semanTIK, Vol.2, No.1, Jan-Jun 2016, pp. 141-154
ISSN : 2502-8928 (Online)

PENGENDALI FUNGSI *POINTER* BERBASIS *HAND GESTURE* MENGGUNAKAN ALGORITMA *CONVEX HULL*

Alif Muqtadir*[1], Bambang Pramono[2], Ika Purwanti Ningrum[3]
[*1,2,3]Jurusan Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kendari
e-mail: ***[1]alifmuqtadir2@gmail.com**, [2]bambangpramono09@gmail.com,
[3]ika.purwanti.n@gmail.com

Abstrak

Perkembangan HCI (*Human Computer Interface*) semakin pesat dan menuju ke arah yang semakin *user-friendly*. Cara memasukkan data secara tradisional menggunakan *keyboard* telah diganti secara bertahap sejak banyak metode baru yang ditemukan. *Touch panel* berkembang dari *single-touch* ke *multi-touch*. Teknologi *body sense* bahkan menghilangkan hambatan pada alat *input* sehingga membuat HCI semakin dekat dengan gerak alami manusia. Bagian yang paling penting dari HCI adalah manipulasi menggunakan tangan. Tujuan dari penelitian ini adalah untuk membangun aplikasi yang dapat memberikan *user* kemampuan untuk mengendalikan *pointer* dengan menggunakan *hand gesture* (isyarat tangan).

Metode yang digunakan aplikasi untuk mengenali *hand gesture* adalah Algoritma *Convex Hull*. Algoritma *Convex Hull* adalah metode yang digunakan untuk mendeteksi kontur atau obyek pada citra dengan menggunakan titik-titik yang dihubungkan dengan garis yang mengelilingi kontur.

Hasil penelitian ini menunjukkan bahwa aplikasi dapat memberkan dapat memberikan *user* kemampuan untuk mengendalikan *pointer* dengan menggunakan *hand gesture*. Akan tetapi, deteksi obyeknya secara keseluruhan masih perlu ditingkatkan karena masih mudah terganggu oleh *background* atau obyek yang memiliki warna sama dengan tangan *user*.

Kata kunci— Pengendali *Pointer*, *Convex Hull, Hand Gesture, Webcam*.

Abstract

The development of HCI (Human Computer Interface) has grown more user-friendly. The traditional way of inputting data using keyboard has been replaced gradually since a lot of new method developed. Touch panel developed from single-touch to multi-touch. Body sense technology even removes obstacles in input device so HCI becomes closer to human nature movement. The most important thing from HCI is manipulating by using hand. The object of this research is to build an application which gives user ability to control pointer by using hand gesture.

Method which is used in the application to recognize hand gesture is Convex Hull Algorithm. Convex Hull Algorithm is a method which is used to detect contour or object in image by using points which are connected by lines that cover the contour.

The result of this research shows that the application could give the user ability to control the pointer by using hand gesture. Nevertheless, the overall object detections are still needs to be improved because they are easily distracted by background or object having the same colors as the user's hand.

***Keywords—** Pointer Controller, Convex Hull, Hand Gesture, Webcam*

1. PENDAHULUAN

Perkembangan HCI (*Human Computer Interface*) semakin pesat dan menuju ke arah yang semakin *user-friendly*. Baru-baru ini ada beberapa terobosan yang inovatif mengenai HCI. Cara memasukkan data secara tradisional menggunakan *keyboard* telah diganti secara bertahap sejak banyak metode baru yang ditemukan. *Touch panel*

berkembang dari *single-touch* ke *multi-touch*. Teknologi *body sense* bahkan menghilangkan hambatan pada alat *input* sehingga membuat HCI semakin dekat dengan gerak alami manusia. Bagian yang paling penting dari HCI adalah manipulasi menggunakan tangan. Oleh karena itu, dibutuhkan sistem yang akurat dalam menemukan beberapa fitur penting dari tangan manusia.

Peralatan digital saat ini berkembang menjadi bermacam-macam tipe yang berbeda, *smartphone* dan komputer *tablet* adalah yang paling banyak digunakan saat ini. Alat-alat tersebut mengubah cara manusia menggunakan komputer. Juga kehadiran Kinect dan Wii mengubah pengetahuan manusia tentang *interface* manipulasi. Kinect adalah alat pendeteksi gerakan yang dikembangkan oleh Microsoft untuk *video game console* Xbox 360 dan Xbox One dan Windows PC yang memungkinkan *user* untuk mengendalikan dan beinteraksi dengan *console* atau komputer tanpa membutuhkan *game controller*. Adapun Wii adalah *video game console* kelima Nintendo yang memiliki *joystick* yang merespon terhadap letaknya dalam ruang fisik tiga dimensi yang terletak di depan televisi. Akibatnya, alat *input* tradisional seperti *keyboard* dan *mouse* mulai tergantikan.

Di masa depan, komputer bukan hanya berupa komputer saja. Banyak alat sehari-hari yang akan diintegrasikan dengan komputer, seperti mobil, televisi, alat-alat rumah tangga, dsb. Dibutuhkan berbagai jenis HCI untuk menyesuaikan dengan tipe alat yang berbeda karena banyaknya tipe alat yang digunakan. Isyarat tangan akan menjadi yang paling populer karena merupakan cara yang paling intuitif bagi manusia untuk berkomunikasi dengan tubuh mereka.

Komunikasi menggunakan tangan menjadikan orang-orang tidak perlu lagi menggunakan peralatan komputer pada lokasi tertentu. Contohnya, jika sebuah TV diintegrasikan dengan fungsi *gesture recognition*, orang-orang bisa mengontrolnya dengan mudah dari jarak yang jauh selama masih di dalam jangkauan sensornya, *remote control* tidak dibutuhkan lagi. Jurang pemisah antara manusia dan komputer akan semakin dekat. Akan ada lebih banyak produk semisal yang muncul sehingga *gesture interface* akan menjadi lebih penting.

Terdapat berbagai macam metode atau algoritma yang digunakan untuk membangun sistem yang memiliki fungsi *gesture recognition*, salah satu di antaranya adalah Algoritma *Convex Hull*. Algoritma *Convex Hull* memiliki kelebihan jika dibandingkan dengan algoritma lain, misalnya seperti Algoritma *Speeded-Up Robust Feature* (SURF). Kelebihan yang dimiliki oleh Algoritma *Convex Hull* jika dibandingkan dengan Algoritma SURF yaitu kecepatan pemrosesan citra oleh Algoritma *Convex Hull* lebih cepat dari pada kecepatan pemrosesan citra oleh Algoritma SURF.

Berdasarkan latar belakang yang telah diuraikan maka penulis mengambil judul tugas akhir **"**PENGENDALI FUNGSI *POINTER* BERBASIS *HAND GESTURE* MENGGUNAKAN ALGORITMA *CONVEX HULL***"** yang akan menjadi solusi bagi *user* untuk mengendalikan *pointer* dengan menggunakan *hand gesture*.

