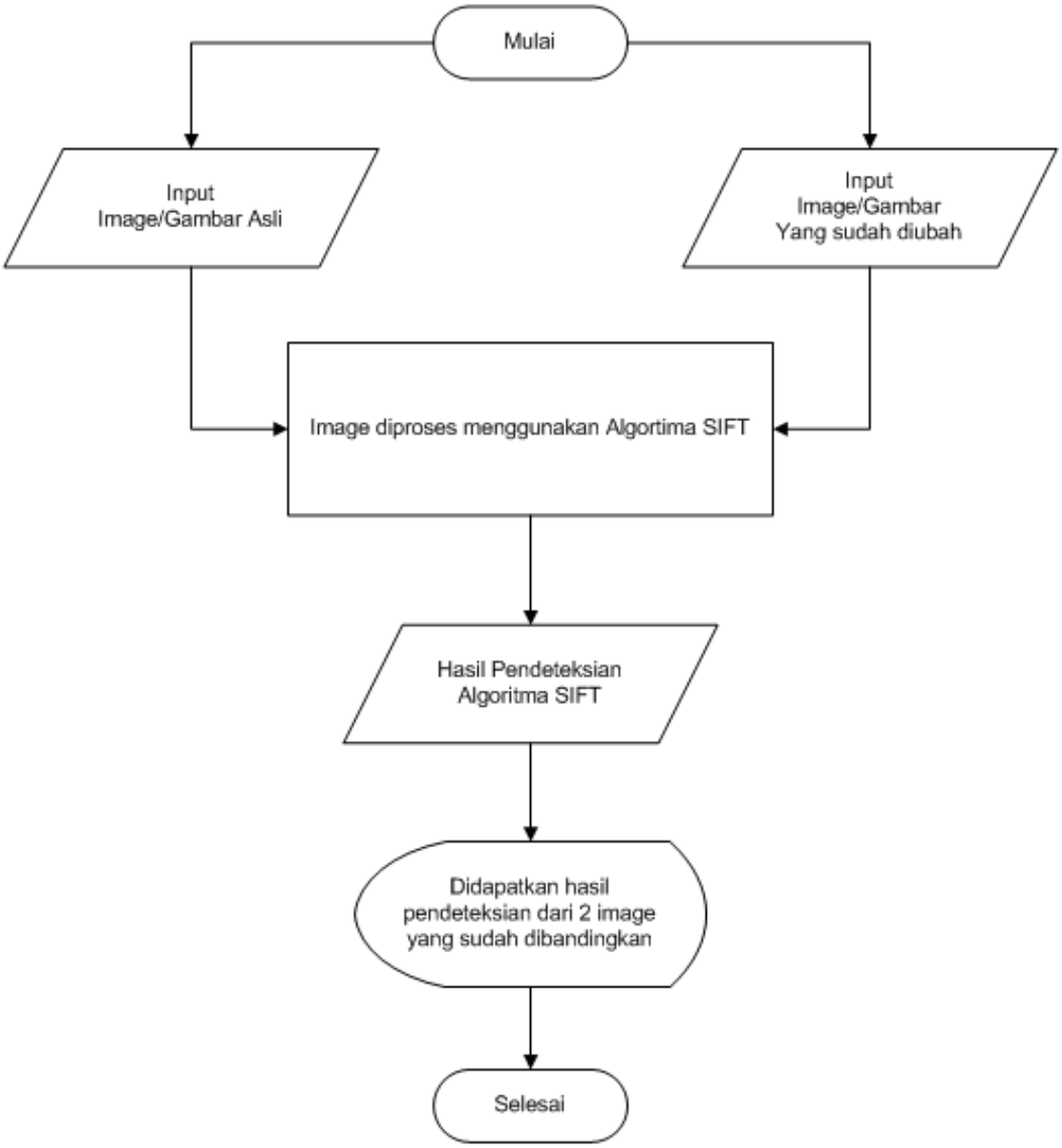
Untuk mendukung impementasi dalam penelitian ini diperlukan adanya perangkat keras dan perangkat lunak sebagai alat, berikut ini beberapa alat yang digunakan untuk melakukan penelitian:

1. Processor AMD E2 APU with AMD Radeon R2 Graphics CPU @1.50 GHz
2. RAM 6 GB
3. Harddisk 500 GB
4. Sistem Operasi Windows 7 Ultimate
5. Matlab R2015a yang berisi source code algoritma SIFT
6. Photoshop CS6 untuk manipulasi atau edit image

Adapun Bahan Penelitian yang digunakan adalah berupa beberapa image asli dan beberapa image yang sudah diubah dari aslinya, menggunakan format JPG / JPEG, baik berupa image warna maupun grayscale, dengan ukuran tidak terbatas pada piksel nya. Alasan kenapa menggunakan format jpg, karena umunya banya yang kita jumpai dengan format tersebut.

3.3 Pengembangan Sistem

Membangun sistem untuk mendeteksi kecocokan objek pada citra digital, menggunakan Algoritma SIFT, dengan mencari dan mencocokan keypoint yang didapat pada kedua image yang akan dianalisis, skenario yang dibuat yaitu dengan menyiapkan dua (2) file image, yang asli dan yang sudah di rubah dari aslinya. Pada *gambar 3.1* berikut ini diagram alir / flowchart dalam mendeteksi kecocokan keaslian image menggunakan Algoritma SIFT pada kode matlab.



Gambar 3.1 Alur Flow Cart Algoritma SIFT

3.4 Implementasi Sistem

Implementasi adalah proses untuk menggunakan metode algoritma SIFT sebagai dasar untuk mendeteksi kecocokan objek pada citra digital, yang diharapkan digunakan untuk mempermudah user / pengguna, dalam hal ini adalah investigator. Implementasi sistem ini dilakukan dengan menggunakan code matlab, kode matlab dibuat menggunakan Algoritma SIFT (Scale Invariant Feature Transformation, mampu mendeteksi skala dan rotasi beberapa perubahan image pada tiap pikselnya. Hasil yang didapat dari kedua image tersebut kemudian dilakukan perbandingan untuk mendapatkan perbedaan image yang asli dengan image yang sudah diubah pada hasil yang sudah didapat, dan akan diketahui dimana letak perubahan image tersebut.

3.4.1 Pengujian

Pengujian dilakukan dengan tujuan untuk mengetahui tingkat akurasi deteksi kecocokan objek pada citra digital dengan menggunakan metode algoritma SIFT.

Tabel 3.1 Rancangan Pengujian

<table>
<tr><th>Image Asli 1</th><th>Image Asli 2</th><th>Image Manipulasi</th><th>Hasil Pencocokan Image</th></tr>
<tr><td>... ... ...</td><td>... ... ...</td><td>... ... ...</td><td>... ... ...</td></tr>
<tr><td>Nilai Piksel :
... ...</td><td>Nilai Piksel :
... ...</td><td>Nilai Piksel :
... ...</td><td rowspan="2">Nilai Keypoint :
... ...
Penjelasan mengenai kecocokan objek yang didapat dari perbandingan image setelah dicari dan ditemukan keypoint</td></tr>
<tr><td colspan="3">Penjelasan mengenai perbandingan nilai piksel dari hasil image yang sudah dilakukan proses analisis</td></tr>
<tr><td colspan="4">Pada image berikut dilakukan pengujian kembali untuk kecocokan objek</td></tr>
<tr><td>... ... ...</td><td colspan="2">... ... ...</td><td>... ... ...</td></tr>
<tr><td>Penjelasan dari hasil temuan keypoint yang didapat, pada perbandingan potongan image yang lain dengan image yang asli</td><td colspan="2">Penjelasan dari hasil temuan keypoint yang didapat, pada perbandingan potongan image yang lain dengan image yang asli</td><td>Penjelasan mngenai kecocokan objek yang didapat, dari bagian potongan image asli dengan image yang sudah dimanipulasi</td></tr>
</table>

BAB 4
Implementasi dan Analisa

4.1 Implementasi

Pada penelitiaan ini menggunakan aplikasi matlab dengan metode Algoritma SIFT, dengan memanggil fungsi yang sudah dibuat pada file matlab. Implementasi inputan data yang akan diproses berupa dua file sampel image dengan format JPG/JPEG, sumber image didapat dari hasil kamera Smartphome EverCoss R40A, Smartphone Xiaomi Redmi 4 dan beberapa sampel image yang didapat dari internet. Dengan melakukan proses beberapa tahapan dan fungsi yang ada di kode matlab tersebut, dilakukan compile dan running dari aplikasi yang menghasilkan keypoint skala dan rotasi sehingga dapat berfungsi untuk mencari kecocokan keypoint yang sama pada masing-masing image.

4.2 Analisa Algoritma SIFT

Pencocokan image adalah aspek paling dasar dari banyak masalah dalam penglihatan komputer, fitur-fitur ini dapat mencocokan ketidak sesuaian skala dan rotasi image. Berikut ini adalah tahapan - tahapan utama perhitungan yang digunakan untuk menghasilkan kumpulan fitur image :

1. **Deteksi ekstraksi ruang skala :** Tahap pertama pencarian komputasi di semua skala dan lokasi image. Dengan menggunakan fungsi difference-of-Gaussian untuk mengidentifikasi titik minat potensial yang tidak sesuai dengan skala dan orientasi.
2. **Lokalisasi keypoint :** Ditentukan lokasi dan skala Keypoint dipilih berdasarkan ukuran stabilitasnya.
3. **Orientasi tugas :** Lebih berorientasi terhadap lokasi penguraian berdasarkan arah gradien. Semua operasi dapat dilakukan pada data citra yang telah ditransformasikan relatif terhadap orientasi, skala, danlokasi untuk setiap fitur nya, sehingga memberikan invarian ke transformasi ini.
4. **Deskriptor Keypoint:** Gradien image diukur pada skala yang dipilih di wilayah sekitar masing-masing keypoint. Ini ditransformasikan menjadi representasi yang dapat memungkinkan tingkat distorsi yang signifikan dan perubahan iluminasi.

