SKRIPSI

DESAIN ALGORITMA DALAM PEMBUATAN MOZAIK FOTO PANORAMA PADA FOTO STEREO

Diajukan Untuk Memenuhi Persyaratan Dalam Mencapai Gelar Sarjana Strata Satu (S-1) Teknik Geodesi

Disusun Oleh :

LENA YUNIARITA SUMBAYAK
09.25.914

JURUSAN TEKNIK GEODESI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
INSTITUT TEKNOLOGI NASIONAL
MALANG

2012

LEMBAR PERSETUJUAN

Desain Algoritma Dalam Pembuatan Mozaik Foto Panorama Pada Foto Stereo

Diajukan Sebagai Salah Satu Syarat Memperoleh Gelar Sarjana Teknik Geodesi

S-1 Institut teknologi Nasional Malang

Disusun Oleh :

Lena Yuniarita Sumbayak

09.25.914

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
M.Edwin Tjahjadi,ST,M.Geom.Sc.,Ph.D	**Ir.M.Nurhadi,MT**

Mengetahui,

Keua Jurusan Teknik Geodesi S-1

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
TEKNIK GEODESI

Ir. Agus Darpono, MT

LEMBAR PENGESAHAN

Desain Algoritma Dalam Pembuatan Mozaik Foto Panorama pada Foto Stereo

SKRIPSI

Dipertahankan dihadapan Majelis Penguji Sidang Skripsi
Jenjang Strata Satu (S-1)

Pada hari : Sabtu
Tanggal : 4 Februari 2012
Dan diterima untuk memenuhi salah satu persyaratan guna memperoleh gelar Sarjana Teknik.

Disusun Oleh :

Lena Yuniarita Sumbayak 09.25.914

Panitian Ujian Tugas Akhir

Ketua	Sekretaris
Ir. Agus Darpono, MT	**Silvester Sari Sai, ST, MT**

Anggota Penguji

Penguji I	Penguji II	Penguji III
Ir.M.Nurhadi,MT	**D.K.Sunaryo,ST,MT**	**M.Edwin Tjahjadi,ST,M.Geom.Sc.,Ph.D**

JURUSAN TEKNIK GEODESI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
INSTITUT TEKNOLOGI NASOIONAL
MALANG
2012

DESAIN ALGORITMA DALAM PEMBUATAN MOZAIK FOTO PANORAMA PADA FOTO STEREO

Lena Yuniarita Sumbayak 09.25.914

Dosen Pembimbing I : M.Edwin Tjahjadi,ST,M.Geom.Sc.,Ph.D
Dosen Pembimbing II : Ir.M.Nurhadi,MT

Abstraksi

Penelitian ini menyajikan suatu teknik untuk membangun mozaik foto panorama *(panoramic image mosaic)* yang dibentuk oleh sepasang foto stereo. Langkah awal dimulai dengan proses *image matching* untuk mendapatkan titik konjugasi antara dua buah foto. Salah satu metode yang digunakan dalam proses *image matching* adalah *area based matching* dimana teknik perhitungannya menggunakan teknik *Normalized Cross Correlation.* Titik-titik konjugasi yang diperoleh akan digunakan untuk menghitung parameter transformasi pada ruang 2D yang biasanya disebut dengan *homography* untuk menggabungkan kedua foto (*image stitching*), dan dilanjutkan dengan proses *image blending.* Hasil dari setiap proses yang disebutkan adalah sebuah foto panorama yang dibentuk oleh sepasang foto stereo.

Kata kunci : panorama, mozaik foto, *homography*

PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan di bawah ini :

Nama : Lena Yuniarita Sumbayak

NIM : 09.25.914

Program Studi : Teknik Geodesi S-1

Fakultas : Fakultas Teknik Sipil Dan Perencanaan

Menyatakan dengan sesungguhnya bahwa Skripsi saya dengan judul :

"Desain Algoritma Dalam Pembuatan Mozaik Foto Panorama Pada Foto Stereo"

adalah hasil karya saya sendiri, bukan merupakan duplikat serta tidak mengutip atau menyadur dari hasil karya orang lain kecuali disebutkan sumbernya.

Malang, 13 Februari 2012
Yang membuat pernyataan

Lena Yuniarita Sumbayak
09.25.914

PERSEMBAHAN

Tulisan ini aku persembahkan untuk :

Yesus..

Allahku yang hidup, sahabat, guru dan sumber inspirasiku..

Bapak, Mamak, Kak Epi, Abang Oki, Adek Andi, dan seluruh keluargaku..
Kado terindah yang aku peroleh selama hidupku..

Dytto..
Best Friend Forever..

Percayalah kepada Tuhan dengan segenap hatimu, dan janganlah bersandar kepada pengertianmu sendiri, akuilah Dia dalam segala lakumu maka Ia akan meluruskan jalanmu.
(Amsal 2:5-6)

KATA PENGANTAR

Puji Syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat rahmat dan karunia-Nya penulis dapat menyelesaikan skripsi yang berjudul **"Desain Algoritma Dalam Pembuatan Mozaik Foto Panorama Pada Foto Stereo"**, dimana penulisan skripsi ini disusun untuk memenuhi salah satu syarat untuk meraih gelar Sarjana Teknik pada Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.

Penulisan ini tidak akan dapat terselesaikan tanpa bantuan dan dukungan berbagai pihak. Oleh karena itu, penenliti ingin mngucapkan terima kasih yang sebesar-besarnya kepada :

1. Bapak Ir. Soeparno Djiwo, M.T, selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. Andrianus Agus Santosa, MT, selaku Dekan Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.
3. Bapak Ir. Agus Darpono, MT, selaku Ketua Jurusan Teknik Geodesi Institut Teknologi Nasional Malang.
4. Bapak M. Edwin Tjahjadi, ST, M.Geom.Sc., Ph.D, selaku Dosen Pembimbing I.
5. Bapak Ir.M.Nurhadi,MT, selaku Dosen pembimbing II.
6. Segenap dosen, staff pengajar dan *recording* Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.

