UNIVERSITAS INDONESIA

IDENTIFIKASI DAN PEMILIHAN JALUR PENGELASAN DENGAN MENGGUNAKAN MESIN *VISION*

TESIS

ALBERTUS RIANTO SURYANINGRAT
0906496163

FAKULTAS TEKNIK
PROGRAM PASCA SARJANA
DEPOK
JULI 2011

UNIVERSITAS INDONESIA

IDENTIFIKASI DAN PEMILIHAN JALUR PENGELASAN DENGAN MENGGUNAKAN MESIN *VISION*

TESIS

Diajukan sebagai salah satu syarat untuk memperoleh gelar Megister Teknik

ALBERTUS RIANTO SURYANINGRAT
0906496163

FAKULTAS TEKNIK
DEPARTEMEN TEKNIK MESIN
DEPOK
JULI 2011

HALAMAN PERNYATAAN ORISINALITAS

Tesis ini adalah karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Albertus Rianto Suryaningrat**
NPM	**: 0906496163**
Tanda Tangan	**:** 
Tanggal	**: 8 Juli 2011**

HALAMAN PENGESAHAN

Tesis ini diajukan oleh :

Nama	: Albertus Rianto Suryaningrat
NPM	: 0906496163
Program Studi	: Departemen Teknik Mesin
Judul	: Identifikasi dan Pemilihan Jalur Pengelasan dengan Menggunakan Mesin Vision

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Megister Teknik pada Program Studi Departemen Teknik Mesin Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Dr. Ir. Gandjar Kiswanto. M.Eng ( )

Penguji : Dr. Ario Sunar Baskoro, ST, MT, M.Eng ( )

Penguji : Ir. Hengky S. Nugroho, MT ( )

Penguji : Dr. Ir. Danardono A.S, DEA ( )

Ditetapkan di : Depok

Tanggal : Juli 2011

KATA PENGANTAR

Syukur kepada Tuhan Yang Maha Esa atas keberhasilan dalam kegiatan penelitian sebagai bagian dari persyaratan kelulusan untuk program megister teknik DTM UI. Di dalam kesempatan kali ini, penulis mengucapkan terimakasih kepada :

1. Dr.Ir. Gandjar Kiswanto, M.Eng selaku dosen pembimbing utama dan Dr.Ario Sunar Baskoro, ST, MT, M.Eng selaku dosen matakuliah sistem mesin *vision* yang keduanya selalu memberikan waktu, semangat, arahan, dan dorongan dalam menyelesaikan penelitian.
2. Seluruh staf pengajar, laboran, dan karyawan di lingkungan Departemen Teknik Mesin atas bantuan akademis, teknis, dan administrasi selama ini.
3. Seluruh mahasiswa pascasarjana angkatan 2009 yang selalu memberikan semangat selama menjalankan penelitian ini.
4. Teguh Santoso, ST, Jediel Billy Ramadhan, Ir. Swarsono, MT, Dede Lia Zariatin, ST, MT , Ir. Hendriko, M.Eng dan para peneliti muda lainnya khususnya di lingkungan laboratorium teknik manufaktur dan otomasi atas sharing ilmu pengetahuan dan bantuan pengambilan data di dalam penelitian selama ini.
5. Rekan kerja di Balai MEPPO yang selalu membantu dari segi dukungan moril dan administrasi selama menjalankan tugas belajar.
6. Keluarga besar Ir. Gregorius Harjanto atas dukungan dan semangatnya selama ini.

Dengan selesainya kegiatan penelitian ini, penulis banyak memperoleh bekal kemampuan untuk dipergunakan didalam kegiatan penelitian di lingkungan BPPT. Demikian yang dapat penulis sampaikan dan atas kesalahan yang pernah penulis lakukan, mohon penulis haturkan maaf yang sedalam-dalamnya.

Depok, 8 Juli 2011

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sevitas akademika Universitas Indonesia, saya yang bertanda tangan di bawah ini :

Nama : Albertus Rianto Suryaningrat
NPM : 0906496163
Program Studi : Pasca Sarjana Teknik Mesin
Departemen : Teknik Mesin
Fakultas : Teknik
Jenis Karya : Tesis

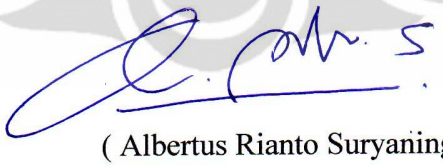
demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Nonekslusive (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul :

Identifikasi dan Pemilihan Jalur Pengelasan dengan Menggunakan Mesin Vision

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-exsklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tugas akhir saya selama mencantumkan nama saya sebagai penulis/pencipta dan pemiliki Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok
Pada tanggal: 8 Juli 2011
Yang menyatakan



(Albertus Rianto Suryaningrat)

ABSTRAK

Nama : Albertus Rianto Suryaningrat
Program Studi : Program Pasca Sarjana / Megister Teknik DTM UI
Judul : Identifikasi dan pemilihan jalur pengelasan dengan menggunakan mesin *vision*

Penggunaan teknologi mesin *vision* pada proses pengelasan telah berkembang seiring dengan kebutuhan akan hasil pengelasan yang lebih konsisten dengan proses pengambilan posisi gerak yang lebih cepat. Aplikasi mesin *vision* untuk melakukan proses analisa obyek dengan pengambilan citra pada benda kerja atau tanpa adanya kontak langsung pada material diharapkan mampu untuk menghasilkan proses yang mudah dan cepat, selama tidak mengurangi sifat keakurasian agar mampu untuk dilakukan pada proses pengelasan. Dengan mengaplikasikan algoritma *hough transform* untuk pendeteksian garis serta didukung proses pengolahan citra yang baik, maka sekumpulan garis yang berhasil terdeteksi akan dapat dipilih jalur pengelasan yang efisien. Konsep pemilihan jalur pengelasan pada penelitian yang dilakukan adalah dengan membuat kombinasi antara jarak maksimal *path* terjauh yang dapat ditempuh dilanjutkan dengan pemilihan jarak minimum pada pergantian *point to point* saat melakukan gerakan *non welding*. Dari hasil pemilihan jalur pengelasan tersebut kemudian dirubah ke bentuk *G-code* yang telah dimodifikasi. Hasil penerapan pemilihan jalur pengelasan mampu untuk dilakukan dengan akurasi tidak lebih dari 0,02 mm.

Kata Kunci :
hough transform, pemilihan jalur pengelasan

ABSTRACT

Name : Albertus Rianto Suryaningrat
Programe : Post Graduate / Master of Engineering DTM UI
Title :Welding path identification and generation employing machine *vision*

The use of machine *vision* technology in the welding process has been developed along with the need for more consistent and more qualified welding results. The application of machine *vision* to perform object analysis without direct contact to the object itself making the whole processes fast while maintaining the accuracy. In this research, *Hough Transform* algorithm is used to detect the welding tracks candidate. Afterward, some modifications to the hough transform is carried to enable finding the exact welding tracks. Once the welding tracks are found, welding sequences (welding path) on the welding tracks are then generated by evaluating all the tracks to produce the longest possible welding path with minimum non-welding motion. When all the welding paths are generated, they are then converted to a form of *G-code* like format which are ready to be sent to the controller unit. The implemented method capable of detecting the exact welding tracks and their appropriate welding path with accuracy not more than 0.02 mm.

Keyword :
hough transform, welding path generation

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

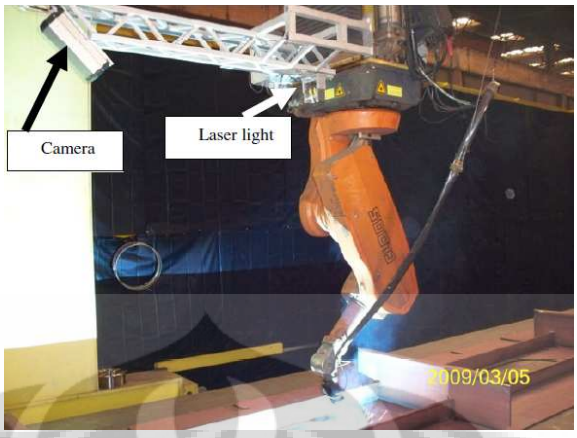
BAB 1
PENDAHULUAN

1.1 Latar Belakang

Mesin *Vision* secara sederhana dapat didefinisikan sebagai mesin untuk menangkap, mengolah dan menganalisa citra. Aplikasi mesin citra untuk melakukan proses analisa obyek tanpa adanya kontak langsung pada material akan mampu untuk menghasilkan proses yang mudah, cepat, dan aman Didalam perkembangan industri manufaktur, kebutuhan mesin citra banyak digunakan didalam sistem inspeksi pada jaminan kualitas (*quality assurance inspection*), *reverse engineering* dan mesin pengukur non kontak (1) (2). Mesin citra yang digunakan biasanya berperan untuk mendeteksi bentuk geometri yang selalu dituntut lebih cepat, lebih akurat dan mampu ulang dengan baik.

Proses pengelasan adalah salah satu proses manufaktur yang didalam perkembangannya selalu dituntut untuk menghasilkan proses pengelasan yang lebih cepat, otonomus dan konsisten untuk menekan biaya produksi. Dengan menggunaan sistem mesin citra diharapkan proses pengelasan akan dapat mengambil keputusan yang terbaik didalam memilih gerakan pengelasan tanpa melibatkan operator di dalam mengidentifikasi jalur pengelasan yang terjadi. Konsep utamanya adalah dengan melakukan deteksi garis oleh sebuah perangkat optik kamera kemudian hasilnya harus mampu untuk dianalisa secara akurat untuk diketahui jalur pengelasan yang terbaik (3) (4). Dengan membuat simulasi jalur lintasan pengelasan secara *off line* diharapkan kesalahan hasil pengelasan yang terjadi dapat dikurangi.

Didalam penelitian sebelumnya, Seyffarth (3) mencoba menggunakan lengan robot untuk melakukan proses pengelasan. Didalam membangun jalur pengelasan digunakan CAD/CAM system. Kamera hanya digunakan sebatas menentuan parameter kecepatan pengelasan. Sementara Micallef (5) mencoba menemukan jalur pengelasan dengan mengidentifikasi citra. Hanya didalam penelitiannya, jalur pengelasan yang diidentifikasi tidak melibatkan adanya perpotongan sehingga tidak melibatkan pemilihan pada jalur pengelasan.



Gambar 1.1 Proses pengelasan menggunakan robot artikulasi dan mesin *vision* (3)

1.2 Perumusan Masalah

Didalam mempercepat dan memperbaiki proses pengelasan dibutuhkan suatu sistem pendeteksi jalur pengelasan yang mana:

1. Konsistensi pengelasan yang sangat rendah jika dilakukan dengan manual atau dengan mengandalakan ketrampilan pengelasan. Hal ini akan sangat mengangu didalam kontrol kualitas hasil pengelasan.
2. Kesulitan didalam menggarahkan (*positioning*) jika dilakukan pada mesin las. Hal ini sangat memakan waktu untuk melakukan pergerakan pengelasan yang tidak seragam.
3. Dapat mengurangi resiko kesalahan didalam proses pengelasan. Hal ini dikarenakan karena proses yang dilakukan dapat disimulasi dan diukur terlebih dahulu didalam memilih jalur pengelasan.

1.3 Tujuan Penelitian

Penelitian tentang identifikasi dan pembuatan jalur pengelasan dengan mesin *vision* mempunyai tujuan khusus sebagai berikut:

1. Mencari teknik pemprosesan citra yang terbaik pada proses pelacakan garis didalam mengidentifikasi jalur pengelasan.
2. Mengembangkan algoritma pemrograman didalam pemilihan jalur lintasan pengelasan.

3. Membuat format untuk dibaca pada mesin las dengan modifikasi G-Code (ISO 6983) dan menganalisa keakurasian yang dihasilkan apakah mampu untuk dilakukan pengelasan pada mesin las.

1.4 Batasan Masalah

Dengan mempertimbangkan ketersediaan alat dan cakupan penelitian, maka didalam penelitian ini ditetapkan suatu batasan masalah sebagai berikut :

1. Pendeteksian jalur pengelasan hanya mampu untuk mendeteksi dalam bentuk garis lurus.
2. Percobaan dilakukan dengan model dua dimensi untuk tiga type jalur pengelasan yang berbeda.
3. Tidak dilakukan analisa yang mendalam didalam teknik pencahayaan.
4. Teknik didalam pengelasan dan analisa hasil pengelasan tidak dilakukan didalam penelitian ini. Analisa hanya dilakukan sebatas gerakan mesin las yang dilakukan.
5. Tidak dilakukan pengelasan berulang untuk jarak gap jalur pengelasan yang terlalu lebar. Algoritma yang dikembangkan tidak dapat mengakusisi Gap yang terlalu lebar secara sempurna.
6. Tidak dilakukan proses pengelasan pada jalur memutar atau tiga garis yang saling memotong. Alasan ini diperkuat bahwa didalam pendeteksian jalur pengelasan, tidak boleh adanya proses pengelasan yang memotong dan menyambung pada dua bagian didalam *region of interest* benda kerja
7. Metode pendeteksian hanya dilakukan dengan konsep melakukan pencarian jalur pengelasan terjauh yang kemudian diikuti dengan pencarian jalur terdekat dari proses yang belum dilalui.

1.5 Metodologi Penelitian

Didalam melakukan penelitian identifikasi dan pembuatan jalur pengelasan dengan mesin *vision*, metode yang dilakukan adalah sebagai berikut :

1. Studi Literatur
2. Pengembangan algoritma dan perangkat lunak uji
3. Ekperimental

Metode eksperimental dilakukan untuk membandingkan *path* yang dihasilkan terhadap hasil pengukuran benda kerja sesungguhnya. Dengan ekperimental ini akan dapat menentukan apakah proses pengelasan mampu untuk dilakukan dengan menggunakan teknologi mesin *vision*.

Sedangkan langkah didalam menjalankan penelitian dilakukan dengan pendekatan langkah sebagai berikut :

1. Mempelajari dan mengamati keterbatasan pada dasar pemrograman pengolahan citra digital, khususnya didalam proses penggunaan operator *image enhancement*, *edge detection*, dan proses binerisasi otomatis.
2. Mencoba mengamati ide dari beberapa jurnal penelitian terkait untuk dilakukan rekomputasi.
3. Bila diperlukan adanya inovasi baru yang dianggap masih relevan untuk dilakukan pada kasus yang dihadapi, maka dilakukanlah percobaan kecil untuk menguji logika yang didapat.
4. Menganalisa suatu masalah untuk didiskusikan pada dosen pembimbing.
5. Melakukan seminar.
6. Pengambilan dan analisa data hasil percobaaan
7. Pembuatan laporan penelitian dan jurnal.

1.6 Sistematika Penulisan Laporan

Laporan penelitian ini disusun dengan sistematika penulisan sebagai berikut :

BAB 1. PENDAHULUAN

Bab ini berisi tentang ide, perumusan masalah, tujuan penelitian, batasan masalah dan metodologi penelitian

BAB 2. MESIN *VISION* DAN PENGOLAHAN CITRA

Bab ini berisi tentang dasar teori, konseptual dan notasi yang menjadi landasan didalam pemikiran pada pembuatan sistem dan algoritma.

BAB 3. PENGEMBANGAN METODE PENDETEKSIAN JALUR PENGELASAN

Bab ini berisi tentang kerangka pemikiran dan konsep yang harus dijalankan.

BAB 4. PERCOBAAAN DAN ANALISA HASIL

Bab ini berisi tentang implementasi algoritma dan analisa hasil. Pada bab ini banyak berisi tentang data percobaan dan uji akurasi.

BAB 5 KESIMPULAN DAN SARAN PENELITIAN LEBIH LANJUT

Bab ini berisi tentang kesimpulan sebagai hasil dari tujuan penelitian dengan saran untuk dilakukan penelitian selanjutnya.

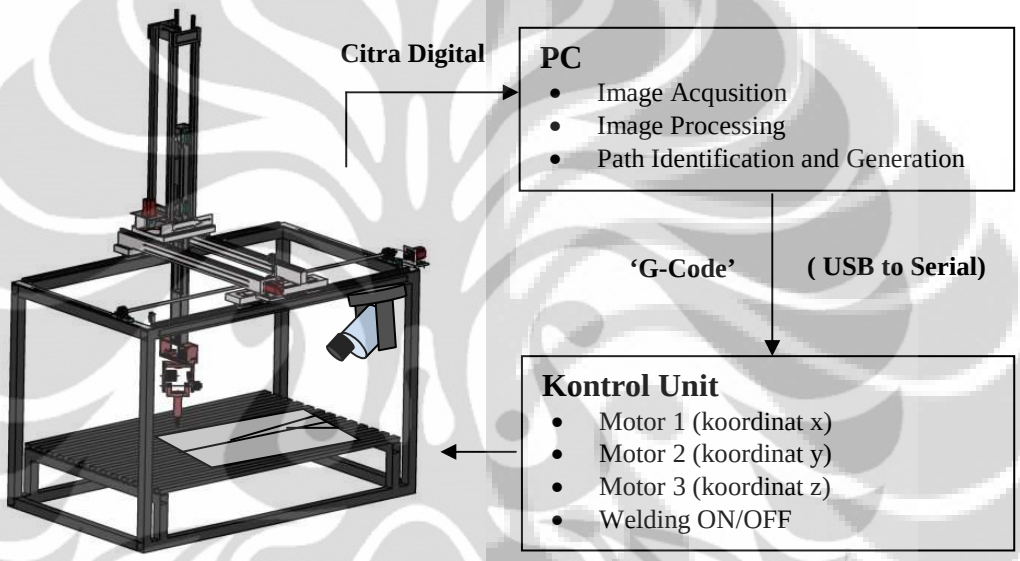
LAMPIRAN

Lampiran banyak diisi dengan hasil pemrograman komputer.

BAB 2
MESIN *VISION* DAN PENGOLAHAN CITRA

2.1 Mesin citra dan teknik penggunaannya

Hasil dari pengolahan citra sangat tergantung dari masukan data (*raw image*) citra yang diambil. Semakin maksimal hasil dari pengambilan citra yang dapat diambil, akan semakin mudah untuk diproses pada pengolahan citra selanjutnya.

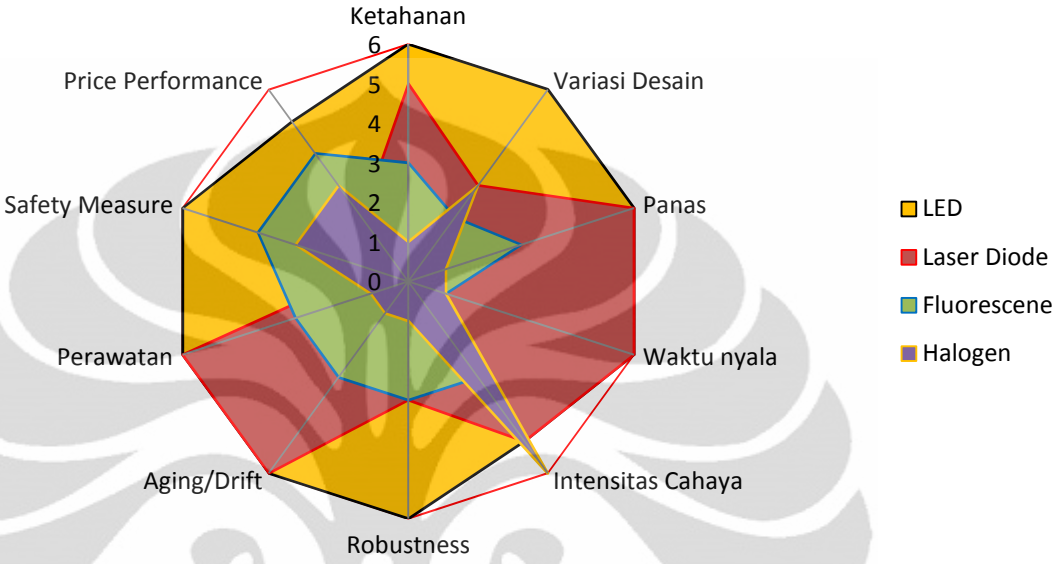


Gambar 2.1 Sistem pada mesin citra (6)

2.1.1 Teknik Pencahayaan

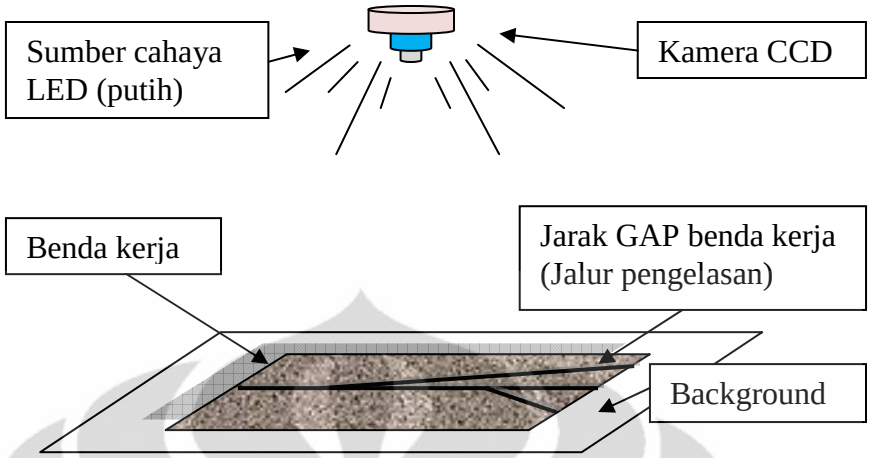
Cahaya adalah hal yang paling penting dalam pengambilan citra. Dengan seberkas cahaya, mata manusia bisa melihat keberadaan suatu benda hanya dengan melihat hasil pantulan cahaya yang dihasilkan benda. Warna dari benda kerja lebih dipengaruhi oleh warna dari cahaya tampak. Sedangkan variasi warna pada pengolahan citra digital lebih didominasi pada sumber cahaya. Pemilihan sumber cahaya dilakukan dengan memilih cahaya yang sesuai dengan kebutuhan. Sedangkan intensitas cahaya lebih dipengaruhi oleh kebutuhan akan intensitas ketajaman per piksel benda kerja yang akan diambil. Permasalahan dalam pemilihan sumber cahaya adalah dengan memilih warna sinar yang mudah untuk

dibedakan antara warna cahaya dengan warna material itu sendiri. Pada gambar 2.2 menunjukkan bagaimana sifat dari sumber cahaya LED (*Light Emitter Diode*), *laser diode*, *fluorescene*, dan *halogan*.