Berdasarkan yang telah dipaparkan sebelumnya, permasalahan yang akan dibahas adalah bagaimana merancang suatu sistem yang dapat mengendalikan *pointer* berdasarkan *input* berupa *hand gesture* dari *user*.

Dari beberapa masalah yang telah diidentifikasikan, permasalahan yang dibahas dalam penelitian ini dibatasi sebagai berikut:

1. Aplikasi dibangun menggunakan Algoritma *Convex Hull*.
2. Aplikasi dapat mendeteksi isyarat dari tangan kiri maupun tangan kanan.
3. Operasi yang dapat dilakukan oleh aplikasi adalah *move*, *left click*, dan *right click*.
4. *User* harus berada di depan jendela dalam keadaan cuaca yang cerah atau di ruangan tertutup yang cukup gelap tapi ada pencahayaan lampu (lampu LED) dari arah depan untuk menerangi tangan *user* agar aplikasi dapat membedakan dengan jelas antara *background* dengan tangan *user* karena aplikasi sensitif terhadap perubahan cahaya.
5. *Background* harus polos atau bisa juga tidak polos tetapi tidak ada obyek yang berwarna sama dengan tangan *user* setelah terkena cahaya.

Jarak antara tangan dengan *webcam* minimal 50 cm dan maksimal 65 cm pada kondisi cahaya yang memadai, sedangkan pada kondisi lain jarak minimal dan maksimal

antara tangan dengan *webcam* dapat berubah sesuai dengan intensitas cahaya.

2. METODE PENELITIAN

2.1 Interaksi Manusia dan Komputer

Interaksi manusia dan komputer adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar yang berhubungan dengannya [1].

Sebagaimana penggunaan komputer menjadi lebih tersebar luas, jumlah peneliti yang mengkhususkan diri mempelajari interaksi antara manusia dan komputer pun ikut bertambah, baik itu mengenai diri sendiri dengan aspek fisik, psikologis, maupun teoritis dari proses ini. Penelitian ini awalnya memiliki nama interaksi manusia dan mesin, tapi berubah menjadi interaksi manusia dan komputer sebagai pengakuan atas kepentingan tertentu dalam komputer dan komposisi populasi pengguna.

Namun, ketika interaksi manusia dan komputer yang dibahas, tidak perlu dibayangkan satu *user* dan satu komputer. *User* dapat berarti seorang pengguna, sekelompok pengguna yang bekerja bersama-sama, atau seurutan pengguna dalam organisasi, masing-masing berurusan dengan beberapa bagian dari tugas atau proses. Pengguna adalah siapa pun yang mencoba menyelesaikan pekerjaannya menggunakan teknologi komputer.

Komputer dapat berarti teknologi apapun berkisar dari komputer *desktop* hingga sistem komputer berskala besar, sebuah sistem kontrol proses dan sistem tertanam. Sistem tersebut dapat mencakup bagian-bagian yang tak terkomputerisasi, termasuk orang lain.

Interaksi dapat berarti komunikasi apa saja antara *user* dan komputer, baik itu secara langsung maupun tidak langsung. Interaksi langsung melibatkan dialog dengan umpan balik dan kontrol seluruh kinerja tugas. Interaksi tidak langsung mungkin melibatkan proses *batch* atau sensor cerdas untuk mengendalikan lingkungan kerja. Hal yang penting adalah bahwa *user* berinteraksi dengan komputer dengan tujuan untuk menyelesaikan sesuatu [2].

2.2 Antarmuka Pengguna

Antarmuka pengguna adalah bagian dari komputer dan perangkat lunaknya yang bisa manusia lihat, dengar, sentuh, berbicara kepadanya, memahami, atau memerintahnya. Antarmuka pengguna memiliki dua komponen utama yaitu *input* dan *output*. *Input* adalah bagaimana seseorang menyampaikan kebutuhan atau keinginannya kepada komputer.

Beberapa komponen *input* yang umum adalah *keyboard*, *mouse*, *trackball*, jari seseorang (untuk *touch-sensitive screen* atau *touch pad*), dan suara seseorang (untuk perintah yang diucapkan). *Output* adalah bagaimana komputer menyampaikan hasil komputasi dan keperluannya pada pengguna. Saat ini mekanisme *output* komputer yang paling umum adalah monitor, diikuti dengan mekanisme yang memanfaatkan kemampuan pendengaran seseorang yaitu suara dan bunyi [3].

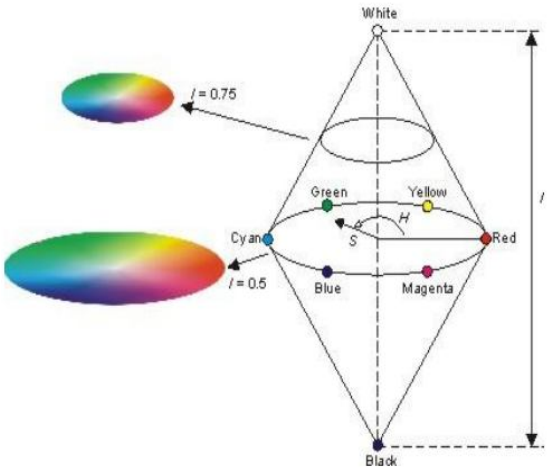
2.3 Deteksi Warna Kulit Menggunakan Ruang Warna HSV

Model HSV terdiri dari tiga komponen warna yaitu *hue*, *saturation*, dan *value*. Ia dapat membedakan *value* dari *hue* dan *saturation*. Hue mewakili gradasi dari warna dalam spektrum optik atau spektrum cahaya yang dapat dilihat. *Saturation* adalah intensitas dari *hue* tertentu yang berdasarkan pada kemurnian sebuah warna. Kepraktisan dari model HSV dapat dilihat dari dua faktor yaitu *value* terpisah dari warna citra, dan *hue* dan *saturation* berhubungan dengan penglihatan manusia sehingga untuk membuat sistem berbasis visi manusia, model HSV adalah metode yang ideal [4].

Komponen *chroma* (*hue* dan *saturation*), dan komponen *luminance* (*value*) dari model warna HSV ditentukan dengan gambar RGB sebagaimana yang ditunjukkan pada Gambar 1 yang terdiri dari tiga warna utama yaitu merah (*red*), hijau (*green*), dan biru (*blue*) yang berada dalam lingkaran kromatik. *Hue* atau H adalah sudut vektor yang sesuai dengan *axis* merah. Ketika H = 0°, warnanya merah. *Saturation* atau S adalah derajat warna yang belum ditambahkan dengan putih. Jarak dari lokasi hingga ke titik tengah sebanding dengan *Saturation*. Semakin jauh jaraknya dari pusat lingkaran kromatik, semakin tinggi pula *saturation*. *Value* atau V diukur dengan garis

yang tegak lurus terhadap lingkaran kromatik dan melewati pusat lingkaran [5].

Gambar 1 menunjukkan model HSV.