4.3 Analisa Fungsi Matlab pada Algoritma SIFT

Pada source code matlab Algoritma SIFT, terdapat tahapan atau fungsi yang berpengaruh terhadap analisa image yang dilakukan, diantaranya :

1. imread, yang berfungsi untuk membaca file image dan format image, dari fungsi inilah image diinput.

```
image = imread(imageFile);
```

2. imresize (image, scala) mendapatkan nilai awal skala yang didapat dari inputan image, baik berupa grayscale, RGB maupun biner.

```
img=imresize(img,[row,colum]);
```

3. rgb2gray (rgb) mengubah image true color RGB ke image intensitas grayscale, dengan menghilangkan informasi hue dan saturation untuk mempertahankan kontras image.

```
img=rgb2gray(img);
```

4. penggabungan fungsi kron, pada array dan looping, untuk menghasilkan matrik yang terbentuk dan nilai elemen pada vektor untuk menentukan dimensi image pada setiap padding nya.

```
sigma0=sqrt(2);
octave=3;%6*sigma*k^(octave*level)<=min(m,n)/(2^(octave-2))
level=3;
D=cell(1,octave);
for i=1:octave
D(i)=mat2cell(zeros(row*2^(2-i)+2,colum*2^(2-i)+2,level),row*2^(2-
i)+2,colum*2^(2-i)+2,level);
end
```

5. mencari DOG (Difference Of Gaussian) map lokalisasi keypoint pada setiap pikselnya menggunakan fungsi interval, extrema, dan looping

```
for i=1:octave
    temp_D=D{i};
    for j=1:level
        scale=sigma0*sqrt(2)^(1/level)^((i-1)*level+j);
        p=(level)*(i-1);
        figure(1);
        subplot(octave,level,p+j);
        f=fspecial('gaussian',[1,floor(6*scale)],scale);
        L1=temp_img;
        if(i==1&&j==1)
        L2=conv2(temp_img,f,'same');
        L2=conv2(L2,f','same');
        temp_D(:,:,j)=L2-L1;
        imshow(uint8(255 * mat2gray(temp_D(:,:,j))));
        L1=L2;
        else
        L2=conv2(temp_img,f,'same');
        L2=conv2(L2,f','same');
        temp_D(:,:,j)=L2-L1;
        L1=L2;
        if(j==level)
            temp_img=L1(2:end-1,2:end-1);
        end
```

```
            imshow(uint8(255 * mat2gray(temp_D(:,:,j))));
            end
        end
        D{i}=temp_D;
        temp_img=temp_img(1:2:end,1:2:end);
        temp_img=padarray(temp_img,[1,1],'both','replicate');
    end
```

6. akurasi lokalisasi keypoint dengan menghilangkan point / titik terendah dari image

```
toc
tic
interval=level-1;
number=0;
```

7. multiple orientasi assignment

```
tic
kpori=zeros(1,36*extr_volume);
minor=zeros(1,36*extr_volume);
f=1;
flag=1;
for i=1:extr_volume
    %search in the certain scale
    scale=sigma0*sqrt(2)^(1/level)^((extrema(4*(i-1)+1)-
1)*level+(extrema(4*(i-1)+2)));
    width=2*round(3*1.5*scale);
    count=1;
    x=floor((extrema(4*(i-1)+3)-1)/(n/(2^(extrema(4*(i-1)+1)-2))))+1;
    y=mod((extrema(4*(i-1)+3)-1),m/(2^(extrema(4*(i-1)+1)-2)))+1;
    %make sure the point in the searchable area
    if(x>(width/2)&&y>(width/2)&&x<(m/2^(extrema(4*(i-1)+1)-2)-
width/2-2)&&y<(n/2^(extrema(4*(i-1)+1)-2)-width/2-2))
        rx=x+1;
        ry=y+1;
        rz=extrema(4*(i-1)+2);
        reg_volume=width*width;%3? thereom
        % make weight matrix
        weight=fspecial('gaussian',width,1.5*scale);
        %calculate region pixels' magnitude and region orientation
        reg_mag=zeros(1,count);
        reg_theta=zeros(1,count);
    for l=(rx-width/2):(rx+width/2-1)
        for k=(ry-width/2):(ry+width/2-1)
            reg_mag(count)=sqrt((D{extrema(4*(i-1)+1)}(l+1,k,rz)-
D{extrema(4*(i-1)+1)}(l-1,k,rz))^2+(D{extrema(4*(i-1)+1)}(l,k+1,rz)-
D{extrema(4*(i-1)+1)}(l,k-1,rz))^2);
            reg_theta(count)=atan2((D{extrema(4*(i-1)+1)}(l,k+1,rz)-
D{extrema(4*(i-1)+1)}(l,k-1,rz)),(D{extrema(4*(i-1)+1)}(l+1,k,rz)-
D{extrema(4*(i-1)+1)}(l-1,k,rz)))*(180/pi);
            count=count+1;
        end
    end
```

8. penggunaan histogram pada SIFT pengskalaan dan rotasi image

```
mag_counts=zeros(1,36);
    for x=0:10:359
        mag_count=0;
       for j=1:reg_volume
           c1=-180+x;
           c2=-171+x;
           if(c1<0||c2<0)
```

```
            if(abs(reg_theta(j))<abs(c1)&&abs(reg_theta(j))>=abs(c2))

mag_count=mag_count+reg_mag(j)*weight(ceil(j/width),mod(j-1,width)+1);
            end
            else

if(abs(reg_theta(j)>abs(c1)&&abs(reg_theta(j)<=abs(c2))))

mag_count=mag_count+reg_mag(j)*weight(ceil(j/width),mod(j-1,width)+1);
                end
            end
        end
           mag_counts(x/10+1)=mag_count;
    end
```

9. mencari nilai maksimal pada bar histogram dengan nilai prosentasi lebih dari 80%

```
[maxvm,~]=max(mag_counts);
     kori=find(mag_counts>=(0.8*maxvm));
     kori=(kori*10+(kori-1)*10)./2-180;
     kpori(f:(f+length(kori)-1))=kori;
     f=f+length(kori);
     temp_extrema=[extrema(4*(i-1)+1),extrema(4*(i-1)+2),extrema(4*(i-
1)+3),extrema(4*(i-1)+4)];

temp_extrema=padarray(temp_extrema,[0,length(temp_extrema)*(length(kor
i)-1)],'post','circular');
     long=length(temp_extrema);
     minor(flag:flag+long-1)=temp_extrema;
     flag=flag+long;
    end
end
idx= minor==0;
minor(idx)=[];
extrema=minor;
```

10. menghapus point - point yang tidak dapat dicari dan menambahkan point - point orientasi sehingga menghasilkan keypoint yang baru, dengan area piksel dimensi 4x4

```
idx= kpori==0;
kpori(idx)=[];
extr_volume=length(extrema)/4;
toc
```

11. deskriptor keypoint

```
descriptor=zeros(1,d*d*8);% feature dimension is 128=4*4*8;
descriptor((area-1)*8+1:area*8)=magcounts;
descriptor=normr(descriptor);
descriptor=normr(descriptor);
```

12. mendapatkan nilai x dan y untuk menentukan lokasi image yang dirotasi dan skala

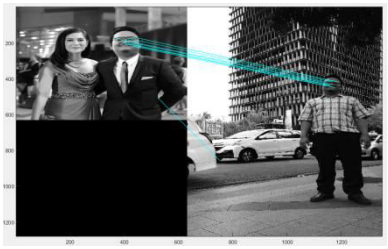
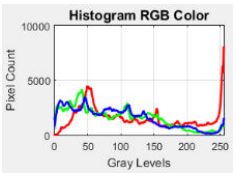
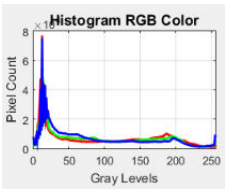
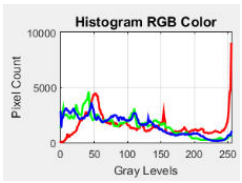
```
feature(:,i)=descriptor';
index=find(sum(feature));
feature=feature(:,index);
toc
```

Pengujian pada Algoritma SIFT melakukan pendeteksian kecocokan image sehingga menghasilkan output image yang sudah di proses oleh kode Matlab pada Algoritma SIFT. Image yang sudah diubah dari keadaan aslinya, dapat terlihat skala dan rotasi, yang akhirnya jika dibandingkan dari hasil kedua image tersebut, terdapat perbedaan dari image yang asli dengan image yang sudah diubah.

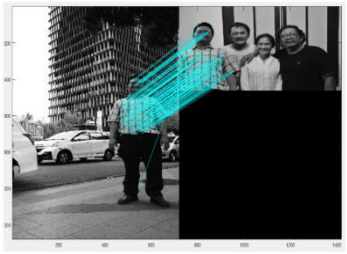
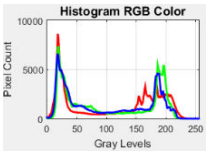
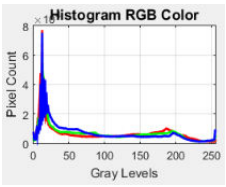
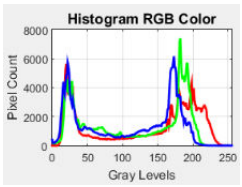
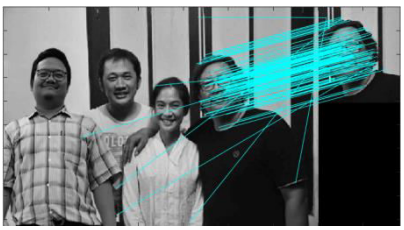
4.4 Hasil Pengujian

Untuk melalukan pengujian ini, penulis menggunakan aplikasi matlab, yang sudah disiapkan source code nya untuk mendeteksi kecocokan objek pada citra digital menggunakan Algoritma SIFT. Pada *tabel 4.1, 4.2, dan 4.3* berikut ini adalah sampel data image asli dan image yang sudah dimanipulasi. File - file tersebut sudah diolah menggunakan photohop CS6, yang disimpan dalam format JPG/JPEG, image-image tersebut diantaranya adalah : dian_sastrowardoyo, sampel, dian_edit, cover_film, agus, agus_oci, bertiga, cewek, cewek_meka, coba1, cover_film, dede_nagita, dian_edit, Image_01, image_01_original, insan_syahrini, jokowi_mark, jokowi_mark_edit, menara_eifel_paris, oci_malay, oci_paris, oshi, sampel, sampel1, trio, dede_nagita_palsu,dian_sastro, dian_sastro1, oshi_agus, dll. Metode Algoritma SIFT yang digunakan adalah fungsi yang dapat membaca dua image, menemukan fitur SIFT mereka, danmenampilkan garis yang menghubungkan keypoint yang cocok. Persamaan yang didapat pada image pertama hanya jika jaraknya kurang dari rasio kali jarak ke arah persamaan terdekat pada image kedua, kemudian mengembalikan jumlah kecocokan yang ditampilkan. Cara kerja Algoritma pada aplikasi ini adalah, setelah melakukan inputan dari dua image, tahap awal mencari *keypoint* pada tiap - tiap image, nilai konstanta di buat pada jarak rasio 0,6, ini dilakukan dengan harapan, cara kerja analisis dari matlab menjadi lebih ringan, descriptor pada image pertama, dicocokkan dengan deskriptor pada image kedua, matriks - matriks pada setiap image dihitung ulang, menghitung vektor pata setiap titik sudut, mengambil nilai inverse cosine dan menampilkan hasilnya, selain mendapatkan keypoint, parameter tambahan lain nya yaitu mendapatkan nilai piksel untuk memperkuat analisis dari image yang diteliti, hasil dari proses tersebut disajikan kedalam *Tabel 4.1, Tabel 4.2 dan Tabel 4.3* berikut ini :

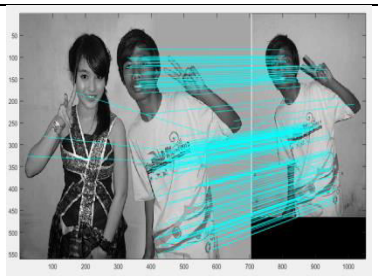
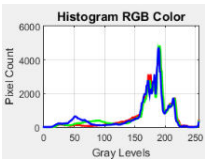
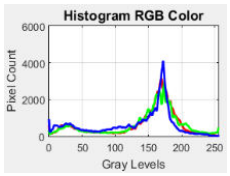
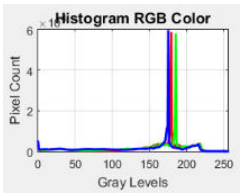
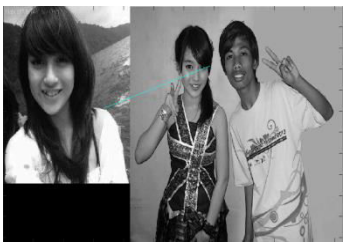
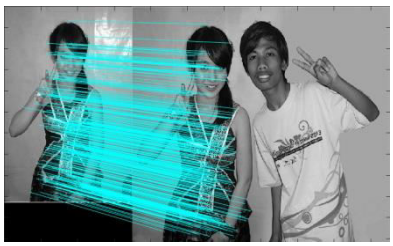
Tabel 4.1 Hasil Pencocokan Objek Pasangan Dian Sastro dengan Algoritma SIFT dan Histogram Warna RGB