7. Bapak, Mama, dan adek-adekku, yang selalu memberikan dukungan, semangat dan doa.
8. Semua teman-teman transferan UGM, *Team Rapid Mapping* dan Deliniasi Garis Pantai, Geodesi ITN 2005-2011, dalam kerjasama dan dukungannya.
9. Teman-teman PMK ITN, saudara dan adek-adek KTBku (Kak Jane, Kak Yetti, Christin, Desi "Butet", Desy, Septi, Dewi) dan kakak-kakak pembimbing.
10. Teman-teman Kos Puspita (Nia, Frilla, Shinta, Tika, Lia, Mb Ika, Yenni, Yana, Nisa).
11. Yohanes Wilhelmus L.L, terimaksih buat dukungannya.
12. Semua pihak yang telah membantu terlaksananya penelitian dan penulisan laporan ini, yang tidak dapat disebutkan satu persatu.

Penulis menyadari masih banyak kekurangan dalam penulisan laporan penelitian ini. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun dari pembaca., dan semoga laporan ini dapat berguna bagi pembacanya.

Malang, 13 Februari 2012

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

I.1 Latar Belakang

Photogrametry adalah sebuah teknik untuk mendapatkan informasi mengenai posisi, ukuran dan bentuk dari sebuah objek dengan mengukur citra foto secara langsung *(K.B.Atkinson, 2001).* Dikarenakan kamera yang digunakan dalam merekan suatu foto memiliki cakupan yang terbatas, maka seringkali sebuah foto tidak dapat mencakup objek atau daerah yang ingin diukur, apalagi jika daerah tersebut sangat luas. Oleh karena itu, untuk mendapatkan informasi yang lengkap, objek atau daerah tersebut direkam dengan lebih dari satu foto.

Untuk menggabungkan foto-foto tersebut maka dibentuklah mosaik foto. Mosaik foto ini terdiri dari beberapa foto yang dijadikan satu menjadi sebuah foto dengan cakupan objek atau luasan yang lebih luas yang disesuaikan dengan kebutuhan pengukuran. Proses pembuatan mosaik foto dapat dilakukan secara manual maupun secara digital dengan bantuan komputer. Syarat foto-foto yang ingin dijadikan mosaik harus memiliki pertampalan atau daerah yang *overlap*. Foto panorama merupakan bentuk dari mosaik foto.

Banyaknya metode yang digunakan dalam pembentukan mosaik foto panorama yang akan berpengaruh pada mosaik foto panorama yang akan dihasilkan. Oleh karena itulah penulis melakukan penenlitian pembuatan mosaik foto panorama dengan memilih sebuah metode dalam setiap prosesnya.

I.2 Perumusan Masalah

Banyak metode yang disajikan dalam proses pembuatan mozaik foto panorama, untuk itu pemilihan metode yang tepat dengan hasil yang baik dan efisiensi perhitungan yang tepat akan mempengaruhi foto panorama yang akan dihasilkan.

I.3 Tujuan Penelitian

Tujuan penelitian ini adalah menyajikan algoritma pembuatan mozaik foto panorama, dengan menggunakan teknik *normalized cross correlation* untuk penentuan titik konjugasi, proses transformasi proyeksi (*homography*) dan diakhiri dengan proses *image blending* dengan metode *linear gradient alpha blending* untuk mendapatkan sebuah foto panorama.

I.4 Batasan Masalah

Penelitian ini dibatasi pada :

1. Pembuatan mozaik foto panorama pada sepasang foto stereo pada bidang planar (bidang datar).
2. Penentuan titik konjugasi menggunakan teknik *normalized cross corelation*, proses transformasi proyeksi (*homography*), dan diakhiri dengan proses *image blending* dengan metode *linear gradient alpha blending*.

I.5 Tinjauan Pustaka

Dalam penelitiannya tentang analisis ketelitian dalam pembuatan mosaik foto udara, Sofwan (2002) membentuk mosaik yang diawali dengan penentuan titik sekutu (*tie point*) kemudian menggabungkan foto-foto tersebut dengan menggunakan metode orthogonal dan metode *polynomial*. Untuk meminimalisir perbedaan antara foto-foto yang digabungkan digunakan metode perentangan kontras (*contras stretching*). Dari kedua metode yang digunakan Sofwan menyimpulkan bahwa pembuatan mosaik foto udara dengan menggunakan metode orthogonal menghasilkan foto mosaik yang lebih teliti.

Shum dan Szeliski dalam laporan penelitiannya yang berjudul *Panoramic Image Mosaics* menyajikan teknik pembuatan foto panorama dari foto-foto yang berurutan. Dengan menggunakan matrik rotasi dan *homography* untuk menggabungkan rentetan foto-foto tersebut dengan nilai fokus kamera yang bebas. Kemudian foto-foto yang sudah di gabungkan (*stitching*) dengan menggunakan transformasi proyeksi 2D dan nilai rotasi foto yang kemudian diproyeksikan dalam bidang silinder.

Pontinen meneliti tentang nilai geometry mozaik foto panorama yang dibentuk oleh foto-foto yang berurutan. Penelitian yang berjudul *On The Geometrical Quality of Panoramic Images*, dalam pengambilan foto yang akan digunakan, Pontinen menempatkan kamera tersebut pada statip dengan posisi yang tetap yang kemudian kameranya akan diputar. Selain proses yang biasa dilakukan dalam pembuatan mozaik foto panorama dalam penelitian ini Pontinen juga mempertimbangkan kalibrasi dari kamera yang digunakan. Dalam penelitian

Pontinen (2000) lain yang berjudul *On The Creation of Panoramic Images From Image Sequances*, Pontinen menggabungkan foto-foto tersebut dengan dua metode. Pertama, dengan nilai rotasi dari setiap foto dan kemudian diproyeksikan ke bidang silinder. Kedua, dengan menggunakan tranformasi proyeksi 2D kemudian ditransformasikan pada bidang silinder. Penelitian ini tidak mempertimbangkan nilai dari parameter kamera.