Gambar 1 Model HSV

Hubungan antara model HSV dan model RGB yaitu:

$$R' = \frac{R}{255}, G' = \frac{G}{255}, B' = \frac{B}{255} \quad (1)$$

$$Cmax = \max(R', G', B') \quad (2)$$

$$Cmin = \min(R', G', B') \quad (3)$$

$$\Delta = Cmax - Cmin \quad (4)$$

$$H = \begin{cases} 0°, & jika\ \Delta = 0 \\ \frac{60° * (G' - B')}{\Delta} mod\ 6, & jika\ Cmax = R' \\ 60° * \left[\frac{B' - R'}{\Delta} + 2\right], & jika\ Cmax = G' \\ 60° * \left[\frac{R' - G'}{\Delta} + 4\right], & jika\ Cmax = B' \end{cases} \quad (5)$$

$$S = \begin{cases} 0, & jika\ Cmax = 0 \\ \frac{\Delta}{Cmax}, & jika\ Cmax \neq 0 \end{cases} \quad (6)$$

$$V = Cmax \quad (7)$$

Keterangan:

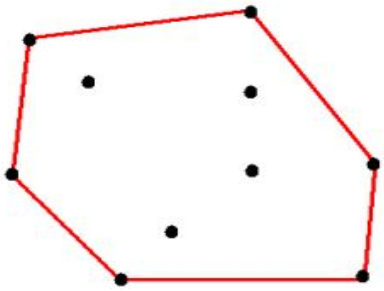
R = intensitas warna merah (*red*)
G = intensitas warna hijau (*green*)
B = intensitas warna biru (*blue*)
R' = *R* dibagi dengan 255 agar rentangnya berkisar antara 0 hingga 1
G' = *G* dibagi dengan 255 agar rentangnya berkisar antara 0 hingga 1
B' = *B* dibagi dengan 255 agar rentangnya berkisar antara 0 hingga 1
Cmax = nilai yang tertinggi antara *R'*, *G'*, dan *B'*
Cmin = nilai yang terendah antara *R'*, *G'*, dan *B'*
Δ = Selisih antara *Cmax* dan *Cmin*
H = *hue*
S = *saturation*
V = *value*

Warna kulit dibedakan dengan yang bukan warna kulit dengan melakukan pengujian terhadap batas atas dan batas bawah rentang *hue*, *saturation*, dan *value* sehingga didapatkan rentang HSV yang sesuai untuk membentuk citra hasil *threshold* untuk tangan. Sebagai contoh, pada Gambar 2 nilai H ditentukan dari 0 sampai 180, nilai S ditentukan dari 0 sampai 41, dan nilai V ditentukan dari 210 sampai 255. Gambar 2 menunjukkan hasil *threshold* dengan nilai rentang HSV.

Gambar 2 Hasil *threshold* dengan nilai rentang HSV

2.4 Convex Hull

Untuk sekumpulan titik pada sebuah bidang, *convex hull* dari kumpulan titik tersebut adalah *convex polygon* terkecil yang mengelilingi semua titik pada kumpulan titik tersebut. Sebagai contoh, pada Gambar 3 terdapat 10 titik, segi enam pada gambar tersebut adalah *convex hull* dari kumpulan titik tersebut. Enam titik yang membentuk segi enam disebut "*hull points*" [5].

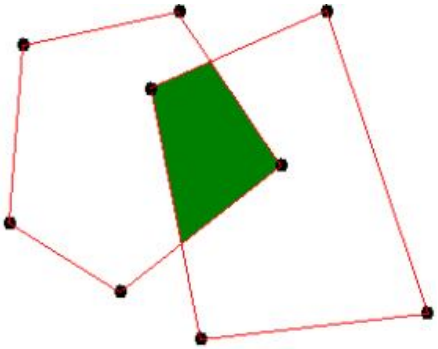


Gambar 3 Contoh *convex hull*

Diketahui bahwa satu himpunan S dalam sebuah bidang atau dalam sebuah ruang adalah *convex polygon* (atau himpunan *convex*) jika dan hanya jika titik X dan Y ada di dalam S, garis XY harus berada di dalam S. Titik potong dari kumpulan manasaja dari himpunan *convex*

adalah *convex* juga, sebagaimana yang ditunjukkan pada Gambar 6. Untuk sembarang himpunan W, *convex hull* dari W adalah titik potong dari semua himpunan *convex* yang berisi W. Batas dari *convex hull* adalah sebuah *polygon* dengan semua titik puncak dalam W [5].

Gambar 4 menunjukkan titik potong dari himpunan *convex*.



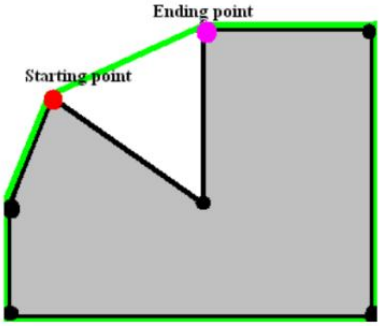
Gambar 4 Titik potong manasaja dari himpunan *convex* adalah juga sebuah himpunan *convex*

2.5 Convexity defect

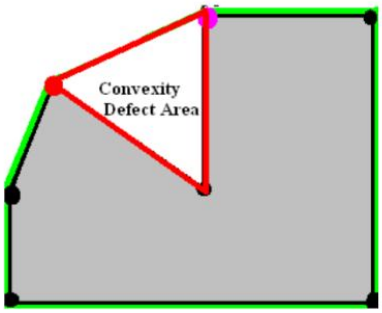
Convex hull dari kontur lengan bawah dihitung untuk mendapatkan *convexity defect* dari kontur. *Convexity defect* menyediakan informasi yang sangat berguna untuk memahami bentuk kontur. Banyak karakteristik dari kontur yang rumit dapat digambarkan dengan *convexity defect*.

Pada pembahasan sebelumnya, dijelaskan bahwa titik-titik yang membentuk *convex hull* harus merupakan bagian dari kontur. Langkah pertama dalam mencari *convexity defect* adalah menemukan titik awal (*starting point*) dari *convexity defect* pada kontur. Titik awal *convexity defect* adalah sebuah titik pada kontur yang juga termasuk dalam titik-titik *convex hull*, tapi titik selanjutnya pada kontur tidak termasuk dalam titik-titik *convex hull*. Gambar 5 menjelaskan titik awal dari *convexity defect*. Kontur dicari dengan jalur searah jarum jam. Titik merah adalah titik pertama yang termasuk dalam *convex hull*, tapi titik selanjutnya tidak termasuk dalam *convex hull* [5].

Setelah definisi titik awal diketahui, titik akhir pun juga demikian. Titik akhir didefinisikan sebagai titik dari kontur yang termasuk dalam titik-titik *convex hull*, tapi titik sebelumnya tidak termasuk dalam titik-titik *convex hull*. Sebagaimana yang ditunjukkan oleh Gambar 5, titik ungu adalah titik akhir dari *convexity defect*. Dengan menghubungkan titik awal, titik akhir, dan titik di antara keduanya, area dari *convexity defect* dapat diketahui sebagaimana ditunjukkan oleh Gambar 6 [5].