Gambar 2.2 Karakteritik dan performa beberapa sumber cahaya (7)

Teknik pencahayaan adalah teknik bagaimana cahaya mengenai benda kerja secara halus. Sudut yang dibentuk akan mempengaruhi bayangan yang dihasilkan profile. Teknik pencahayaanya bisa dilakukan langsung maupun dengan filter optik. Efek dari kesalahan didalam memilih sumber cahaya dapat membuat ketajaman dari warna material menjadi kurang jelas. Sedangkan efek kesalahan didalam memilih teknik pencahayaan adalah kemampuan deteksi yang kurang sempurna karena adanya bayangan (*silhouette effect*) yang sifatnya lebih menjadi ganguan (*noise*) saat proses pengambilan citra. Pada gambar 2.3 dapat dilihat teknik pencahayaan dengan sumber cahaya langsung yang terpasang pada perangkat kamera. Sumber cahaya berada di sekeliling perangkat kamera. Menurut Hornberg (7), dengan teknik seperti ini, bayangan yang dihasilkan akan saling mengurangi



Gambar 2.3 Teknik pencahayaan

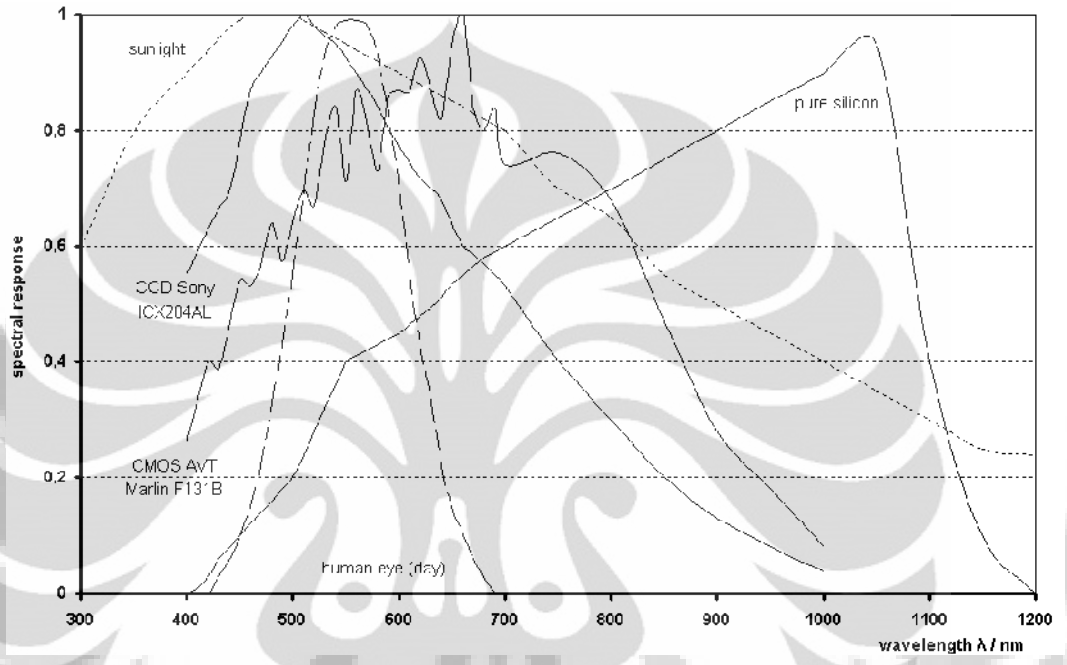
2.1.2 Kamera Digital

Kamera digital adalah alat optik (*optic device*) untuk mengambil suatu citra dari benda kerja secara digital. Sedangkan *chip* yang sering digunakan untuk pengolahan warna digital adalah CCD atau CMOS. Menurut Hornberg (7) didalam memilih kamera perlu diingat bahwa setiap optik kamera yang digunakan selalu mempunyai keterbatasan didalam melakukan pengukuran. Hal ini dikarenakan efek titik fokus yang berbeda untuk setiap titik berdimensi dua. Kamera juga memperlukan waktu *warming up* untuk pengambilan gambar yang maksimal. Kebutuhan peralatan tambahan seperti tripod atau alat penyangga yang stabil sangat diperlukan didalam pengambilan citra yang baik. Spesifikasi penggunaan kabel pertukaran data seperti USB/Serial atau didalam penggunaan *card* tambahan pada *slot* PC juga perlu dipertimbangkan guna mempermudah didalam pengambilan data.

Didalam memilih kamera digital, parameter produk sebagai bahan pertimbangan adalah :

- Adaptor kamera (dcam, winvideo) dan hardwere interface.
- Jumlah frame yang mampu dihasilkan per detik (*frame/sec*)
- Bentuk citra yang dikirim (*return color space*)
- Resolusi atau jumlah pixel per luas area (*dpi*)

Sedangkan alasan didalam pemiliahan menggunakan chip sensor menggunakan CCD daripada penggunaan chip sensor CMOS adalah selain harganya lebih murah karena kemampuan didalam merespon terhadap warna RGB lebih maksimal atau mendekati kemampuan mata manusia seperti terlihat pada gambar 2.4.



Gambar 2.4 *Spectral response* vs panjang gelombang warna pada kamera CCD dan kamera CMOS (7)

Kamera CCD

Kamera CCD adalah kamera yang menggunakan chip sensor CCD untuk mengolah warna, Kamera jenis ini sangat banyak digunakan pada aplikasi *webcam*, dan pada kamera CCTV untuk keperluan keamanan.

2.1.3 Teknik Pengambilan Citra (*Image Data Acqusition*)

Teknik pengambilan citra adalah teknik untuk menentukan kapan dan bagaimana citra tersebut harus diambil. Didalam teknik pengambilan citra, hal yang paling menentukan adalah karakteristik data citra yang akan digunakan pada proses pengolahan citra. Indikator yang sering digunakan adalah :

- Adaptor antara *Image acqusition device* dengan *PC*
- Penyimpanan data

- Teknik pengambilan *frame* baik yang dilakukan pada jarak interval kamera atau pada *frame per trigger*
- Area citra yang diambil (*Region of Interest*)

2.1.4 Arsitektur Pengolahan Citra

Arsitektur pengolahan citra berhubungan dengan bagaimana suatu sistem perangkat berupa sumber cahaya, kamera, dan komputer bersinergi menjadi suatu sistem mesin citra. Didalam mendesain sistem tersebut diperlukan beberapa parameter desain sebagai berikut :

- Kesinambungan kerja mesin citra
- Tingkat keakurasian
- Tingkat kemampuan komputer dalam pengolah dan menganalisa citra

2.2 Pengolahan Citra

Pengolahan citra berfungsi untuk memaksimalkan kontras fitur yang diperlukan (*foreground*) dan meminimalkan kontras fitur yang tidak diperlukan (*background*). Beberapa bagian dari pengolahan citra antara lain adalah representasi citra, teknik perbaikan citra, pelacakan tepi obyek, teknik binerisasi, pelacakan garis, dan masih banyak yang lainnya.

2.2.1 Representasi Citra

Davies (8) menggambarkan bahwa citra dapat di presentasikan ke dalam bentuk array tiga layer (RGB/YUYV/HSI/YcbCr/CMY) atau hanya dalam bentuk satu layer saja atau sering disebut sebagai citra keabu-abuan. Data citra pada perangkat kamera menentukan seperti apa representasi citra yang akan dikirimkan ke komputer. Dengan algoritma pengolahan citra, hasil representasi citra dari kamera bisa dirubah menjadi representasi citra yang lain sesuai dengan kebutuhan. Namun demikian juga ada kamera yang mempunyai kemampuan mengeluarkan citra langsung dalam bentuk satu layer. Sementara untuk merubah warna RGB (3 layer) ke citra keabu-abuan (1 layer) digunakan persamaan 2.1 sebagai berikut:

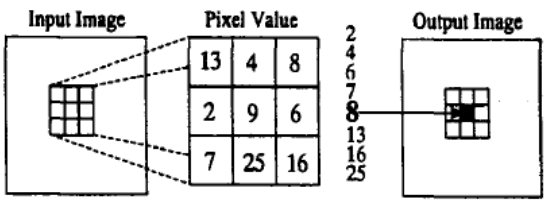
$$Intensity = 0.299R + 0.587G + 0.114B \quad (2.1)$$

2.2.2 *Filtering* dan *Noise Suppression*

Kebutuhan akan informasi citra yang bebas ganguan adalah hal yang dibutuhkan dalam pengolahan citra. Teknik *filtering* adalah bagian dari teknik perbaikan citra yang mana akan dilakukan berdasarkan kebutuhan yang diinginkan. Menurut Davies (8), jika menginginkan proses untuk mengeliminasi gangguan citra pada frekuensi tinggi, maka dapat digunakan *low pass filter*. Sementara jika menginginkan untuk mengeliminasi citra pada frekuensi rendah, dapat digunakan metode *gaussian* untuk *high pass filter*. Dengan metode pengaburan (*blurring*), akan dapat mengeliminasi ganguan (*noise*) yang terjadi. *Filter* yang sering digunakan didalam pengolahan citra antara lain adalah *median filter, mean filter* dan *gaussian fillter*. *Median filter* mempunyai sifat yang sangat halus didalam penggaburannya, diikuti dengan *mean filter*, dan terakhir *gaussian filter* yang sifat penggaburannya sangat tinggi. Semakin nilai penggaburannya tinggi, citra semakin beresiko untuk kehilanggan informasi sebagai bagian dari pemprosesan *filter*. Metode *filter* juga dapat digunakan dengan mengkonversikan citra ke parameter imajiner dengan teknik *fast forier transform* yang kemudian mem-*filter* sebagian pada hasil parameter imajiner tersebut untuk dilihat hasilnya pada citra sesungguhnya. Biasanya teknik *fast forier transform* digunakan pada gangguan yang bersifat pengulangan.

Median Filter

Menurut Davies (8), *median filter* mempunyai konsep untuk mengubah nilai pixel yang terlalu extrem karena efek goresan atau perubahan warna yang mendadak yang bersifat sangat lokal. Berbeda dengan teknik *gaussian*, didalam *median filter* teknik pengaburannya sangat halus dan bersifat marginal sehingga pixel yang bersifat informasi tidak hilang dalam pendeteksian .



Gambar 2.5 Konsep median filter (9)

Algoritma untuk median filter sebagai berikut :

Start
- *Baca image greyscale*
- *Cari dimensi image*
- *Buat temporary matrik dengan ukuran sama dengan image*
  For (jumlah pixel)
  - *Cari nilai tengah dari 9 nilai pada kernel [3 3]*
  - *Ganti nilai pada tengah kernel dengan nilai tengah*
  - *Simpan pada temporary matrix*

  End
- *Ganti Image greyscale dangan temporary matrix*
  Selesai

2.2.3 Pelacakan Tepi Obyek (*Edge Detection*)

Edge detection digunakan untuk mendeteksi keberadaan tepi pada bentuk material dalam suatu citra. Garis tepi pada suatu objek akan terdeteksi dengan melihat perubahan intensitas warna yang sangat kontras yang bersifat lokal dan nyata. Menurut Davies (8), gradien tersebut adalah hasil pengukuran perubahan intensitas yang merupakan persamaan dua dimensi dari turunan pertama sehingga hasil pendekatan numeriknya dapat didefinisikan pada persamaan 2.2.

$$G[f(\mathrm{x},\mathrm{y})] = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix}$$

$$Gx[f(x,y)] \cong f(x+1,y) - f(x,y)$$

$$Gy[f(x,y)] \cong f(x,y+1) - f(x,y) \qquad (2.2)$$

Besaran G[f(x,y)] menunjukan arah penambahan laju maksimal pada fungsi f(x,y) sementara besarnya gradien mempresentasikan laju dari fungsi f(x,y) per satuan jarak dalam arah G. Untuk mempresentasikan kedua arah deteksi vertikal dan horizontal dapat digunakan dengan hukum *phytagoras* seperti terlihat pada persamaan 2.3.

$$G[f(x,y)] = \sqrt{(Gx)^2 + (Gy)^2} \qquad (2.3)$$

Nilai gradien mempunyai sifat bebas terhadap arah akibat dari posisi tepi yang mengelilingi obyek. Hal ini dikenal dengan sifat isentropik obyek. Terdapat

dua operator hasil turunan pertama yang dikenal mempunyai karakteristik pelacakan tepi yang baik yakni *Sobel* dan *Prewitt.* Sedangkan untuk menemukan garis tepi pada suatu obyek yang idealnya hanya garis tipis, maka diperlukan suatu operator yang dibangun dengan turunan kedua. Namun demikian, operator yang dibangun dengan turunan ke dua sangat peka terhadap gangguan citra. Operator turunan kedua yang sering dipakai dalam pelacakan tepi adalah operator *Laplacian of Gaussian*.

Operator Sobel

Sifat dari operator Sobel adalah mempunyai sensitifitas pelacakan tepi yang sangat jelas dibanding dengan pelacakan lainnya khususnya didalam mendeteksi garis tepi yang sederhan. Dibanding dengan operator *Prewitt*, operator *Sobel* mempunyai nilai error numerik pada pendeteksi kemiringan sebesar 2,06 deg sementara operator *Prewitt* mempunyai nilai error sebesar 4,5 deg (Lyvers and Mitchell, 1998). Menurut Davies (8), operator Sobel sangat cocok digunakan didalam pendeteksian pada citra satu layer dengan resolusi yang kecil. Hasil keakurasian pada pendeteksian sangat seimbang dengan beban komputasi yang dilakukan.

$$Sx = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad Sy = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} \tag{2.4}$$

Algoritma dari operator Sobel adalah sebagai berikut :

Start

- *Baca image greyscale*
- *Cari dimensi image*
- *Buat matrik temporary dengan dimensi sama*
- *Buat kernel sobel operator pada arah x dan y*

  For (semua pixel)
  - *Nilai(x;y) = Σ(Kernel image* kernel sobel)*
  - *Gunakan pytagoras untuk nilai(x;y)*
  - *Ganti nilai tengah kernel matrik temporary dengan nilai ini.*

  End

Selesai

2.2.4 Binerisasi Otomatis

Binerisasi (*binerization*) adalah teknik segmentasi atau suatu cara untuk memisahkan antara pixel benda kerja dengan latar belakang (*background)*. Dengan teknik binerisasi, hasil gambar akan lebih ringan untuk diolah (1 bit/pixel) daripada gambar pada citra keabu-abuan (8 bit/pixel).

Kebutuhan akan teknik binerisasi otomatis didalam mesin citra akan sangat penting mengingat peranannya pada sistem dituntut untuk cepat (*real time*). Oleh karena itu di dalam memutuskan suatu nilai batas ambang (*Threshold*), teknik binerisasi sudah tidak membutuhkan interferensi dari operator.

Banyak algoritma yang dibangun berdasarkan nilai keseluruhan histogram yang dihasilkan pada citra (*Non-Parametric Histogram Thresholding*). Semula algoritma ini dibangun hanya dengan melihat secara langsung bentuk histogram seperti algoritma untuk mencari nilai minimum sebagai nilai ambang batas antara dua mode. Kemudian dikembangkan algoritma pencarian intensitas distribusi pada kemiringan maksimal. Namun kedua algoritma ini sangat tidak konsisten untuk dipergunakan pada kondisi nyata yang mana yang mana akan banyak terinterferensi terhadap kerataan distribusi cahaya, bayangan yang dihasilkan, dan warna material yang menyilaukan.

Karena kendala didalam teknik binerisasi tersebut maka diperlukan penggunaan teknik binerisasi yang yang lebih adaptif (*adaptive thresholding*) dengan untuk menangulangi laju histogram pada nilai pixel obyek yang sifatnya tidak seragam dan terpusat pada suatu area sempit. Algoritma ini dimulai dengan pembuatan histogram terpisah yang mana histogram tersebut dibangun kembali dengan kaedah *gaussian* (Chow and Kaneko, 1972), namun didalam suatu kondisi pada pemrosesan resolusi tinggi masih mengalami banyak kendala pada kecepatan komputasi. Kemudian mulai digunakan nilai *mean, median, momen* sebagai variabel untuk menentukan nilai bineriasi yang bebas ganguan (*noise*), namun kelemahan dari algoritma ini pada kemampuan menghilangkan *noise* yang berlebihan yang mengakibatkan bagian *crack* pada jalur pengelasan akan hilang.

Didalam perkembangan selanjutnya, dikembangkanlah suatu algoritma dengan berbasis pencarian nilai maksimal variasi antar kelas dengan menggunakan nilai minimum antar variasi tersebut (Otsu, 1979), Selain itu

terdapat algoritma pencarian nilai *maximum entrophy* (Kapur et al., 1985), dan yang terakhir adalah algoritma dengan *maximum likelihood* atau teknik binerisasi dengan *training set* untuk mengenali bentuk histogram. Ketiga teknik binerisasi tersebut sangat *robustness* dengan komputasi yang cepat. Menurut Davies (8), metode yang terakhir ini tidak dapat digunakan karena tidak ada training set yang tersedia. Sedangkan metode *maximum entrophy* terkendala ketika nilai histogram sangat fluktuasi disuatu jarak dan mulai menghilang disuatu jarak tertentu. Pada penelitian Nacereddine (10) dilakukan pengujian proses binerisasi hasil pengelasan yang mana metode *otsu* menjadi kandidat ke dua setelah proses binerisasi dengan metode *maximum entrophy* dengan nilai error 0,33.

Metode Otsu

Metode otsu adalah metode untuk menghitung nilai biner berdasarkan perubahan persentase nilai histogram yang terjadi. Menurut Davies (8), metode ini sangat cepat menggingat hanya membutuhkan sekali pembentukan nilai histogram dan menganalisa sekali proses propagasi nilai histogram.

Di dalam menentukan nilai biner yang terjadi dilakukan dengan meng-iterasi-kan nilai 0 sampai 255 kedalam persamaan 2.5 untuk mencari nilai berat. Kemudian dari nilai berat yang didapat, di iterasikan kembali pada persamaan 2.6 untuk mendapatkan nilai perubahan variasi maksimal pada benda kerja. Ketika mendapatkan nilai perubahan variasi maksimal maka dapat dibandingkan dengan nilai perubahan variasi maksimal yang terdaat pada *background*. Setelah ditemukan kedua nilai variasi tersebut kemudian gunakan selisih nilai variasi antar kelas yang paling minimal yang mana di titik iterasi tersebut kemudian digunakan menjadi nilai ambang batas (*thresholding*).

Karena proses pencarian perubahan variasi maksimal pada benda kerja dan *background* dengan pencarian selisih antara keduanya bisa dilakukan pada satu kali iterasi, maka proses ini akan sangat pas untuk dilakukan pada pengolahan citra pada kondisi nyata, *adaptif*, *robustness*, dan *real time*.

$$p(i) = \frac{Histogram(i)}{jumlah\ pixel\ citra} \tag{2.5}$$

$$\omega(k) = \sum_{i=1}^{k} p(i)$$

$$\mu(k) = \sum_{i=1}^{k} i * p(i)$$

$$\mu T = \sum_{i=1}^{L} i * p(i)$$

$$\sigma_B^2(k) = \frac{[\mu_{T*}\omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]} \quad (2.6)$$

Algoritma binerisasi dengan metode otsu adalah sebagai berikut :

- *Start*
- *Baca image greyscale*
  For (i = nilai histogram dari 1 s/d 256)
  - *Temukan nilai rata-rata(*$\omega(k) = p(i) = \frac{Histogram(i)}{jumlah\ pixel\ citra}$ *)*

  End
  For (i = nilai histogram dari 1 s/d 256)

  $$\mu(k) = \sum_{i=1}^{k} i * p(i)$$

  $$\mu T = \sum_{i=1}^{L} i * p(i)$$

  $$\sigma_B^2(k) = \frac{[\mu_{T*}\omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]}$$

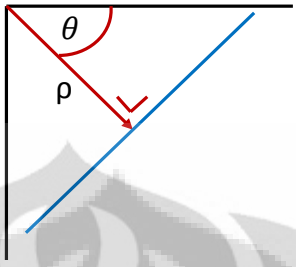
  If (nilai perubahan $\sigma_B^2(k)$ *> nilai perubahan maksimal)*
  - *Gunakan nilai histogram(i) untuk nilai threshold*

  End
  End
- *Selesai*

2.2.6 Algoritma Hough Transform

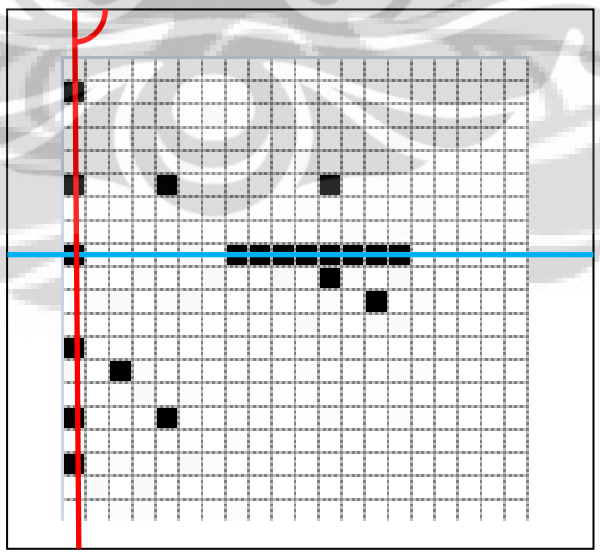
Algoritma HT sangat populer untuk digunakan pada pendeteksian garis, lingkaran ataupun kurva pada suatu citra. Algoritma ini mengubah citra gambar menjadi citra dengan parameter *rho* dan *theta*. Pada persamaan 2.7 didapatkan bahwa untuk melakukan deteksi garis pada panjang (ρ) dari koordinat awal f(0,0) dengan sudut(θ) maka akan dapat dideteksi keberadaan posisi *pixel* biner pada arah garis tegak lurus dengan arah panjang *rho* seperti pada gambar 2.6. Nilai *rho* akan lebih bersifat mengikuti gerakan nilai *theta* pada saat proses deteksi sudut dilakukan.

$$\rho = x * \cos\theta + y * \sin\theta \quad (2.7)$$



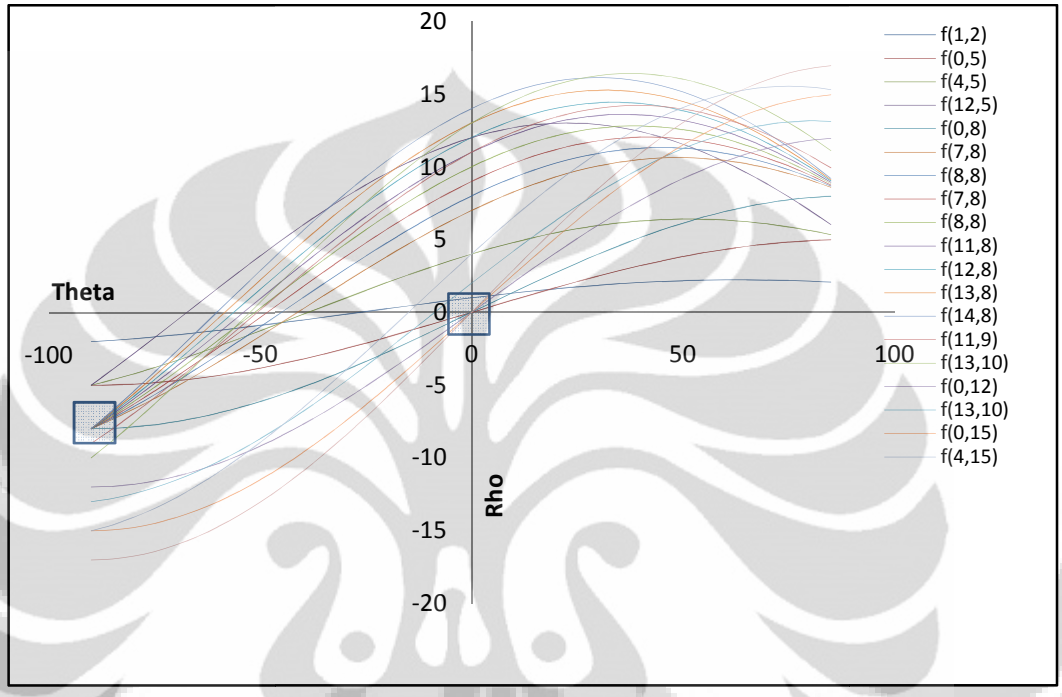
Gambar 2.6 Sistem koordinat polar

Kemudian dari kumpulan *pixel* biner, dicari lokasi koordinat f(x,y) untuk di subtitusikan dengan ada variabel *theta* ke dalam persamaan 2.7 . Dengan nilai resolusi *theta*, maka akan dapat menentukan seberapa detail akurasi arah yang diambil. Dari sekumpulan kurva yang berhasil terbentuk pada parameter HT kemudian dicari lokasi koordinat *rho* dan *theta* yang terdapat konsentrasi adanya perpotongan kurva maksimal. Dari nilai *rho* dan *theta* pada daerah perpotongan tersebut dapat diindikasikan bahwa pada citra biner tersebut terdapat sekumpulan *pixel* yang teridikasi membentuk garis pada arah *rho* dan *theta* . Sebagai contohnya adalah citra biner dengan resolusi 20 x 20 pada gambar 2.7.