Recognising Panoramas adalah judul penelitian Brown dan Lowe (2003) yang mengkaji tentang pembuatan mozaik foto panorama. Untuk menentukan titik konjugasi pada foto-foto yang bertampalam mereka menggunakan *feature matching* dengan teknik SIFT dan metode RANSAC untuk menghitung nilai *homographynya*. Brown dan Lowe juga menambahkan teknik *bundle adjustment* untuk mengetahui parameter kameranya.

BAB II

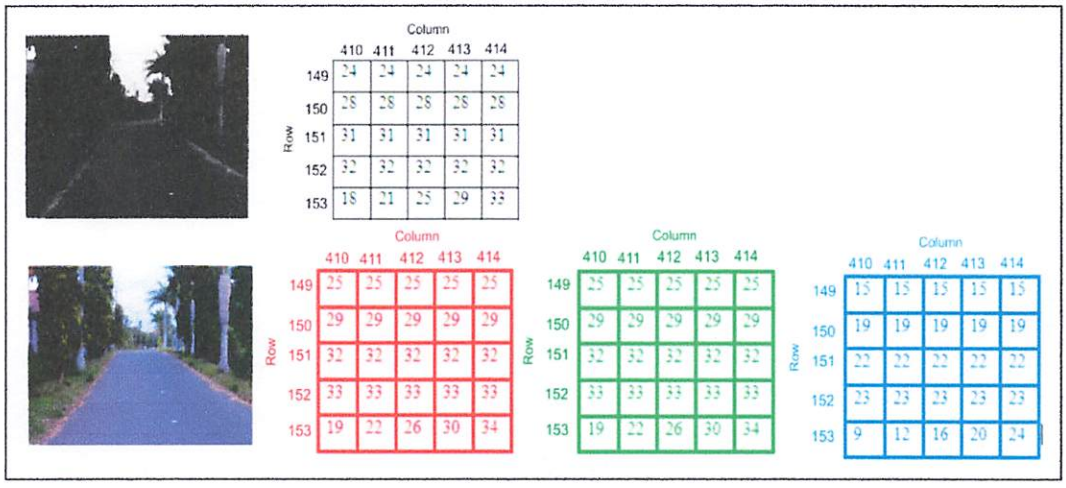
DASAR TEORI

Algoritma untuk menggabungkan foto-foto *(images stitching)* untuk membentuk suatu mosaik foto panorama sudah lama dan dikembangkan sangat luas dalam berbagai bidang ilmu. Berbagai metode digunakan untuk mendapatkan algoritma *image stitching* yang diharapkan dapat membentuk mosaik foto panorama dengan kualitas yang sangat baik.

Dalam komunitas fotogrametri pembuatan mosaik foto panorama terlebih dahulu dimulai dengan penentuan titik-titik kontrol tanah *(ground control point)* atau dengan meregistrasikan titik ikat *(tie point)* pada foto yang bertampalan secara manual. Selain pemilihan titik-titik ikat, terdapat metode-metode lainnya yang dibutuhkan dalam pembentukan mosaik foto panorama. Untuk jelasnya metode-metode yang dipakai dalam penelitian ini akan dijelaskan pada bab 2 ini.

II.1 Citra Digital

Citra digital adalah foto (citra) yang direpresentasikan pada komputer, dimana citra tersebut dibagi dalam suatu grid yang disebut dengan *"picture element"* atau piksel *(Wolf and Dewitt, 2000)*. Setiap citra terdiri dari sekumpulan bilangan bulat yang biasa disebut sebagai *"digital numbers". Digital numbers* mengukur tingkat keabuan *(gray level)* atau derajat kegelapan dari foto tersebut *(Wolf and Dewitt, 2000)*. Setiap nilai piksel pada citra merepresentasikan nilai intensitas cahaya (Gambar 2.1),



Gambar 2.1 Citra hitam & putih dan citra berwarna beserta matrik yang merepresentasikannya

Pada umumnya nilai piksel pada citra bernilai antara 0-255, dimana nilai 0 untuk berwarna lebih gelap (hitam) dan 255 untuk yang berwarna putih. Nilai 0-255 dimuat dalam *1 byte* yang berisi 8 pasang digit, atau *bits*. Nilai 8 bit dapat menyimpan 2^8 atau 256 (0-255) degradasi warna.

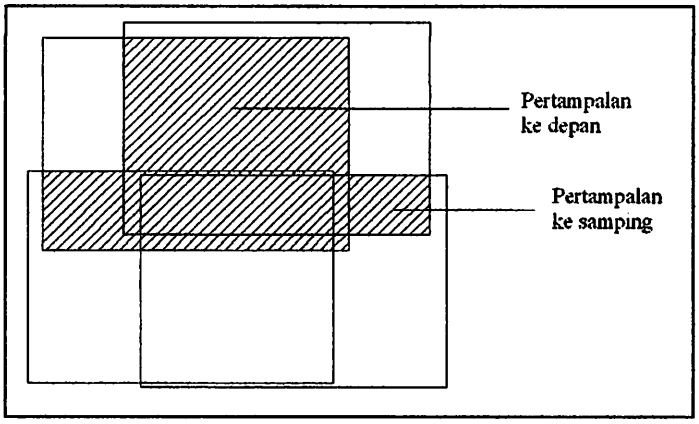
Citra digital dapat diperoleh dengan dua cara yaitu melalui sensor kamera atau dengan proses secara tidak langsung yakni dengan memindai (*scanning*) foto analog untuk menjadi citra digital. Informasi tentang obyek yang ditampilkan dalam citra kemudian dianalisa atau yang lebih dikenal dengan 'proses data citra'. Kualitas citra digital tergantung pada parameter-parameter dari CCD (C*harge Coupled Device*), bagian-bagian *photosensitive scanner*, kamera digital dan sensornya.