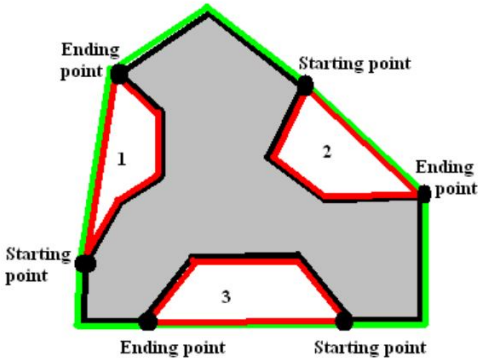


Gambar 5 Titik awal (*starting point*) dan titik akhir (*ending point*) *convexity defect*



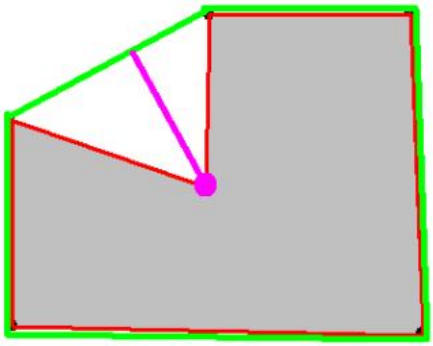
Gambar 6 Area *convexity defect*

Setelah semua titik pada kontur telah dicari, berbagai macam *convexity defect* dapat ditemukan sebagaimana yang ditunjukkan oleh Gambar 7. Setiap *convexity defect* terdiri dari titik awal, titik akhir, dan titik di antara keduanya [5].



Gambar 7 Terdapat tiga *convexity defect* pada kontur

Informasi berguna lainnya yang dapat diperoleh dari *convexity defect* selain titik awal dan titik akhir adalah *depth of defect* dan *depth point*. *Depth of defect* adalah jarak terjauh dari semua titik pada *defect* hingga ke tepi *convex hull* dari *defect*. Titik pada *defect* yang memiliki jarak terjauh ke tepi *convex hull* dari *defect* adalah adalah *depth point* [5]. Gambar 8 menunjukkan *depth point* dari *convexity defect.*



Gambar 8 *Depth point* dari *convexity defect*

Sebagaimana yang ditunjukkan oleh Gambar 8, hanya terdapat satu *convexity defect*. Titik ungu adalah titik yang memiliki jarak terpanjang hingga ke tepi *convex hull* dari *defect* karena panjang dari garis ungu adalah yang terpanjang maka ia termasuk *depth of defect* dan titik ungu adalah *depth point*. [5]

2.6 OpenCV

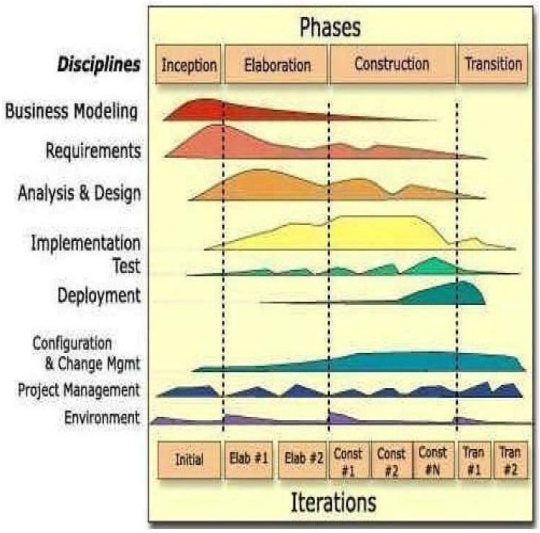
OpenCV (*Open Source Computer Vision*) adalah *open source library* yang berisi lebih dari 500 algoritma teroptimasi untuk analisa citra atau *video*. Sejak diperkenalkan pada tahun 1999, *OpenCV* sebagian besar telah diadopsi sebagai alat pengembangan utama oleh komunitas peneliti dan pengembang dalam bidang komputer visi. *OpenCV* pada awalnya dikembangkan di Intel oleh tim yang dipimpin oleh Gary Bradski sebagai inisiatif untuk memajukan penelitian dalam visi dan mempromosikan pengembangan aplikasi yang kaya dan berbasis visi-CPU-intensif. Setelah *OpenCV* versi *Beta* diluncurkan, versi 1.0 diluncurkan pada tahun 2006. Rilis besar kedua terjadi pada tahun 2009 dengan diluncurkannya *OpenCV* 2 yang menawarkan banyak perubahan penting [6].

Library OpenCV adalah suatu cara untuk membangun komunitas *open source vision* yang akan lebih memanfaatkan kesempatan *up-to-date* untuk menerapkan komputer visi di lingkungan PC yang sedang tumbuh. *OpenCV* menyediakan satu set fungsi pengolahan citra, serta fungsi analisis pola dan citra.

OpenCV didesain untuk digunakan bersama dengan *Intel Image Processing Library* (IPL) dan memperluas fungsionalitas terhadap citra dan pola analisis. Oleh karena itu, *OpenCV* berbagi format (*iplImage*) citra yang sama dengan IPL. Selain itu, *OpenCV* juga menggunakan *Intel Integrated Performance Primitive* (IPP) pada level yang lebih rendah, jika lokasi biner IPP bisa ditemukan pada *startup* [7].

2.7 Rational Unified Process (RUP)

Rational Unified Process (RUP) merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *best practises* yang terdapat dalam industri pengembangan perangkat lunak. Ciri utama metode ini adalah menggunakan *use case driven* dan pendekatan iteratif untuk siklus pengembangan perangkat lunak. RUP menggunakan konsep *object oriented*, dengan aktivitas yang berfokus pada pengembangan model dengan menggunakan *Unified Modeling Language* (UML) [8]. Gambar 9 menunjukkan arsitektur *Rational Unified Process*.



Gambar 9 Arsitektur *Rational Unified Process*

Fase RUP terdiri dari beberapa langkah yaitu:

1. *Inception*/Insepsi
   - Menentukan ruang lingkup proyek.
   - Membuat *Business Case.*
   - Menjawab pertanyaan "apakah yang dikerjakan dapat menciptakan *good*

business sense" sehingga proyek dapat dilanjutkan.

2. *Elaboration*/Elaborasi
   - Menganalisa berbagai persyaratan dan resiko.
   - Menetapkan *base line.*
   - Merencanakan fase berikutnya yaitu *construction.*
3. *Construction*/Konstruksi
   - Melakukan sederetan iterasi.
   - Pada setiap iterasi akan melibatkan beberapa proses yaitu: analisa desain, implementasi, dan *testing*.
4. *Transition*/Transisi
   - Membuat apa yang sudah dimodelkan menjadi suatu produk jadi.
   - Dalam fase ini dilakukan:
     a. *Beta and performance testing.*
     b. Membuat dokumentasi tambahan seperti: *training*, *user guides*, dan *sales kit.*
     c. Membuat rencana peluncuran produk ke komunitas pengguna.