Gambar 2.7 Simulasi citra biner dengan grid pixel

Dengan menentukan koordinat pixel f(x,y) dan mengakumulasikan pada persamaan 2.7, akan didapat grafik sebagai fungsi dari koordinat x sebagai *theta* (θ) dan koordinat y sebagai *rho* (ρ) seperti terlihat pada gambar 2.8.



Gambar 2.8 *Hough tranform* parameter

Dari parameter HT terlihat bahwa nilai *peak thresholding* yang dilakukan dapat mendeteksi adanya garis di *theta* (θ) = 90 dengan *rho* (ρ) ≅ 9 dan *theta* (θ) = 0 dengan *rho* (ρ) ≅ 0. Sedangkan nilai parameter yang dibutuhkan untuk membangun algoritma *hough transform* adalah sebagai berikut :

- Resolusi pencarian *theta* (θ) pada setiap nilai pixel biner bernilai 1
- Resolusi pencarian *rho* (ρ) pada setiap nilai pixel biner bernilai 1
- Nilai perpotongan minimum pada HT parameter yang masih teridentifikasi terdapat bentuk garis.
- Besar dimensi kernel sebagai eliminasi matrik (*NhoodSize*)
- Jumlah maksimal garis yang mampu diproses

Didalam membangun algoritma HT selalu dihindari adanya inputan manual dari operator. Nilai perpotongan minimum (*peak thresholding*) dan

dimensi eliminasi kernel merupakan dua inputan yang harus dilakukan secara otomatis. Menurut Gonzelez (11) didalam menentukan nilai *peak thresholding* dan eliminasi kernel dapat ditentukan dengan membuat persamaan empiriz seperti pada persamaan 2.8 dan persamaan 2.9.

$$T_{min} = 0.5 * (biner)_{max} \tag{2.8}$$

NhoodSize mempunyai fungsi untuk mengeliminasi nilai *peak thresholding* dan nilai pixel yang berada disekitarnya pada saat proses iterasi dijalankan. Teknik eliminasi ini dilakukan karena nilai *peak thresholding* tidak selamanya terpusat pada satu titik. Di dalam pembuatan dimensi eliminasi kernel (*NhoodSize*), Gonzeles (9) menyarankan untuk dilakukan secara proporsional terhadap dimensi parameter hough (θ,ρ) yang dihasilkan. *NhoodSize* akan terbentuk selalu berjumlah ganjil. Di dalam persamaan 2.9 didapat pembuatan bentuk matrik yang sebangun dengan dimensi *HT* parameter (θ, ρ) tersebut.

$$NHoodSize = \left[2 * ceil\ \frac{\left[\frac{size(hough)}{50}\right]}{2} + 1{,}1\right]_{max} \tag{2.9}$$

Sebagai tambahan algoritma dapat dimasukkan jumlah garis maksimal yang akan teridentifikasi yang fungsi utamanya untuk membatasi banyaknya iterasi yang akan dilakukan. Didalam menentukan nilai jumlah garis, perlu diperhatikan bentuk kerumitan garis dengan kemampuan komputasi yang dimiliki.

2.2.7 Segmentasi Citra

Karena garis yang teridentifikasi pada parameter HT hanya menyatakan arah (θ) dan jarak (ρ), maka diperlukan algoritma segmentasi yang fungsinya akan mengidentifikasi kapan garis tersebut masih berada pada sebuah citra atau tidak. Garis ini dibentuk dengan melihat keberadaan nilai pixel biner pada citra dengan nilai hough (θ,ρ) yang telah teridentifikasi. Teknik segmentasi ini semula dibentuk dengan langsung melakukan pengecekan silang pada kondisi citra biner dengan

hasil garis yang didapat. Namun teknik ini banyak kendala ketika garis mulai kehilangan keterhubungan. Liao (12) menyarankan untuk membuat algoritma segmentasi dengan pengkondisian keterhubungan garis yang tidak kaku. Didalam algoritma yang dibangun, diperlukan adanya *fillgap* untuk mengakusisi apakah garis tersebut menjadi satu segmen atau lebih. Perlu diingat bahwa menentukan nilai *fillgap* yang berlebihan juga dapat membuat ganguan (*noise*) ketika mendeteksi bagian pixel yang tidak berhubungan.

Gambar 2.9 Hasil penggunaan segmentasi

Gambar (a) adalah garis hasil citra biner, (b) merupakan hasil segmentasi hanya dengan adanya pixel pada citra biner, (c) merupakan segmentasi dengan *fillgap* kecil menjadikan deteksi dua segmentasi garis, dan (d) adalah segmentasi dengan *fillgap* berlebih, yang menjadikan satu segmen garis tersambung.

Didalam pendeteksian sering terjadi garis yang menyimpang yang sifatnya sangat menggangu yang disebabkan efek pendeteksian secara menyeluruh. Oleh karena itu teknik segmentasi juga perlu dilakukan eliminasi panjang garis yang nilainya terlalu kecil untuk menjadi segmen garis tersendiri.

Untuk mengidentifikasi garis diperlukan persamaan garis seperti pada persamaan 2.10 yang merupakan *gradien* (m) dari kemiringan pada sebuah garis, sehingga jika persamaan 2.10 dapat di turunkan maka akan menjadi persamaan garis seperti pada persamaan 2.11.

$$\frac{y2-y1}{x2-x1} = \frac{y-y1}{x-x1} = m \tag{2.10}$$

$$y = m * x + c \tag{2.11}$$

Sedangkan untuk mencari koordinat perpotongan antara dua garis dapat dilakukan dengan menggunakan persamaan matrik pada persamaan 2.12.

$$\begin{vmatrix} x \\ y \end{vmatrix} = \begin{vmatrix} -lines(1).m & 1 \\ -lines(2).m & 1 \end{vmatrix}^{-1} * \begin{vmatrix} lines(1).c \\ lines(2).c \end{vmatrix} \quad (2.12)$$

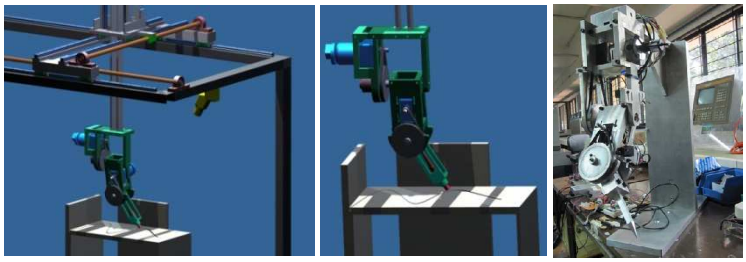
Untuk menghitung jarak antara dua titik pada berbagai posisi kemiringan dapat dilakukan dengan metode *Euclidian* seperti pada persamaan 2.13.

$$L\_garis = \sqrt[2]{[x_2 - x]^2 + +[y_2 - y_1]^2} \quad (2.13)$$

2.3 Interprestasi Citra

3.3.1 Mesin Las

Mesin las adalah suatu aktuator yang menggunakan gerakan 3 axis untuk melakukan proses pengelasan. Berbasis pada teknologi mesin perkakas CNC, mesin las juga digerakkan dengan teknologi NC (*Numerical Control*) yang dikontrol dengan memasukan data format G-Code (ISO 6983) yang telah dimodifikasi. Didalam penelitian yang akan dilakukan, mesin las digunakan untuk menghasilkan gerakan motor *stepper* yang akan dikontrol dengan *NC-File* dan aktifasi *torch* itu sendiri. Gerakan penetrasi pada pengelasan dilakukan oleh motor *stepper* pada axis z, sedangkan gerakan pengelasan (x,y) pada proses pengelasan dilakukan oleh dua buah motor step untuk mengerakan meja penumpu *jig fixture*. Pada penelitian Kiswanto, et al (6) didalam pembuatan mesin las dapat dirancang dengan membangun 5 derajad kebebasan seperti terlihat pada gambar 2.10



Gambar 2.10 Mesin las 5 derajad kebebasan (6)

3.3.2 NC-File

NC-File adalah *format* data pergerakan secara numerik yang digunakan untuk membuat suatu gerakan pada axis tanpa bantuan operator. Dengan teknologi *NC-File*, akan dapat langsung diterapkan pada gerakan 3 axis tanpa harus melakukan perintah gerakan satu persatu pada motor untuk digerakan secara simultan. Hal ini akan sangat membantu ketika terjadi gerakan *interpolasi* pada motor untuk melakukan gerakan miring atau gerakan berbentuk kurva. Sedangkan *format* standar yang biasa dipakai untuk gerakan NC pada mesin perkakas adalah G-Code.

2.3.3 Penskalaan

Teknik penskalaan dilakukan untuk men-transformasi nilai pada resolusi gambar menjadi nilai sesungguhnya. Karena gambar yang diambil pada mesin *vision* tidak mempunyai nilai resolusi yang besar maka proses pengelasan yang tidak banyak dituntut mempunyai keakurasian yang baik seperti pada keakurasian pada proses permesinan. Didalam penelitian Seyffarth (3) dijelaskan bahwa akurasi gerak sebesar 1 mm adalah akurasi yang berelebihan didalam proses pengelasan.

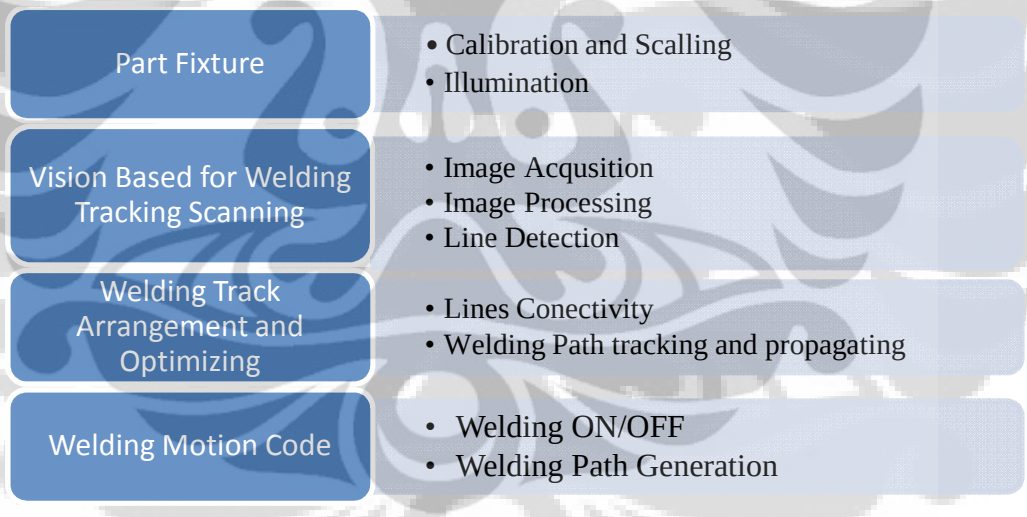
Didalam teknik penskalaan dilakukan tanpa melibatkan adanya efek kecembungan kamera. Kamera yang digunakan tidak menggunakan fokus otomatis, maka didalam proses pangembilan jarak ditentukan dengan melihat hasil terbaik dengan mengubah variabel jarak antara kamera dengan benda kerja. Benda kerja yang dilakukan untuk melakukan pen-skalaan dapat dilakukan dengan kertas milimeter. Teknik penskaalan ini dilakukan juga pada penelitian Chen (13).

Gambar 2. Teknik penskalaan pada penelitian Chen (11)

BAB 3
PENGEMBANGAN METODE PENDETEKSIAN JALUR PENGELASAN

3.1 Kerangka Pemikiran

Metode yang dikembangkan adalah metode pendeteksian jalur pengelasan dengan menggunakan sebuah kamera CCD dengan pencahayaan LED yang menjadi satu dengan kamera. Dari hasil gambar yang didapat kemudian dikirim ke sebuah PC untuk diolah citra nya hingga mendapatkan deteksi garis. Jika garis tersebut lebih daris satu lintasan atau saling memotong, maka akan dicari jalur yang paling efisien untuk melakukan proses penggelasan. Algoritma yang dikembangkan didekati dengan teknologi yang sudah digunakan pada teknologi *CAD/CAM* ditambah dengan memanfaatkan algoritma HT sebagai algoritma untuk melakukan mendeteksian garis.



Gambar 3.1 Diagram alur pada langkah pemrograman

Algoritma 1 Welding Path Generation using Machine *Vision*

Input : *Image*
Output: *G-Code*
Process: *Start*

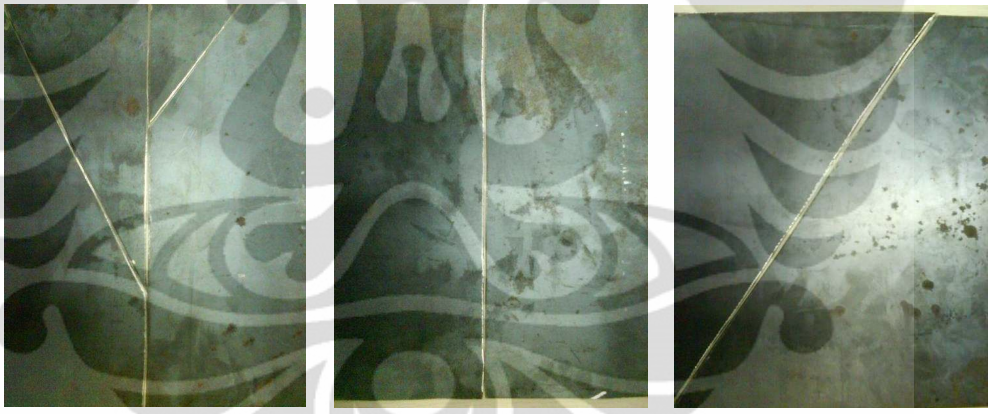
- *Baca image greyscale*
- *Median Filter untuk low pass filtering*
- *Sobel Operator untuk pelacakan tepi dengan binerisasi otsu*
- *Hough Transform untuk pendeteksian garis*
- *Filtering untuk menghilangkan deteksi garis terluar*

- *Path Identification*
- *Path Routing Selection*
- *G-Code Generation*

Selesai

3.2 Pemilihan Benda Uji

Pemilihan dalam proses pembuatan algoritma dilakukan dengan percobaan awal pada tiga benda kerja berdimensi dua seperti terlihat pada gambar 3.2 dengan membuat karakteristik jalur pengelasan yang beragam. Dari ketiga benda kerja tersebut akan diambil citra gambar yang kemudian dilakukan proses pengolahan citra agar mampu teridentifikasi dengan baik jalur pengelasan. Material benda uji menggunakan plat besi dengan tebal 2 mm, sedangkan benda uji yang dianggap mewakili variasi arah untuk dilakukan percobaan adalah benda uji type 1.



TYPE 1 TYPE 2 TYPE 3

Gambar 3.2 Tiga type benda kerja yang akan diidentifikasi jalur pengelasan

3.3 Metode yang dikembangkan

3.3.1 Parameter Hough Transform

Didalam metode untuk menentukan parameter HT dilakukan dengan mengganti benda uji yang mempunyai hasil terbaik untuk pendeteksian garis. Parameter yang digunakan disesuaikan pada kondisi pengolahan citra yang

dibutuhkan agar mampu untuk mendeteksi terhadap bentuk garis sebenarnya. Resolusi *theta* dan resolusi *rho* dinilai optimal pada nilai 1 untuk mendeteksi garis pada resolusi citra 480x640 pixel. Nilai ini diambil dari serangkaian percobaaan pendeteksian garis pada ketiga benda uji dengan perubahan pencahayaan. Nilai resolusi ini tidak begitu terpengaruh terhadap pencahayaan, namun sangat terpengaruh terhadap resolusi gambar yang diproses.

$$Theta(\theta)Resolusi = 1$$
$$Rho(\rho)Resolusi = 1$$

Sedangkan untuk melakukan pendeteksian nilai puncak (*peak thresholding*) pada parameter HT diperlukan algoritma yang mampu untuk beradaptasi dalam memberikan hasil yang konsisten meskipun terdapat perubahan resolusi *rho* dan resolusi *theta*. Kekonsistensian didalam menemukan nilai *peak threshoding* dengan melakukan perubahan resolusi hanya untuk menunjukan bahwa algoritma mempunyai kemampuan beradaptasi karena didalam kondisi sebenarnya, nilai resolusi *theta* dan *rho* adalah bernilai konstan. Sedangkan didalam pemilihan nilai *peak thresholding* yang otonomus dapat digunakan dengan memodifikasi persamaan empiriz *Gonzalez* (11) dengan mengganti nilai konstanta 0.5 menjadi 0.75. Nilai ini diambil dari serangkaian percobaan pendeteksian garis pada ketiga benda uji.

$$T_{min} = \mathbf{0.75} * (biner)_{max}$$

Sedangkan untuk membentuk dimensi eliminasi kernel dilakukan pengamatan hasil HT parameter yang terbentuk untuk ketiga bercobaan material uji. Dari hasil pengamatan HT parameter dapat disimpulkan bahwa nilai 71 dapat mengeliminasi perpotongan matrik HT yang tidak sempurna.. Ukuran kernel ini selalu berjumlah ganjil dengan nilai *peak thresholding* yang selalu berada ditengah kernel. Ketidaksempurnaan perpotongan HT parameter banyak disebabkan oleh pengambilan nilai resolusi *theta* dan *rho* dengan nilai resolusi pengambilan citra yang rendah atau senilai 640 x 480 pixel.

$$NHoodSize = [\,71\ \ 71]$$

Dengan membatasi banyaknya iterasi akan dapat membatasi jumlah garis yang akan terdeteksi. Didalam menetukan banyaknya jumlah garis perlu diperhatikan bentuk kerumitan garis dan kemampuan komputasi yang dimiliki. Jumlah deteksi garis yang akan dilakukan tidak lebih dari 10. Nilai ini pada dasarnya tidak akan menggangu jumlah deteksi yang akan dilakukan selama garis yang akan terdeteksi masih kurang dari nilai 10 garis. Nilai ini bisa digunakan sebagai penggaman fungsi komputasi ketika harus melakukan identifikasi jalur pengelasan.

$$Numlines = 10$$

3.3.2 Algoritma Segmentasi Garis (*Lines Segmentation*)

Algoritma segmentasi dibangun dengan mempertimbangkan keterbentukan garis deteksi pada HT yang mana belum tersegmentasi dengan benar. Dengan mengamati data gambar yang kurang sempurna akan diketahui kebutuhan *fillgap* untuk mengakusisi garis apakah garis tersebut menjadi satu segmen, dua segment, atau tidak menjadi satu segmen sama sekali. Didalam penentuan nilai parameter *fillgap* perlu diamati efek negatif terhadap pelacakan garis terutama terhadap keberadaan pixel yang bukan merupakan bagian dari garis sesungguhnya. Dari beberapa percobaan, diketahui bahwa penggunaan *fillgap* antara 50 s/d 100 cukup akurat untuk citra berdimensi 640 x 480 pixel. *Fillgap* ini harus mampu untuk menggukur keterputusan garis deteksi pada kondisi miring sehingga dapat digunakan teori phytagoras untuk menghitungnya.

$$Fillgap = 75$$

$$Fillgap^2 = (gap\_x)^2 + (gap\_y)^2$$

Sedangkan nilai *minlength* digunakan untuk mengeliminasi garis pendek yang bukan merupakan data segmentasi garis yang digunakan. Garis pendek ini sering ditemukan karena teknik pendeteksian yang dilakukan adalah dengan melakukan deteksi per pixel pada gambar biner secara keseluruhan.