<table>
<tr><th>Image Asli 1</th><th>Image Asli 2</th><th>Image Manipulasi</th><th>Hasil Pencocokan Image</th></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>Nilai Keypoint :
Finding keypoints...
1218 keypoints found.
Finding keypoints...
5149 keypoints found.
Found 12 matches.
Ditemukan kecocokan objek sebanyak 12 keypoint pada perbandingan image yang dimanipulasi dengan yang asli</td></tr>
<tr><td colspan="3">Pada Perbandingan tiga image diatas menghasilkan nilai histogram dengan jumlah piksel yang berbeda - beda, artinya nilai histogram tersebut bisa dijadikan acuan sebagai parameter tambahan untuk melakukan pencocokan objek.</td></tr>
<tr><td colspan="4">Pada image berikut dilakukan pengujian kembali untuk kecocokan objek</td></tr>
<tr><td> 
Finding keypoints...
1065 keypoints found.
Finding keypoints...
1218 keypoints found.
Found 0 matches.</td><td> 
Finding keypoints...
1027 keypoints found.
Finding keypoints...
1218 keypoints found.
Found 0 matches.</td><td> 
Finding keypoints...
524 keypoints found.
Finding keypoints...
1218 keypoints found.
Found 354 matches.</td></tr>
<tr><td>Ditemukan 0 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Dian Sastro yang lain</td><td>Ditemukan 0 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Dian Sastro yang lain</td><td>Terdapat kecocokan objek pada wajah Dian Sastro sebanyak 354 kecocokan keypoint, setelah mengambil bagian dari wajah dian sastro yang asli.</td></tr>
</table>

Tabel 4.2 Hasil Pencocokan Objek Pasangan All Artis dengan Algoritma SIFT dan Histogram Warna RGB

<table>
<tr><th>Image Asli 1</th><th>Image Asli 2</th><th>Image Manipulasi</th><th>Hasil Pencocokan Image</th></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>Nilai Keypoint :
Finding keypoints...
5149 keypoints found.
Finding keypoints...
1640 keypoints found.
Found 148 matches.
Ditemukan kecocokan objek sebanyak 148 keypoint pada perbandingan image yang dimanipulasi dengan yang asli.</td></tr>
<tr><td colspan="3">Pada Perbandingan tiga image diatas menghasilkan nilai histogram dengan jumlah piksel yang berbeda - beda, artinya nilai histogram tersebut bisa dijadikan acuan sebagai parameter tambahan untuk melakukan pencocokan objek.</td></tr>
<tr><td colspan="4">Pada image berikut dilakukan pengujian kembali untuk kecocokan objek</td></tr>
<tr><td> 
Finding keypoints...
1640 keypoints found.
Finding keypoints...
544 keypoints found.
Found 0 matches.</td><td> 
Finding keypoints...
1640 keypoints found.
Finding keypoints...
922 keypoints found.
Found 1 matches.</td><td> 
Finding keypoints...
1640 keypoints found.
Finding keypoints...
222 keypoints found.
Found 145 matches.</td><td></td></tr>
<tr><td>Ditemukan 0 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Didi Petet yang lain</td><td>Ditemukan 1 keypoint, tetapi tidak ada kecocokan objek setelah mengambil sampel image pada wajah Didi Petet yang lain</td><td>Terdapat kecocokan objek pada wajah Didi Petet sebanyak 145 kecocokan keypoint, setelah mengambil bagian dari wajah Didi Petet yang asli.</td><td></td></tr>
</table>

Tabel 4.3 Hasil Pencocokan Objek Pasangan Member Oshi Nabila JKT48 dengan Algoritma SIFT dan Histogram Warna RGB

<table>
<tr><th>Image Asli 1</th><th>Image Asli 2</th><th>Image Manipulasi</th><th>Hasil Pencocokan Image</th></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>Nilai Keypoint :
Finding keypoints...
2114 keypoints found.
Finding keypoints...
448 keypoints found.
Found 165 matches.
Ditemukan kecocokan objek sebanyak 165 keypoint pada perbandingan image yang dimanipulasi dengan yang asli</td></tr>
<tr><td colspan="3">Pada Perbandingan tiga image diatas menghasilkan nilai histogram dengan jumlah piksel yang berbeda - beda, artinya nilai histogram tersebut bisa dijadikan acuan sebagai parameter tambahan untuk melakukan pencocokan objek.</td></tr>
<tr><td colspan="4">Pada image berikut dilakukan pengujian kembali untuk kecocokan objek</td></tr>
<tr><td> 
Finding keypoints...
601 keypoints found.
Finding keypoints...
2114 keypoints found.
Found 1 matches.</td><td> 
Finding keypoints...
1752 keypoints found.
Finding keypoints...
2114 keypoints found.
Found 2 matches.</td><td> 
Finding keypoints...
797 keypoints found.
Finding keypoints...
2114 keypoints found.
Found 364 matches.</td></tr>
<tr><td>Hanya ditemukan 1 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Nabilah JKT48 yang lain</td><td>Hanya ditemukan 2 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Nabilah JKT48 yang lain</td><td>Terdapat kecocokan objek pada image Nabilah JKT48 sebanyak 364 kecocokan keypoint, setelah mengambil bagian dari image Nabilah JKT48 yang asli.</td></tr>
</table>

4.5 Analisis Hasil Pengujian

Pada hasil pengujian pada metode Algoritma SIFT, dilakukan pada satu set dari 16 (enam belas) image dari 5 (lima) image yang asli, 5 (lima) yang telah dimanipulasi menggunakan photosop CS6, dan 6 (enam) image pembanding lainnya. Image ini dipilih secara acak dari hasil kamera Smartphome EverCoss R40A, Smartphone Xiaomi Redmi 4 dan beberapa sampel image dari internet. Resolusi image terletak pada kisaran piksel ke piksel. Image yang dimanipulasi adalah hasil potongan atau croping dari image yang juga mengandung persegi atau segi empat yang ditempa melalui penyisipan berupa rotasi, copy-move, penskalaan (simetris atau asimetris) atau bahkan kombinasi dari ketiganya.

Penggunaan algoritma SIFT dipilih sebagai metode ekstraksi ciri karena metode ini invarian terhadap perubahan skala, rotasi, translasi, dan iluminasi. SIFT digunakan untuk memperoleh ciri dari pola keypoint yang didapatkan. Harapannya dengan menerapkan metode ini, dari keypoint yang didapat pada masing - masing image tersebut, mendapatkn kecocokan objek yang akurat. Tahap pertama dalam menentukan keypoint yang invarian terhadap perubahan skala pada image adalah mencari nilai ekstrim pada ruang skala. Untuk mendapatkan lokasi keypoint dalam suatu ruang skala secara efisien, digunakan fungsi Difference-of-Gaussian (DoG), nilai untuk fungsi DoG diperoleh dari selisih antara citra Gaussian dengan skala k berbeda.

Setiap keypoint yang tidak tereliminasi akan diberikan orientasi sehingga tidak akan terpengaruh dengan adanya rotasi pada citra. Pada penelitian ini, fitur-fitur berupa lokasi keypoint dan juga vektor ciri dari keypoint tersebut telah didapatkan. Kemudian klasifikasi pun telah dilakukan, sehingga dapat disimpulkan bahwa sistem ini mampu mendeteksi kecocokan objek pada citra digital secara akurat. Untuk memperkuat hasil analisa pada pengujian diatas, ditambahkan juga pengujian menggunakan grafik histogram warna RGB (Red, Green, Blue). Dimana tingkat perbedaan pada image bisa didapatkan dari perubahan jumlah piksel dan warna RGB yang ditampilkan berdasarkan grafik histogram.

BAB 5
Kesimpulan dan Saran

5.1 Kesimpulan

Dari hasil penelitian dan pengujian yang telah dilakukan, maka menghasilkan kesimpulan sebagai berikut :

1. Penerapkan Algoritma SIFT (Scale Invariant Feature transform) dan histogram warna RGB dalam menentukan kesamaan letak keypoint dan jumlah piksel pada image, untuk mendapatkan kecocokan objek pada citra digital pada matlab berhasil diimplementasikan.
2. Berdasarkan hasil pengujian didapatkan keakuratan dalam mengidentifikasi image pada citra digital menggunalan Algoritma SIFT (Scale Invarian Feature Transform), bahwa penelitian telah berjalan baik dan lancar.

5.2 Saran

Saran yang diberikan dari penelitian ini untuk penelitian selanjutnya sebagai berikut :

1. Penambahan pendeteksi kecocokan objek pada citra digital perlu ditingkatkan lagi, misalnya refleksi, JPEG Kompres, Blurring, dan lain - lain.
2. Untuk mendapatkan hasil yang lebih baik lagi dalam mendeteksi kecocokan objek pada citra digital, dapat dilakukan penggabungan dua algoritma atau lebih.

Daftar Pustaka

Amerini, I., Ballan, L., Caldelli, R., Del, A., Del, L., & Serra, G. (2013). Copy-Move Forgery Detection and Localization by Means of Robust Clustering with J-Linkage.

Anantharaj, N. (2014). Tampering and Copy-Move Forgery Detection Using Sift Feature. *International Journal of Innovative Research in Computer and Communication Engineering*, *2*(1), 2132–2137.

Bhullar, L. K., Budhiraja, S., & Dhindsa, A. (2014). DWT and SIFT Based Passive Copy-Move Forgery Detection. *International Journal of Computer Applications*, *95*(23), 14–18.

Fan, B., Kong, Q., Wang, X., Wang, Z., Xiang, S., Pan, C., & Fua, P. (2017). A Performance Evaluation of Local Features for Image Based 3D Reconstruction, 1–13. Retrieved from http://arxiv.org/abs/1712.05271

Ghosh, P., Pandey, A., & Pati, U. C. (2015). Comparison of Different Feature Detection Techniques for Image Mosaicing, (1), 1–7.

Guan, W., You, S., & Angeles, L. (2013). Robust Image Matching with Line Context. *Bmvc2013*, 1–11. https://doi.org/10.5244/C.27.34

Hassaballah, M., Abdelmgeid, A. A., & Alshazly, H. A. (2016). *Image Feature Detectors and Descriptors* (Vol. 630). https://doi.org/10.1007/978-3-319-28854-3

Inoue, K., Hara, K., & Urahama, K. (2017). RGB Color Cube-Based Histogram Specification for Hue-Preserving Color Image Enhancement. *Journal of Imaging*, *3*(3), 24. https://doi.org/10.3390/jimaging3030024

Introduction to MATLAB. (2016).

Jayanthi, N., & Indu, S. (n.d.). Comparison of Image Matching Techniques, (3), 396–401. https://doi.org/10.21172/1.73.552

Jolhip, M. I., Minoi, J., & Lim, T. (n.d.). A Comparative Analysis of Feature Detection and Matching Algorithms for Aerial Image Stitching, *9*(2), 2.