2.8 C#

C# (dibaca *C-Sharp*) adalah bahasa pemrograman untuk .Net *Environment*. C# adalah bahasa baru yang miskin kompatibilitas tapi memiliki banyak fitur yang menarik dan menjanjikan. C# adalah bahasa pemrograman berbasis obyek yang memiliki inti, banyak kemiripannya dengan Java, C++, dan VB. Kenyataannya, C# menggabungkan efisiensi dari C++, desain berbasis obyek yang sederhana dan bersih dari Java dan penyederhanaan bahasa dari *Visual Basic.*

Seperti Java, C# juga tidak membolehkan *multiple inhetitance* atau penggunaan *pointer* (pada *safe*/*managed code*), tapi menyediakan *garbage memory collection* pada *runtime*, tipe dan pengecekan akses *memory*. Akan tetapi, berlawanan dengan Java, C# mempertahankan operasi berguna yang unik pada C++ seperti *operator overloading*, *enumeration*, *pre-processor directive*, *pointer* (pada *unmanaged/unsafe code*), *function pointer* (pada pengutusan *form*) dan janji untuk memiliki *support template* pada versi selanjutnya [10].

2.9 Flowchart

Bagan alir (*flowchart*) adalah bagan (*chart*) yang menunjukkan alir (*flow*) di dalam program atau prosedur sistem secara logika. Bagan alir digunakan terutama untuk alat bantu komunikasi dan untuk dokumentasi.

2.10 Metode Pengumpulan Data dan Informasi

Adapun metode pengumpulan data dan informasi adalah studi pustaka, yaitu teknik pengumpulan data dengan menghimpun dan menganalisis dokumen. Dokumen yang termasuk di dalamnya yaitu penelitian terdahulu, buku, artikel, dan jurnal yang berkaitan dengan objek penelitian. Secara rinci yang penulis lakukan pada studi pustaka ini adalah:

1. Mempelajari buku-buku atau dokumen-dokumen dan artikel-artikel yang berhubungan dengan Algoritma *Convex Hull* dan aplikasinya.
2. Menganalisa bahasa pemrograman yang sesuai untuk mengkontrol *pointer* dengan *hand gesture* dengan bantuan *webcam.*
3. Mempelajari penelitian sebelumnya yaitu mengenali *hand gesture* dengan menggunakan metode *Speeded-Up Robust Feature* (SURF), *Hue Saturation Value color space*, dan *Convex Hull* yang akan diimplementasikan pada penelitian ini.

2.11 Analisis dan Perancangan

A. Analisis Sistem

Analisis sistem merupakan suatu tahapan yang bertujuan untuk mengetahui dan mengamati apa saja yang terlibat dalam suatu sistem. Pembahasan yang ada pada analisis sistem ini yaitu analisis masalah, analisis algoritma, analisis kebutuhan nonfungsional, dan analisis kebutuhan fungsional.

B. Analisis Masalah

Berdasarkan hasil pengamatan sementara diperoleh permasalahan yaitu penggunaan *mouse* dalam mengendalikan *pointer* sudah merupakan hal yang biasa atau tradisional dan sudah terdapat peralatan digital berbasis *gesture* yang dapat menggantikannya. Peralatan digital tersebut saat ini digunakan untuk meningkatkan kemudahan dan kenyamanan *user* dalam mengendalikan *pointer*.

Salah satu cara untuk meningkatkan kemudahan dan kenyamanan *user* dalam mengendalikan *pointer* adalah dengan menggunakan *webcam* sebagai pendeteksi *hand gesture* kemudian mengolahnya menjadi

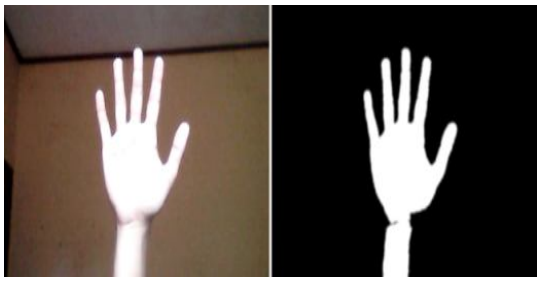
gerakan atau eksekusi dari *pointer* dengan menggunakan metode tertentu. Salah satu metode yang dapat digunakan dan handal dalam mengolah *hand gesture* adalah Algoritma *Convex Hull*.

C. Analisis Ruang Warna HSV

Ruang warna HSV (HSV *color space*) terdiri dari tiga komponen warna yaitu *hue*, *saturation*, dan *value*. HSV memiliki kelebihan jika diterapkan dalam sistem berbasis visi manusia yaitu *value* terpisah dari warna citra, dan *hue* dan *saturation* berhubungan dengan penglihatan manusia.

Ruang warna HSV biasanya digunakan untuk mendeteksi warna kulit manusia sehingga dalam penelitian ini, HSV digunakan untuk melakukan deteksi terhadap warna kulit telapak tangan hingga lengan sehingga hasil *threshold*-nya hanya menampakkan telapak tangan hingga sedikit bagian dari lengan. Warna kulit dibedakan dengan yang bukan warna kulit dengan melakukan pengaturan terhadap batas atas dan batas bawah rentang *hue*, *saturation*, dan *value* sehingga membentuk citra hasil *threshold*.

Gambar 10 menunjukkan citra hasil *threshold* dengan nilai rentang HSV.



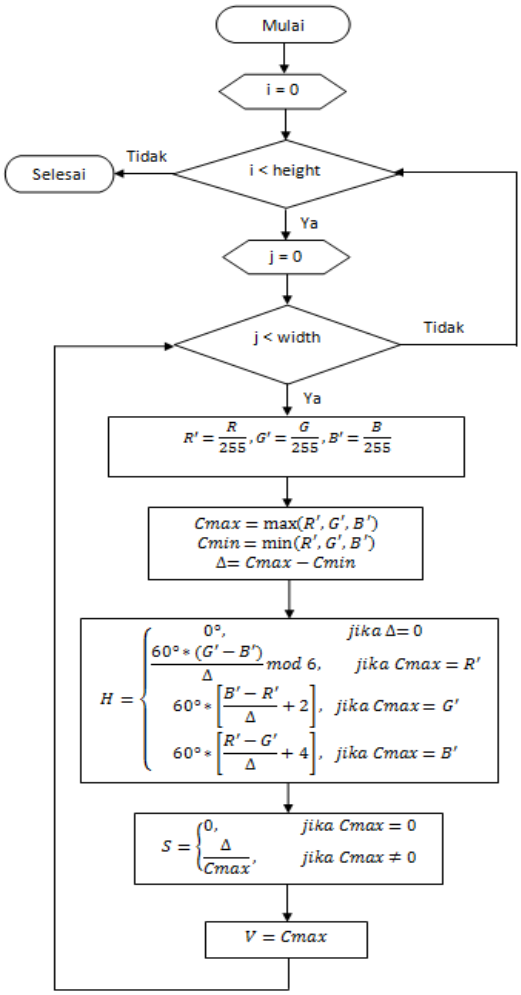
Gambar 10 Hasil *threshold* dengan nilai rentang HSV

Nilai HSV dibentuk dari nilai RGB (*red*, *green*, *blue*) yang telah dikonversi. Sebagai contoh, terdapat citra 2x2 dengan nilai RGB yang sama, yaitu R = 70, G = 255, B = 34. Gambar 11 menunjukkan citra RGB 2x2.