II.1.1 Citra Bertampalan *(Overlap)*

Pertampalan foto udara adalah foto udara yang dipotret (direkam) pada suatu daerah yang di potret 2x atau lebih dari posisi stasiun pemotretan yang berbeda. Syarat pertampalan foto udara adalah 60% untuk pertampalan ke depan dan 30% untuk pertampalan ke samping *(Michail,2001)*. Ketentuan besarnya pertampalan foto udara dan mengapa diperlukan pertampalan foto udara adalah agar tidak terjadinya gap, selain itu dengan adanya pertampalan foto udara maka foto tersebut dapat dilakukan stereo (pandangan 3 dimensi).

Pertampalan foto udara ada 2 macam yaitu :

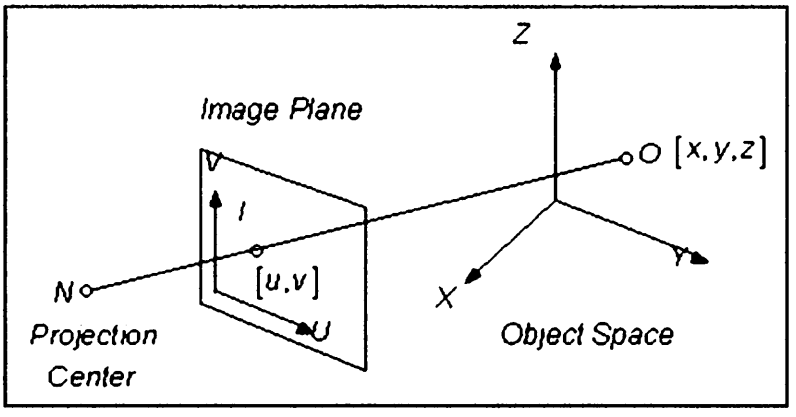
1. Pertampalan depan (*overlap*) bila pertampalannya mengikuti arah jalur terbang. Pertampalan kedepan menerangkan tentang hubungan antar batas-batas foto yang terekam di sekitar garis penerbangan*(Michail,2001l)*.
2. Pertampalan samping (*sidelap*) bila pertampalan satu foto dengan foto yang lain pada jalur di sebelahnya. Pertampalan ke samping menerangkan tentang hubungan sepasang foto pada dua batas garis penerbangan *(Michail,2001)*.



Gambar 2.2 Gambar bertampalan

II.1.2 Orientasi dan Georeferensi Citra

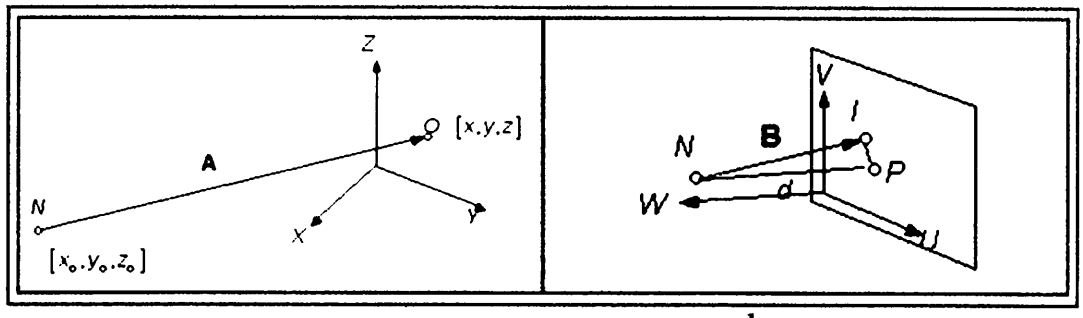
Untuk menentukan koordinat objek pada titik-titik yang akan diukur pada citra merupakan tugas pokok dalam fotogrametri. Hubungan geometris antara citra dengan sistem koordinat objek ditunjukkan pada Gambar 2.3,



Gambar 2.3 Kondisi kesegarisan (Kolinear) (Kwon, 1998)

Pada Gambar 2.3 sebuah titik objek *O* pada bidang objek *(object space)* diproyeksikan ke bidang foto secara langsung. Bidang foto *(image plane)* disebut sebagai bidang proyeksi dimana titik *N* adalah pusat proyeksi *(projection center)*.

Pada gambar terdapat referensi bidang objek *(object space reference)* dalam sistem *X,Y* dan *Z* dan referensi bidang foto dalam sistem *u* dan *v*. Ketika kamera merekam objek *O* pada sistem koordinat objek *(X,Y,Z)* maka objek *O* tersebut akan terekam dalam bidang foto dengan titik *I* yang memiliki sistem *(u,v)*. Titik *I*, *N* dan *O* adalah kolinear (segaris). Kondisi kolinear merupakan kondisi dimana kedudukan titik pemotretan, titik objek dan gambaran titik pada foto seluruhnya terletak pada satu garis lurus *(Mikhail et al., 2001;Wolf & dewitt, 2000)*.



Gambar 2.4 Gambar posisi vektor A dan vektor B (Kwon, 1998)

Persamaan kondisi kolinear diadopsi dari sebuah persamaan dasar transformasi sebagai berikut :

$$x' = Hx \qquad \text{Persamaan (2.1)}$$

dimana :

x' : vektor bidang hasil transformasi

H : matrik tranformasi

x : vektor bidang yang akan ditransformasi

Menurut *Michail,et all (2001)* titik O pada sistem koordinat objek dihubungkan dengan sistem koordinat foto dengan menggunakan rotasi matrik omega, phi, kappa *(ω,φ,K)* dan ditambahakan dengan skala. Berdasarkan pada pengertian ini maka hubungan titik O pada bidang foto dan bidang objek dengan menggunakan persamaan 2.1 dan dijabarkan dalam bentuk matrik, maka persamaan 2.1 di atas dapat ditulis sebagai berikut :

$$\begin{bmatrix} x \\ y \\ -d \end{bmatrix} = SR \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \text{Persamaan (2.2)}$$