$Minlength = 100$

$Minlength^2 = (length\_x)^2 + (length\_y)^2$

Algoritma 2 Hough transform untuk pendeteksian garis

Input :

- *Image biner*

Output :

- *lines (structure database)*
  Point1
  Point2
  Theta
  Rho
  M (lines gradien)
  C (lines constanta)

Process :

Mulai

- *Baca image biner*
- *Masukan nilai rho resolusi = 1*
- *Masukan nilai theta resolusi = 1*
- *Cari dimensi image biner*
- *Ukur diagonal image dengan pytagoras pada dimensi image*
- *Hitung jumlah rho parameter index dengan persamaan*
  *Nrho = 2 * (diagonal image / rho resolusi)*
- *Buat rho paramater index dengan membuat persamaan*
  rho = linspace(-(diagonal image / resolusi rho,(diagonal image/ resolusi rho), Nrho)
- *Buat theta parameter index*
- *Temukan koordinat image(x,y) yang bernilai 1*
  For (semua nilai x,y yang bernilai 1)
    For (semua nilai theta)
      - *Gunakan persamaan rho = x*cos (theta) + y*sin(theta)*
      - *Masukan nilai rho pada setiap theta pada HT parameter yang telah dibuat*

    End
  End
- *Gunakan Algoritma3 untuk melakukan proses masking*
- *Tentukan nilai peak thresholding dengan mengkalikan 0,75 pada nilai HT maksimal yang terdeteksi*
- *Gunakan numlines = 10 untuk membatasi deteksi garis*
- *Buat HT parameter baru sebagai temporary*
  While (menemukan nilai HT parameter > peak Thresholding)

- *Temukan nilai HT parameter dan simpan nilai theta dan rho pada struktur database*
- *Gunakan kernel NhoodSize = [71 71] untuk mengeliminasi daerah disekitar lokasi nilai maksimal pada HT parameter (temporary) menjadi bernilai 0. Gunakan juga sifat asymetris theta jika menemukan lokasi di ujung HT parameter*
  If (deteksi peak > numlines)
  - *Hentikan while pada proses berikutnya*

  End

End

- *Menentukan nilai fillgap = 100 pixel*
- *Menentukan nilai minlength = 100 pixel*
- *Buat array nilai koordinat(x,y) pada citra biner*

For (hasil deteksi garis)

For (nilai theta hasil peak thresholding)

- *Buat array rho dengan persamaan rho = x*cos (theta) + y*sin(theta)*

End

If (garis mendekati vertikal)

- *Buat nilai index dengan mengurutkan nilai koordinat x dari yang terkecil terlebih dahulu*

Else

- *Buat nilai index dengan mengurutkan nilai koordinat y dari yang terkecil terlebih dahulu*

End

- *Buat array hasil selisih antara nilai koordinat (x,y) dengan nilai koordinat (x+1,y+1) pada nilai pangkatkan 2*
- *Jumlahkan antara x dan y pada setiap selisih sebagai konsep phytagoras untuk menghitung jarak pixel miring*
- *Hitung selisih hasil penjumlahan sebagai nilai gap*
- *Cari nilai gap > fillgap*
- *Simpan nomor index pada nilai gap yang berlebih pada data antara nilai index 1 sampai index terakhir sebagai proses segmentasi dua garis*

If (panjang garis hasil segmentasi > minlength)

- *Simpan nilai titik 1, titik 2, theta, rho pada struktur database garis*
- *Hitung nilai M dan C dengan persamaan*

$$M\ (\textit{lines gradien}) = \frac{titik2(y2) - titik1(y1)}{titik2(x2) - titik1(x1)}$$

C (lines constanta) = (y) – M(x)*

- *Simpan nilai M dan C untuk melengkapi struktur database garis*

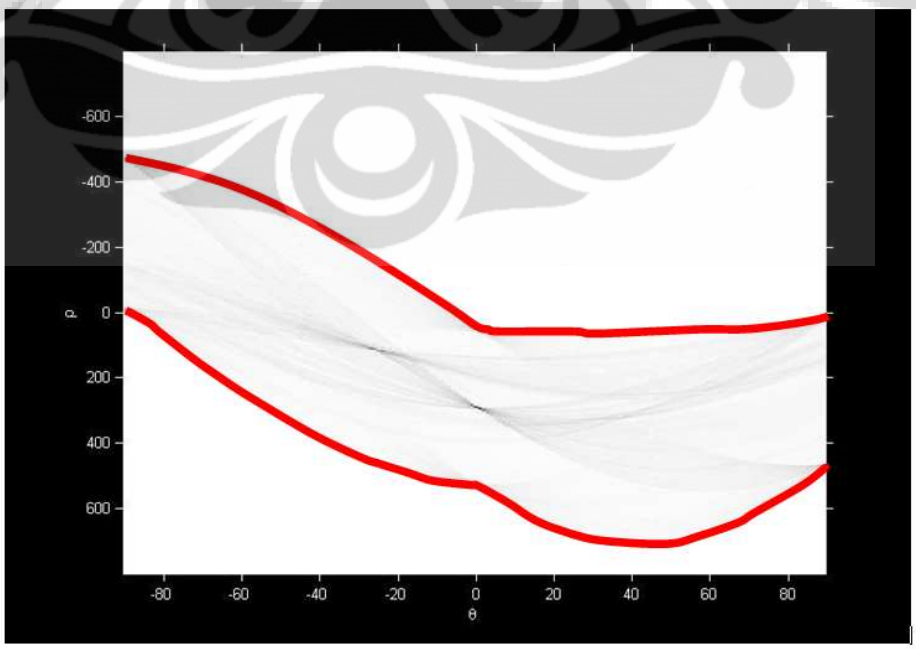
End

Selesai

3.3.3 Algoritma deteksi bagian dalam benda kerja (*Region of Interest)*

Metode yang digunakan adalah dengan memfilter hasil HT parameter yang diberikan. Dengan berbasis pada hasil pengamatan dari serangkaian percobaan didapatkan kesimpulan bahwa dengan memfilter pada bagian permukaan nilai *rho* pada parameter HT terluar akan didapatkan eliminasi garis yang mana garis yang tereliminasi adalah garis tepi permukaan material. Didapat teori bahwa permukaan garis terluar pada sebuah obyek selalu berada pada posisi *rho* terpendek dan posisi *rho* terjauh pada setiap nilai *theta*. Sedangkan penentuan nilai filter dilakukan sebatas menghilangkan deteksi peak yang akan terjadi di daerah tersebut.Pada hasil percobaan diketahui bahwa panjang eliminasi *rho* adalah bernilai antara 15 s/d 20 pixel yang mana nilai tersebut digunakan sebagai nilai filter pada *theta* sesaat. Nilai filter ini harus mampu untuk menghilangkan nilai *peak thresholding* terluar dari HT yang dihasilkan tanpa diproses filter. Teknik filter ini ditambahkan pada nilai *rho*(1) dan dikurangkan pada nilai *rho*(end) pada setiap deteksi nilai *theta*.

$$masking\_rho\_HT = \pm 20$$



Gambar 3.3 Masking rho pada HT parameter terluar

Algoritma 3 Filtering untuk menghilangkan deteksi garis terluar

Input :

- *HT Parameter*

Output :

- *HT parameter baru*

Process :

start

- *masking_rho_HT = 20*

  for (jumlah index theta)

  - *temukan nilai rho terluar sebagai H_rho*
  - *masking H_rho dengan menambahkan/mengurangi nilai masking_rho_HT menjadi bernilai 0*

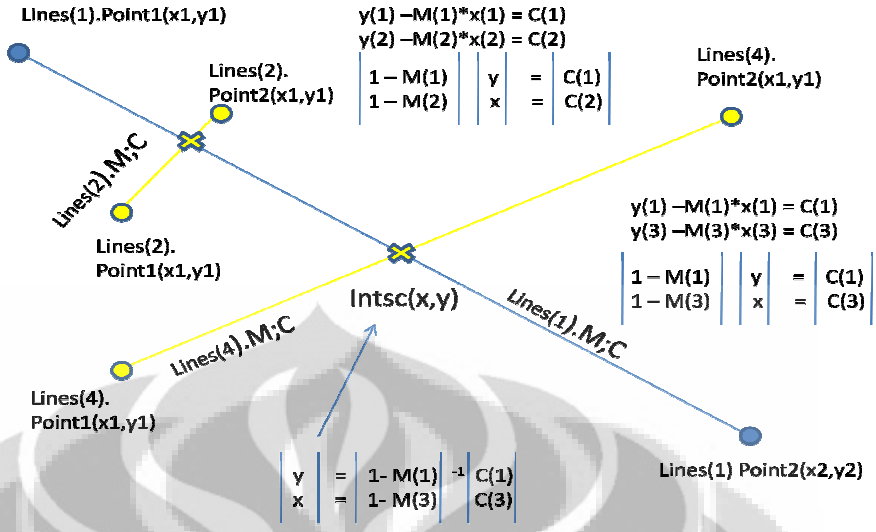
  End

selesai

3.3.4 Algoritma Keterhubungan Garis (*Lines Conectivity*)

Metode yang digunakan untuk mengetahui keterhubungan (*conectivity*) garis yang satu dengan garis yang lainnya adalah dengan mancari perpotongan pada kedua garis. Dengan menganalisa perpotongan semua garis yang terdeteksi terhadap keberadaan garis terdeteksi yang lainya, maka akan dapat diketahui keterhubungan antar garis tersebut dengan garis lainnya.

Dimana (a,b) adalah nomor garis yang saling memotong, **lines(a).M** adalah gradien pada garis lines a dan **lines(a).C** adalah nilai konstanta garis pada lines(a) seperti terlihat pada contoh perpotongan pada gambar 3.3. Dari perpotongan tersebut kemudian dibuat suatu matrik berdimensi jumlah garis yang mana setiap koordinat (x,y) pada matrik dimasukan nilai perpotongannya $(x,y)_{intsection}$. Fungsi dari matrik tersebut selain untuk penggambilan data koordinat titik potong, digunakan juga untuk proses *flaging* saat proses komputasi pencarian jalur pengelasan dilakukan.



Gambar 3.4 Konsep deteksi perpotongan pada intersection

Untuk mempermudah pendataan garis, setiap garis yang terdeteksi dibuat struktur data atau *lines database* yang mana pada setiap garis akan terdapat informasi sebagai berikut :

- Koordinat point awal (x1,y1)
- Koordinat point akhir (x2,y2)
- *Hough Transform* (θ,ρ)
- Gradien kemiringan garis (M)
- Konstanta garis (C)
- Perpotongan pada garis lain (*Intsc)*

```
Algoritma 4 Path Identification
Input :
• Lines stucture database ( M,C )
Output :
• Lines structure database (Line Conectifity)
Process :
Start
  for (semua garis yang terdeteksi)
    If (menemukan garis vertikal pada garis 1)
    • y = (lines2(M)*lines1(rho)) + lines2(C)
    • Intsc = [ lines2(rho) y ]
    Elseif (menemukan garis vertikal pada garis 2)
    • y = (lines1(M)*lines2(rho)) + lines1(C)
    • Intsc = [ lines1(rho) y ]
    Else
```

$$intersection = \begin{bmatrix} -lines1(M) & 1 \\ -lines2(M) & 1 \end{bmatrix}^{-1} * \begin{bmatrix} lines1(C) \\ lines2(C) \end{bmatrix}$$

End
If (koordinat intersection diluar segmentasi)
- *Hapus nilai intersection*

Else
- *Simpan nilai intersection pada matrix connectivity*

End
End
Selesai

3.3.5 Algoritma Deteksi Jalur Pengelasan

Dengan memilih jalur garis yang terpanjang dari serangkaian keterhubungan garis, dapat dipilih bagaimana jalur pengelasan dimulai. Kemudian pendeteksian dilanjutkan untuk mancari jarak terdekat dari posisi terakhir jalur pengelasan yang sudah dilakukan. Sedangkan untuk pendeteksian lanjut, dapat lakukan dengan menganalisa *path lines* berikutnya tanpa melihat *path* yang sudah dilalui. Menggingat batasan yang dilakukan bahwa pengelasan dilakukan pada titik awal, bukan pada koordinat intersection maka proses pengelasan dapat dilakukan lebih efektif. Hal ini dilakukan untuk menghindari gerakan berulang ketika dua garis bertemu pada pada titik intersection.

Algoritma 5 Path Routing Selection

Input :
- *Lines structure database*
- *Matrix connectivity*

Output :
- *Jalur pengelasan terpanjang*

Process :
Start
- *Buat matrik kosong sebagai matrik temporary*

For (semua garis yang terdeteksi)
- *Matrik temporary = matrix connectivity*
- *Lakukan start awal pada titik 1*

While (stop jika proses propagasi garis selesai)
If (menemukan titik intersection)
- *Simpan jalur path dan panjangnya*
- *Ubah start awal dengan nilai intersetion*

Else

- *Simpan jalur path dan panjangnya untuk titik P1 dan P2 garis terakhir secara bersamaan*
- *Lakukan masking untuk intersection terakhir*

If (garis diawali pada P1)

- *Lakukan propagasi ulang untuk garis P2*
- *Kembalikan matrix temporary = matrix connectivity*

Else (garis diawali pada P2)

- *Kembalikan matrix temporary = matrix connectivity*
- *Menghentikan While pada saat proses berikutnya*

End

End

If (menemukan jarak lebih jauh)

- *Ubah jarak terjauh dengan jarak terbaru*
- *Simpan kombinasi path dan nomor index path nya*

End

End

End

- *Buat Segmentasi untuk keseluruhan line database garis*

For (jumlah lines)

- *Lakukan segmentasi per path tiap lines*

End

- *Masking Segmentasi untuk path yang sudah dilalui*

While (masih ada segmentasi yang belum diolah)

- *Gunakan teknik propagasi lanjut dengan mengukur jarak terakhir terhadap jarak point yang belum dilalui.*
- *Gunakan point terdekat untuk melakukan proses berikutnya*
- *Posisi awal = point terdekat*

While (tidak ada keterhubungan segment)

- *Cari nilai posisi terhadap posisi berikutnya pada tabel segmentasi*
- *Save posisi awal pada array path*
- *Masking tabel segmentasi untuk posisi awal dan akhir tersebut*

If (ditemukan keterhubungan segmentasi)

- *Rubah posisi awal = posisi akhir*

Else

- *Hentikan while*

End

End

- *Save array path yang didapat*

End

3.3.6 Pembuatan G-Code

Pembuatan G-Code dilakukan dengan melihat bentuk *path axis* yang dapat teridentifikasi. Dengan membuat struktur *database* lintasan, maka akan dapat di-petakan kondisi jalur pengelasan yang akan dilakukan. Konsep pembuatan G-code adalah dengan melihat hasil perpindahan antar *path* hasil deteksi jalur pengelasan dengan menambahkan gerakan *positioning* antar *path*. *Positioning* dilakukan dengan menghentikan proses pengelasan, mengangkat *torch*, melakukan perpindahan koordinat antar *path*, menurunkan *torch* kembali, dan yang terakhir melakukan pengelasan kembali pada posisi *path* selanjutnya. Sedangkan untuk jarak vertikal gerakan mengangkat *torch* dapat diinputkan ke komputasi, hal ini lebih disebabkan oleh kondisi posisi angkat mesin yang berbeda.

Untuk pembuatan G-Code perlu dilakukan penskalaan kamera terlebih dahulu untuk menyesuaikan *transformasi* koordinat pixel ke koordinat sesunggunya. Dalam penelitian digunakan nilai skala yang bersifat *universal* atau tanpa memperhatikan efek dari kecembungan kamera.

Sedangkan untuk masukan ke *machine controller* diperlukan data format untuk memerintahkan suatu pergerakan (*axis of motion*) mesin welding NC. Data format berguna untuk memerintahkan mesin welding NC bergerak ke posisi yang ditetapkan sekaligus menunjukan suatu besaran perubahan pergerakan tersebut. Data format juga berguna untuk aktifasi welding dan kelengkapan proses lainnya. Di dalam penelitian dirancang suatu data format yang mengikuti dan memodifikasi data format dari proses permesinan NC (ISO 6983) yang selanjutnya disebut G-Code.

Tabel 3.1 Daftar modifikasi perintah G-Code

Fungsi	Kode	Keluaran	Satuan
Menggerakan Motor 1	X	Gerakan sejajar dgn koordinat x	mm
Menggerakan Motor 2	Y	Gerakan sejajar dgn koordinat y	mm
Menggerakan Motor 3	Z	Gerakan sejajar dengan sumbu vertikal	mm
Menyalakan Pengelasan	G01	Mesin las dinyalakan	
Mematikan Pengelasan	G00	Mesin las dimatikan	
FeedRate	F	Kecepatan gerakan motor	mm/det

Dari tabel 3.1 dapat dilihat bahwa untuk menyalakan dan menghentikan aktifitas *torch* pada proses pengelasan dilakukan dengan memodifikasi standar G-code (ISO 6983) dengan menginisialisasi kode G01 sebagai gerakan proses las, dan G00 sebagai gerakan non las atau gerakan bebas.

Algoritma 6 G-Code Generation

Input :
- *Path axis database*

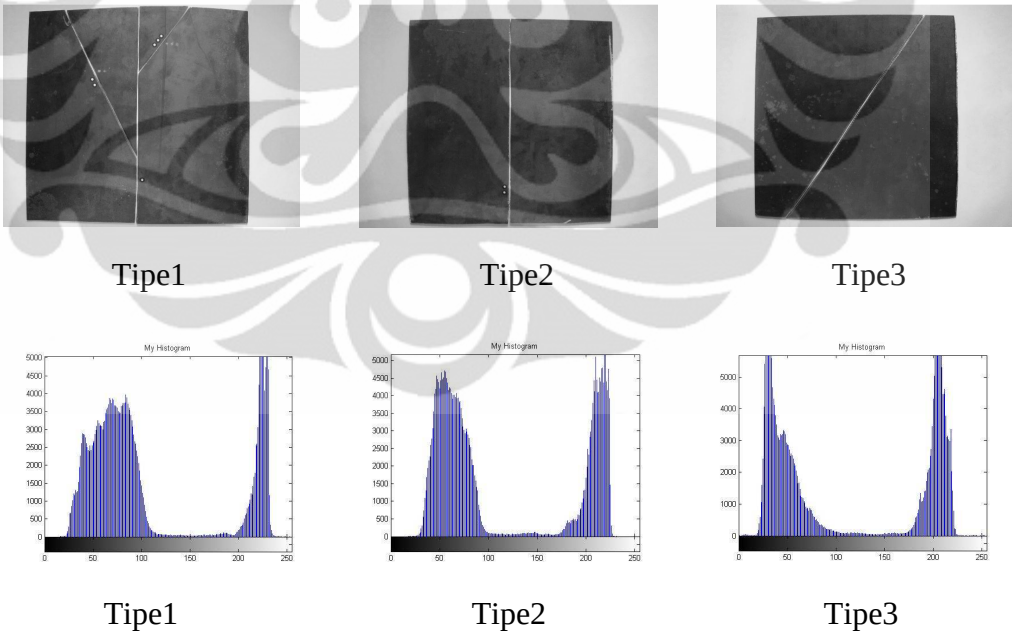
Output :
- *G-Code*

Process :
Start
For (jumlah path yang didapat)
    For (jumlah path.axis yang didapat)
        - *Posisi axis diturunkan ke benda kerja*
    End
        - *Posisi axis dinaikan*
    If (masih ada path)
        - *Menuju posisi berikutnya*
    Else
        - *Menuju Home Coordinate Mesin*
STOP
End
Selesai

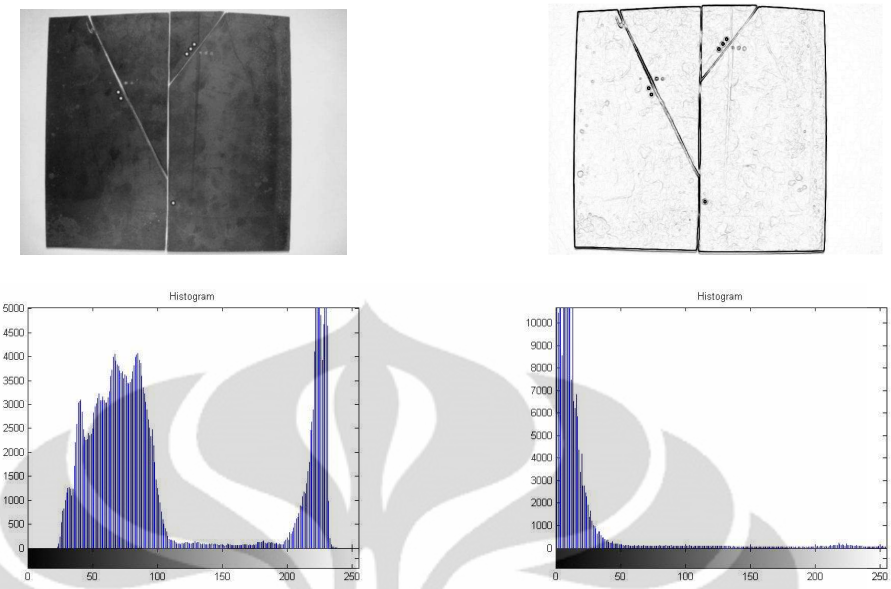
BAB 4
PERCOBAAN DAN ANALISA HASIL

4.1 Hasil Percobaan

Percobaan awal dilakukan dengan sebuah perangkat kamera CCD kamera dengan resolusi 460x640 pixel dengan menggubah gambar 3 layer menjadi *Intensity*. Sumber cahaya yang digunakan adalah LED (*Light Emitter Diode*) dengan daya 3 Watt warna putih dengan arah pencahayaan langsung dari arah vertikal benda kerja. Latar belakang yang digunakan adalah material dengan kondisi warna terang, sedangkan material yang akan diuji adalah plat besi 1.2 mm warna gelap dengan kondisi warna yang tidak homogen. Plat yang digunakan berjumlah 3 tipe, dengan material dan dimensi sama (30 x 30 mm). Didalam pembahasan hasil percobaan yang dilakukan, dapat digunakan satu plat yang dianggap mewakili identifikasi garis ke semua arah.

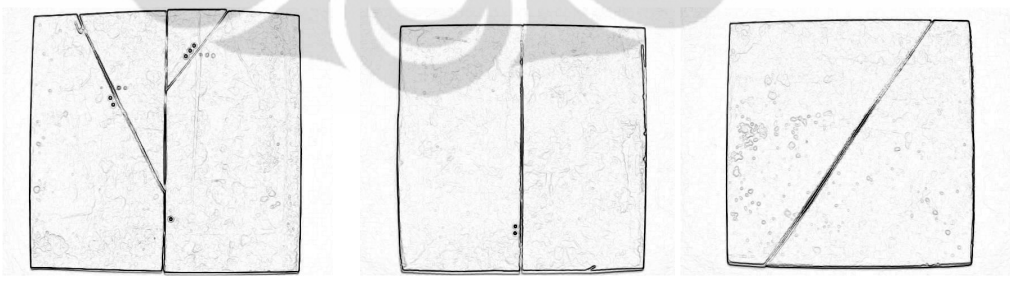


Gambar 4.1 Pengujian histogram pada tiga benda uji



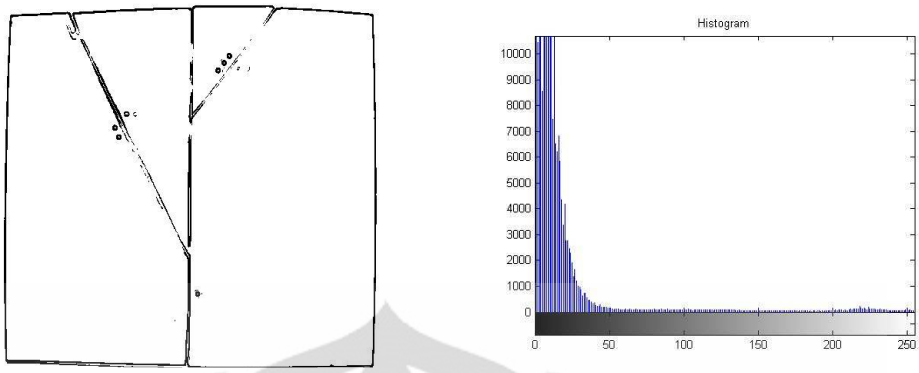
Gambar 4.2 Hasil median filter dan deteksi operator Sobel

Hasil dari percobaan tipe 1 pada gambar 4.2 dapat diketahui bahwa terjadi sedikit pergeseran nilai histogram karena operasi *filter median*. Pada gambar 4.2 juga dapat dilihat bahwa penggunaan operator *Sobel* akan menggeser nilai histogram ke arah kiri, namun demikian citra yang terbentuk masih pada posisi *intensity* (8 bit/pixel) . Sedangkan hasil pengujian untuk ketiga tipe dapat dilihat pada gambar 4.3.



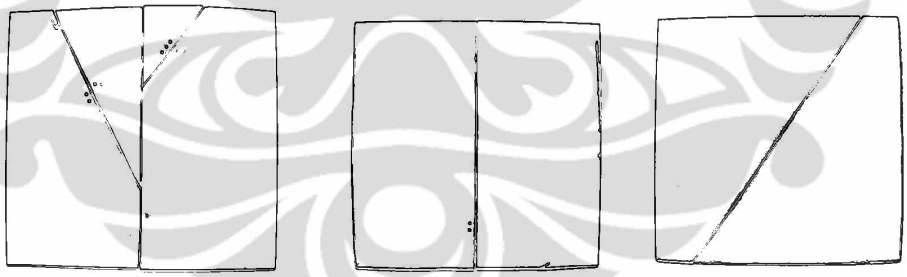
Tipe1 Tipe2 Tipe3

Gambar 4.3 Pengujian algoritma pelacakan tepi dengan sobel operator



Gambar 4.4 Binerisasi *Otsu* (T = 117)

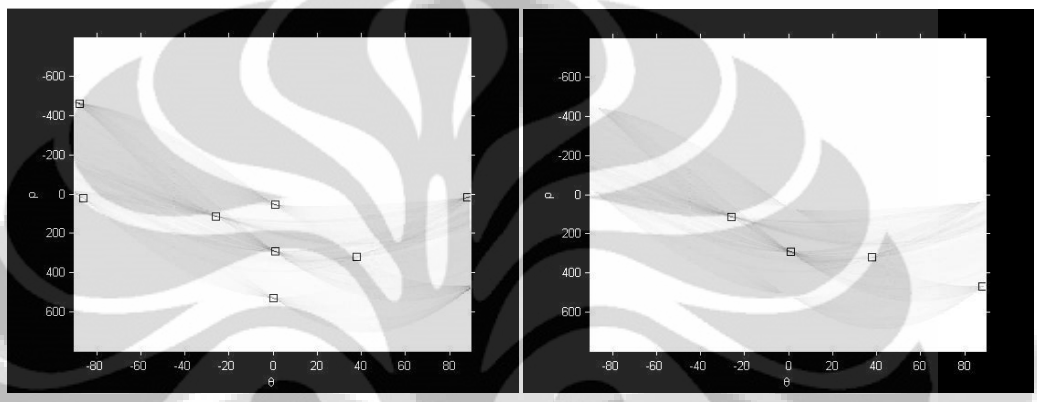
Pada gambar 4.4 dapat dilihat bahwa dengan metode binerisasi *otsu* setelah dilakukan *filter median*, terjadi sedikit keterputusan garis, namun demikian fungsi *filter median* dapat mengeliminasi adanya gangguan pada citra berfrekuensi tinggi (*low pass filter*). Hasil binerisasi pada proses binerisasi *otsu* untuk pengujian ketiga tipe mempunyai hasil seperti terlihat pada gambar 4.5.