R=70 G=255 B=34	R=70 G=255 B=34
R=70 G=255 B=34	R=70 G=255 B=34

Gambar 11 Citra RGB 2x2

Kemudian akan dilakukan konversi ke HSV terhadap citra tersebut dengan langkah-langkah dalam yang terdapat dalam *flowchart* yang ditunjukkan oleh Gambar 12.



Gambar 12 *Flowchart* konversi RGB ke HSV

Citra 2x2 pada Gambar 11 terdapat 4 *pixel*, jika diurutkan berdasarkan koordinat *width* (lebar) dan *height* (tinggi)-nya maka yang pertama *pixel* 0,0, kedua *pixel* 0,1, ketiga *pixel* 1,0 dan yang keempat *pixel* 1,1. Pemrosesan citra akan dimulai secara berurutan dari 0,0 hingga 1,1. Keempat *pixel* memiliki nilai RGB yang sama sehingga perhitungannya cukup sekali saja untuk mewakili yang lain ketika dikonversi ke HSV. Perhitungannya yaitu:

$$R = 70;\ G = 255;\ B = 34$$

$$R' = \frac{70}{255} = 0{,}27;$$

$$G' = \frac{255}{255} = 1;$$

$$B' = \frac{34}{255} = 0{,}13$$

dirumuskan pada Persamaan (1),

$$Cmax = \max(0{,}27, 1, 0{,}13) = 1$$

dirumuskan pada Persamaan (2),

$$Cmin = \min(0{,}27, 1, 0{,}13) = 0{,}13$$

dirumuskan pada Persamaan (3),

$$\Delta = Cmax - Cmin = 1 - 0{,}13 = 0{,}87$$

dirumuskan pada Persamaan (4), karena $Cmax = G'$, maka

$$H = 60° * \left[\frac{B' - R'}{\Delta} + 2\right] = 60° * \left[\frac{0{,}13 - 0{,}27}{0{,}87} + 2\right] = 110{,}34°$$

dirumuskan pada Persamaan (5), karena $Cmax \neq 0$, maka

$$S = \frac{\Delta}{Cmax} = \frac{0{,}87}{1} = 0{,}87$$

dirumuskan pada Persamaan(6),

$$V = Cmax = 1$$

dirumuskan pada Persamaan (7).

Jadi, $H = 110{,}34°$, $S = 0{,}87$, $dan\ V = 1$.

Gambar 13 menunjukkan citra HSV 2x2 hasil konversi dari citra RGB 2x2.

H = 110,34 ° S = 0,87 V = 1	H = 110,34° S = 0,87 V = 1
H = 110,34 ° S = 0,87 V = 1	H = 110,34° S = 0,87 V = 1

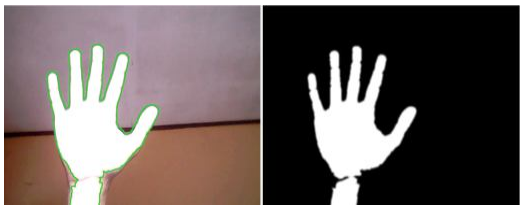
Gambar 13 Citra HSV 2x2 hasil konversi dari citra RGB 2x2

D. Analisis Algoritma Convex Hull

Convex hull adalah metode yang digunakan untuk mendeteksi kontur atau obyek pada citra. *Convex hull* terdiri dari titik-titik yang dihubungkan dengan garis yang mengelilingi kontur (obyek pada citra). Sebelum mencari *convex hull* pada obyek, pertama-tama obyek harus dipilih dari obyek-obyek yang masih tersisa dari hasil *filter* HSV. Obyek adalah kontur berwarna putih yang dihasilkan setelah proses *thresholding*. Obyek yang dipilih adalah obyek yang terbesar di antara semua obyek yang ada pada citra hasil *threshold*.

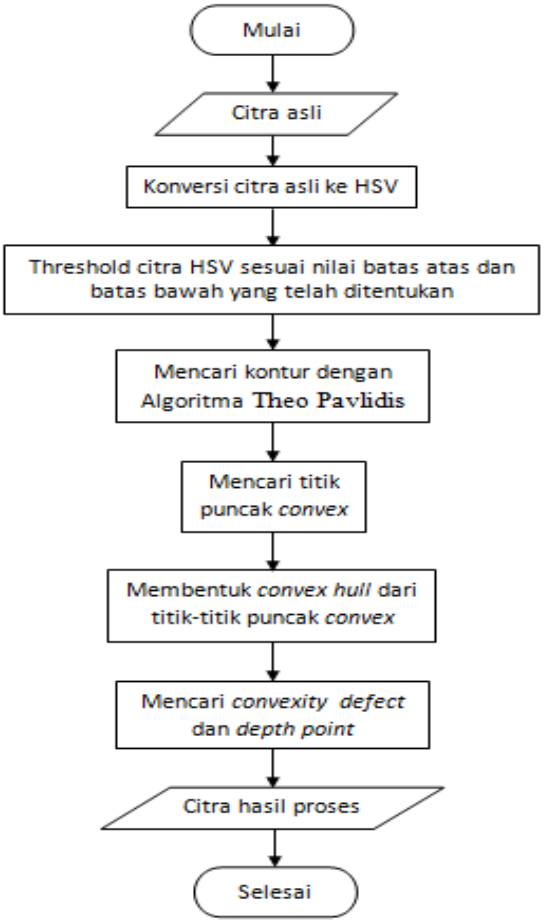
Obyek yang dimaksud di sini adalah tangan *user*. Tangan *user* diletakkan di depan *webcam* kemudian akan dilakukan *thresholding*, maka akan tampak hasil *threshold*. Dari hasil *threshold* tersebut akan muncul satu atau beberapa area yang berwarna putih yang disebut dengan obyek.

Setelah kontur-kontur didapatkan, dari kontur-kontur tersebut akan dicari kontur yang paling besarnya (obyek tangan harus yang paling besar) sebelum dilakukan pencarian *convex hull*-nya. Setelah obyek yang paling besar ditemukan, maka obyek tersebut akan ditandai dengan garis warna hijau yang mengelilingi seluruh sisinya, seperti yang ditunjukkan oleh Gambar 14.



Gambar 14 Obyek tangan ditandai dengan garis hijau

Setelah obyek tangan ditandai dengan garis hijau, maka akan dilakukan pencarian *convex hull*. Gambar 15 menunjukkan *flowchart* Algoritma *Convex Hull*.