Korman, S., Reichman, D., Tsur, G., & Avidan, S. (2017). Fast-Match: Fast Affine Template Matching. *International Journal of Computer Vision*, *121*(1), 111–125. https://doi.org/10.1007/s11263-016-0926-1

Lin, W., Khan, S. U., Yow, K. C., Qazi, T., Madani, S. A., Xu, C.-Z., … Hayat, K. (2013). Survey on blind image forgery detection. *IET Image Processing*, *7*(7), 660–670. https://doi.org/10.1049/iet-ipr.2012.0388

Lindeberg, T. (2015). *Image Matching Using Generalized Scale-Space Interest Points*. *Journal of Mathematical Imaging and Vision* (Vol. 52). Springer US. https://doi.org/10.1007/s10851-014-0541-0

Lindeberg, T., & Science, C. (2014). Scale Selection, 701–713. https://doi.org/10.1007/978-0-387-31439-6.Scale

Manikandan, V. M., & Masilamani, V. (n.d.). Copy-Move Forgery Detection using Histogram Based Package Clustering and Sorted Consecutive Local Binary Patterns.

Miksik, O., & Mikolajczyk, K. (2012). Evaluation of Local Detectors and Descriptors for Fast Feature Matching. *Pattern Recognition (ICPR), 2012 21st International Conference on*, (Icpr), 2681–2684. https://doi.org/978-1-4673-2216-4

Putri, G., Agung, T., & Siti, S. (2015). Analisis dan Implementasi Scale Invariant Feature Transform (SIFT) pada Sistem Autentikasi Menggunakan Pembuluh Vena, *2*(1), 1353–1361.

Salahat, E., & Qasaimeh, M. (2017). Recent advances in features extraction and description algorithms: A comprehensive survey. *Proceedings of the IEEE*

International Conference on Industrial Technology, 1059–1063. https://doi.org/10.1109/ICIT.2017.7915508

Scholar, E. A. P. G., Assistant, S. D. M., & Vijayalakshmi, K. (2013). A Forensic Method for Detecting Image Forgery Using Codebook, *3*(3), 1–5.

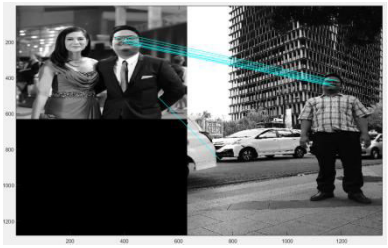
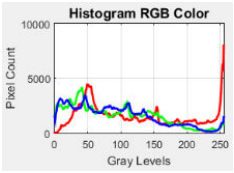
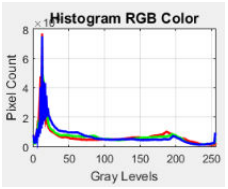
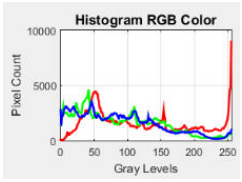
Susan Oommen, R., & M, ayamohan. (2015). A Hybrid Copy-Move Forgery Detection Technique Using Regional Similarity Indices. *International Journal of Computer Science and Information Technology*, *7*(4), 157–134. https://doi.org/10.5121/ijcsit.2015.7411

Trzcinski, T., Christoudias, M., & Lepetit, V. (n.d.). Learning Image Descriptors with Boosting, 1–14.

Ushma, A., Scholar, M., & Shanavas, P. A. R. M. (2014). Object Detection In Image Processing Using Edge Detection Techniques. *IOSR Journal of Engineering*, *4*(3), 10–13.

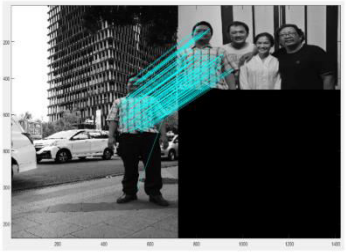
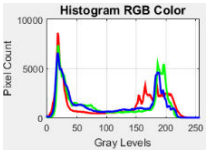
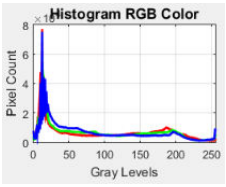
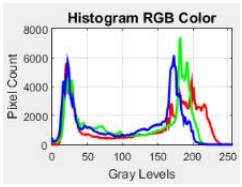
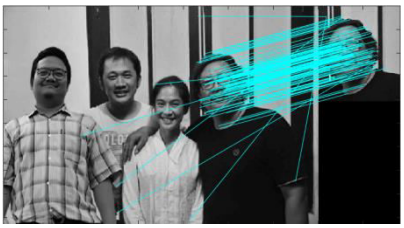
Lampiran 1 Hasil Pengujian Metode Algoritma SIFT

Tabel Hasil Pengujian Kecocokan Objek Pada Wajah

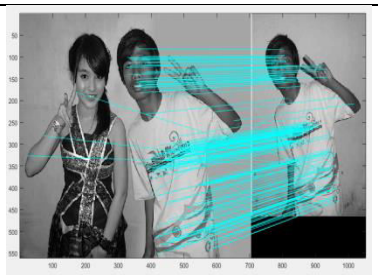
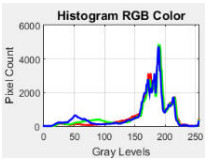
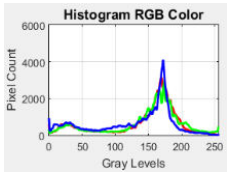
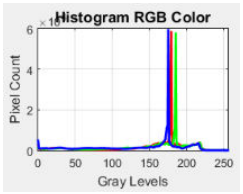
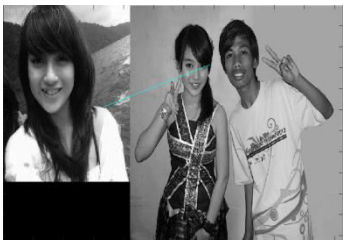
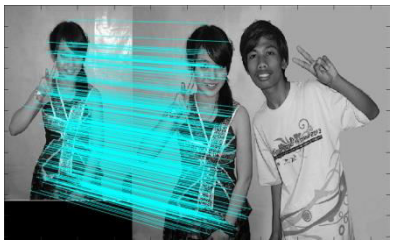
<table>
<tr><th>Image Asli 1</th><th>Image Asli 2</th><th>Image Manipulasi</th><th>Hasil Pencocokan Image</th></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>Nilai Keypoint :
Finding keypoints...
1218 keypoints found.
Finding keypoints...
5149 keypoints found.
Found 12 matches.

Ditemukan kecocokan objek sebanyak 12 keypoint pada perbandingan image yang dimanipulasi dengan yang asli</td></tr>
<tr><td colspan="3">Pada Perbandingan tiga image diatas menghasilkan nilai histogram dengan jumlah piksel yang berbeda - beda, artinya nilai histogram tersebut bisa dijadikan acuan sebagai parameter tambahan untuk melakukan pencocokan objek.</td></tr>
<tr><td colspan="4">Pada image berikut dilakukan pengujian kembali untuk kecocokan objek</td></tr>
<tr><td> 
Finding keypoints...
1065 keypoints found.
Finding keypoints...
1218 keypoints found.
Found 0 matches.</td><td> 
Finding keypoints...
1027 keypoints found.
Finding keypoints...
1218 keypoints found.
Found 0 matches.</td><td> 
Finding keypoints...
524 keypoints found.
Finding keypoints...
1218 keypoints found.
Found 354 matches.</td></tr>
<tr><td>Ditemukan 0 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Dian Sastro yang lain</td><td>Ditemukan 0 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Dian Sastro yang lain</td><td>Terdapat kecocokan objek pada wajah Dian Sastro sebanyak 354 kecocokan keypoint, setelah mengambil bagian dari wajah dian sastro yang asli.</td></tr>
</table>

Tabel Hasil Pengujian Kecocokan Objek Setengah Badan

<table>
<tr><th>Image Asli 1</th><th>Image Asli 2</th><th>Image Manipulasi</th><th>Hasil Pencocokan Image</th></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>Nilai Keypoint :
Finding keypoints...
5149 keypoints found.
Finding keypoints...
1640 keypoints found.
Found 148 matches.
Ditemukan kecocokan objek sebanyak 148 keypoint pada perbandingan image yang dimanipulasi dengan yang asli.</td></tr>
<tr><td colspan="3">Pada Perbandingan tiga image diatas menghasilkan nilai histogram dengan jumlah piksel yang berbeda - beda, artinya nilai histogram tersebut bisa dijadikan acuan sebagai parameter tambahan untuk melakukan pencocokan objek.</td></tr>
<tr><td colspan="4">Pada image berikut dilakukan pengujian kembali untuk kecocokan objek</td></tr>
<tr><td> 
Finding keypoints...
1640 keypoints found.
Finding keypoints...
544 keypoints found.
Found 0 matches.</td><td> 
Finding keypoints...
1640 keypoints found.
Finding keypoints...
922 keypoints found.
Found 1 matches.</td><td> 
Finding keypoints...
1640 keypoints found.
Finding keypoints...
222 keypoints found.
Found 145 matches.</td></tr>
<tr><td>Ditemukan 0 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Didi Petet yang lain</td><td>Ditemukan 1 keypoint, tetapi tidak ada kecocokan objek setelah mengambil sampel image pada wajah Didi Petet yang lain</td><td>Terdapat kecocokan objek pada wajah Didi Petet sebanyak 145 kecocokan keypoint, setelah mengambil bagian dari wajah Didi Petet yang asli.</td></tr>
</table>

Tabel Hasil Pengujian Kecocokan Objek Full Badan

<table>
<tr><th>Image Asli 1</th><th>Image Asli 2</th><th>Image Manipulasi</th><th>Hasil Pencocokan Image</th></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>HISTOGRAM RGB :
  </td><td>Nilai Keypoint :
Finding keypoints...
2114 keypoints found.
Finding keypoints...
448 keypoints found.
Found 165 matches.
Ditemukan kecocokan objek sebanyak 165 keypoint pada perbandingan image yang dimanipulasi dengan yang asli</td></tr>
<tr><td colspan="3">Pada Perbandingan tiga image diatas menghasilkan nilai histogram dengan jumlah piksel yang berbeda - beda, artinya nilai histogram tersebut bisa dijadikan acuan sebagai parameter tambahan untuk melakukan pencocokan objek.</td></tr>
<tr><td colspan="4">Pada image berikut dilakukan pengujian kembali untuk kecocokan objek</td></tr>
<tr><td> 
Finding keypoints...
601 keypoints found.
Finding keypoints...
2114 keypoints found.
Found 1 matches.</td><td> 
Finding keypoints...
1752 keypoints found.
Finding keypoints...
2114 keypoints found.
Found 2 matches.</td><td> 
Finding keypoints...
797 keypoints found.
Finding keypoints...
2114 keypoints found.
Found 364 matches.</td></tr>
<tr><td>Hanya ditemukan 1 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Nabilah JKT48 yang lain</td><td>Hanya ditemukan 2 keypoint, artinya tidak ada kecocokan objek setelah mengambil sampel image pada wajah Nabilah JKT48 yang lain</td><td>Terdapat kecocokan objek pada image Nabilah JKT48 sebanyak 364 kecocokan keypoint, setelah mengambil bagian dari image Nabilah JKT48 yang asli.</td></tr>
</table>