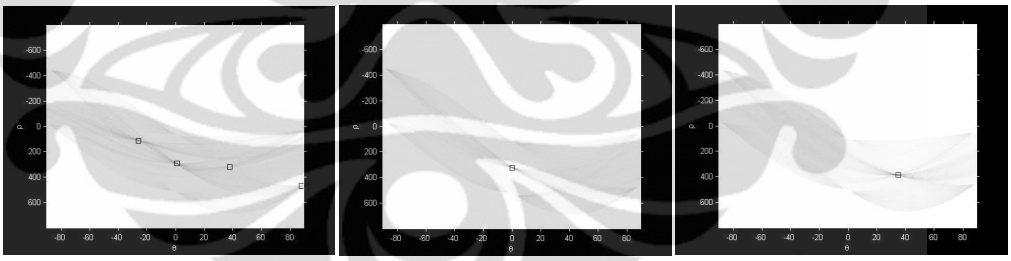
Tipe 1 (*Threshold*=117) | Tipe 2 (*Threshold*=108) | Tipe 3 (*Threshold*=108)

Gambar 4.5 Pengujian algoritma binerisasi dengan metode Otsu

Sedangkan pada gambar 4.6 dapat dilihat bahwa nilai *peak thresholding* pada parameter HT setelah dan sebelum dilakukan filter mempunyai jumlah *peak* yang berbeda. Sedangkan untuk hasil pengujian HT parameter setelah difilter untuk ketiga benda uji dapat dilihat pada gambar 4.7. Hal ini sedikit membuktikan bahwa nilai filter yang dilakukan pada rho dapat menghilangkan *peak* untuk ketiga kondisi yang berbeda.

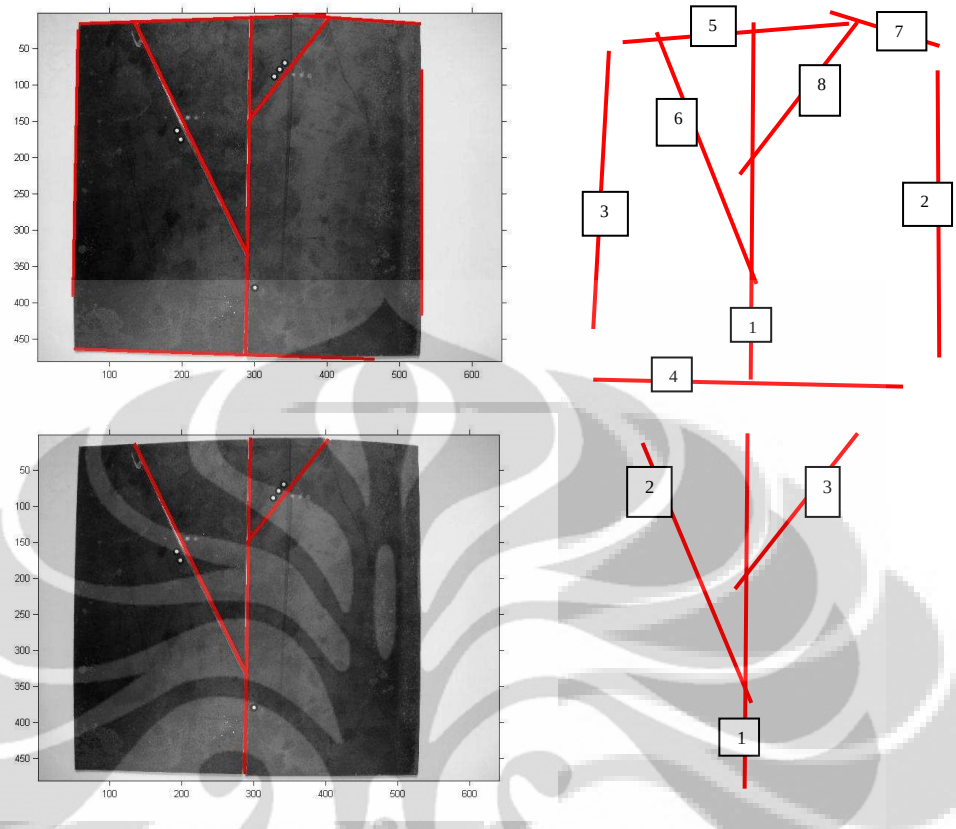


Gambar 4.6 HT sebelum dan sesudah diproses filter



Tipe 1 Tipe 2 Tipe 3

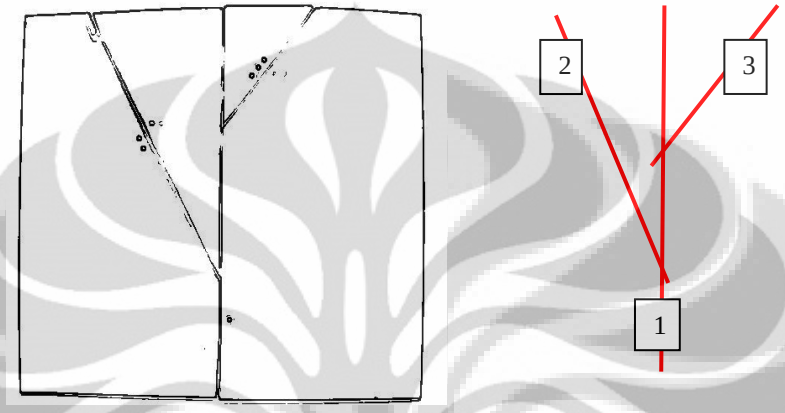
Gambar 4.7. Hasil pengujian HT setelah di filter pada ketiga tipe



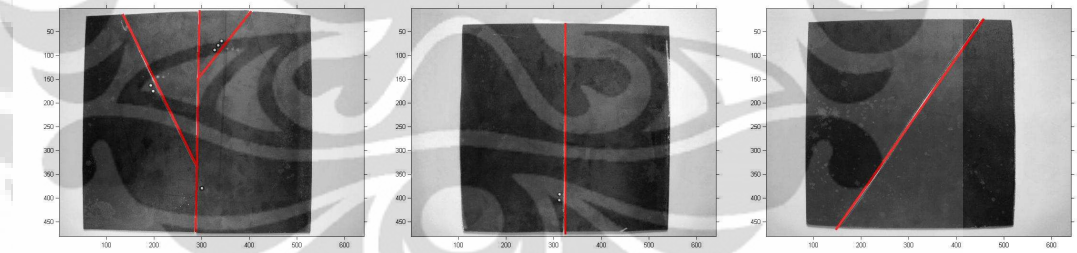
Gambar 4.8. Segmentasi garis yang didapat sebelum (atas) dan sesudah (bawah) di proses filter untuk percobaan pada benda uji tipe 1

Pada gambar 4.8 dapat dibuktikan bahwa pendeteksian garis yang dihasilkan sangat sesuai dengan kebutuhan. Dari semula terdeteksi delapan garis dengan melakukan filter akan menjadi terdeteksi tiga garis, dan dari ketiga garis yang berhasil dideteksi tersebut benar menunjukan jalur pengelasan yang sesungguhnya.

Pada gambar 4.9 dapat dilihat bahwa algoritma segmentasi garis dengan *fillgap* dengan nilai 100 dan *minlength* dengan nilai 100 mampu untuk melakukan segmentasi dan mampu melakukan eliminasi garis pendek sesuai kebutuhan. Hasil pengujian segmentasi garis untuk ketiga tipe dapat dilihat pada gambar 4.10.



Gambar 4.9 Labeling untuk segmentasi garis sebelum dan sesudah di *filter*



Tipe 1 Tipe 2 Tipe 3

Gambar 4.10 Hasil pengujian segmentasi garis

Pada tabel 4.1 dapat dilihat bahwa hasil database garis yang dihasilkan pada proses pendeteksian HT sejumlah 3 garis dengan posisi titik awal di P1 dan berakhir di P2 pada arah *theta* dan *rho* dengan dapat dilihat keterhubungan antara garis yang satu dengan garis lainnya sehingga akan mempermudah dalam pembuatan tabel 4. 2 sebagai matrik koordinat *intersection*.

Tabel 4.1 Hasil database untuk segmentasi garis

Lines	1	2	3
P1(x,y)	[295 5]	[134 13]	[403 7]
P2(x,y)	[287 471]	[291 336]	[290 151]
Θ (deg)	1	-26	38
ρ	294	114	320
M	-58.2500	2.0573	-1.2743
C	1.7189e+004	-262.6815	520.5575
Intsc	[2 3]	1	1

Tabel 4.2 Hasil intersection antara dua garis pada koordinat (x,y)

	1	2	3
1		290, 333	293, 148
2	290, 333		---
3	293, 148	---	

Hasil iterasi pencarian garis pada tabel 4.3 dilakukan untuk semua titik pada setiap garis jalur pengelasan yang mungkin. Data yang dihasilkan didekati dengan nilai *integer* terdekat. Dari hasil pencarian garis didapatkan bahwa *index* 11 dan 21 mempunyai hasil jarak yang sama yang disebabkan oleh efek keterbalikan pada pendeteksian jalur pengelasan. Didalam batasan masalah dalam penelitian ini maka hanya diambil nilai *path* terjauh yang pertama kali terdeteksi tanpa menganalisa hasil yang dilakukan pada keterbalikan gerakan.

Tabel 4.3. Hasil iterasi jalur pengelasan

No.*Path*	1	2	3	4	5	6	7
Path (x,y)	295 5 290 333 134 13	295 5 290 333 291 336	295 5 293 148 403 7	295 5 293 148 290 151	295 5 287 471	287 471 290 333 134 13	287 471 290 333 291 336
Jarak	684.0381	331.2004	321.8463	147.2566	466.0687	494.0326	141.1949

Tabel 4.3. Hasil iterasi jalur pengelasan (sambungan)

No.*Path*	8	9	10	11	12	13	14
Path (x,y)	287 471 293 148 403 7	287 471 293 148 290 151	287 471 295 5	134 13 290 333 293 148 403 7	134 13 290 333 293 148 290 151	134 13 290 333 295 5	134 13 290 333 287 471
Jarak	501.888	327.2984	466.0687	719.8566	545.267	684.0381	494.0326

Tabel 4.3. Hasil iterasi jalur pengelasan (sambungan)

No.*Path*	15	16	17	18	19	20	21
Path (x,y)	134 13 291 336	291 336 290 333 293 148 403 7	291 336 290 333 293 148 290 151	291 336 290 333 295 5	291 336 290 333 287 471	291 336 134 13	403 7 293 148 290 333 134 13
Jarak	359.1351	367.0189	192.4292	331.2004	141.1949	359.1351	719.8566

Tabel 4.3. Hasil iterasi jalur pengelasan (sambungan)

No.*Path*	22	23	24	25	26	27	28
Path (x,y)	403 7 293 148 290 333 291 336	403 7 293 148 295 5	403 7 293 148 287 471	403 7 290 151	290 151 293 148 290 333 134 13	290 151 293 148 290 333 291 336	290 151 293 148 295 5
Jarak	367.0189	321.8463	501.888	183.0437	545.267	192.4292	147.2566

Tabel 4.3. Hasil iterasi jalur pengelasan (sambungan)

No.*Path*	29	30
Path (x,y)	290 151 293 148 287 471	290 151 403 7
Jarak	327.2984	183.0437

Kemudian dari nilai *path* tersebut kemudian dicari jarak nilai terdekat dari posisi terakhir gerakan. Hasil titik yang didapat kemudian dicari apakah masih ada keterhubungan garis terhadap titik tersebut. Jika ada, maka masukan ke koordinat selanjutnya dan jika tidak ada, maka akan dibuat *array* pada *path* berikutnya dengan mencari kembali jarak titik terdekat dari posisi titik terakhir. Hasil percobaan yang didapat dapat dilihat pada tabel 4.4.

Tabel 4.4 Hasil *path* yang terdeteksi untuk gerakan pengelasan

Path(1).axis		*Path*(2).axis		*Path*(3).axis	
x	y	x	y	x	y
134	13	295	5	291	336
290	333	293	148	290	333
293	148	290	151	287	471
403	7				

Untuk penggunaaan pada mesin NC perlu adanya kegiatan penskalaan antara koordinat mesin dengan pixel kamera. Konsep penskalaan dilakukan dengan melihat kertas milimeter untuk diletakan pada jarak fokus dari titik tengah pada lensa kamera. Setiap jarak pada kertas milimeter dibandingkan dengan jarak pixel yang dihasilkan. Hasil penskalaan dapat diberikan pada angka **0,625** mm/pixel yang dilakukan pada jarak fokus **560** mm dari posisi lensa kamera ke benda kerja dengan resolusi gambar 480 x 640 pixel. Sedangkan jarak gerakan mengangkat *torch* dapat diinputkan ke algoritma senilai **-100 sd -200**, hal ini lebih dikarenakan dari spesifikasi mesin NC sebagai alat percobaan.

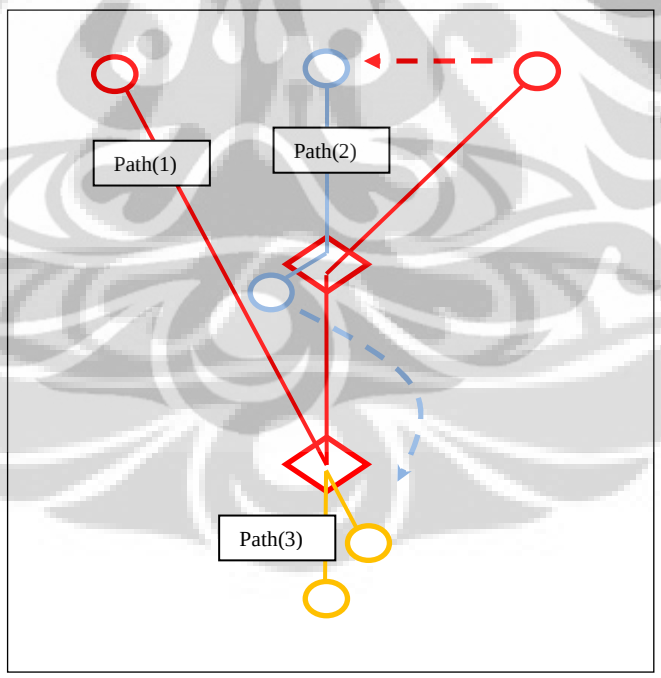
Kemudian hasil dari penskalaan yang telah dilakukan disesuaikan terhadap G-Code yang dihasilkan. Hasilnya dapat dilihat pada gambar 4.11 Untuk nilai *feeding* proses pengelasan, tidak ditentukan secara spesifik didalam penelitian ini.

START							mesin diaktifkan
GO HOME							mesin dikembalikan ke posisi home pada mesin
G00	X	84	Y	8	Z	-200	OFF, keposisi awal *path* 1
G00	X	84	Y	8	Z	-100	OFF, torch turun ke benda kerja
G01	X	181	Y	208	Z	-100	ON, pengelasan dimulai, bergerak ke koordinat selanjutnya
G01	X	183	Y	93	Z	-100	ON, bergerak ke koordinat selanjutnya
G01	X	252	Y	4	Z	-100	ON, bergerak ke koordinat selanjutnya
G00	X	252	Y	4	Z	-200	OFF, torch diangkat ke posisi aman
G00	X	184	Y	3	Z	-200	OFF, bergerak ke posisi *path* 2
G00	X	184	Y	3	Z	-100	OFF, torch turun ke benda kerja
G01	X	183	Y	155	Z	-100	ON, pengelasan dimulai, bergerak ke koordinat selanjutnya
G01	X	181	Y	94	Z	-100	ON, bergerak ke koordinat selanjutnya
G00	X	181	Y	94	Z	-200	OFF, torch diangkat ke posisi aman
G00	X	182	Y	210	Z	-200	OFF, bergerak ke posisi *path* 3
G00	X	182	Y	210	Z	-100	OFF, torch turun ke benda kerja
G01	X	181	Y	208	Z	-100	ON, pengelasan dimulai, bergerak ke koordinat selanjutnya
G01	X	179	Y	294	Z	-100	ON, bergerak ke koordinat selanjutnya
G00	X	179	Y	294	Z	-200	OFF, torch diangkat ke posisi aman
GO HOME							kembali ke posisi home mesin
STOP							mesin selesai

Gambar 4.11 Hasil G-Code dengan skala dan keterangannya

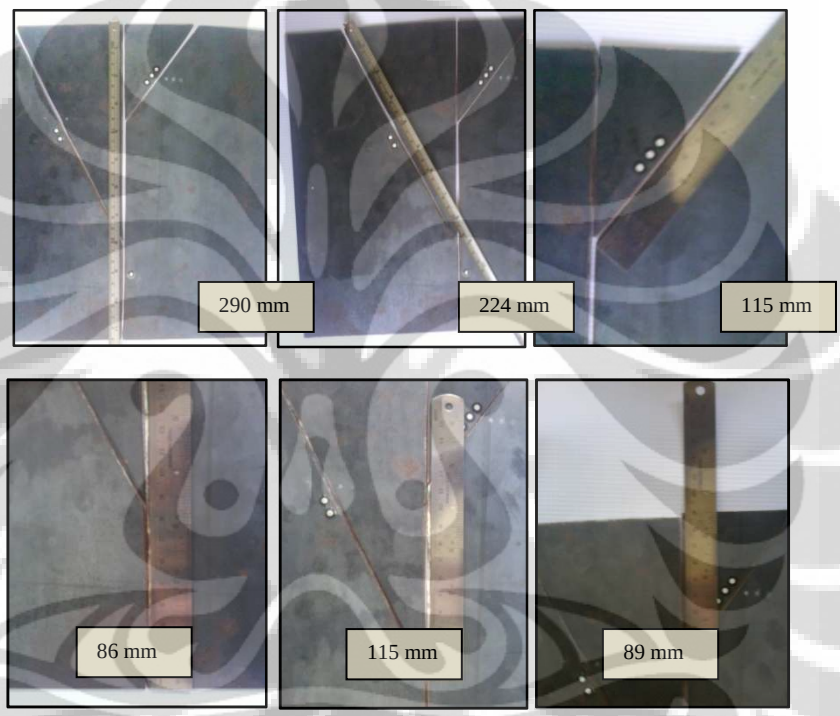
4.2 Pembahasan dan Analisa

Dari hasil iteasi garis pada gambar 4.12 dapat disimpulkan bahwa jumlah iterasi *path* dilakukan sebanyak **30** kali dengan menghasilkan panjang line terjauh sebesar **719,8566** pixel . Dari hasil analisa diatas didapat kesimpulan bahwa *path* yang akan dimulai dari **lines(2).P1 [403, 7]** kemudian bergerak ke **intersection_2 [293,148]** diikuti pergerakan ke arah **intersection_1 [290,333]**. Dari posisi Intersection_1 kemudian diakhiri ke **lines(3).P1 [134,13]** sebagai bagian dari gerakan *path* pertama. Dari posisi terakhir *path* tersebut, kemudian posisi Z ditarik keatas sejauh **-100**, dan kemudian dilakukan menuju ***path*(2).axis [295 5]** menuju **intersection_2 [293,148]** diikuti ke posisi **lines(3).P2 [290,151]** dan berhenti. Sama seperti kegiatan sebelumnya, pengelasan dimulai **pada lines(1).P2 [291 336]** menuju **intsection_1 [290 333]** dan diakhir pada **lines(1).P2 [287 471]**. Sebelum menyelesaikan proses secara keseluruhan, posisi Z ditarik keatas untuk kemudian di kembalikan ke posisi koordinat mesin.



Gambar 4.12 *Welding Path*

Dari hasil penyesuaian nilai koordinat yang didapat, lalu dibandingkan dengan jarak antar *path* pada hasil pengukuran material sebenarnya seperti terlihat pada gambar 4.13 Hasil perbandingan antara pengukuran benda nyata dengan hasil *path* yang dihasilkan dapat dilihat pada tabel 4.5. Dari hasil perbandingan dapat disimpulkan bahwa akurasi terendah yang didapat adalah **0,02** yang terjadi pada titik antara **intersection_1 [290 333]** ke titik **lines(3).P2 [290 151].**



Gambar 4.13 Pengukuran benda sebenarnya

Tabel 4.5 Nilai akurasi yang diperoleh pada *path*

PATH (1)	Tanpa Skala			*0,625 mm/pixel	Dengan Skala			Real Geometri	
	Koordinat		Jarak		Koordinat		Jarak	Jarak	Akurasi
	x	y	(pixel)		x	y	(mm)	(mm)	
	134	13	356		84	8	222.2814	224	0.0077
	290	333	185.02432		181	208	115.0174	115	0.0002
	293	148	178.83232		183	93	112.6144	115	0.0207
	403	7			252	4			
	Total Jarak		720		Total Jarak		450	454	0.0088

Tabel 4.5 Nilai akurasi yang diperoleh pada *path* (sambungan)

	Tanpa Skala				Dengan Skala			Real Geometri	
	Koordinat		Jarak		Koordinat		Jarak	Jarak	Akurasi
PATH(2)	x	Y	(pixel)	*0,625 mm/pixel	x	y	(mm)	(mm)	
	295	5	143.01399		184	3	90.00556	89	0.0113
	293	148	4.2426407		183	93	2.236068	0	
	290	151			181	94			
	Total Jarak		147		Total Jarak		92	89	0.0337

Tabel 4.5 Nilai akurasi yang diperoleh pada *path* (sambungan)

	Tanpa Skala				Dengan Skala			Real Geometri	
	Koordinat		Jarak		Koordinat		Jarak	Jarak	Akurasi
PATH(3)	x	Y	(pixel)	*0,625 mm/pixel	x	y	(mm)	(mm)	
	291	336	3.1622777		182	210	2.236068	0	
	290	333	138.0326		181	208	86.02325	86	0.0003
	287	471			179	294			
	Total Jarak		141		Total Jarak		88	86	0.0233

Sedangkan nilai akurasi G-Code dapat diukur dari G-Code yang dihasilkan dibandingkan dengan hasil pengukurannya. Akurasi ini berpotensi tidak sama dengan akurasi per *path* seperti yang telah dilakukan pada tabel 4.5 , hal ini banyak disebabkan karena pendekatan nilai integer yang terjadi. Hasil perbandingannya dapat dilihat pada tabel 4.6. Akurasi terendah yang dapat dilakukan adalah **0,02** yang secara kebetulan bernilai sama pada tempat yang sama dengan nilai akurasi per *path*.