Gambar 15 *Flowchart* algoritma *Convex Hull*

E. Analisis Kebutuhan Fungsional

Analisis kebutuhan nonfungsional adalah salah satu langkah dalam pembangun aplikasi untuk menganalisis sumber daya yang dibutuhkan untuk membangun suatu aplikasi. Analisis kebutuhan nonfungsional dibagi dalam dua tahap, yaitu analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak.

1. Analisis Kebutuhan Perangkat Keras

Perangkat keras yang digunakan dalam membangun Aplikasi *Hand Gesture Pointer* ditunjukkan oleh Tabel 1.

Tabel 1 Spesifikasi perangkat keras

No	Nama Perangkat	Spesifikasi
1	*Processor*	*Intel Core* i3 2.40 GHz
2	*Monitor*	*Monitor* LCD (1366x768)
3	*Memory*	RAM 2 GB DDR3
4	*Disk Drive*	120 GB SSD

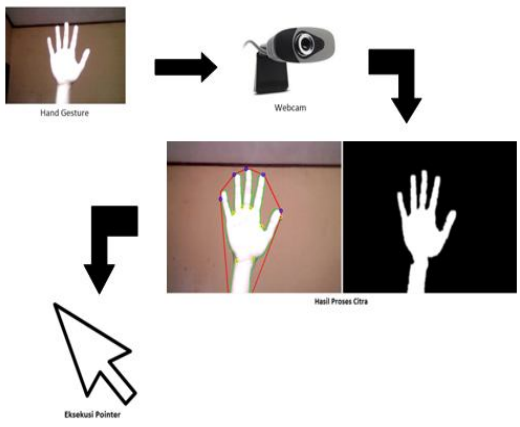
2. Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan dalam membangun aplikasi ini ditunjukkan oleh Tabel 2.

No	Nama Perangkat	Spesifikasi
1	*Operating System*	*Windows 7 Ultimate 32-bit*
2	*Framework*	*Microsoft .Net Framework* 4.5.2
3	*Application* IDE	*SharpDevelop* 5.1 RC

F. Arsitektur Sistem

Adapun arsitektur sistem ditunjukkan oleh Gambar 16.

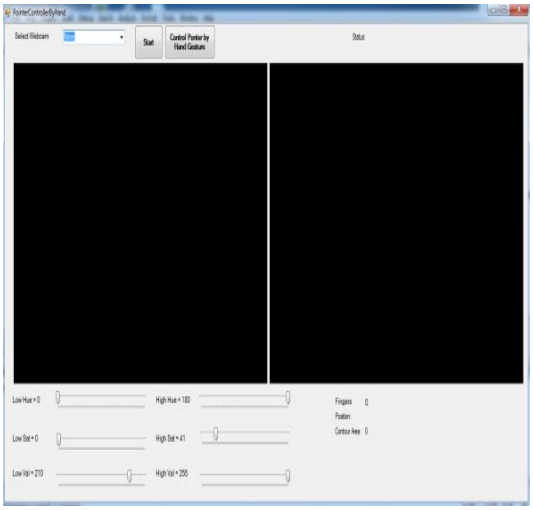


Gambar 16 Arsitektur sistem

3. HASIL DAN PEMBAHASAN

3.1 Implementasi

Implementasi dari Aplikasi *Hand Gesture Pointer* ditunjukkan oleh Gambar 17.



Gambar 17 *Form* aplikasi

Pada *form* aplikasi, pertama-tama *user* memilih *webcam*. Kemudian *user* menekan tombol *Start* untuk memulai pengambilan *frame video*. Setelah itu meletakkan telapak tangannya di depan *webcam* sambil mengatur rentang HSV hingga hasil *threshold* hanya mencakup telapak tangan dan sedikit bagian dari lengan. *User* dapat melihat hasil pemrosesan citra pada *imageBox*. Jika hasil *threshold* telah sesuai, maka *user* sudah dapat menekan tombol *Control Pointer by Hand Gesture*. Setelah itu *user* dapat melihat *pointer* bergerak (melakukan eksekusi *move*) seiring dengan gerakan telapak tangannya dan *user* juga dapat memberi perintah *left click* dengan memberi *hand gesture* angka satu dan *right click* dengan memberi *hand gesture* angka dua. Untuk perintah *move*, *user* dapat menggunakan *hand gesture* angka satu, dua, tiga, empat, dan lima.

3.2 Pengujian

Pada pengujian Aplikasi *Hand Gesture Pointer* ini, bahan uji coba yaitu tangan *user*nya adalah tangan penyusun skripsi ini sendiri. Uji coba dilakukan untuk mengetahui apakah *hand gesture* (jumlah jari tangan) dari *user* akan memberikan perintah *pointer* yang sesuai dengan yang telah dirancang atau tidak, serta pengaruh *background* dan obyek selain tangan yang tertangkap oleh *webcam* terhadap

deteksi *hand gesture*. Setelah dilakukan beberapa kali pengujian, didapatkan nilai rentang HSV yang optimal yaitu H = 0 – 180, S = 0 – 41, dan V = 210 – 255. Nilai rentang tersebut dijadikan nilai rentang *default* untuk aplikasi. Hasil pengujian ditampilkan dalam tabel yang memperlihatkan data hasil uji coba.

Skenario 1: Uji coba dilakukan pada *background* yang polos dan kondisi pencahayaan yang baik, yaitu di depan jendela pada siang hari dan cuaca cerah. Hasil pengujian ditunjukkan oleh Tabel 3,

Tabel 3 Data hasil uji coba skenario 1

No	Jumlah Jari Tangan yang terdeteksi	Perintah *Pointer*	Keterangan
1	1	*Left click* atau *move*	Stabil
2	2	*Right click* atau *move*	Stabil
3	3	*Move*	Stabil
4	4	*Move*	Stabil
5	5	*Move*	Stabil

Skenario 2: Uji coba dilakukan pada *background* yang polos tapi ada obyek (*Tablet* PC) yang warnanya sama dengan tangan ketika terkena cahaya, dan kondisi pencahayaan yang bagus, yaitu di depan jendela pada siang hari dan kondisi cuaca cerah. Hasil pengujian ditunjukkan oleh Tabel 4.

Tabel 4 Data hasil uji coba skenario 2

No	Jumlah Jari Tangan yang terdeteksi	Perintah *Pointer*	Keterangan
1	1	*Left click* atau *move*	Tidak stabil, jari tangan berubah-ubah
2	2	*Right click* atau *move*	Tidak stabil, jari tangan berubah-ubah
3	3	*Move*	Tidak stabil, jari tangan berubah-ubah
4	4	*Move*	Tidak stabil, jari tangan berubah-ubah
5	5	*Move*	Tidak stabil, jari tangan berubah-ubah

Skenario 3: Uji coba dilakukan pada *background* yang polos tetapi pencahayaan yang kurang baik, yaitu di ruangan tertutup dan jendela ditutup dengan gorden. Hasil pengujian ditunjukan oleh Tabel 5.