Lampiran 2 Source Code pada Aplikasi Matlab

match.mat

```
% num = match(image1, image2)
%
% Fungsi ini untuk membaca 2 image, mencari fitur-fitur SIFT dan
%   menampilkan berupa garis yang terhubung pada keypoint yang sama.
% Contoh: match('gambar1.jpeg','gambar2.jpeg');

function num = match(image1, image2)

% Mencari SIFT keypoint pada setiap image
[im1, des1, loc1] = sift(image1);
[im2, des2, loc2] = sift(image2);

distRatio = 0.6;

des2t = des2';
for i = 1 : size(des1,1)
   dotprods = des1(i,:) * des2t;
   [vals,indx] = sort(acos(dotprods));

   if (vals(1) < distRatio * vals(2))
      match(i) = indx(1);
   else
      match(i) = 0;
   end
end

im3 = appendimages(im1,im2);

figure('Position', [100 100 size(im3,2) size(im3,1)]);
colormap('gray');
imagesc(im3);
hold on;
cols1 = size(im1,2);
for i = 1: size(des1,1)
  if (match(i) > 0)
    line([loc1(i,2) loc2(match(i),2)+cols1], ...
         [loc1(i,1) loc2(match(i),1)], 'Color', 'c');
  end
end
hold off;
num = sum(match > 0);
fprintf('Mendapat %d kecocokan.\n', num);
```

match.c

```
/*****************************************************************
Demo software: Invariant keypoint matching.
Author: David Lowe

match.c:
This file contains a sample program to read images and keypoints, then
   draw lines connecting matched keypoints.
*****************************************************************
/
```

```
#include "defs.h"

/* ------------------- Local function prototypes ----------------------
- */

void FindMatches(Image im1, Keypoint keys1, Image im2, Keypoint keys2);
Keypoint CheckForMatch(Keypoint key, Keypoint klist);
int DistSquared(Keypoint k1, Keypoint k2);
Image CombineImagesVertically(Image im1, Image im2);


/*---------------------------- Routines ------------------------------
--*/

/* Top level routine.  Read PGM images and keypoints from files given
   in command line arguments, then call FindMatches.
*/
int main (int argc, char **argv)
{
    int arg = 0;
    Image im1 = NULL, im2 = NULL;
    Keypoint k1 = NULL, k2 = NULL;

    /* Parse command line arguments and read given files.  The command
       line must specify two input images and two files of keypoints
       using command line arguments as follows:
          match -im1 i1.pgm -k1 k1.key -im2 i2.pgm -k2 k2.key > result.v
    */
    while (++arg < argc) {
      if (! strcmp(argv[arg], "-im1"))
      im1 = ReadPGMFile(argv[++arg]);
      else if (! strcmp(argv[arg], "-im2"))
      im2 = ReadPGMFile(argv[++arg]);
      else if (! strcmp(argv[arg], "-k1"))
      k1 = ReadKeyFile(argv[++arg]);
      else if (! strcmp(argv[arg], "-k2"))
      k2 = ReadKeyFile(argv[++arg]);
      else
      FatalError("Invalid command line argument: %s", argv[arg]);
    }
    if (im1 == NULL || im2 == NULL || k1 == NULL || k2 == NULL)
      FatalError("Command line does not specify all images and keys.");

    FindMatches(im1, k1, im2, k2);
    exit(0);
}


/* Given a pair of images and their keypoints, pick the first keypoint
   from one image and find its closest match in the second set of
   keypoints.  Then write the result to a file.
*/
void FindMatches(Image im1, Keypoint keys1, Image im2, Keypoint keys2)
{
    Keypoint k, match;
    Image result;
    int count = 0;

    /* Create a new image that joins the two images vertically. */
    result = CombineImagesVertically(im1, im2);
```

```
    /* Match the keys in list keys1 to their best matches in keys2.
    */
    for (k= keys1; k != NULL; k = k->next) {
      match = CheckForMatch(k, keys2);

      /* Draw a line on the image from keys1 to match.  Note that we
       must add row count of first image to row position in second so
       that line ends at correct location in second image.
      */
      if (match != NULL) {
      count++;
      DrawLine(result, (int) k->row, (int) k->col,
             (int) (match->row + im1->rows), (int) match->col);
      }
    }

    /* Write result image to standard output. */
    WritePGM(stdout, result);
    fprintf(stderr,"Found %d matches.\n", count);
}


/* This searches through the keypoints in klist for the two closest
   matches to key.  If the closest is less than 0.6 times distance to
   second closest, then return the closest match.  Otherwise, return
   NULL.
*/
Keypoint CheckForMatch(Keypoint key, Keypoint klist)
{
    int dsq, distsq1 = 100000000, distsq2 = 100000000;
    Keypoint k, minkey = NULL;

    /* Find the two closest matches, and put their squared distances in
       distsq1 and distsq2.
    */
    for (k = klist; k != NULL; k = k->next) {
      dsq = DistSquared(key, k);

      if (dsq < distsq1) {
      distsq2 = distsq1;
      distsq1 = dsq;
      minkey = k;
      } else if (dsq < distsq2) {
      distsq2 = dsq;
      }
    }

    /* Check whether closest distance is less than 0.6 of second. */
    if (10 * 10 * distsq1 < 6 * 6 * distsq2)
      return minkey;
    else return NULL;
}


/* Return squared distance between two keypoint descriptors.
*/
int DistSquared(Keypoint k1, Keypoint k2)
{
    int i, dif, distsq = 0;
    unsigned char *pk1, *pk2;
```

```
    pk1 = k1->descrip;
    pk2 = k2->descrip;

    for (i = 0; i < 128; i++) {
      dif = (int) *pk1++ - (int) *pk2++;
      distsq += dif * dif;
    }
    return distsq;
}


/* Return a new image that contains the two images with im1 above im2.
*/
Image CombineImagesVertically(Image im1, Image im2)
{
    int rows, cols, r, c;
    Image result;

    rows = im1->rows + im2->rows;
    cols = MAX(im1->cols, im2->cols);
    result = CreateImage(rows, cols);

    /* Set all pixels to 0,5, so that blank regions are grey. */
    for (r = 0; r < rows; r++)
      for (c = 0; c < cols; c++)
      result->pixels[r][c] = 0.5;

    /* Copy images into result. */
    for (r = 0; r < im1->rows; r++)
      for (c = 0; c < im1->cols; c++)
      result->pixels[r][c] = im1->pixels[r][c];
    for (r = 0; r < im2->rows; r++)
      for (c = 0; c < im2->cols; c++)
      result->pixels[r + im1->rows][c] = im2->pixels[r][c];

    return result;
}
```

sift.m

```
% [image, descriptors, locs] = sift(imageFile)
%
% This function reads an image and returns its SIFT keypoints.
%   Input parameters:
%     imageFile: the file name for the image.
%
%   Returned:
%     image: the image array in double format
%     descriptors: a K-by-128 matrix, where each row gives an invariant
%         descriptor for one of the K keypoints.  The descriptor is a
vector
%         of 128 values normalized to unit length.
%     locs: K-by-4 matrix, in which each row has the 4 values for a
%         keypoint location (row, column, scale, orientation).  The
%         orientation is in the range [-PI, PI] radians.
%
% Credits: Thanks for initial version of this program to D. Alvaro and
%          J.J. Guerrero, Universidad de Zaragoza (modified by D. Lowe)

function [image, descriptors, locs] = sift(imageFile)
```

```
% Load image
image = imread(imageFile);

% If you have the Image Processing Toolbox, you can uncomment the
following
%   lines to allow input of color images, which will be converted to
grayscale.
% if isrgb(image)
%    image = rgb2gray(image);
% end

[rows, cols] = size(image);

% Convert into PGM imagefile, readable by "keypoints" executable
f = fopen('tmp.pgm', 'w');
if f == -1
    error('Could not create file tmp.pgm.');
end
fprintf(f, 'P5\n%d\n%d\n255\n', cols, rows);
fwrite(f, image', 'uint8');
fclose(f);

% Call keypoints executable
if isunix
    command = '!./sift ';
else
    command = '!siftWin32 ';
end
command = [command ' <tmp.pgm >tmp.key'];
eval(command);

% Open tmp.key and check its header
g = fopen('tmp.key', 'r');
if g == -1
    error('Could not open file tmp.key.');
end
[header, count] = fscanf(g, '%d %d', [1 2]);
if count ~= 2
    error('Invalid keypoint file beginning.');
end
num = header(1);
len = header(2);
if len ~= 128
    error('Keypoint descriptor length invalid (should be 128).');
end

% Creates the two output matrices (use known size for efficiency)
locs = double(zeros(num, 4));
descriptors = double(zeros(num, 128));

% Parse tmp.key
for i = 1:num
    [vector, count] = fscanf(g, '%f %f %f %f', [1 4]); %row col scale ori
    if count ~= 4
        error('Invalid keypoint file format');
    end
    locs(i, :) = vector(1, :);

    [descrip, count] = fscanf(g, '%d', [1 len]);
    if (count ~= 128)
```

```
        error('Invalid keypoint file value.');
    end
    % Normalize each input vector to unit length
    descrip = descrip / sqrt(sum(descrip.^2));
    descriptors(i, :) = descrip(1, :);
end
fclose(g);
```

appendimages.m

```
% im = appendimages(image1, image2)
%
% Return a new image that appends the two images side-by-side.

function im = appendimages(image1, image2)

% Select the image with the fewest rows and fill in enough empty rows
%   to make it the same height as the other image.
rows1 = size(image1,1);
rows2 = size(image2,1);

if (rows1 < rows2)
     image1(rows2,1) = 0;
else
     image2(rows1,1) = 0;
end

% Now append both images side-by-side.
im = [image1 image2];
```