Tabel 4.6 Nilai akurasi G-Code yang dihasilkan

Image		Dikalikan dengan faktor skala sebesar 0,625	G-CODE setelah diskala		Didekati pd nilai integer	Gerak		Jarak	Didekati pd nilai integer	Jarak (*nearest*)	Akurasi
(Pixel)			(mm)			(mm)		(mm)		(mm)	
x	y		X	y		x	y	SQRT(x^2 + y^2)			
134	13		84	8		84	8	222.28	222	224	0.01
134	13		84	8		181	208	115.02	115	115	-
290	333		181	208		183	93	112.61	113	115	0.02
293	148		183	93		252	4	68.01	68	Reposisi	
403	7		252	4		184	3	90.01	90	89	0.01
403	7		252	4		183	93	2.24	2	0	tdk ada
295	5		184	3		181	94	116.00	116	Reposisi	
295	5		184	3		182	210	2.24	2	0	tdk ada
293	148		183	93		181	208	86.02	86	86	-
290	151		181	94		179	294				
290	151		181	94							
291	336		182	210							
291	336		182	210							
290	333		181	208							
287	471		179	294							
287	471		179	294							

Dari nilai akurasi terendah dari proses pembuatan 'G-code' sebesar **0.02** didapatkan bahwa proses pengelasan dengan bantuan mesin *vision* mampu untuk melakukan gerakan pengelasan dimana masih jauh diatas dari akurasi proses penggelasan itu sendiri (~1mm).

BAB 5
KESIMPULAN DAN SARAN PENELITIAN LEBIH LANJUT

Pada penelitian identifikasi dan pembuatan jalur pengelasan dengan menggunakan mesin *vision* yang telah dilakukan maka didapat hasil kesimpulan dan saran penelitian lebih lanjut sebagai berikut :

5.1 Kesimpulan

Dari serangkaian hasil percobaan yang telah dilakukan sebagai apresiasi hasil dari tujuan penelitian didapatkan kesimpulan bahwa :

1. Penggunaan *median filter* paling sesuai untuk digunakan sebagai *low pass filter* pada variasi bentuk citra yang sederhana.
2. Penggunaan operator Sobel mempunyai hasil paling jelas dan tebal untuk dilihat bentuk *edge* benda kerja dan bentuk lintasan las.
3. *Filter* yang dilakukan pada parameter HT terbukti mampu untuk mendeteksi garis hanya pada bagian lintasan las selama benda kerja dan *background* cukup memiliki nilai kontras.
4. Algoritma yang dibangun mampu untuk memilih jalur pengelasan terpanjang dan untuk diikuti dengan pemilihan jarak lokasi terpendek dari lintasan terakhir.
5. Kepresisian yang dihasilkan didalam percobaan layak untuk dilakukan pada proses pengelasan.

5.3 Saran penelitian lebih lanjut

Untuk penelitian pada tema manufaktur berbasis pada pemanfaatan teknologi mesin *vision* khususnya penggunaan didalam proses pengelasan, panulis menyarankan untuk dilakukan penelitian sebagai berikut :

1. Pemilihan jalur pengelasan pada bentuk bukan garis lurus.
2. Pembuatan jalur pengelasan pada benda kerja pada bentuk 3 dimensi yang prosesnya dapat dilakukan dengan artikulasi robot 5 derajat kebebasan.

DAFTAR REFERENSI

1. *A Prototype System of Three Dimensional Non Contact Measurement.* **Huang, SJ and Lin, YW.** s.l. : International Journal of Advance Manufacturing Technology, 1996, Vol. 11, pp. 336-342.

2. *A Three Dimensional Non-Contact Measurement System.* **Huang, SJ, Lin and Chang, C.** s.l. : International Journal of Advanced Manufacturing Technology, 1997, Vol. 13, pp. 419-425.

3. *Image Processing for Automated Robotic Welding.* **P, Seyffarth and R, Gaede.** s.l. : Springer-Verlg BErlin Heidelberg, 2011, Robotic Welding, Intelligence and Automation, pp. LNEE 88, 15-21.

4. *Usulan Sistem Monitor Jalur Pengelasan pada Robot Las menggunakan Machine Vision.* **Baskoro, AS, Kiswanto, G and Santoso, T.** Palembang : SNTTM, 2010, SNTTM IX, Vol. IX, p. 145 :155. ISBN 978-602-97742-0-7.

5. *Automatic Seam Detection and Path Planning in Robotic Welding.* **Micallef, K, Fang, Gu and Dinham, M.** s.l. : Springer-Verlag Berlin Heidelberg, 2011, Journal of Robotic Welding, Intelligence and Automation, pp. LNEE 88 (23-32). ISBN 978-3-642-19958-5.

6. *Pengembangan Robot 2-Axis Rotasi untuk Robot Las dengan 5 Derajad Kebebasan.* **Kiswanto, G and Baskoro, AS, Santoso, T.** Palembang : s.n., Oktober 2010, SNTTM, Vol. IX, pp. 139-144. ISBN 978-602-97742-0-7.

7. **Hornberg, A.** *Handbook of Machine Vision.* Weirhem : Willey VCH Verlag Gmbh Co KGaA, 2006. ISBN-13: 978-3-527-40584-8.

8. **ER, Davies.** *Machine Vision ; Teorema, Algoritm, Practicalities.* 3rd Edition. San Francisco : Elsevier, 2005.

9. *Estimation of Moving Information for Tracking Moving Object.* **Kang, SK and JK, Park.** s.l. : KSME Journal, 2001, Vol. 15, pp. No.3 ,300-308.

10. *Non-Parameter Histogram-Based Thresholding Methods for Weld Defect Detetion in Radiography.* **Nacereddine, N, et al.** s.l. : World Academy of Science, Engineering and Technology, 2005, Vol. 9, pp. 213-217.

11. **Gonzalez, RC Wood & Eddins.** *Digital Image Processing Using Matlab.* s.l. : Prentice Hall, 2000.

12. *Line Segment and Dominate Points Detection Based from Hough Transform.* **Liao, ZW and Hu, SX.** s.l. : Springer-Verlag Berlin Heidelberg, Vol. II, pp. LNAI 3802 (917-922).

13. *Acquisition of Weld Seam Dimensional Position Information for Arc Welding Robot Based on Vision Computing.* **Chen, SB, Chen, XZ and Qiu, T.** s.l. : Springer, 2005, Journal of Intelligent and Robotic Systems, pp. 43, 77-97.

LAMPIRAN
KEGIATAN KOMPUTASI

Kegiatan	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Pemilihan jalur pengelasan yang terbaik untuk mempercepat waktu proses
Perumusan Masalah	Melakukan propagasi berulang terhadap pemilihan proses jalur pengelasan akan menimbulkan kebutuhan proses baik dari segi keakurasian hasil, kebutuhan memori dan waktu komputasi.
Identifikasi Masalah	Hasil dari pemilihan jalur pengelasan akan kita bandingkan terhadap parameter akurasi data, kebutuhan memori dan waktu yang diperlukan sebagai biaya komputasi. Dengan mencari parameter tepat sesuai dengan kebutuhan akan kita lakukan untuk mencapai *low cost computation*.
Metodologi yang digunakan	Mencoba melakukan proses komputasi dengan menganti variabel data seperti : pixel pengambilan citra, dan tidak menutup kemungkinan untuk menganti urutan proses yang sudah dilakukan.

```
%IDENTIFIKASI DAN PEMBUATAN JALUR PENGELASAN DENGAN MACHINE VISION
% ------------ OLEH : ALBERTUS RIANTO SURYANINGRAT, ST ----------
% ---------------------- NPM  : 0906496163 ----------------------
% -------- DOSEN PEMBIMBING : DR. IR. GANDJAR KISWANTO, M.ENG ----
% ----- DEPARTEMEN TEKNIK MESIN FAK.TEKNIK UNIVERSITAS INDONESIA -
% ------------------------------- 2011 ----------------------------
clc;close all; clear;
fig_num = 0;
%% ----------------------- IMAGE DATA AKUSISI -------------------
% ---------------------- Konsep yang dikembangkan ----------------
%Image data akusisi digunakan untuk mensetting data akusisi kamera
%yang mana pada teknik pengambilan gambar memperlukan parameter
%yang terbaik untuk diambil citranya.
% ----------------------------------------------------------------
% ------------------------ Kesimpulan Sementara ------------------
%Didapat kesimpulan bahwa penggunaan pixel 640x480 cukup untuk
%menemukan lintasan las. Dengan melakukan warming up kamera
%membuat citra yang diambil sangat jelas.
%Penggunaan Greyscale terbukti mengurangi komputasi yang akan
%dilakukan dalam proses selanjutnya.
% ----------------------------------------------------------------
%%
% --------------------- IMAGE DIAMBIL DARI KAMERA ----------------
image = citra_kamera(2);
fig_num = fig_num + 1; figure(fig_num),imshow(image),title('Citra
Kamera');
imwrite(image,'image1.jpg');
```

```
% ------------------------------------------------------------------
%%
% ------------------ IMAGE DIAMBIL DARI DATA MEMORY --------------
image=imread('image1.jpg');
% ------------------------------------------------------------------
%% ---------------- PERBAIKAN CITRA ( IMAGE ENHANCEMENT) ---------
%  ----------------------- Konsep yang dikembangkan ----------------
% Perbaikan citra digunakan untuk memperbaiki hasil identifikasi
% jalur pengelasan. Dengan Perbaikan citra diharapkan akan
% mengurangi adanya noise yang terjadi karena perubahan warna
% yang mendadak(low pass filter)
% ------------------------------------------------------------------
% ------------------------ Kesimpulan Sementara --------------------
% Penggunaan Filter Median lebih jelas teridentifikasi adanya
% sebuah garis pada permukaan plat dibanding dengan penggunaan pd
% filter Gaussian 1/16 dan Mean Gaussian.
% ------------------------------------------------------------------
%%
% ----------------- PERBAIKAN CITRA DENGAN GAUSSIAN 1/16 ----------
image = gaussian_16(image);
fig_num = fig_num + 1; figure(fig_num),imshow(image),title('Gauss
1/16');
imwrite(image,'gaussian_16.jpg');
%%
% ----------------- PERBAIKAN CITRA DENGAN MEAN GAUUSIAN ----------
image = mean_gauss(image);
fig_num = fig_num + 1; figure(fig_num),imshow(image),title('Mean
Gauss');
% imwrite(image, 'mean gaussian.jpg');
% ------------------------------------------------------------------
%%
% ----------------- PERBAIKAN CITRA DENGAN FILTER MEDIAN ----------
image = filter_median(image);
fig_num = fig_num + 1; figure(fig_num),imshow(image),title('Med
Filter');
imwrite(image,'median filter.jpg');
% ------------------------------------------------------------------
%% ------------------------- EDGE DETECTION ----------------------
% ------------------------ Konsep yang dikembangkan ----------------
% Edge Detection digunakan untuk mencari batas antar dua bidang.
% ------------------------------------------------------------------
% ------------------------ Kesimpulan Sementara --------------------
% Didapat bahwa dengan sobel operation, garis yang teridentifikasi
% terlihat sangat jelas (tebal ) dibanding dengan prewitt
operation
% ------------------------------------------------------------------
%%
% ----------------- EDGE DETECTION DENGAN SOBEL OPERATION ---------
image = pelacak_sobel(image);
fig_num = fig_num + 1; figure(fig_num), imshow(image),title('Sobel
Magnitude');
imwrite(image, 'sobel magnitude.jpg');
% ------------------------------------------------------------------
%%
% --------------- EDGE DETECTION DENGAN PREWITT OPERATION ---------
image = pelacak_prewitt(image);
fig_num = fig_num + 1; figure(fig_num),
imshow(image),title('Prewitt Magnitude');
imwrite(image, 'prewitt magnitude.jpg');
```

```
% ------------------------------------------------------------------
%%
% --------------------- EDGE DETECTION DENGAN LOG ----------------
image = pelacak_log(image);
fig_num = fig_num + 1; figure(fig_num), imshow(image),title('Log
Magnitude');
imwrite(image, 'log magnitude.jpg');
% ------------------------------------------------------------------
%%
% ------------------- EDGE DETECTION DENGAN KRISCH ---------------
image = pelacak_krisch(image);
fig_num = fig_num + 1; figure(fig_num),
imshow(image),title('Krisch Magnitude');
imwrite(image, 'krisch magnitude.jpg');
% ------------------------------------------------------------------
%%
% ------------------- EDGE DETECTION DENGAN ROBERT ---------------
image = pelacak_robert(image);
fig_num = fig_num + 1; figure(fig_num),
imshow(image),title('Robert Magnitude');
imwrite(image, 'robert magnitude.jpg');
% ------------------------------------------------------------------
%%
% ----------------- EDGE DETECTION DENGAN PELACAK ARAH -----------
image = pelacak_arah(image);
fig_num = fig_num + 1; figure(fig_num),
imshow(image),title('Palacak Arah');
imwrite(image, 'pelacak arah magnitude.jpg');
% ------------------------------------------------------------------
% ------------------ THRESHOLDING DENGAN METODE OTSU -------------
image = otsu(image);
fig_num = fig_num + 1; figure(fig_num),
imshow(image),title('Otsu');
imwrite(image, 'otsu.jpg');
% ------------------------------------------------------------------
%%
% ---------- LINE DETECTION DENGAN ALGORITMA HOUGH TRANSFORM -----
% DETEKSI YANG DIGUNAKAN MENGGUNAKAN PARAMETER SPACE THETA DAN RHO
% --------------------------- data parameter HT --------------------
b_biner = image;
RhoRes= 1;
ThetaRes= 1;
NHoodSize = [ 71 71];
threshold = 90;
numpeaks=10;
fillgap=75;
minlength=100;
%%
% ----------------- MEMBUAT KOORDINAT PARAMETER HT --------------
%  ----------------------- Dengan Syntac Matlab ------------------
[H,theta,rho]   = hough(b_biner,'RhoResolution',...
RhoRes,'ThetaResolution',ThetaRes);
[H,theta,rho]   = hough(b_biner,'RhoResolution',...
RhoRes,'ThetaResolution',ThetaRes);
% ------------------------------------------------------------------
% ------------------------ Masking rho pada HT -------------------
for i = 1: numel(theta);
    mat = find(H(:,i)~=0);
    a = mat(1);
```

```
    b = mat(end);
    H(a:(a+20),i) = 0;
    H((b-20):b,i) = 0;
end
% ------------------------------------------------------------------
%%
% ------------------ MEMBUAT KOORDINAT PARAMETER HT --------------
% ---------------------- Dengan Syntac Sendiri---------------------
% [H, theta,rho] = hough_transform(b_biner, RhoRes, ThetaRes);
% ------------------------------------------------------------------
% ------------------ Kelemahan yang belum terpecahkan ------------
% Untuk membuat koordinat parameter HT, belum mempunyai algoritma
% yang cepat untuk dilakukan iterasi.
% ------------------------------------------------------------------
% ------------------------------------------------------------------
% ------------------ MENCARI PEAK PADA PARAMETER HT --------------
% --------------------- Dengan Syntac Sendiri ---------------------
peaks = hough_threshold ( H,numpeaks,threshold,NHoodSize);
fig_num=fig_num+1; figure(fig_num),
imshow(H,[],'XData',theta,'YData',...
    rho,'InitialMagnification','fit');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
plot(theta(peaks(:,2)),rho(peaks(:,1)),'s','color','white');
%  ------------------------------------------------------------------
%%
% ----------------- SEGMENTASI GARIS HASIL PEAKS -----------------
% -------------------- Konsep yang dikembangkan --------------------
% Dengan membuat algoritma segmentasi garis akan didapat garis
yang mampu
% untuk disegmentasi terpisah atau dihubungkan pada setiap
pelacakan.
% Selain mensubrtact garis satu persatu pada gambar utama,
dilakukan juga
% pembuatan database pada setiap garis yang terlacak secara
pararel.
% Database berupa koordinat(x,y) titik awal dan akhir garis, nilai
theta
% dan rho pada titik pusat NHoodSize, gradien garis(M) dan
kontanta (C).
% ------------------------------------------------------------------
%%
% ------------------ MEMBUAT SEGMENTASI GARIS ----------------
% -------------------- Dengan Syntac Sendiri -------------------
lines = hough_garis(b_biner,theta,rho,peaks, fillgap, minlength);
fig_num=fig_num+1; figure(fig_num), imshow(image,[]);
axis on, axis normal, hold on;
%  ------------------------------------------------------------------
%%
% ----- Algoritma yang terkombinasi dalam pembuatan struktur data
% ------------------------------------------------------------------
for k   = 1:numel(lines)
    x1  = lines(k).point1(1);
    y1  = lines(k).point1(2);
    x2  = lines(k).point2(1);
    y2  = lines(k).point2(2);
    plot([x1 x2],[y1 y2],'Color','r','LineWidth', 3)
    lines(k).M = (y2-y1) / (x2-x1);
    lines(k).C = y1 - lines(k).M * x1;
```

```
end
hold off
% ------------------------------------------------------------------
%% ------------------- PEMILIHAN JALUR PENGELASAN ----------------
%  --------------------- Konsep yang dikembangkan -----------------
% Pencarian jalur pengelasan dilakukan dengan mempropagasi satu
% persatu panjang jalur pengelasan yang dimungkinkan. Propagasi
% ini akan diketahui seberapa jauh lintasan dapat ditempuh untuk
% setiap jalur yang dimungkinkan. Algoritma ini dibangun dengan
% mempertimbangkan kecepatan proses yang mana pada penelitian ini
% diasumsikan mengambil lintasan terpanjang yang kemudian dicari
% jarak terpendek dari lintasan terakhir. Untuk mempermudah dalam
% proses flagging, dibuatlah database intersection sedangkan
% untuk mendatabase garis, digunakan struktur data yang berisi
% koordinat path dan jarak tempuh per path yang dipropagasi.
% ------------------ Kelemahan yang belum terpecahkan ------------
% Masih sulitnya mengakusisi adanya perpotongan segitiga untuk
tiga garis
% ------------------------------------------------------------------
%%
% -------------------- PEMBUATAN DATA INTERSECTION ----------------
% Pembuatan data intersection dilakukan dengan membentuk matrik 3
dimensi
% sebagai jumlah garis x jumlah garis x nilai koordinat itu
sendiri.
% Nilai koordinat adalah nilai titik perpotongan x,y antara 2
garis.
% ------------------ Kelemahan yang belum terpecahkan ------------
% Belum dapat mendatabase interkoneksi antara 3 garis saling
memotong
% ------------------------------------------------------------------
% ----------------------------- Algoritma ------------------------
% ------------------------ Dengan Syntac Sendiri-------------------
[intsc_xy lines ] = intersection(lines);

[lines, line, segment_garis, list_intsc,  list_point] =
segmentasi(intsc_xy, lines);

segment_garis_idx = indexing(segment_garis,list_point);

list_intsc_idx = indexing(list_intsc, list_point);

%  ------------------------------------------------------------------
%%
% ---------------- PENCARIAN JALUR PENGELASAN TERPANJANG ---------
% ---------------------- Konsep yang dikembangkan ----------------
% Cara yang dilakukan adalah dengan mempropagasi garis per garis
% secara bolak-balik (P1 dan P2) untuk dicari konektifitas yang %
% terdapat didalam garis tersebut. Ketika menemukan konektifitas,
% cari garis yang terkoneksi berikutnya dan dicari konektifitas
% berikutnya sampai ujung path tak terkoneksi. Simpan path
% propagasi untuk mencapai P1 dan P2 terakhir dari P1 dan P2 awal
% sebagai path pertama dan kedua. Hitung jarak pixel untuk
% mencapai path tersebut. Masking pada konektifitas terakhir.
% Simpan path didalam database. Titik awal selalu bermula pada
% ujung garis dan bukan pada intersection.
% ------------------ Kelemahan yang belum terpecahkan ------------
% Belum dapat mempropagasi interkoneksi antara 3 garis saling
```

```
% memotong
% ------------------------------------------------------------------
% ------------- Algoritma untuk pencarian jalur pengelasan -------
% ------------------------ Dengan Syntac Sendiri -------------------
[prop idx_max_prop] = tracking (intsc_xy, lines);
% ------------------------------------------------------------------

path_point = prop(idx_max_prop).path;
path_point_idx = indexing(path_point,list_point);
path = struct;
num_path = 1;
path(num_path).axis = path_point;
mask_segment = zeros(numel(path_point(:,1))-1,2);
for k = 1 : numel(path_point(:,1)) - 1
    mask_segment(k,:) = sort([ path_point_idx(k,1)
path_point_idx(k+1,1)],2);
end
path_segment_idx = mask_segment;
% ------ mencari segment semuanya -----------
temp = line(1).segment;
for i = 2 : numel(lines)
    temp = [ temp ; line(i).segment];
end
all_segment_idx = temp;
% ----- mencari index path yang belum dilalui --------------------
temp = all_segment_idx;
temp2 = all_segment_idx;
for i = 1 : numel(path_segment_idx(:,1))
 a = path_segment_idx(i,1);b = path_segment_idx(i,2);
 idx = find(all_segment_idx(:,1)== a & all_segment_idx(:,2) == b);
 temp(idx,:) = 0;
end
idx = find(temp(:,1) == 0 & temp(:,2) == 0 );
temp2(idx,:) = [];
no_path_segment_idx = temp2;
% ----- no_path_segment_idx = index path yang belum dilalui ------
% ----- no_path_segment_idx - list_intsc_idx ---------------------
stop = 0;
while stop ~= 1
list_next_point_idx = no_path_segment_idx(:);
    for i = 1 : numel(list_intsc_idx)
        idx = list_intsc_idx(i) == no_path_segment_idx(:) ;
        list_next_point_idx(idx) = 0;
    end
    idx = find(list_next_point_idx == 0 );
    list_next_point_idx(idx) = [];
% - next_point_idx = no_path_segment_idx - list_intsc_idx ---
    j = path_segment_idx(end);
    [m,n] = size(image);
    min_length = sqrt(m^2 + n^2);
    for i = 1 : numel(list_next_point_idx(:))
        length = sqrt((list_point(j,1)-
list_point(list_next_point_idx(i),1)).^2 + ...
            (list_point(j,2)-
list_point(list_next_point_idx(i),2)).^2);
        if length < min_length
            min_length = length;
            next_point_idx = list_next_point_idx(i);
        end
```

```
    end
done = 0;
start = next_point_idx;
num = 1;
next_path(num) = start;
    while done ~= 1
        if find(start == no_path_segment_idx)
            [r,c] = find(start == no_path_segment_idx);
            if c == 1
                not_c = 2;
            else
                not_c = 1;
            end
            num = num + 1;
            next_path(num) = no_path_segment_idx(r,not_c);
            start = no_path_segment_idx(r,not_c);
            no_path_segment_idx(r,:) = [];
        else% jika sudah tidak ditemukan segmentasi yang mungkin
            % maka torch di angkat dan dipindah ke path berikutnya
        done = 1;
        end
    end
        if isempty(no_path_segment_idx);
            stop = 1;
        else
        end
        temp = zeros(numel(next_path)-1,2);
% ------ masukan dalam koordinat pada database path ------
    for i = 1 : numel(next_path)-1;
        temp(i:i+1,:) = [ list_point(next_path(i),:) ;
            list_point(next_path(i+1),:)];
    end
    num_path = num_path + 1; path(num_path).axis = temp;
end
for i = 1 : numel(path)
    for j = 1 : numel(path(i).axis(:,1));
        disp (['G01 X' num2str(path(i).axis(j,1)) ' Y'
num2str(path(i).axis(j,2)) ' Z100'])
    end
    disp (['G00 X' num2str(path(i).axis(j,1)) ' Y'
num2str(path(i).axis(j,2)) ' Z200'])
    if i ~= numel(path)
        disp (['G00 X' num2str(path(i+1).axis(1,1)) ' Y'
num2str(path(i+1).axis(1,2)) ' Z200'])
    else
        disp(['STOP']);
    end
end
```

```
function y = indexing (temp, list_point)
y= zeros(size(temp(:,1)));
    for i = 1 : numel(temp(:,1))
        a = temp(i,1);b = temp(i,2);
        idx = find(list_point(:,1) == a & list_point(:,2) == b);
        y(i,:) = idx;
    end
end
```

```
function [lines, line, segment_garis, list_intsc,  list_point] =
segmentasi(intsc_xy, lines)
% -------------------------------------------------------------
% ----- Buat segmentasi untuk semua garis yang terdeteksi ------
for i = 1 :numel(lines)
    P1 = lines(i).point1 ;
    P = P1;
    P2 = lines(i).point2;
    temp_intsc = lines(i).intsc;
    length_min = sqrt((P1(1)- P2(1))^2 + (P1(2) - P2(2))^2);
    done = 0;
    while done ~=1
        for j = 1  : numel(temp_intsc);
            intsc(1,:) = intsc_xy(i,temp_intsc(j),:);
            length = sqrt ((P1(1) - intsc(1))^2 + (P1(2) -
intsc(2))^2);
            if length < length_min
                length_min = length;
                I = intsc;
                idx_temp = j;
            end
        end
            P = [ P ; I];
        temp_intsc(idx_temp) = [];
        if isempty(find(temp_intsc, 1))
            P = [ P ; P2];
            done = 1;
        elseif find(temp_intsc) == 1
            intsc(1,:) = intsc_xy(i,temp_intsc(1),:);
            I = intsc;
            P = [ P ; I];
            P = [ P ; P2];
            done = 1;
        end
        lines(i).segment = P;
    end
end
% ------- garis sudah tersegmentasi pd database -----------
temp = lines(1).segment;
for i = 1 : numel(lines)-1
    temp2 = lines(i+1).segment;
    temp = [ temp ; temp2];
end
segment_garis = temp;
% ----- garis sudah tersegmentasi pd bantuk array ---------
% ----- mencari koordinate intersection -------------------------
num = 0;
list_intsc = zeros((numel(find(intsc_xy(:,:,1) ~= 0)))*0.5, 2);
temp4 = intsc_xy;
for i = 1 : numel(lines)
    for j = 1 : numel(lines(i).intsc)
        if temp4(i,lines(i).intsc(j),1)~=0
            num = num + 1;
            list_intsc(num,:) = [temp4(i,lines(i).intsc(j),1)
                temp4(i,lines(i).intsc(j),2)];
            temp4(lines(i).intsc(j),i,1) = 0 ;
```

```
        end
    end
end
% ----- intsc_coordinate = koordinate intersection ---------------
% ----- list koordinat semua titik  -------------
temp2 = sortrows([temp(:,1) temp(:,2)],[1 2]);
diff2 = diff(temp2,1);
temp3 = temp2(1,:);
num = 0;
for i = 2 : numel(segment_garis(:,1))
    if diff2(i-1,2)~=0
        num = num +1;
         temp3 = [temp3 ; temp2(i,:)];
    else
    end
end
list_point = temp3;
% ----- list_point = koordinat semua titik -----
% ----- buat struktur database line ------------
line = struct;
for i = 1 : numel(lines)
    for j = 1 : numel(lines(i).segment(:,1))
        a = lines(i).segment(j,1); b = lines(i).segment(j,2);
        line(i).point(j) = find(list_point(:,1) == a &
list_point(:,2) == b);
    end
    for k = 1 : numel(lines(i).segment(:,1))-1
        line(i).segment(k,:) = sort([ line(i).point(k)
line(i).point(k+1)],2);
    end
end
```