Dimana :

x,y : *koordinat foto*

d : *panjang fokus kamera*

S : *parameter skala*

R : *parameter matrik rotasi dengan dimensi 3x3*

X,Y,Z : *parameter koordinat objek dalam ruang tiga dimensi*

berdasarkan persamaan (2.2) maka hubungan titik koordinat objek pada gambar 2.3 adalah :

$$\begin{bmatrix} U - U_o \\ V - V_o \\ -d \end{bmatrix} = c \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X - X_o \\ Y - Y_o \\ Z - Z_o \end{bmatrix} \qquad \text{Persamaan (2.3)}$$

persamaan (2.3) dikembangkan menjadi,

$$U - U_o = S - d\frac{r_{11}(X - X_0) + r_{12}(Y - Y_0) + r_{13}(Z - Z_0)}{r_{31}(X - X_0) + r_{32}(Y - Y_0) + r_{33}(Z - Z_0)}$$

Persamaan (2.4a)

$$V - V_o = S - d\frac{r_{21}(X - X_0) + r_{22}(Y - Y_0) + r_{23}(Z - Z_0)}{r_{31}(X - X_0) + r_{32}(Y - Y_0) + r_{33}(Z - Z_0)}$$

Persamaan (2.4b)

Keterangan :

U,V : *koordinat foto pada titik P*

U_0, V_0 : *pusat sistem koordina foto (principle point)*

X,Y,Z : *koordinat objek pada titik P dalam ruang 3 dimensi*

X_0, Y_0, Z_0 : *koordinat posisi kamera pada saat melakukan pemotretan*

d : *panjang fokus kamera*

S : *skala*

$r_{11}, r_{12}...r_{33}$: *elemen matrik rotasi*

Persamaan 2.4a dan 2.4b disebut dengan persamaan kolinear. Koordinat foto dari *principle point*, *principle distance (d)* dan distorsi lensa disebut sebagai parameter *interior orientation*, sedangkan koordinat posisi kamera dan parameter rotasi disebut parameter *exterior orientation*. Dalam proses perhitungan kedua parameter ini harus dipecahkan terlebih dahulu. Parameter *interior orientation* biasanya diambil dari informasi kalibrasi kamera. *Interior orientation* juga dapat ditentukan melalui proses perhitungan pada *exterior orientation (self calibration)*. Dalam kasus pemindahan foto *(scanned analogue photograph)*, hubungan antar koordinat foto dan sistem koordinat piksel harus diselaraskan. Karena semua pengukuran dilakukan dalam sistem koordinat piksel sedangkan dalam mengoreksi distorsi lensa dan kemungkinan menggunakan persamaan kolinear, maka diperlukan pengukuran posisi kedalam sistem koordinat foto.

II.1.3 Pencocokan Citra (*Image matching*)

Menemukan titik konjugasi secara otomatis yang terdapat pada dua atau lebih foto yang bertampalan dikenal dengan *image matching* atau pencocokan citra. (*Schenk, 1999*). Beberapa teknik yang digunakan dalam proses *image matching* adalah *area based, feature based* dan *symbolic matching* (*Schenk, 1999: Wolf* dan *Dewit, 2000*). Tabel berikut menunjukan hubungan metode pencocokan citra dengan entitasnya.,

Metode Pencocokan Citra	Teknik Perhitungan Pencocokan Citra	Entitas
Area Based	*Normalized Cross Correlation, Least Square Matching*	Derajat keabuan *(gray level)*
Feature Based	Fungsi *cost*	Titik tepi, daerah
Symbolic Based	Fungsi *cost*	Keterangan symbol

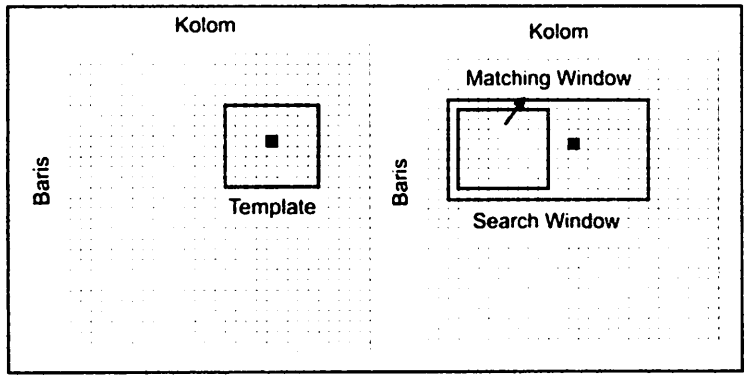
Tabel 2.1 Metode Image Matching (Schenk,1999)

Pencocokan citra dengan penentuan titik-titik konjugasi dalam 2 foto pada penelitian ini menggunakan teknik *area based* dengan teknik perhitungannya menggunakan *Normalized cross correlation.*

II.1.3.1 Metode *Area Based*

Entitas yang digunakan dalam *area based matching* adalah nilai keabuan citra (*gray value*). Pada foto pertama dibentuklah sebuah *image patch* (jendela sasaran) yang disebut sebagai *template* yang memuat titik piksel yang akan dicari pasangannya pada foto kedua. Ukuran dari *template* biasanya m x n piksel atau m = n piksel. Karena pada umumnya pusat *template* berada ditengah *template,* sehingga biasanya *template* berukuran ganjil. Pada foto kedua dibentuk daerah selidik dengan ukuran yang lebih besar dari *template* yang disebut dengan *search window*. Di dalam *search window* ini dibentuk pula jendela/daerah sub selidik atau *matching window* yang memiliki ukuran yang sama dengan *template. Matching window* ini akan bergerak (*moving window*) dengan *increment* 1 piksel sepanjang setiap baris dan kolom di daerah selidik (*search window*). Korelasi nilai

keabuan yang tertinggi dan melebihi nilai *threshold* antara *template* dengan *matching window* dinyatakan sebagai daerah yang paling berkesesuaian. Penjelasan ini diilustrasikan dalam gambar berikut :