SIFT_feature.m

```
%this code is the Matlab implimentation of David G. Lowe,
%"Distinctive image features from scale-invariant keypoints,"
%International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.
%this code should be used only for academic research.
%any other useage of this code should not be allowed without Author
agreement.
% if you have any problem or improvement idea about this code, please
% contact with Xing Di, Stevens Institution of Technology.
xdi2@stevens.edu

%%initial image
tic
clear;
clc;
row=256;
colum=256;
img=imread('jokowi_mark.jpeg');
img=imresize(img,[row,colum]);
img=rgb2gray(img);
% img=histeq(img);
img=im2double(img);
origin=img;
% img=medfilt2(img);
toc
%% Scale-Space Extrema Detection
tic
% original sigma and the number of actave can be modified. the larger
% sigma0, the more quickly-smooth images
```

```
sigma0=sqrt(2);
octave=3;%6*sigma*k^(octave*level)<=min(m,n)/(2^(octave-2))
level=3;
D=cell(1,octave);
for i=1:octave
D(i)=mat2cell(zeros(row*2^(2-i)+2,colum*2^(2-i)+2,level),row*2^(2-
i)+2,colum*2^(2-i)+2,level);
end
% first image in first octave is created by interpolating the original
one.
temp_img=kron(img,ones(2));
temp_img=padarray(temp_img,[1,1],'replicate');
figure(2)
subplot(1,2,1);
imshow(origin)
%create the DoG pyramid.
for i=1:octave
    temp_D=D{i};
    for j=1:level
        scale=sigma0*sqrt(2)^(1/level)^((i-1)*level+j);
        p=(level)*(i-1);
        figure(1);
        subplot(octave,level,p+j);
        f=fspecial('gaussian',[1,floor(6*scale)],scale);
        L1=temp_img;
        if(i==1&&j==1)
        L2=conv2(temp_img,f,'same');
        L2=conv2(L2,f','same');
        temp_D(:,:,j)=L2-L1;
        imshow(uint8(255 * mat2gray(temp_D(:,:,j))));
        L1=L2;
        else
        L2=conv2(temp_img,f,'same');
        L2=conv2(L2,f','same');
        temp_D(:,:,j)=L2-L1;
        L1=L2;
        if(j==level)
            temp_img=L1(2:end-1,2:end-1);
        end
        imshow(uint8(255 * mat2gray(temp_D(:,:,j))));
        end
    end
    D{i}=temp_D;
    temp_img=temp_img(1:2:end,1:2:end);
    temp_img=padarray(temp_img,[1,1],'both','replicate');
end
toc
%% Keypoint Localistaion
% search each pixel in the DoG map to find the extreme point
tic
interval=level-1;
number=0;
for i=2:octave+1
    number=number+(2^(i-octave)*colum)*(2*row)*interval;
end
extrema=zeros(1,4*number);
flag=1;
for i=1:octave
    [m,n,~]=size(D{i});
    m=m-2;
    n=n-2;
```

```
    volume=m*n/(4^(i-1));
    for k=2:interval
        for j=1:volume
            % starter=D{i}(x+1,y+1,k);
            x=ceil(j/n);
            y=mod(j-1,m)+1;
            sub=D{i}(x:x+2,y:y+2,k-1:k+1);
            large=max(max(max(sub)));
            little=min(min(min(sub)));
            if(large==D{i}(x+1,y+1,k))
                temp=[i,k,j,1];
                extrema(flag:(flag+3))=temp;
                flag=flag+4;
            end
            if(little==D{i}(x+1,y+1,k))
                temp=[i,k,j,-1];
                extrema(flag:(flag+3))=temp;
                flag=flag+4;
            end
        end
    end
end
idx= extrema==0;
extrema(idx)=[];
toc
[m,n]=size(img);
x=floor((extrema(3:4:end)-1)./(n./(2.^(extrema(1:4:end)-2))))+1;
y=mod((extrema(3:4:end)-1),m./(2.^(extrema(1:4:end)-2)))+1;
ry=y./2.^(octave-1-extrema(1:4:end));
rx=x./2.^(octave-1-extrema(1:4:end));
figure(2)
subplot(1,2,2);
imshow(origin)
hold on
plot(ry,rx,'r+');
%% accurate keypoint localization
%eliminate the point with low contrast or poorly localised on an edge
% x:|,y:-- x is for vertial and y is for horizontal
% value comes from the paper.
tic
threshold=0.1;
r=10;
extr_volume=length(extrema)/4;
[m,n]=size(img);
secondorder_x=conv2([-1,1;-1,1],[-1,1;-1,1]);
secondorder_y=conv2([-1,-1;1,1],[-1,-1;1,1]);
for i=1:octave
    for j=1:level
        test=D{i}(:,:,j);
        temp=-1./conv2(test,secondorder_y,'same').*conv2(test,[-1,-
1;1,1],'same');
        D{i}(:,:,j)=temp.*conv2(test',[-1,-1;1,1],'same')*0.5+test;
    end
end
for i=1:extr_volume
    x=floor((extrema(4*(i-1)+3)-1)/(n/(2^(extrema(4*(i-1)+1)-2))))+1;
    y=mod((extrema(4*(i-1)+3)-1),m/(2^(extrema(4*(i-1)+1)-2)))+1;
    rx=x+1;
    ry=y+1;
    rz=extrema(4*(i-1)+2);
    z=D{extrema(4*(i-1)+1)}(rx,ry,rz);
```

```
    if(abs(z)<threshold)
        extrema(4*(i-1)+4)=0;
    end
end
idx=find(extrema==0);
idx=[idx,idx-1,idx-2,idx-3];
extrema(idx)=[];
extr volume=length(extrema)/4;
x=floor((extrema(3:4:end)-1)./(n./(2.^(extrema(1:4:end)-2))))+1;
y=mod((extrema(3:4:end)-1),m./(2.^(extrema(1:4:end)-2)))+1;
ry=y./2.^(octave-1-extrema(1:4:end));
rx=x./2.^(octave-1-extrema(1:4:end));
figure(2)
subplot(2,2,3);
imshow(origin)
hold on
plot(ry,rx,'g+');
for i=1:extr_volume
    x=floor((extrema(4*(i-1)+3)-1)/(n/(2^(extrema(4*(i-1)+1)-2))))+1;
    y=mod((extrema(4*(i-1)+3)-1),m/(2^(extrema(4*(i-1)+1)-2)))+1;
    rx=x+1;
    ry=y+1;
    rz=extrema(4*(i-1)+2);
        Dxx=D{extrema(4*(i-1)+1)}(rx-1,ry,rz)+D{extrema(4*(i-
1)+1)}(rx+1,ry,rz)-2*D{extrema(4*(i-1)+1)}(rx,ry,rz);
        Dyy=D{extrema(4*(i-1)+1)}(rx,ry-1,rz)+D{extrema(4*(i-
1)+1)}(rx,ry+1,rz)-2*D{extrema(4*(i-1)+1)}(rx,ry,rz);
        Dxy=D{extrema(4*(i-1)+1)}(rx-1,ry-1,rz)+D{extrema(4*(i-
1)+1)}(rx+1,ry+1,rz)-D{extrema(4*(i-1)+1)}(rx-1,ry+1,rz)-D{extrema(4*(i-
1)+1)}(rx+1,ry-1,rz);
        deter=Dxx*Dyy-Dxy*Dxy;
        R=(Dxx+Dyy)/deter;
        R_threshold=(r+1)^2/r;
        if(deter<0||R>R_threshold)
            extrema(4*(i-1)+4)=0;
        end

end
idx=find(extrema==0);
idx=[idx,idx-1,idx-2,idx-3];
extrema(idx)=[];
extr_volume=length(extrema)/4;
x=floor((extrema(3:4:end)-1)./(n./(2.^(extrema(1:4:end)-2))))+1;
y=mod((extrema(3:4:end)-1),m./(2.^(extrema(1:4:end)-2)))+1;
ry=y./2.^(octave-1-extrema(1:4:end));
rx=x./2.^(octave-1-extrema(1:4:end));
figure(2)
subplot(2,2,4);
imshow(origin)
hold on
plot(ry,rx,'b+');
toc
%% Orientation Assignment(Multiple orientations assignment)
tic
kpori=zeros(1,36*extr_volume);
minor=zeros(1,36*extr_volume);
f=1;
flag=1;
for i=1:extr_volume
    %search in the certain scale
```

```
    scale=sigma0*sqrt(2)^(1/level)^((extrema(4*(i-1)+1)-
1)*level+(extrema(4*(i-1)+2)));
    width=2*round(3*1.5*scale);
    count=1;
    x=floor((extrema(4*(i-1)+3)-1)/(n/(2^(extrema(4*(i-1)+1)-2))))+1;
    y=mod((extrema(4*(i-1)+3)-1),m/(2^(extrema(4*(i-1)+1)-2)))+1;
    %make sure the point in the searchable area
    if(x>(width/2)&&y>(width/2)&&x<(m/2^(extrema(4*(i-1)+1)-2)-width/2-
2)&&y<(n/2^(extrema(4*(i-1)+1)-2)-width/2-2))
        rx=x+1;
        ry=y+1;
        rz=extrema(4*(i-1)+2);
        reg_volume=width*width;%3? thereom
        % make weight matrix
        weight=fspecial('gaussian',width,1.5*scale);
        %calculate region pixels' magnitude and region orientation
        reg_mag=zeros(1,count);
        reg_theta=zeros(1,count);
    for l=(rx-width/2):(rx+width/2-1)
        for k=(ry-width/2):(ry+width/2-1)
            reg_mag(count)=sqrt((D{extrema(4*(i-1)+1)}(l+1,k,rz)-
D{extrema(4*(i-1)+1)}(l-1,k,rz))^2+(D{extrema(4*(i-1)+1)}(l,k+1,rz)-
D{extrema(4*(i-1)+1)}(l,k-1,rz))^2);
            reg_theta(count)=atan2((D{extrema(4*(i-1)+1)}(l,k+1,rz)-
D{extrema(4*(i-1)+1)}(l,k-1,rz)),(D{extrema(4*(i-1)+1)}(l+1,k,rz)-
D{extrema(4*(i-1)+1)}(l-1,k,rz)))*(180/pi);
            count=count+1;
        end
    end
    %make histogram
    mag_counts=zeros(1,36);
    for x=0:10:359
        mag_count=0;
       for j=1:reg_volume
           c1=-180+x;
           c2=-171+x;
           if(c1<0||c2<0)
           if(abs(reg_theta(j))<abs(c1)&&abs(reg_theta(j))>=abs(c2))
               mag_count=mag_count+reg_mag(j)*weight(ceil(j/width),mod(j-
1,width)+1);
           end
           else
               if(abs(reg_theta(j)>abs(c1)&&abs(reg_theta(j)<=abs(c2))))

mag_count=mag_count+reg_mag(j)*weight(ceil(j/width),mod(j-1,width)+1);
               end
           end
       end
          mag_counts(x/10+1)=mag_count;
    end
    % find the max histogram bar and the ones higher than 80% max
    [maxvm,~]=max(mag_counts);
     kori=find(mag_counts>=(0.8*maxvm));
     kori=(kori*10+(kori-1)*10)./2-180;
     kpori(f:(f+length(kori)-1))=kori;
     f=f+length(kori);
     temp_extrema=[extrema(4*(i-1)+1),extrema(4*(i-1)+2),extrema(4*(i-
1)+3),extrema(4*(i-1)+4)];

temp_extrema=padarray(temp_extrema,[0,length(temp_extrema)*(length(kori)-
1)],'post','circular');
```

```
     long=length(temp_extrema);
     minor(flag:flag+long-1)=temp_extrema;
     flag=flag+long;
    end
end
idx= minor==0;
minor(idx)=[];
extrema=minor;
% delete unsearchable points and add minor orientation points
idx= kpori==0;
kpori(idx)=[];
extr_volume=length(extrema)/4;
toc
%% keypoint descriptor
tic
d=4;% In David G. Lowe experiment,divide the area into 4*4.
pixel=4;
feature=zeros(d*d*8,extr_volume);
for i=1:extr_volume
    descriptor=zeros(1,d*d*8);% feature dimension is 128=4*4*8;
    width=d*pixel;
    %x,y centeral point and prepare for location rotation
    x=floor((extrema(4*(i-1)+3)-1)/(n/(2^(extrema(4*(i-1)+1)-2))))+1;
    y=mod((extrema(4*(i-1)+3)-1),m/(2^(extrema(4*(i-1)+1)-2)))+1;
    z=extrema(4*(i-1)+2);
        if((m/2^(extrema(4*(i-1)+1)-2)-
pixel*d*sqrt(2)/2)>x&&x>(pixel*d/2*sqrt(2))&&(n/2^(extrema(4*(i-1)+1)-2)-
pixel*d/2*sqrt(2))>y&&y>(pixel*d/2*sqrt(2)))
        sub_x=(x-d*pixel/2+1):(x+d*pixel/2);
        sub_y=(y-d*pixel/2+1):(y+d*pixel/2);
        sub=zeros(2,length(sub_x)*length(sub_y));
        j=1;
        for p=1:length(sub_x)
            for q=1:length(sub_y)
                sub(:,j)=[sub_x(p)-x;sub_y(q)-y];
                j=j+1;
            end
        end
        distort=[cos(pi*kpori(i)/180),-
sin(pi*kpori(i)/180);sin(pi*kpori(i)/180),cos(pi*kpori(i)/180)];
    %accordinate after distort
        sub_dis=distort*sub;
        fix_sub=ceil(sub_dis);
        fix_sub=[fix_sub(1,:)+x;fix_sub(2,:)+y];
        patch=zeros(1,width*width);
        for p=1:length(fix_sub)
        patch(p)=D{extrema(4*(i-1)+1)}(fix_sub(1,p),fix_sub(2,p),z);
        end
        temp_D=(reshape(patch,[width,width]))';
        %create weight matrix.
        mag_sub=temp_D;
        temp_D=padarray(temp_D,[1,1],'replicate','both');
        weight=fspecial('gaussian',width,width/1.5);
        mag_sub=weight.*mag_sub;
        theta_sub=atan((temp_D(2:end-1,3:1:end)-temp_D(2:end-1,1:1:end-
2))./(temp_D(3:1:end,2:1:end-1)-temp_D(1:1:end-2,2:1:end-1)))*(180/pi);
        % create orientation histogram
        for area=1:d*d
        cover=pixel*pixel;
        ori=zeros(1,cover);
        magcounts=zeros(1,8);
```

```
        for angle=0:45:359
          magcount=0;
          for p=1:cover;
              x=(floor((p-1)/pixel)+1)+pixel*floor((area-1)/d);
              y=mod(p-1,pixel)+1+pixel*(mod(area-1,d));
              c1=-180+angle;
              c2=-180+45+angle;
              if(c1<0||c2<0)
                  if
(abs(theta_sub(x,y))<abs(c1)&&abs(theta_sub(x,y))>=abs(c2))

                      ori(p)=(c1+c2)/2;
                      magcount=magcount+mag_sub(x,y);
                  end
              else

if(abs(theta_sub(x,y))>abs(c1)&&abs(theta_sub(x,y))<=abs(c2))
                      ori(p)=(c1+c2)/2;
                      magcount=magcount+mag_sub(x,y);
                  end
              end
          end
          magcounts(angle/45+1)=magcount;
        end
        descriptor((area-1)*8+1:area*8)=magcounts;
        end
        descriptor=normr(descriptor);
        % cap 0.2
        for j=1:numel(descriptor)
            if(abs(descriptor(j))>0.2)
            descriptor(j)=0.2;
            end
        end
        descriptor=normr(descriptor);
        else
            continue;
        end
        feature(:,i)=descriptor';
end
index=find(sum(feature));
feature=feature(:,index);
toc
```