Data Percobaan

No.Path	11	21
Path (x,y)	134 13 290 333 293 148 403 7	403 7 293 148 290 333 134 13
Jarak	719.8566	719.8566

```
START
G01 X134 Y13  Z100
G01 X290 Y333 Z100
G01 X293 Y148 Z100
G01 X403 Y7   Z100
G00 X403 Y7   Z200
G00 X295 Y5   Z200
G01 X295 Y5   Z100
G01 X293 Y148 Z100
G01 X290 Y151 Z100
G00 X290 Y151 Z200
G00 X291 Y336 Z200
G01 X291 Y336 Z100
G01 X290 Y333 Z100
G01 X287 Y471 Z100
G00 X287 Y471 Z200
STOP
```

Kesimpulan — Dari data percobaan didapat kesimpulan bahwa penggunaan function dalam pemrograman akan mengurangi kebutuhan memori dalam menjalankan komputasi. Terutama memori yang bersifat temporary.

Kekurangan — Kekurangan yang dapat diambil adalah dengan mengintegrasikan beberapa function akan mengalami kesulitan dalam mengetahui kesalahan per step yang akan dilakukan.

Kegiatan Penelitian	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Pembuatan sistem *Image Data Acqusition* yang optimal.
Perumusan Masalah	Didalam pengambilan citra, kita harus melihat keterbatasan alat yang digunakan dengan kebutuhan citra yang dibutuhkan. Dengan memilih parameter yang tepat akan dihasilkan citra yang optimal.
Identifikasi Masalah	Dengan mencoba melihat hasil yang didapat dari penggunaan parameter *Image Data Acqusition* yang berbeda akan kita ketahui bagaimana citra harus terambil. Hal ini tentu juga harus dilihat spesifikasi peralatan optik yang akan digunakan. Karena pengambilan citra adalah diam, maka keterbatasan alat sangat dominan dalam menentukan parameter. Parameter tersebut adalah Color Space yang dihasilkan, dan teknik pengambilan frame sebagai konsekuensi dari waktu pemanasan kamera.
Metodologi Penelitian	Penelitian dilakukan dengan menggunakan YUY 640 x 480 sesuai dengan kemampuan adaptor kamera. Sedangkan input diambil dalam bentuk intensity dan teknik pengambilan data dilakukan dengan metode membuang data gambar pertama.

```
function b = citra_kamera(a1)
% -------------------------- Citra Kamera ----------------------
% Setup kamera digunakan untuk mensetting data akusisi kamera yang
mana
% pada teknik pengambilan gambar memperlukan parameter yang
terbaik untuk
% diambil citranya.
% --------------- dibuat oleh : albertus rianto ------------------
% ---------------- INPUT PARAMETER PADA KAMERA -------------------
vid1=videoinput('winvideo',a1,'YUY2_640x480');
set(vid1,'ReturnedColorSpace', 'grayscale')
set(vid1,'FramesPerTrigger', 1,'LoggingMode','memory');
% ---- Time out jika > 20 detik dan blm mengambil data -----------
set(vid1,'Timeout',20);
% ---- Gambar diambil setiap 50 frame pd kamera ------------------
set(vid1, 'FrameGrabInterval',50);
% ---- Trigger diulangi sekali (15 x 2) --------------------------
set(vid1,'TriggerRepeat',1)
set(vid1,{'TimerFcn', 'Timerperiod'},{@imaqcallback,25});
% ---- Mengganti Nama RGB24_160x120 --> MyObject -----------------
set(vid1,'Name','MyObject');
% ------------------- PENGAMBILAN GAMBAR PADA KAMERA -------------
set(vid1,'SelectedSourceName','input1');
%  Pengambilan set up kamera dilajutkan dengan menyalakan kamera -
vid_src = getselectedsource(vid1);
```

```
start(vid1);
% ---- Pemanasan kamera (data pertama dibuang, data kedua
disimpan)---
b(:,:,1) = getdata(vid1);
b(:,:,1) = getdata(vid1);
% -------- citra dikeluarkan dan kemudian kamera dimatikan -------
stop(vid1);
imshow
```

Kesimpulan	Dapat diambil kesimpulan bahwa dengan teknik pengambilan citra yang baik akan menentukan hasil citra yang optimal. Pixel (adaptor): 640 x 480 (winvideo YUY2) Frame per trigger : 1 Data Logger : Memory Frame grab interval : 50 Teknik penggambilan citra : Data citra pertama didummie dengan data citra kedua dan diambil untuk diolah.
Kekurangan	----

Kegiatan Penelitian	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Pemilihan *Image Enhancement*
Perumusan Masalah	Proses pengambilan citra banyak mengandung noise yang banyak disebabkan karena keterbatasan kamera yang digunakan, teknik pencahayaan yang kurang sempurna, dan warna yang tidak homogen pada material yang diidentifikasi.
Identifikasi Masalah	Pemilihan *Image Enhancement* akan menentukan hasil yang terbaik didalam proses *filtering* noise. Namun demikian didalam pemilihan filter juga tidak jarang akan menghilangkan informasi yang dibutuhkan.
Tujuan komputasi	Menguji hasil pelacakan garis terhadap pemilihan operasi low pass filter didalam *Image Enhancement*. Operasi tersebut antara lain adalah operasi **filter median, filter gaussian 1/16**, dan **filter mean gaussian**

```
function b = filter_median(a)
% ------------------------- filter median ----------------------
% Metode filter median adalah metode untuk menganti nilai tengah
% dari nilai pixel tetangganya. Filter ini sangat effektif untuk
% menghilangkan salt and papper dan juga dapat sebagai pass low
% filter yang sangat effektif tanpa mengaburkan batas perbedaan
% nilai intensitas pd citra seperti yang terjadi pada filter
% rata-rata(mean_gauss)
% ------------------- dibuat oleh : albertus rianto -------------
[m,n,o]=size(a);
a=im2double(a);
iter1=1;
b=a;
for i=1:m-2
    for j=1:n-2
        for k=1:o
        a0=a(i+1,j+1,k);
        a_sort=[a0 a(i+1,j+2,k) a(i,j+2,k)...
                a(i,j+1,k) a(i,j,k) a(i+1,j,k)...
                a(i+2,j,k) a(i+2,j+1,k) a(i+2,j+2,k)];
        b(i+1,j+1,k)=median(a_sort);
        iter1=iter1+1;
            if (iter1 > 5.0e6)
          disp('Gambar yang di unduh terlalu besar utk di proses')
            return
            end
        end
    end
end
```

```
function b = gaussian_16(a)
mask= 1/16*  [  1 2 1
                2 4 2
                1 2 1];
[m,n,o]=size(a); a=im2double(a); b=a; iter=1;
```

```
if o ~=1
    disp ('gambar belum intensity utk diproses mean gaussian')
    return
end
for i=1:m-2
    for j=1:n-2
    b(i+1,j+1)= a(i,j)*mask(1,1)  +a(i+1,j)  *mask(2,1)     +...
                a(i+2,j)*mask(3,1)+a(i,j+1)  *mask(1,2)     +...
                a(i,j+2)*mask(1,3)+a(i+1,j+2)*mask(2,3)     +...
                a(i+2,j+1)*mask(3,2)+a(i+2,j+2)*mask(3,3)   +...
                a(i+1,j+1)*mask(2,2);
        iter=iter+1;
        if (iter > 5.0e5)
            disp('gambar terlalu besar utk di proses')
            return
        end
    end
end
b = im2uint8(b);
```

```
function b = mean_gauss(a)
mask= 1/9*  [   1 1 1
                1 1 1
                1 1 1];
[m,n,o]=size(a); a=im2double(a); b=a; iter=1;
if o ~=1
    disp ('gambar belum intensity utk diproses mean gaussian')
    return
end
for i=1:m-2
    for j=1:n-2
     b(i+1,j+1)= a(i,j)  *mask(1,1) +a(i+1,j)  *mask(2,1)     +...
                 a(i+2,j)*mask(3,1) +a(i,j+1)  *mask(1,2)     +...
                 a(i,j+2)*mask(1,3) +a(i+1,j+2)*mask(2,3)     +...
                 a(i+2,j+1)*mask(3,2)  +a(i+2,j+2)*mask(3,3)  +...
                 a(i+1,j+1)*mask(2,2);
        iter=iter+1;
        if (iter > 5.0e5)
            disp('gambar terlalu besar utk di proses mean')
            return
        end
    end
end
b = im2uint8(b);
```

Kesimpulan	Dapat diambil kesimpulan bahwa *Image Enhancement* yang dilakukan oleh *Median Filter* sebagai low pass filter tidak merusak informasi yang ada.
Kekurangan	----

Kegiatan Penelitian	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Pemilihan *Edge Detection*
Perumusan Masalah	Untuk mengetahui adanya garis batas pada objek material adalah dengan dilakukan proses *Edge Detection* yang tepat.
Identifikasi Masalah	Operasi *Edge Detection* sangat banyak variasinya, dan teknik pemilihannya pun akan sulit jika tidak kita bandingkan terhadap hasil thresholding yang akan kita input kemudian.
Tujuan komputasi	Mencoba melakukan pemilihan teknik *edge detection* dengan membandingkan hasil thresholding pada metode *otsu*. Dalam hal ini metode *otsu* adalah metode yang didefault sebagai inisialisasi data pada proses binerisasi. Pengujian dilakukan pada **Sobel**, **Prewitt**, **Robert**, **Krisch**, **Pelacak Arah**.

```
function b =pelacak_sobel(a)
% --------------------------- pelacak sobel ----------------------
% Metode pelacak sobel adalah metode edge detection dengan
memberikan % bobot
% lebih pada titik pusat karnel.
%
%           Sx =  -1  0   1          Sy =   1   2   1
%                 -2  0   2                 0   0   0
%                 -1  0   1                -1  -2  -1
%
% kemudian disatukan dengan  S = abs( sqrt(Sx^2 + Sy^2))
%
% ------------------- dibuat oleh : albertus rianto -------------

[m,n,o]=size(a);b = zeros(m,n);
if o ~=1
    disp ('Gambar belum intensity')
    return
end
a=im2double(a); iter=1;
sx=[-1       0       1
    -2       0       2
    -1       0       1];
sy=[ 1       2       1
     0       0       0
    -1      -2      -1];
for i=1:m-2
    for j=1:n-2
b1x =sx(1,1)*a(i,j)  +sx(1,2)*a(i,j+1)  + sx(1,3)*a(i,j+2)  +...
     sx(2,1)*a(i+1,j)+sx(2,2)*a(i+1,j+1)+ sx(2,3)*a(i+1,j+1)+...
     sx(3,1)*a(i+2,j)+sx(3,2)*a(i+2,j+1)+ sx(3,3)*a(i+2,j+2);
b1y =sy(1,1)*a(i,j)  + sy(1,2)*a(i,j+1)   + sy(1,3)*a(i,j+2)  +...
     sy(2,1)*a(i+1,j)+ sy(2,2)*a(i+1,j+1) + sy(2,3)*a(i+1,j+1)+...
     sy(3,1)*a(i+2,j)+ sy(3,2)*a(i+2,j+1) + sy(3,3)*a(i+2,j+2);
b(i+1,j+1)  =   abs(((b1x^2) + (b1y^2))^0.5);
```

```
        if (iter > 5.0e5)
            disp('Gambar yang di unduh terlalu besar utk di
proses’)
            return
        end
    end
end
b = im2uint8(b);
```

```
function b =pelacak_prewitt(a)
% ------------------------- pelacak prewitt --------------------
%Metode pelacak prewitt adalah metode edge detection dengan
%memberikan bobot sama pada setiap pelacakan.
%
%           Px =  -1   0   1        Py =   1   1   1
%                 -1   0   1               0   0   0
%                  -1   0   1             -1  -1  -1
%
%kemudian disatukan dengan  P = abs( sqrt(Px^2 + Py^2))
%
% ------------------- dibuat oleh : albertus rianto -------------
[m,n,o]=size(a);b = zeros(m,n);
if o ~=1
    disp ('Gambar belum intensity utk diproses mean gaussian')
    return
end
a=im2double(a); iter=1;
sx=[-1      -1      -1
     0       0       0
     1       1       1];
sy=[ -1      0       1
     -1      0       1
     -1      0       1];
for i=1:m-2
    for j=1:n-2
b1x = sx(1,1)*a(i,j)+sx(1,2)*a(i,j+1)+sx(1,3)*a(i,j+2)    +...
      sx(2,1)*a(i+1,j)+sx(2,2)*a(i+1,j+1)+sx(2,3)*a(i+1,j+1)  +...
      sx(3,1)*a(i+2,j)+ sx(3,2)*a(i+2,j+1)+ sx(3,3)*a(i+2,j+2)  ;
b1y = sy(1,1)*a(i,j)+sy(1,2)*a(i,j+1) + sy(1,3)*a(i,j+2)    +...
      sy(2,1)*a(i+1,j)+sy(2,2)*a(i+1,j+1)+sy(2,3)*a(i+1,j+1)  +...
      sy(3,1)*a(i+2,j)+ sy(3,2)*a(i+2,j+1) + sy(3,3)*a(i+2,j+2)  ;
b(i+1,j+1)  =   abs(((b1x^2) + (b1y^2))^0.5);
        if (iter > 5.0e5)
            disp('Gambar yang di unduh terlalu besar utk di proses
pelacak sobel')
            return
        end
    end
end
b = im2uint8(b);
```

Kesimpulan	Dapat diambil kesimpulan bahwa Edge Detection yang dilakukan oleh *Sobel Magnitude* sangat tajam.
Kekurangan	----

Kegiatan Penelitian	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Pemilihan Teknik Binerisasi
Perumusan Masalah	Teknik binerisasi sangat menentukan bagaimana pixel yang dihasilkan citra biner sesuai dengan yang diharapkan.
Identifikasi Masalah	Teknik binerisasi ini dilakukan dengan mencari nilai minimal antar class variance
Tujuan komputasi	Mencoba seperti apa algoritma untuk membangun metode otsu

```
function b = otsu(a)
% ---------------------------- Metode Otsu ----------------------
% Metode untuk menentukan nilai binerisasi antara dua atau lebih
kelompok
% yang muncul secara alami hal ini akan dapat memisahkan antara
objek
% dengan background secara maksimal.
% Metode dilakukan adalah dengan memaksimalkan nilai perubahan :
%
% perubahan = (((t_mean* momen_komulatif) - komulatif_i)^2) / ...
%             ( momen_komulatif * (1 - momen_komulatif));
% yang mana nilai perubahan akan teriterasi dari 1s/d256 dengan :
%
% ------------------- dibuat oleh : albertus rianto --------------
t_mean = 0; b = a;
perubahan_max = 0;
komulatif_i = 0 ; momen_komulatif = 0;
[m,n] = size(a);
histogram = imhist(a);
for k = 1 : 256
    t_mean = t_mean + ( (k*histogram(k))/(m*n));
end
for k = 1 : 256
momen_komulatif = momen_komulatif + ((histogram(k)) /(m*n));
komulatif_i  = komulatif_i + ((k* histogram(k)) / (m*n));
perubahan = (((t_mean* momen_komulatif) - komulatif_i)^2) / ...
            ( momen_komulatif * (1 - momen_komulatif));
    if perubahan > perubahan_max
        perubahan_max = perubahan;
        T = k;
    end
end
for i=1:m
    for j=1:n
        if a(i,j) <=T;
           b(i,j) = 0;
        elseif a(i,j)> T
           b(i,j) =255;
        end
    end
end
```

Kegiatan Penelitian	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Pembuatan Parameter Hough Transform
Perumusan Masalah	Didalam mendeteksi garis, kita dapat melihat hasil dari parameter HT. Parameter HT dibuat dengan persamaan koordinat polar dan dimasukan kedalam fungsi koordinat HT.
Identifikasi Masalah	Membentuk ukuran dari parameter HT dan algoritma didalam mengaplikasikan persamaan polar.
Tujuan komputasi	Membuat algoritma pembuatan HT parameter.

```
function [ h,theta,rho] = hough_transform(bw,RhoRes,ThetaRes)
% ------------------------ Hough Transform ----------------------
% Hough Transform adalah tranformasi pengubah parameter citra
% menjadi parameter hough(rho,theta).
% Input     = citra biner, RhoResolusi, ThetaResolusi
% Output    = matrik hough(theta,rho)
% ------------------- dibuat oleh : albertus rianto --------------
% ----------------------------------------------------------------
% ----------------------- Membuat Nilai Theta --------------------
% ---- Nilai Theta adalah sudut -90 s/d 90 dengan Slope = ThetaRes
theta = linspace(-90,0,ceil(90/ThetaRes) + 1);
theta = [theta -fliplr(theta(2:end -1))];
% ----------------------------------------------------------------
% ----------------------- Membuat Nilai Rho ----------------------
% ------- Nilai Rho adalah -diag s/d diag dengan Slope = RhoRes --
[m,n] = size(bw);
diag = sqrt((m-1)^2 + (n-1)^2);
q = ceil(diag/RhoRes);
nrho = 2*q + 1;
rho = linspace(-q*RhoRes, q*RhoRes, nrho);
% ----------------------------------------------------------------
% -------------------- Membuat Parameter Hough -------------------
% ---- Temukan koordinat(x,y) pada setiap pixel yang bernilai 1 --
% -- Iterasi satu persatu pixel biner(i) pada persamaan koordinat
polar -h = zeros(numel(rho),numel(theta));
[y,x] = find (bw);
for i = 1 : numel(x)
    for idx_theta = 1 : numel(theta)
        rho_j = ceil(x(i)*cos(deg2rad(theta(idx_theta))) + ...
            y(i)*sin(deg2rad(theta(idx_theta))));
        idx_rho = find(rho == rho_j);
        h(idx_rho,idx_theta) = h(idx_rho,idx_theta) + 1;
    end
end
```

Kesimpulan	Dapat diambil kesimpulan bahwa komputasi yang dilakukan melibatkan banyak iterasi elementer
Kekurangan	Masih kurang cepat melakukan iterasi dibanding dengan syntac matlab