Gambar 2.5. Konsep Area Based Matching (Schenk,1999)

II.1.3.2 *Normalized Cross Correlation* (NCC)

Normalized Cross Correlation merupakan teknik penghitungan dalam *area based matching (Schenk, 1999: Wolf dan Dewit, 2000)*. Teknik *normalized cross correlation* ini membandingkan nilai keabuan antara *template* pada foto pertama dan *matching window* pada foto kedua. Koefisien korelasi dihitung dengan menggunakan persamaan berikut *(Schenk, 1999: Wolf dan Dewit, 2000)* :

$$c = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}[(A_{ij}-\bar{A})(B_{ij}-\bar{B})]}{\sqrt{\left[\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij}-\bar{A})^2\right]\left[\sum_{i=1}^{m}\sum_{j=1}^{n}(B_{ij}-\bar{B})^2\right]}} \quad \text{Persamaan (2.5)}$$

Dimana :

c : *koefisien normalized cross correlation*

A dan B : *template dan matching window*

$\bar{A}$ *dan* $\bar{B}$: *nilai rata-rata dari template dan matching window*

i dan j : *baris dan kolom dari image patch*

Apabila pada foto pertama ditentukan sebuah objek sebagai titik acuan pencarian, maka komputer harus menentukan objek tersebut pada foto kedua dengan mengamati sekumpulan nilai *gray value* pada kedua foto. Dalam domain digital, citra tersebut dipresentasikan sebagai variasi nilai piksel yang membentuk dimensi m x n piksel atau m = n piksel. Apabila pada foto pertama ditentukan *template* yang berdimensi 5 x 5 disekeliling titik objek maka *template* ini berisi sekumpulan (25) nilai piksel disekeliling titik acuan. Pada foto kanan dibentuk *matching window* yang memiliki dimensi yang sama dengan *template*, dimana *matching window* ini terdapat dalam *search window*. Sampai tahap ini akan diperoleh nilai dua buah matrik (*template* dan *matching window*) dengan dimensi yang sama.

Penempatan *matching window* dimulai dari posisi ujung kiri atas dari *search window*. Kemudian *matching window* akan bergeser menelusuri citra kolom demi kolom kearah kanan sampai mencapai batas dari *search window*. Setelah itu, *matching window* akan bergeser ke bawah sebanyak satu baris dan kembali menelusuri sepanjang baris tersebut ke arah kiri, demikian seterusnya. Proses penelusuran dilakukan sampai ke seluruh *search window*. Untuk setiap tahap penelusuran, nilai *c* dihitung dan dicatat oleh sistem komputer.

Koefisien *normalized cross correlation (c)* memiliki nilai antara $-1 \leq c \geq 1$. Nilai +1 diindikasikan sebagai korelasi yang sempurna dan nilai -1 diperoleh ketika ada kecocokan dari citra positif dan negatif. Nilai koefisien yang mendekati nilai 0 diidentifikasikan sebagai nilai yang tidak memiliki korelasi *(mismatch)*.

II.2 Proyeksi Geometri 2D

Menurut gambar 2.3 titik objek O dalam bidang 3D diproyeksikan pada bidang foto *(image plane).* Bidang tempat diproyeksikannya suatu titik disebut dengan bidang proyeksi. Titik pada suatu bidang, baik itu 2D atau 3D diproyeksikan pada bidang proyeksi dengan sebuah garis lurus. Menurut Hartley dan Zisserman (2003) sebuah titik x $(x,y)^T$ yang terletak pada bidang 2D berada pada suatu *garis* l $(a,b,c)^T$ jika dan hanya jika $ax+by+c=0$. Persamaan ini direpresentasikan dalam bentuk vektor menjadi $(x,y,1)(a,b,c)^T=(x,y,1)l=0$, hal ini menunjukkan bahwa suatu titik pada bidang 2 dimensi direpresentasikan dengan $x=(x,y,1)^T$. Jika pada persamaan $(x,y,1)l=0$, ditambahkan suatu konstanta k maka persamaan tersebut menjadi $(kx,ky,k)l=0$, untuk berbagai jenis nilai k harus merepresentasikan titik $(x,y)^T$ pada bidang 2 dimensi. Atau dengan kata lain $x = \frac{kx}{k}$ dan $y = \frac{ky}{k}$. Koordinat dengan bentuk $(x,y,1)^T$ dikenal dengan koordinat *homogeneous.*

II.2.1 Transformasi Proyeksi 2D

Proyeksi transformasi di antara bidang proyeksi disebut dengan *homography*, baik itu bidang 2 dimensi ataupun 3 dimensi *(Heuel.S,1973; Estrada.F.J et al, 2004)*. Transformasi proyeksi direpresentasikan dengan persamaan dasar transformasi *(Heuel.S,1973)* yang diuraikan pada persamaan 2.1.

Jika vektor x' dan x berbentuk *homogeneous* maka transformasi ini adalah transformasi *homogenenous* dengan skala sembarang (*up to scale*) *(Heuel.S,1973).*

Menurut Hartley dan Zisserman *(2003),* transformasi proyeksi pada bidang planar (2D) adalah sebuah transformasi yang bersifat linear pada vektor yang bersifat *homogeneous* dan direpresentasikan dengan matrik yang berukuran 3x3 dan matrik tersebut bersifat non singular. Matrik proyeksi transformasi ini dibentuk oleh sekumpulan titik-titik yang berkesesuaian antara dua buah bidang yang akan dilakukan proses transformasi. Dari persamaan 2.1 dikembangkan menjadi,

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{pmatrix} X \\ Y \\ w \end{pmatrix} \qquad \text{Persamaan (2.6)}$$

Pada persamaan 2.6 matrik *H* memiliki 9 elemen, namun karena matrik *H* tersebut adalah matrik dengan skala sembarang *(up to scale)* maka matrik *H* memiliki 8 DOF dengan elemen $h_9=1$ dan $w'=w=1$ karena persamaan ini menggunakan koordinat *homogeneous.* Karena matrik H ini memiliki 8 DOF dan setiap titik memberikan 2 buah persamaan, 1 persamaan untuk *x* dan 1 persamaan untuk *y*, maka untuk menyelesaikan matrik *H* pada transformasi suatu titik pada bidang dua dimensi dibutuhkan minimal 4 titik yang berkesesuaian pada dua buah bidang proyeksi. Dengan titik-titik berkesesuaian yang lebih banyak dari jumlah minimal akan menghasilkan matrik *H* yang lebih baik.