HistogramColorDetection.m

```
% deteksi warna RGB
% by ImageAnalyst
function SimpleColorDetection()
clc;  % Perintah untuk membersihkan layar.
clear;      % menghapus semua vriabel.
close all;  % menutup semua windows, kecuali yang dibuat oleh imtool.
% imtool close all;     % Close all figure windows created by imtool.
workspace;  % Make sure the workspace panel is showing.

% Change the current folder to the folder of this m-file.
% (The "cd" line of code below is from Brett Shoelson of The Mathworks.)
if(~isdeployed)
      cd(fileparts(which(mfilename))); % From Brett
```

```
end

ver % Display user's toolboxes in their command window.

% Introduce the demo, and ask user if they want to continue or exit.
% message = sprintf('This demo will illustrate very simple color
detection in RGB color space.\nIt requires the Image Processing
Toolbox.\nDo you wish to continue?');
message = sprintf('Aplikasi ini dapat menganalisa bagaimana cara
mendeteksi warna pada RGB color.\nTools Matlab yang diperlukan adalah
Image Processing Toolbox.\nKlik OK untuk melanjutkan?');
reply = questdlg(message, 'Run Demo?', 'OK','Cancel', 'OK');
if strcmpi(reply, 'Cancel')
      % User canceled so exit.
      return;
end

try
      % Check that user has the Image Processing Toolbox installed.
      versionInfo = ver; % Capture their toolboxes in the variable.
      hasIPT = false;
      for k = 1:length(versionInfo)
            if strcmpi(versionInfo(k).Name, 'Image Processing Toolbox') >
0
                  hasIPT = true;
            end
      end
      if ~hasIPT
            % User does not have the toolbox installed.
            message = sprintf('Sorry, but you do not seem to have the
Image Processing Toolbox.\nDo you want to try to continue anyway?');
            reply = questdlg(message, 'Toolbox missing', 'Yes', 'No',
'Yes');
            if strcmpi(reply, 'No')
                  % User said No, so exit.
                  return;
            end
      end

      % Continue with the demo.  Do some initialization stuff.
      close all;
      fontSize = 16;
      figure;
      % Maximize the figure.
      set(gcf, 'Position', get(0, 'ScreenSize'));

      % Change the current folder to the folder of this m-file.
      % (The line of code below is from Brett Shoelson of The Mathworks.)
      if(~isdeployed)
            cd(fileparts(which(mfilename)));
      end

      % Ask user if they want to use a demo image or their own image.
      % message = sprintf('Do you want use a standard demo image,\nOr
pick one of your own?');
    message = sprintf('Apakah anda akan menggunakan standar bawaan image
dari Matlab,\natau menggunakan gambar sendiri?');
      reply2 = questdlg(message, 'Pilih gambar yang akan digunakan?',
'Demo Image Matlab', 'Gambar Sendiri', 'Demo');
      % Open an image.
      if strcmpi(reply2, 'Demo')
```

```
            % Read standard MATLAB demo image.
      %     fullImageFileName = 'peppers.png';
            message = sprintf('Gambar mana yang akan anda gunakan?');
            selectedImage = questdlg(message, 'Which Demo Image?',
'Onions', 'Peppers', 'Canoe', 'Onions');
            if strcmp(selectedImage, 'Onions')
                  fullImageFileName = 'onion.png';
            elseif strcmp(selectedImage, 'Peppers')
                  fullImageFileName = 'peppers.png';
            else
                  fullImageFileName = 'canoe.tif';
            end
      else
            % They want to pick their own.
            % Change default directory to the one containing the standard
demo images for the MATLAB Image Processing Toolbox.
            originalFolder = pwd;
            folder = 'C:\Program
Files\MATLAB\R2010a\toolbox\images\imdemos';
            if ~exist(folder, 'dir')
                  folder = pwd;
            end
            cd(folder);
            % Browse for the image file.
            [baseFileName, folder] = uigetfile('*.*', 'Specify an image
file');
            fullImageFileName = fullfile(folder, baseFileName);
            % Set current folder back to the original one.
            cd(originalFolder);
            selectedImage = 'My own image'; % Need for the if threshold
selection statement later.

      end

      % Check to see that the image exists.  (Mainly to check on the demo
images.)
      if ~exist(fullImageFileName, 'file')
            message = sprintf('This file does not exist:\n%s',
fullImageFileName);
            uiwait(msgbox(message));
            return;
      end

      % Read in image into an array.
      [rgbImage storedColorMap] = imread(fullImageFileName);
      [rows columns numberOfColorBands] = size(rgbImage);
      % If it's monochrome (indexed), convert it to color.
      % Check to see if it's an 8-bit image needed later for scaling).
      if strcmpi(class(rgbImage), 'uint8')
            % Flag for 256 gray levels.
            eightBit = true;
      else
            eightBit = false;
      end
      if numberOfColorBands == 1
            if isempty(storedColorMap)
                  % Just a simple gray level image, not indexed with a
stored color map.
                  % Create a 3D true color image where we copy the
monochrome image into all 3 (R, G, & B) color planes.
                  rgbImage = cat(3, rgbImage, rgbImage, rgbImage);
```

```
            else
                  % It's an indexed image.
                  rgbImage = ind2rgb(rgbImage, storedColorMap);
                  % ind2rgb() will convert it to double and normalize it
to the range 0-1.
                  % Convert back to uint8 in the range 0-255, if needed.
                  if eightBit
                        rgbImage = uint8(255 * rgbImage);
                  end
            end
      end
      % Display the original image.
      subplot(3, 4, 1);
      imshow(rgbImage);
      drawnow; % Make it display immediately.
      if numberOfColorBands > 1
            title('Original Color Image', 'FontSize', fontSize);
      else
            caption = sprintf('Original Indexed Image\n(converted to true
color with its stored colormap)');
            title(caption, 'FontSize', fontSize);
      end

      % Extract out the color bands from the original image
      % into 3 separate 2D arrays, one for each color component.
      redBand = rgbImage(:, :, 1);
      greenBand = rgbImage(:, :, 2);
      blueBand = rgbImage(:, :, 3);
      % Display them.
      subplot(3, 4, 2);
      imshow(redBand);
      title('Red Band', 'FontSize', fontSize);
      subplot(3, 4, 3);
      imshow(greenBand);
      title('Green Band', 'FontSize', fontSize);
      subplot(3, 4, 4);
      imshow(blueBand);
      title('Blue Band', 'FontSize', fontSize);
      message = sprintf('These are the individual color bands.\nNow we
will compute the image histograms.');
      reply = questdlg(message, 'Continue with Demo?', 'OK','Cancel',
'OK');
      if strcmpi(reply, 'Cancel')
            % User canceled so exit.
            return;
      end

      fontSize = 13;

      % Compute and plot the red histogram.
      hR = subplot(3, 4, 6);
      [countsR, grayLevelsR] = imhist(redBand);
      maxGLValueR = find(countsR > 0, 1, 'last');
      maxCountR = max(countsR);
      bar(countsR, 'r');
      grid on;
      xlabel('Gray Levels');
      ylabel('Pixel Count');
      title('Histogram of Red Band', 'FontSize', fontSize);

      % Compute and plot the green histogram.
```

```
      hG = subplot(3, 4, 7);
      [countsG, grayLevelsG] = imhist(greenBand);
      maxGLValueG = find(countsG > 0, 1, 'last');
      maxCountG = max(countsG);
      bar(countsG, 'g', 'BarWidth', 0.95);
      grid on;
      xlabel('Gray Levels');
      ylabel('Pixel Count');
      title('Histogram of Green Band', 'FontSize', fontSize);

      % Compute and plot the blue histogram.
      hB = subplot(3, 4, 8);
      [countsB, grayLevelsB] = imhist(blueBand);
      maxGLValueB = find(countsB > 0, 1, 'last');
      maxCountB = max(countsB);
      bar(countsB, 'b');
      grid on;
      xlabel('Gray Levels');
      ylabel('Pixel Count');
      title('Histogram of Blue Band', 'FontSize', fontSize);

      % Set all axes to be the same width and height.
      % This makes it easier to compare them.
      maxGL = max([maxGLValueR,  maxGLValueG, maxGLValueB]);
      if eightBit
                  maxGL = 255;
      end
      maxCount = max([maxCountR,  maxCountG, maxCountB]);
      axis([hR hG hB], [0 maxGL 0 maxCount]);

      % Plot all 3 histograms in one plot.
      subplot(3, 4, 5);
      plot(grayLevelsR, countsR, 'r', 'LineWidth', 2);
      grid on;
      xlabel('Gray Levels');
      ylabel('Pixel Count');
      hold on;
      plot(grayLevelsG, countsG, 'g', 'LineWidth', 2);
      plot(grayLevelsB, countsB, 'b', 'LineWidth', 2);
      title('Histogram RGB Color', 'FontSize', fontSize);
      maxGrayLevel = max([maxGLValueR, maxGLValueG, maxGLValueB]);
      % Trim x-axis to just the max gray level on the bright end.
      if eightBit
            xlim([0 255]);
      else
            xlim([0 maxGrayLevel]);
      end

      % Now select thresholds for the 3 color bands.
      message = sprintf('Now we will select some color threshold
ranges\nand display them over the histograms.');
      reply = questdlg(message, 'Continue with Demo?', 'OK','Cancel',
'OK');
      if strcmpi(reply, 'Cancel')
            % User canceled so exit.
            return;
      end

      % Assign the low and high thresholds for each color band.
      if strcmpi(reply2, 'My Own') || strcmpi(selectedImage, 'Canoe') > 0
```

```
            % Take a guess at the values that might work for the user's
image.
            redThresholdLow = graythresh(redBand);
            redThresholdHigh = 255;
            greenThresholdLow = 0;
            greenThresholdHigh = graythresh(greenBand);
            blueThresholdLow = 0;
            blueThresholdHigh = graythresh(blueBand);
            if eightBit
                  redThresholdLow = uint8(redThresholdLow * 255);
                  greenThresholdHigh = uint8(greenThresholdHigh * 255);
                  blueThresholdHigh = uint8(blueThresholdHigh * 255);
            end
      else
            % Use values that I know work for the onions and peppers demo
images.
            redThresholdLow = 85;
            redThresholdHigh = 255;
            greenThresholdLow = 0;
            greenThresholdHigh = 70;
            blueThresholdLow = 0;
            blueThresholdHigh = 90;
      end