Kegiatan Penelitian	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Pelacakan Peaks
Perumusan Masalah	Pencarian peaks dilakukan untuk menentukan posisi garis sebagai fungsi rho dan theta.
Identifikasi Masalah	Didalam mendeteksi peak banyak membutuhkan parameter yang harus dicari nilai yang sesuai dengan kondisi yang dibutuhkan.
Tujuan komputasi	Membuat algoritma dan menentukan parameter peaks lines

```
function peaks = hough_threshold (h,numpeaks,threshold,nhood)
% -------------------------- Hough Threshold ---------------------
% Hough threshold adalah cara untuk menthreshold hough parameter
% Input = hough parameter
% Output= array peaks
% ------------------- dibuat oleh : albertus rianto --------------
% ----------------------------------------------------------------
%%
% --------- Inisialisasi jika numpeaks, threshold, nhood kosong --
if isempty(numpeaks)
   numpeaks = 1;
end
if isempty(threshold)
   threshold = 0.5 * max(h(:));
end
if isempty(nhood)
  nhood = size(h)/50;
  nhood = max(2*ceil(nhood/2) + 1, 1);
end
% ---------------------- Sumber :  R.C. Gonzales ----------------
% ----------------------------------------------------------------
%%
% -------------------------- Membangun Looping -------------------
done = false;
hnew = h;
% ---- Menentukan titik pusat karnel ----------------------------
nhood_center = (nhood-1)/2;
peaks = [];
% ----------------------------------------------------------------
% ------------ Membangun looping setiap garis yang terdeteksi ----
%ind2sub digunakan utk mencari koordinat nilai maksimal index
%berbasis matrik size(hnew) dengan p = row, dan q = kolom.
%Jika terdapat nilai max lebih dari satu, kita lakukan satu
%saja,hal ini dimungkinkan karena pada looping ke dua,nilai max
%pertama sudah dieliminasi,sehingga hanya menyisakan satu lokasi
%saja (jika ada 2)
%----------------------------------------------------------------
while ~done
  [dummy max_idx] = max(hnew(:));
  [p, q] = ind2sub(size(hnew), max_idx);
  p = p(1); q = q(1);
  if hnew(p, q) >= threshold
%------------ Membuat data peak dalam bentuk array ---------------
%Peaks akan keluar peringatan krn perubahan dimensi setiap iterasi
%hal ini kita abaikan, mengingat sudah kita lakukan batasan dengan
```

```
%numpeaks. Sebagai catatan, input data peaks dengan metode ini
%memang tidak disarankan karena akan memperlambat proses looping.
%-------------------------------------------------------------------
    peaks = [peaks; [p q]];
%   -------------------- Eliminasi Karnel ---------------------
%   Guna dari eliminasi karnel ini adalah untuk mengatasi
%   perpotongan kurva yang tidak tepat.  Hal ini diakibatkan krn
%   faktor resolusi rho dan theta.
%   -----------------------------------------------------------
    p1 = p - nhood_center(1); p2 = p + nhood_center(1);
    q1 = q - nhood_center(2); q2 = q + nhood_center(2);
    [qq, pp] = meshgrid(q1:q2, max(p1,1):min(p2,size(h,1)));
    pp = pp(:); qq = qq(:);
%    -------------------- Sifat Asymetris theta ------------------
%Karena bentuk yang berulang pada theta propagasi ( -90 s/d 90)
%maka ketika hough parameter di batas kanan, maka eliminasi karnel
%akan dilakukan dengan menambah elementer karnel paling kiri dari
%hough parameter ( sifat asymetris theta ).
%-------------------------------------------------------------------
    theta_too_low = find(qq < 1);
    qq(theta_too_low) = size(h, 2) + qq(theta_too_low);
    pp(theta_too_low) = size(h, 1) - pp(theta_too_low) + 1;
    theta_too_high = find(qq > size(h, 2));
    qq(theta_too_high) = qq(theta_too_high) - size(h, 2);
    pp(theta_too_high) = size(h, 1) - pp(theta_too_high) + 1;
    %------------------------ ind2sub ----------------------------
%Kebalikan dari syntac ind2sub, sub2ind adalah mencari nilai dari
%row dan kolom pada sebuah matrik size(hnew).
%Dengan menganti nilai row dan kolom pada titik yang sudah didata
%dengan nilai 0, maka pd saat pp(row) dan qq(colomb)pd matrik hnew
%dipropagasi ulang, maka tidak akan menemukan adanya hmax (pp,qq).
%-------------------------------------------------------------------
    hnew(sub2ind(size(hnew), pp, qq)) = 0;
    done = size(peaks,1) == numpeaks;
  else
    done = true;
  end
end
% peaks
%%
%----------------------------- Keterangan ------------------------
%Hasil yang dilakukan untuk mencoba algoritma adalah dengan
%langsung membandingkan hasil yang didapat dengan sebuah syntac
%matlab. Hasil yang didapat sama hasil peaks nya. Dan waktu
%iterasi relative sama.Elapsed time is 0.031647 s dibanding
% Elapsed time is 0.050701 s.
%-------------------------------------------------------------------
```

Kesimpulan	Dapat diambil kesimpulan bahwa
Kekurangan	Masih kurang cepat melakukan iterasi dibanding dengan syntac matlab

Kegiatan Penelitian	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Segmentasi Garis
Perumusan Masalah	Segmentasi garis dilakukan untuk menentukan seperti apa garis yang sebenarnya terdeteksi yakni dengan membandingkan dengan citra biner
Identifikasi Masalah	Didalam mendeteksi peak banyak membutuhkan parameter yang harus dicari nilai yang sesuai dengan kondisi yang dibutuhkan. Parameter ini adalah *fillgap* dan *minlength*.
Tujuan komputasi	Membuat algoritma dan menentukan parameter untuk segmentasi garis

```
function lines = hough_garis
(b_biner,theta,rho,peaks,fillgap,minlength)
% ---------------------------- Hough Garis -----------------------
% Hough garis adalah cara untuk membuat garis hasil hough
parameter
% kedalam citra biner dalam bentuk garis yang tersegmentasi.
% Input  =  citra biner
%           theta
%           rho
%           peaks
%           fillgap
%           minlength
% Output =  struktur data lines ( P_start , P_end, theta, rho )
%
% ------------------------ Segmentasi Garis ----------------------
% Segmentasi garis dilakukan dengan mensubstract hasil
thresholding nilai hough (theta dan rho) ke dalam garis pada
citra biner. Dgn membandingkan hasil thresholding akan dapat
diketahui bagaimana garis tersebut akan tersegmentasi. Seperti
diketahui bahwa kegiatan thresholding yang dilakukan untuk
menentukan peaks pada parameter hough hanya akan mendapatkan nilai
theta dan rho sehingga garis deteksi yang didapat hanya merupakan
garis lurus panjang tanpa tersegmentasi sesuai dengan garis
sebenarnya. Langkah pertamanya adalah dengan mensorting jarak,
kemudian mencari jarak point to point, kemudian menemukan lebar
gap, diikuti dgn eliminasi panjang segment minimal, dan terakhir
pembuatan struktur data garis.
% ---------------------------------------------------------------
% ------------------- dibuat oleh : albertus rianto -------------
% ---------------------------------------------------------------
% -------------------- Inisialisasi data ------------------------
%  panjang_min_sq = panjang yang di input dipangkat 2 (pytagoras)
%  fillgap_sq = fillgap yang diinput dipangkat 2 (pytagoras)
%  num = untuk iterasi jumlah peaks.
%  done = false; kebutuhan iterasi num dengan perintah while
%  lines = struct ; digunakan membuat structure data
%  numlines = untuk penomoran database garis
%  ---------------------------------------------------------------
panjang_min_sq = minlength^2;
```

```
fillgap_sq = fillgap^2;
numlines = 0;
lines = struct;
done = false;
num = 0;
% ------------------------------------------------------------------
% ------------------  Mencari posisi x dan y -------------------
% Membuat (rho bin index )atau Index dari nilai rho hasil
subtitusi
% nilai biner dgn bantuan slope. Dari nilai index tersebut
kemudian
% dicari nilai index yang nilainya sama dgn nilai rho hasil peaks.
% Dari hasil pencarian tersebut kemudian diterjemahkan pada nilai
% kolom dan baris pada citra untuk diketahui posisinya.
% ------------------------- Keterangan --------------------------
% Slope adalah offset dr jmlh parameter hough(rho).Nilai slope
akan
% selalu bernilai konstan terhadap jumlahnya.
% theta_peaks  = theta hasil proses thresholding (peaks(num,2)).
% rho_subtitusi= rho hasil substitusi pers hough pd nilai
biner(x,y)
% dgn hasil theta_peaks. Nilai ini tidak sama dengan
% nilai rho hasil peaks (peaks(num,1))
% ------------------------------------------------------------------
[y, x] = find(b_biner);
x = x - 1;      y = y - 1;
nrho = length(rho);
slope = (nrho - 1)/(rho(end) - rho(1));
while ~done
num = num + 1;
theta_peaks = theta(peaks(num,2)) * pi / 180;
rho_subtitusi = x*cos(theta_peaks) + y*sin(theta_peaks);
rho_bin_index = round(slope*(rho_subtitusi - rho(1)) + 1);
idx = find(rho_bin_index == peaks(num,1));
baris = y(idx) + 1; kolom = x(idx) + 1;
% ------------------------------------------------------------------
% -------------------------- Sorting jarak ----------------------
% Jika (jarak baris > jarak kolom) maka diurutkan terlebih dahulu
% array barisnya, dan jika (jarak baris < jarak kolom) maka
diurutkan
% terlebih dahulu array kolomnya.
% Tujuan dari kegiatan sorting ini adalah untuk menghitung selisih
% jarak antar koordinat_xy pd saat proses segmentasi. Jika bentuk
% garis cenderung mendekati vertikal, maka propagasi jarak yang
% dihitung adalah jarak baris atau y(idx)
% --------------------------- Keterangan -----------------------
% sortrow  = digunakan untuk membuat orientasi pengurutan array
% [b_c_new]= baris dan kolom yang array nya sudah diurutkan.
% Sifat dari [b_c_new] hanya sebagai temporary array
% utk menginput nilai kolom dan baris ke koordinat_xy
% ------------------------------------------------------------------
jarak_baris = max(baris) - min(baris);
jarak_kolom = max(kolom) - min(kolom);
if jarak_baris > jarak_kolom
    sorting_order = [1 2];
else
    sorting_order = [2 1];
end
[b_c_new] = sortrows([baris kolom], sorting_order);
```

```
baris_new = b_c_new(:,1); kolom_new = b_c_new(:,2);
r = baris_new; c = kolom_new;
% ------------------------------------------------------------------
% --------------- Algoritma Segmentasi Garis ---------------------
% Algoritma yang akan dikembangkan adalah dengan menentukan jarak
gap minimal yang mana jika terdapat gap garis terputus masih akan
teridentifikasi menjadi satu segment atau lebih dari satu segment.
Karena sering dalam suatu kasus, hasil segmentasi yang dilakukan
mengakibatkan segmentasi kecil-kecil, maka diperlukan thresholding
untuk mengeliminasi panjang garis yang lebih kecil.
Untuk mempermudah didalam menghitung panjang gap dan thresholding
maka dilakukan mengoperasikan nilai pangkat dua sebagai konsep
pytagoras sederhana (jarak^2 =x^2+ y^2 )
% ---------------------------- Keterangan -----------------------
% koordinat_xy = koordinat y(idx) atau kolom dan x(idx) atau baris
%                yang sudah diurutkan menurut bentuk garis.
% koordinat_xy adalah koordinat dengan system origin pada
koordinat(1,1).
% jarak_xy_sq  = selisih antar array dipangkat dua
% dist_x_sq    = jumlah jarak_xy_sq (digunakan utk pytagoras)
% ------------------------------------------------------------------
% ------------- Mencari jarak point to point pada koordinat y ----
koordinat_xy = [c r];
jarak_xy_sq = diff(koordinat_xy,1,1).^2;
dist_x_sq = sum(jarak_xy_sq,2);
% ------------------------------------------------------------------
% ------- Menemukan lebar gap yang lebih besar dari threshold ----
  fillgap_idx = find(dist_x_sq > fillgap_sq);
  idx = [0; fillgap_idx; size(koordinat_xy,1)];
% ------------------------------------------------------------------
  for p = 1:length(idx) - 1
    titik_1 = koordinat_xy(idx(p) + 1,:);
    titik_2 = koordinat_xy(idx(p + 1),:);
% ----- Eliminasi panjang segmentasi dibawah nilai thresholding --
    panjang_garis_sq = sum((titik_2-titik_1).^2);
    if panjang_garis_sq >= panjang_min_sq
% ------------------------------------------------------------------
    numlines = numlines + 1;
% ------------------- Pembuatan Database -------------------------
      lines(numlines).point1 = titik_1;
      lines(numlines).point2 = titik_2;
      lines(numlines).theta = theta(peaks(num,2));
      lines(numlines).rho = rho(peaks(num,1));
      lines(numlines).M =(titik_2(:,2)- titik_1(:,2)) / ...
                         (titik_2(:,1)- titik_1(:,1));
      lines(numlines).C = titik_1(:,2)- lines(numlines).M *
titik_1(:,1);
% ------------------------------------------------------------------
    end
  end
done = num == numel(peaks(:,1));
end
% ------------------------------------------------------------------
% ------------------------- Tips & Trick -------------------------
% Untuk mempermudah membuat algoritma, kita menggunakan excel
untuk
% memahami operasi array yang terjadi.
% Untuk lebih memahami logika proses, sebaiknya gunakan satu data
% peaks.
```

```
% ---------------------------- Percobaan ------------------------
% Kita akan membandingkan dengan kecepatan dengan menggunkaan
syntac matlab.
% ---------------------------- Kesimpulan ------------------------
%  Hasil datanya adalah sama. Kecepatan Iterasi juga relative
sama.
%  0.010267 s (function)  :  0.011829 seconds (syntac)
% ----------------------------------------------------------------
```

Kesimpulan	Dapat diambil kesimpulan bahwa segmentasi yang dilakukan cukup effektif untuk dilakukan pada hasil yang sebenarnya.
Kekurangan	---

Kegiatan Penelitian	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Algoritma konektifitas garis
Perumusan Masalah	Keterhubungan antar garis dapat ditentukan dengan mencari perpotongan garis tersebut
Identifikasi Masalah	Karena setiap garis tidak selalu berpotongan, maka perlu dilakukan identifikasi segmentasi garis.
Tujuan komputasi	Membuat algoritma konektifitas

```
function [intsc_xy lines] = intersection(lines)
% ------------------------- intersection ----------------------
% Intersection digunakan untuk menentukan keterhubungan antar
garis.
% Input = structure data garis
% Output= structure data dengan menambah parameter keterhubungan
%         matrik koordinat perpotongan (intsc_xy)
% ------------------- dibuat oleh : albertus rianto -------------
intsc_xy = zeros(numel(lines),numel(lines));
for a = 1 : numel(lines);
    num = 0;
    for b = 1 : numel(lines);
        if a ~= b
        % ----- Mencari koordinat perpotongan pada dua garis -----
        % Koordinat dicari dengan persamaan matrik dan ketika
        % menemukan garis vertikal lurus (M dan C = inf) maka
        % perpotongan dicari dengan persamaan garis biasa.
        % --------------------------------------------------------
            if lines(b).M == inf
               y = (lines(a).M * lines(b).rho) + lines(a).C ;
               intsc = [ lines(b).rho  y];
            elseif lines(a).M == inf
               y = (lines(b).M* lines(a).rho) + lines(b).C ;
               intsc = [ lines(a).rho  y];
            else
               intsc =([ -(lines(a).M) 1; -(lines(b).M) 1]^-1 *...
                   [lines(a).C ; lines(b).C])';
            end
        %  ---- Menentukan apakah perpotongan berada didalam garis
        %  Untuk menentukan perpotongan didalam garis atau tidak,
        %  dilakukan penentuan titik awal dan akhir tiap garis.
        %  Jika perpotongan tidak berada di daerah tersebut, maka
        %  tidak perlu dimasukan didalam database.
if isfinite(intsc)~=0
      if lines(a).point1(:,1) < lines(a).point2(:,1);
   xmax_a = lines(a).point2(:,1); xmin_a = lines(a).point1(:,1);
      else
   xmax_a = lines(a).point1(:,1); xmin_a = lines(a).point2(:,1);
      end
      if lines(a).point1(:,2) < lines(a).point2(:,2);
   ymax_a = lines(a).point2(:,2); ymin_a = lines(a).point1(:,2);
      else
   ymax_a = lines(a).point1(:,2); ymin_a = lines(a).point2(:,2);
      end
      if lines(b).point1(:,1) < lines(b).point2(:,1);
```

```
    xmax_b = lines(b).point2(:,1); xmin_b = lines(b).point1(:,1);
        else
    xmax_b = lines(b).point1(:,1); xmin_b = lines(b).point2(:,1);
        end
        if lines(b).point1(:,2) < lines(b).point2(:,2);
    ymax_b = lines(b).point2(:,2); ymin_b = lines(b).point1(:,2);
        else
    ymax_b = lines(a).point1(:,2); ymin_b = lines(a).point2(:,2);
        end
        if  intsc(:,1) >= xmin_a && intsc(:,1) <= xmax_a && ...
            intsc(:,2) >= ymin_a && intsc(:,2) <= ymax_a && ...
            intsc(:,1) >= xmin_b && intsc(:,1) <= xmax_b && ...
            intsc(:,2) >= ymin_b && intsc(:,2) <= ymax_b
            num=num+1; lines(a).intsc(num) = b;
            intsc_xy(a,b,1) = intsc(1); intsc_xy(a,b,2) = intsc(2);
        end
end
end
end
```

Kesimpulan	Dapat diambil kesimpulan bahwa konektifitas yang dilakukan sangat sesuai dengan koordinat sesungguhnya
Kekurangan	Belum mampu mendatabase kan perpotongan segitiga

Kegiatan Penelitian	Welding Path Generation and Identification Using Machine Vision
Sub Kegiatan	Algoritma path propagation
Perumusan Masalah	Untuk menemukan panjang lintasan garis maksimal, perlu dilakukan path propagation untuk setiap kemungkinan.
Identifikasi Masalah	Teknik flagging dan masking untuk path propagation
Tujuan komputasi	Membuat algoritma path propagation

```
function [prop idx_max_prop] = tracking (intsc_xy, lines)
% -------------------- Pencarian Jalur Terpanjang ----------------
% Digunakan untuk menentukan path yang mungkin dilakukan dalam
welding
% Input  = intsc_xy, lines
% Output = propagasi path (prop), index porpagasi yg mempunyai
panjang
%          maksimal
% -------------------- dibuat oleh : albertus rianto -------------
% ----------------------------------------------------------------
% --------- Inisialisasi data path -----------
prop = struct;
idx_prop = 0;
jarak_terpanjang = 0;
% ------------------------------------------
% -------- Buat binerisasi intsc_xy ----------
% Tujuan dari binerisasi ini adalah simplyfikasi
temp_intsc = zeros(numel(lines),numel(lines));
for i=1:numel(lines)
    for j=1:numel(lines)
        if intsc_xy(i,j) > 0;
            temp_intsc(i,j) = 1;
        else
            temp_intsc(i,j) = 0;
        end
    end
end
% ---------------- Default --------------------
% Digunakan untuk mengembalikan temp_intsc saat
% jalur dipropagasi terbalik.
% ---------------------------------------------
default = temp_intsc;
% ---------------------------------------------
for i = 1: numel (lines);
    % ---------------- Keterangan -----------------
    % i_start digunakan untuk mengembalikan i awal
    % mundur = 0 berarti dilakukan propagasi maju
    % stop = 0, inisialiasi untuk masuk ' while '
    % ---------------------------------------------
    i_start = i;
    mundur = 0;
    stop = 0;
% ----------------------- Keterangan While ----------------------
% Propagasi semua track yang mungkin setiap garis(i) baik
maju/mundur
```

```
% Jika stop = 1, maka iterasi akan berhenti untuk garis(i) dan
% kemudian akan dilakukan propagasi pada line berikutnya (i+1)
% -------------------------------------------------------------------
    while stop ~=1
       num = 0; i = i_start;
% ---------------------- Keterangan IF --------------------------
% Jika propagasi maju, line akan diawali dengan point1, dan
% diakhiri dengan point2, jika mundur, dilakukan sebaliknya.
% -------------------------------------------------------------------
       if mundur ~= 1
           p_start = lines(i).point1;
           p_end = lines(i).point2;
       else
           p_start = lines(i).point2;
           p_end = lines(i).point1;
       end
       num = num + 1;
       path(num,:) = p_start;
 % ---------------------- Keterangan IF --------------------------
% Karena selama iterasi akan dilakukan masking satu kali pd setiap
 % propagasi, maka hasil akhir hanya akan menyisakan garis lurus
 % tanpa adanya intersection didalamnya.
 % Dan saat kondisi tersebut kita lakukan pergantian arah mundur
% atau jika sudah ke arah mundur, kita lakukan pergantian ke garis
 % i berikutnya pada perintah for i = 1 : numel(lines)
 % -------------------------------------------------------------------
       if  numel(find(temp_intsc(i,:)~=0)) == 0
           num = num + 1;
           path(num, :) = p_end;
           % ------ Hitung jarak path ----------------
           selisih = diff(path,1,1).^2;
           jarak_step  = sqrt(sum(selisih,2));
           jarak_total = sum(jarak_step,1);
           % -----------------------------------------
           % ---- Structure data path propagation ----
           idx_prop = idx_prop + 1;
           prop(idx_prop).path = path;
           prop(idx_prop).jarak = jarak_total;
           % -----------------------------------------
           % ------------ Path Terpanjang ------------
           if jarak_total > jarak_terpanjang
               jarak_terpanjang = jarak_total;
               idx_max_prop = idx_prop;
           end
           % -------------------------------------------------------
           % --------------------- Keterangan IF ----------------
         % Jika maju, ubah ke mundur dan jika mundur, ubah kemaju
           % untuk langsung hentikan iterasi agar dilakukan
           % pergantian propagasi pada garis berikutnya.
         % -------------------------------------------------------
           if mundur ~=1;
               mundur = 1;
               temp_intsc = default;
           else
               stop = 1;
               temp_intsc = default;
           end
       % --------------- Keterangan Else -------------
       % Lakukan iterasi selama menemukan intersection,
```

```
% baik intersection pertama, kedua, dst.
% -----------------------------------------
 else
     done = 0;
% ------------------ Ket : intsc_line(1) ------------------
% Lakukan pada intersection 1 saja karena setiap propagasi,
% intersection akan selalu dimasking sehingga jumlah
% intersection akan bernilai surut.
% ---------------------------------------------------------
% ---------------------- Ket : While ----------------------
% Digunakan untuk propagasi intersection bercabang
% ---------------------------------------------------------
     temp2_intsc = temp_intsc;
     while done ~= 1
         if numel(find(temp2_intsc(i,:,1)~= 0)) ~= 0
             intsc_line = find(temp2_intsc(i,:,1)~= 0);
             x_intsc = ceil(intsc_xy(intsc_line(1), i, 1));
             y_intsc = ceil(intsc_xy(intsc_line(1), i, 2));
             num = num + 1;
             path(num,:) = [x_intsc  y_intsc];
             temp2_intsc = temp_intsc;
             i_lama = i;
             i = intsc_line(1);
             % -------------- temp2_intsc ----------------
             % Digunakan untuk masking intersection(i)
             % untuk digunakan saat propagasi berjalan
             % -------------------------------------------
             temp2_intsc(i,i_lama) = 0 ;
             temp2_intsc(i_lama,i) = 0 ;
             % -------------------------------------------
         else
             % --------------- temp_intsc ----------------
             % Berbeda dgn temp2_intsc, masking yang akan
             % dilakukan lebih bersifat permanen. Masking
             % ini digunakan untuk menghilangkan intsction
             % saat propagasi diawali dari depan. Seolah-
             % olah menggurangi intsction di line terakhir
             % -------------------------------------------
             temp_intsc(i_lama,i) = 0;
             temp_intsc(i,i_lama) = 0;
             % -------------------------------------------
             num = num + 1;
             path(num,:) = lines(i).point1;
             % ----------- Hitung jarak path ------------
             selisih = diff(path,1,1).^2;
             jarak_step  = sqrt(sum(selisih,2));
             jarak_total = sum(jarak_step,1);
             % -------------------------------------------
             % ---- Structure data path propagation ------
             idx_prop = idx_prop + 1;
             prop(idx_prop).path = path;
             prop(idx_prop).jarak = jarak_total;
             % -------------------------------------------
             % ------------ Path Terpanjang --------------
             if jarak_total > jarak_terpanjang
                 jarak_terpanjang = jarak_total;
                 idx_max_prop = idx_prop;
             end
             % -------------------------------------------
```

```
                path(num,:) = lines(i).point2;
                % ------------ Hitung jarak path ------------
                selisih = diff(path,1,1).^2;
                jarak_step  = sqrt(sum(selisih,2));
                jarak_total = sum(jarak_step,1);
                % -------------------------------------------
                % ---- Structure data path propagation ------
                idx_prop = idx_prop + 1;
                prop(idx_prop).path = path;
                prop(idx_prop).jarak = jarak_total;
                % -------------------------------------------
                % ----------- Path Terpanjang ---------------
                if jarak_total > jarak_terpanjang
                    jarak_terpanjang = jarak_total;
                    idx_max_prop = idx_prop;
                end
                % -------------------------------------------
                done = 1;path = zeros(1,2);
            end
        end
    end
end
end
```

Kesimpulan	Dapat diambil kesimpulan bahwa algoritma yang dilakukan sangat sesuai dengan hasil yang diinginkan
Kekurangan	Belum mampu mengeksekusi perpotongan segitiga