II.2.1.1 Metode *Direct Linear Transformation* (DLT)

Pada umumnya kegunaan metode DLT ini adalah untuk menyelesaikan persamaan kolinear pada persamaan (2.4) dalam menentukan matrik rotasi pada hubungan antara bidang 3D dengan bidang 2D *(Remondino.F, 2002).* Namun

metode DLT ini juga dapat digunakan untuk menghitung matrik proyeksi transformasi *(Heuel.S,1973; Hartley, R, & Zisserman, A, 2003; Belo, 2009)*. Persamaan transformasi (2.1) dengan perkalian *cross product* akan menjadi (*Hartley, R, & Zisserman, A, 2003;Belo, 2009*):

$$x_i' \text{ x } Hxi = 0 \qquad \text{Persamaan (2.7)}$$

jika baris dari matrik H dinotasikan dengan h^j, dimana j menandakan baris dari elemen h, maka,

$$Hx_i = \begin{pmatrix} h^1 x_i \\ h^2 x_i \\ h^3 x_i \end{pmatrix} \qquad \text{Persamaan (2.8)}$$

dan $x_i' = (x_i', y_i', w_i')^T$, $x_i = (x_i, y_i, w_i)^T$ dengan perkalian *cross* maka persamaan (2.7) menjadi :

$$x_i' \text{ x } Hxi = \begin{pmatrix} y_i' h^3 x_i - w_i' h^2 x_i \\ w_i' h^1 x_i - x_i' h^3 x_i \\ x_i' h^2 x_i - y_i' h^1 x_i \end{pmatrix} \qquad \text{Persamaan (2.9)}$$

untuk j=1,…,3, maka persamaan (2.9) dapat dituliskan menjadi :

$$\begin{bmatrix} 0 & -w_i' x_i & y_i' x_i \\ w_i' x_i & 0 & -x_i' x_i \\ -y_i' x_i & x_i' x_i & 0 \end{bmatrix} \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix} = 0 \qquad \text{Persamaan (2.10)}$$

Persamaan (2.10) menggunakan koordinat *homogeneous* oleh karena itu persamaan (2.10) dapat diselesaikan dengan sebuah persamaan *linear homogeneous* untuk menentukan parameter yang tidak diketahui pada matrik *homography*, yaitu :

$$Ah = 0 \qquad \text{Persamaan (2.11)}$$

dimana A adalah matrik koefisien dan h merupakan 9 elemen matrik parameter yang tidak diketahui,

$$h = \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix}, \qquad H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_5 \\ h_6 & h_7 & h_8 \end{bmatrix} \qquad \text{Persamaan (2.12)}$$

dari persamaan (2.10) nilai koefisien untuk matrik A adalah,

$$A = \begin{pmatrix} 0 & 0 & 0 & -x_1 & -y_1 & -1 & y_1'x_1 & y_1'y_1 & y_1'w_1 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1'w_1 \\ -y_1'x_1 & -y_1'y_1 & -y_1'w_1 & x_1'x_1 & x_1'y_1 & x_1'w_1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -x_n & -y_n & -1 & y_n'x_n & y_n'y_n & y_1'w_1 \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_1 & -x_1'w_1 \\ -y_n'x_n & -y_n'y_n & -y_n'w_n & x_n'x_n & x_n'y_n & x_n'w_n & 0 & 0 & 0 \end{pmatrix}$$

Persamaan (2.13)

$$h = (h_1 \;\; h_2 \;\; h_3 \;\; h_4 \;\; h_5 \;\; h_6 \;\; h_7 \;\; h_8 \;\; h_9)^T \qquad \text{Persamaan (2.14)}$$

dimana A adalah matrik koefisien dan h merupakan 9 elemen matrik parameter yang tidak diketahui. Untuk mendapatkan solusi dalam penentuan elemen matrik h dapat menggunakan Metode *Singular Value Decomposition* (SVD) *(Hartley, R, & Zisserman, A, 2003, Kriegman.D, 2007, Belo, 2009)*. Metode SVD ini dipakai untuk memecah matrik A menjadi tiga buah matrik singular yaitu U,S,V, persamaan tersebut adalah sebagai berikut :

$$_mA_n = {}_mU_m \; {}_mS_n \; {}_nV_n{}^T \qquad \text{Persamaan (2.15)}$$

Keterangan :

$_mA_n$: *Matrik koefisien*

$_mU_m$, $_mS_n$: *Matrik Orthogonal hasil dekomposisi*

$_nV_n$:*Matrik diagonal dengan tanpa elemen bernilai negative hasil dekomposisi*

Dari persamaan di atas, solusi yang didapat untuk matrik *h* pada persamaan (2.14) merupakan nilai dari kolom matrik *V* yang berkorespondensi terhadap nilai matrik singular *S* terkecil atau biasanya nilai matrik *h* berada pada kolom terakhir dari matrik *V*.