      % Show the thresholds as vertical red bars on the histograms.
      PlaceThresholdBars(6, redThresholdLow, redThresholdHigh);
      PlaceThresholdBars(7, greenThresholdLow, greenThresholdHigh);
      PlaceThresholdBars(8, blueThresholdLow, blueThresholdHigh);

      message = sprintf('Now we will apply each color band threshold
range to the color band.');
      reply = questdlg(message, 'Continue with Demo?', 'OK','Cancel',
'OK');
      if strcmpi(reply, 'Cancel')
            % User canceled so exit.
            return;
      end

      % Now apply each color band's particular thresholds to the color
band
      redMask = (redBand >= redThresholdLow) & (redBand <=
redThresholdHigh);
      greenMask = (greenBand >= greenThresholdLow) & (greenBand <=
greenThresholdHigh);
      blueMask = (blueBand >= blueThresholdLow) & (blueBand <=
blueThresholdHigh);

      % Display the thresholded binary images.
      fontSize = 16;
      subplot(3, 4, 10);
      imshow(redMask, []);
      title('Is-Red Mask', 'FontSize', fontSize);
      subplot(3, 4, 11);
      imshow(greenMask, []);
      title('Is-Not-Green Mask', 'FontSize', fontSize);
      subplot(3, 4, 12);
      imshow(blueMask, []);
      title('Is-Not-Blue Mask', 'FontSize', fontSize);
      % Combine the masks to find where all 3 are "true."
      % Then we will have the mask of only the red parts of the image.
      redObjectsMask = uint8(redMask & greenMask & blueMask);
```

```
      subplot(3, 4, 9);
      imshow(redObjectsMask, []);
      caption = sprintf('Mask of Only\nThe Red Objects');
      title(caption, 'FontSize', fontSize);

      % Tell user that we're going to filter out small objects.
      smallestAcceptableArea = 100; % Keep areas only if they're bigger
than this.
      message = sprintf('Note the small regions in the image in the lower
left.\nNext we will eliminate regions smaller than %d pixels.',
smallestAcceptableArea);
      reply = questdlg(message, 'Continue with Demo?', 'OK','Cancel',
'OK');
      if strcmpi(reply, 'Cancel')
            % User canceled so exit.
            return;
      end

      % Open up a new figure, since the existing one is full.
      figure;
      % Maximize the figure.
      set(gcf, 'Position', get(0, 'ScreenSize'));

      % Get rid of small objects.  Note: bwareaopen returns a logical.
      redObjectsMask = uint8(bwareaopen(redObjectsMask,
smallestAcceptableArea));
      subplot(3, 3, 1);
      imshow(redObjectsMask, []);
      fontSize = 13;
      caption = sprintf('bwareaopen() removed objects\nsmaller than %d
pixels', smallestAcceptableArea);
      title(caption, 'FontSize', fontSize);

      % Smooth the border using a morphological closing operation,
imclose().
      structuringElement = strel('disk', 4);
      redObjectsMask = imclose(redObjectsMask, structuringElement);
      subplot(3, 3, 2);
      imshow(redObjectsMask, []);
      fontSize = 16;
      title('Border smoothed', 'FontSize', fontSize);

      % Fill in any holes in the regions, since they are most likely red
also.
      redObjectsMask = uint8(imfill(redObjectsMask, 'holes'));
      subplot(3, 3, 3);
      imshow(redObjectsMask, []);
      title('Regions Filled', 'FontSize', fontSize);

      message = sprintf('This is the filled, size-filtered mask.\nNow we
will apply this mask to the original image.');
      reply = questdlg(message, 'Continue with Demo?', 'OK','Cancel',
'OK');
      if strcmpi(reply, 'Cancel')
            % User canceled so exit.
            return;
      end


      redObjectsMask = cast(redObjectsMask, class(redBand));
```

```
      maskedImageR = redObjectsMask .* redBand;
      maskedImageG = redObjectsMask .* greenBand;
      maskedImageB = redObjectsMask .* blueBand;
      % Show the masked off red image.
      subplot(3, 3, 4);
      imshow(maskedImageR);
      title('Masked Red Image', 'FontSize', fontSize);
      % Show the masked off green image.
      subplot(3, 3, 5);
      imshow(maskedImageG);
      title('Masked Green Image', 'FontSize', fontSize);
      % Show the masked off blue image.
      subplot(3, 3, 6);
      imshow(maskedImageB);
      title('Masked Blue Image', 'FontSize', fontSize);
      % Concatenate the masked color bands to form the rgb image.
      maskedRGBImage = cat(3, maskedImageR, maskedImageG, maskedImageB);
      % Show the masked off, original image.
      subplot(3, 3, 8);
      imshow(maskedRGBImage);
      fontSize = 13;
      caption = sprintf('Masked Original Image\nShowing Only the Red
Objects');
      title(caption, 'FontSize', fontSize);
      % Show the original image next to it.
      subplot(3, 3, 7);
      imshow(rgbImage);
      title('The Original Image (Again)', 'FontSize', fontSize);

      % Measure the mean RGB and area of all the detected blobs.
      [meanRGB, areas, numberOfBlobs] = MeasureBlobs(redObjectsMask,
redBand, greenBand, blueBand);
      if numberOfBlobs > 0
            fprintf(1, '\n-----------------------------------------------
\n');
            fprintf(1, 'Blob #, Area in Pixels, Mean R, Mean G, Mean
B\n');
            fprintf(1, '-----------------------------------------------
\n');
            for blobNumber = 1 : numberOfBlobs
                  fprintf(1, '#%5d, %14d, %6.2f, %6.2f, %6.2f\n',
blobNumber, areas(blobNumber), ...
                        meanRGB(blobNumber, 1), meanRGB(blobNumber, 2),
meanRGB(blobNumber, 3));
            end
      else
            % Alert user that no red blobs were found.
            message = sprintf('No red blobs were found in the
image:\n%s', fullImageFileName);
            fprintf(1, '\n%s\n', message);
            uiwait(msgbox(message));
      end

      subplot(3, 3, 9);
      ShowCredits();
      message = sprintf('Done!\n\nThe demo has finished.\n\nLook the
MATLAB command window for\nthe area and color measurements of the %d
regions.', numberOfBlobs);
      msgbox(message);

catch ME
```

```
      errorMessage = sprintf('Error running this m-file:\n%s\n\nThe error
message is:\n%s', ...
            mfilename('fullpath'), ME.message);
      errordlg(errorMessage);
end
return; % from SimpleColorDetection()
% ---------- End of main function ---------------------------------


% Measure the mean intensity and area of each blob in each color band.
function [meanRGB, areas, numberOfBlobs] = MeasureBlobs(maskImage,
redBand, greenBand, blueBand)
      [labeledImage numberOfBlobs] = bwlabel(maskImage, 8);     % Label
each blob so we can make measurements of it
      if numberOfBlobs == 0
            % Didn't detect any yellow blobs in this image.
            meanRGB = [0 0 0];
            areas = 0;
            return;
      end
      % Get all the blob properties.  Can only pass in originalImage in
version R2008a and later.
      blobMeasurementsR = regionprops(labeledImage, redBand, 'area',
'MeanIntensity');
      blobMeasurementsG = regionprops(labeledImage, greenBand, 'area',
'MeanIntensity');
      blobMeasurementsB = regionprops(labeledImage, blueBand, 'area',
'MeanIntensity');

      meanRGB = zeros(numberOfBlobs, 3);  % One row for each blob.  One
column for each color.
      meanRGB(:,1) = [blobMeasurementsR.MeanIntensity]';
      meanRGB(:,2) = [blobMeasurementsG.MeanIntensity]';
      meanRGB(:,3) = [blobMeasurementsB.MeanIntensity]';

      % If redBand etc. are double, the intensities will be in the range
of 0-1.
      % Multiply by 255 to get them back into the uint8 range of 0-255.
      if ~strcmpi(class(redBand), 'uint8')
            meanRGB = meanRGB * 255.0;
      end
      % Now assign the areas.
      areas = zeros(numberOfBlobs, 3);  % One row for each blob.  One
column for each color.
      areas(:,1) = [blobMeasurementsR.Area]';
      areas(:,2) = [blobMeasurementsG.Area]';
      areas(:,3) = [blobMeasurementsB.Area]';
      return; % from MeasureBlobs()
plots.
function PlaceThresholdBars(plotNumber, lowThresh, highThresh)
      % Show the thresholds as vertical red bars on the histograms.
      subplot(3, 4, plotNumber);
      hold on;
      maxYValue = ylim;
      maxXValue = xlim;
      hStemLines = stem([lowThresh highThresh], [maxYValue(2)
maxYValue(2)], 'r');
      children = get(hStemLines, 'children');
      set(children(2),'visible', 'off');
      % Place a text label on the bar chart showing the threshold.
      fontSizeThresh = 14;
```

```
      annotationTextL = sprintf('%d', lowThresh);
      annotationTextH = sprintf('%d', highThresh);
      % For text(), the x and y need to be of the data class "double" so
let's cast both to double.
      text(double(lowThresh + 5), double(0.85 * maxYValue(2)),
annotationTextL, 'FontSize', fontSizeThresh, 'Color', [0 .5 0],
'FontWeight', 'Bold');
      text(double(highThresh + 5), double(0.85 * maxYValue(2)),
annotationTextH, 'FontSize', fontSizeThresh, 'Color', [0 .5 0],
'FontWeight', 'Bold');

highThresh/maxXValue(2)],[0.7 0.7]);

      return; % from PlaceThresholdBars()

% Display the MATLAB logo.
function ShowCredits()
      logoFig = subplot(3,3,9);
      caption = sprintf('A MATLAB Demo\nby ImageAnalyst');
      text(0.5,1.15, caption, 'Color','r', 'FontSize', 18,
'FontWeight','b', 'HorizontalAlignment', 'Center') ;
      positionOfLowerRightPlot = get(logoFig, 'position');
      L = 40*membrane(1,25);
      logoax = axes('CameraPosition', [-193.4013 -265.1546  220.4819],...
            'CameraTarget',[26 26 10], ...
            'CameraUpVector',[0 0 1], ...
            'CameraViewAngle',9.5, ...
            'DataAspectRatio', [1 1 .9],...
            'Position', positionOfLowerRightPlot, ...
            'Visible','off', ...
            'XLim',[1 51], ...
            'YLim',[1 51], ...
            'ZLim',[-13 40], ...
            'parent',gcf);
      s = surface(L, ...
            'EdgeColor','none', ...
            'FaceColor',[0.9 0.2 0.2], ...
            'FaceLighting','phong', ...
            'AmbientStrength',0.3, ...
            'DiffuseStrength',0.6, ...
            'Clipping','off',...
            'BackFaceLighting','lit', ...
            'SpecularStrength',1.1, ...
            'SpecularColorReflectance',1, ...
            'SpecularExponent',7, ...
            'Tag','TheMathWorksLogo', ...
            'parent',logoax);
      l1 = light('Position',[40 100 20], ...
            'Style','local', ...
            'Color',[0 0.8 0.8], ...
            'parent',logoax);
      l2 = light('Position',[.5 -1 .4], ...
            'Color',[0.8 0.8 0], ...
            'parent',logoax);

      return; % from ShowCredits()
```