Tabel 5 Data hasil uji coba skenario 3

No	Jumlah Jari Tangan yang terdeteksi	Perintah *Pointer*	Keterangan
1	0	Tidak ada	Tidak terdeteksi
2	0	Tidak ada	Tidak terdeteksi
3	0	Tidak ada	Tidak terdeteksi
4	0	Tidak ada	Tidak terdeteksi
5	0	Tidak ada	Tidak terdeteksi

Skenario 4: Uji coba dilakukan pada *background* yang tidak polos dan pencahayaan kurang baik yaitu di ruangan tertutup dan jendela pun ditutup dengan gorden. Hasil pengujian ditunjukkan oleh Tabel 6.

Tabel 6 Data hasil uji coba skenario 4

No	Jumlah Jari Tangan yang terdeteksi	Perintah *Pointer*	Keterangan
1	0	Tidak ada	Tidak terdeteksi
2	0	Tidak ada	Tidak terdeteksi
3	0	Tidak ada	Tidak terdeteksi
4	0	Tidak ada	Tidak terdeteksi
5	0	Tidak ada	Tidak terdeteksi

Skenario 5: Uji coba dilakukan pada *background* yang polos, dan pencahayaan dari arah depan *user* dengan menggunakan lampu LED di ruangan yang tertutup dan jendela ditutup dengan gorden. Hasil pengujian ditunjukkan oleh Tabel 7.

Tabel 7 Data hasil uji coba skenario 5

No	Jumlah Jari Tangan yang terdeteksi	Perintah *Pointer*	Keterangan
1	1	*Left click* atau *move*	Stabil
2	2	*Right click* atau *move*	Stabil
3	3	*Move*	Stabil

4	4	*Move*	Stabil
5	5	*Move*	Stabil

Skenario 6: Uji coba dilakukan pada jarak lebih dari 65 cm yaitu 80 cm dari depan *webcam*, di depan jendela pada siang hari. Hasil pengujian ditunjukkan oleh Tabel 8.

Tabel 8 Data hasil uji coba skenario 6

No	Jumlah Jari Tangan yang terdeteksi	Perintah *Pointer*	Keterangan
1	1	*Left click* atau *move*	Tidak stabil, jari tangan berubah-ubah
2	2	*Right click* atau *move*	Tidak stabil, jari tangan berubah-ubah
3	3	*Move*	Tidak stabil, jari tangan berubah-ubah
4	4	*Move*	Tidak stabil, jari tangan berubah-ubah
5	5	*Move*	Tidak stabil, jari tangan berubah-ubah

Skenario 7: Uji coba dilakukan pada jarak kurang dari 50 cm yaitu 30 cm dari depan *webcam*, di depan jendela pada siang hari. Hasil pengujian ditunjukkan oleh Tabel 9.

Tabel 9 Data hasil uji coba skenario 7

No	Jumlah Jari Tangan yang terdeteksi	Perintah *Pointer*	Keterangan
1	0	Tidak terdeteksi	Tidak terdeteksi
2	0	Tidak terdeteksi	Tidak terdeteksi
3	0	Tidak terdeteksi	Tidak terdeteksi
4	0	Tidak terdeteksi	Tidak terdeteksi
5	0	Tidak terdeteksi	Tidak terdeteksi

Skenario 8: Uji coba dilakukan dengan menggunakan sarung tangan berwarna hitam di depan jendela pada siang hari. Hasil pengujian ditunjukkan oleh Tabel 10.

Skenario 9: Uji coba dilakukan dengan jari dirapatkan, di depan jendela di siang hari. Hasil pengujian ditunjukkan oleh Tabel 11.

Tabel 10 Data hasil uji coba skenario 8

No	Jumlah Jari Tangan yang terdeteksi	Perintah *Pointer*	Keterangan
1	0	Tidak ada	Tidak terdeteksi
2	0	Tidak ada	Tidak terdeteksi
3	0	Tidak ada	Tidak terdeteksi
4	0	Tidak ada	Tidak terdeteksi
5	0	Tidak ada	Tidak terdeteksi

Tabel 11 Data hasil uji coba skenario 9

No	Jumlah Jari Tangan yang terdeteksi	Perintah *Pointer*	Keterangan
1	1	*Left click* atau *move*	Stabil
2	2	*Right click* atau *move*	Stabil
3	3	*Move*	Stabil
4	4	*Move*	Stabil
5	5	*Move*	Stabil

4. KESIMPULAN

Dari penelitian dan pembahasan Aplikasi *Hand Gesture Pointer* dengan Algoritma *Convex Hull*, dapat ditarik kesimpulan yaitu aplikasi yang dibangun dapat memberikan *user* kemampuan untuk mengendalikan *pointer* dengan menggunakan *hand gesture*. Namun, pengendaliannya secara keseluruhan belum stabil karena aplikasi masih memiliki kekurangan yaitu masih mudah terganggu oleh *background* atau obyek yang memiliki warna sama dengan tangan *user*.

5. SARAN

Saran yang perlu diperhatikan untuk penelitian lebih lanjut adalah:

1. Metode pemrosesan citra dapat ditingkatkan sehingga aplikasi dapat lebih mudah dalam mengenali *hand gesture* dari *user* yaitu tidak terpengaruh lagi oleh *background* misalnya dengan menggunakan *background removal*.
2. Metode lain dapat digabungkan dengan Algoritma *Convex Hull* untuk mendapatkan

hasil pengolahan citra yang lebih baik dan lebih stabil dalam memproses citra.

DAFTAR PUSTAKA

[1] Hewett, T. T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G. dan Verplank, W., 1992, *ACM SIGCHI Curricula for Human Computer Interaction*, ACM, New York.

[2] Dix, A., Finlay, J., Abowd, G. D. dan Beale, R., 2004, *Human-Computer Interaction*, Third Edition, Prentice Hall, England.

[3] Galitz, W. O., 2007, *The Essential Guides to User Interface Design*, Third Edition, Wiley Publishing, Inc., Indiana.

[4] Gonzalez, R.C. dan Woods, R.E., 1992, *Digital Image Processing*, Third Edition, Addison Wesley, Massachusetts.

[5] Chen, W. Z., 2011, Real-Time Palm Tracking and Hand Gesture Estimation Based on Fore-Arm Contour, *Tesis*, Department of Computer Science and Information Engineering, University of Science and Technology, Taiwan.

[6] Laganiere, R., 2011, *OpenCV 2 Computer Vision Application Programming Cookbook*, Packt Publishing, Birmingham.

[7] Intel Corp., 2001, Open Source Computer Vision Library - Reference Manual, *www.cs.unc.edu/~stc/FAQs/OpenCV/OpenCVReferenceManual.pdf*, diakses pada tanggal 1 Februari 2016.

[8] Ambler, S.W., 2005, *A Manager's Introduction to the Rational Unified Process* (RUP).

[9] Munansyah, A., 2007, *Rational Unified Process*, STIKMI LIKMI, Bandung.

[10] Rasheed, F., 2006, 14 Lessons to Get You Started With C# and .Net, *http://ebooks.programmersheaven.com/csharp_ebook.pdf, diakses pada tanggal* 4 Januari 2016.