Jika dalam menentukan nilai dari matrik H menggunakan lebih dari empat titik, maka persamaan 2.14 menjadi tidak terdefenisi *(over determined)*. Jika posisi dari titik-titik hasil pengukuran tidak memiliki kesalahan maka matrik A tetap memiliki *rank* bernilai 8 dengan satu dimensi matrik nol, namun bila titik-titik pengukuran memiliki kesalahan $Ah=\epsilon$, dimana ϵ adalah aljabar kesalahan vektor *(algebraic error vector)* maka tidak ada penyelesaian yang pasti dalam bentuk persamaan 2.11. Aljabar kesalahan vektor ini dihubungkan dengan titik-titik yang berkesesuaian dan nilai *homographyn*ya yang akan menghasilkan sebuah skalar yang disebut dengan jarak aljabar *(algebraic distance)*. Untuk meminimalisir nilai dari jarak aljabar tersebut dilakukanlah proses normalisasi pada kedua foto *(Hartley, R, & Zisserman, A, 2003).*

Langkah-langkah normalisasi titik-titik tersebut adalah :

1. Normalisasi titik-titik pada foto pertama (x) dengan transformasi T dimana setiap *centroid*nya berada pada origin dari foto pertama dan jarak rata-rata dari origin adalah $\sqrt{2}$. Ubah titik-titik tersebut dengan persamaan

$$\tilde{x} = Tx \qquad \text{Persamaan (2.16)}$$

Lakukan hal yang sama dengan langkah 1 terhadap titik-titik pada foto 2 (x') dalam hal ini transformasi U, kemudian transformasi titik-titik tersebut dengan persamaan

$$\widetilde{x'} = Ux_i' \qquad \text{Persamaan (2.17)}$$

2. Lakukan cara DLT untuk menghitung nilai dari matrik *homography* $\widetilde{H}$ dari titik-titik yang sudah dinormalisasi.
3. Denormalisasi matrik $\widetilde{H}$ untuk mendapatkan matrik H *update* yang sesuai dengan titik-titik yang asli ($x_i \leftrightarrow x_i'$) dengan persamaan

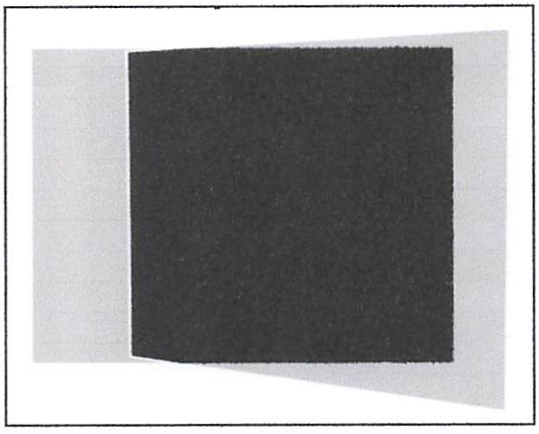
$$H = U^{-1}\widetilde{H}T \qquad \text{Persamaan (2.18)}$$

II.3 *Image Blending*

Dalam beberapa kasus, seringkali garis tepi dari foto-foto yang digabungkan tidaklah berkesesuaian, sehingga perbedaan antara kedua foto terlihat sangat mencolok *(visible seam),* adanya daerah yang kabur *(blur)* dan bayangan *(ghosting) (Szeliski, 2006).* Untuk menghilangkan efek tersebut dibutuhkan sebuah metode *blending* (pencampuran) warna pada kedua foto.

Dalam tahap ini, peneliti menggunakan metode *linear gradient alpha blending* yang dihitung mulai dari pusat salah satu foto terhadap foto yang lain.

Gradient blending bekerja dengan membentuk perubahan *gradual alpha channel* yang disambungkan pada pusat dari dua foto tersebut.



Gambar 2.6 Representasi proses blending antara dua buah foto (Souza. Cesar., 2009-2010)

Gambar 2.6, pada bagian yang paling kiri menunjukkan foto sebelah kiri (foto pertama), sedangkan bagian yang paling kanan menunjukkan foto sebelah kanan (foto kedua). Bagian yang didalam (ditengah) menunjukkan daerah yang bertampalan, daerah yang akan dilakukan operasi *blending.*

Pada daerah yang bertampalan algoritma *blending* digunakan untuk menghitung besarnya kontribusi gambar pertama dan gambar kedua pada setiap piksel. Foto-foto dicampurkan dengan menggunakan rumus sebagai berikut *(Rankov et all, 2005;Porter and Duff, 1984)* :

$$N(x,y) = \alpha I(x,y) + (1-\alpha)\, C(x,y) \qquad \text{Persamaan (2.19)}$$

Dimana :

$N(x,y)$ = *Piksel foto yang baru*

α = *Nilai bobot*

$C(x,y)$ = *Piksel foto pertama*

$I(x,y)$ = *Piksel foto kedua*

Nilai bobot pada setiap piksel dihitung dengan menormalisasi terlebih dahulu daerah yang bertampalan. Proses normalisasi merupakan proses untuk merubah nilai intensitas jangkauan piksel yang memiliki nilai sangat beragam sehingga di dapatkan nilai maksimum 1 dan nilai minimum 0, dimana proses normalisai ini bersifat linear *(Wikipedia, 2011)*. Proses normalisasi akan menghasilkan daerah yang ber*gradient*, yang memiliki dimensi yang konstan. Maka, jika terdapat daerah yang bertampalan dari dua foto yang berbeda dikenai proses normalisasi, daerah tersebut akan memiliki karakteristik atau ciri-ciri yang sama pada lokasi spasial yang sama *(Wikipedia, 2011)*. Normalisasi $(\bar{d})$ dilakukan dengan membagi selisih nilai jarak antara piksel yang berada pada daerah yang bertampalan dan pusat dari dua foto tersebut *(p)* dengan selisih antara jarak piksel yang paling dekat dengan pusat foto *(N)*, yang dinyatakan dengan persamaan :

$$\bar{d} = \frac{p}{N} \qquad \text{Persamaan (2.20)}$$

Jarak *(d)* antara dua buah titik piksel *p(x,y)* dan *q(s,t)* dihitung dengan menggunakan rumus *(Gonzales. R. C., Woods. R. E, 2008)*,

$$d = [(x-s)^2 + (y-t)^2]^{\frac{1}{2}} \qquad \text{Persamaan (2.21)}$$

Dimana :

d : *Jarak antara dua buah piksel*

x : *Nilai koordinat titik pertama pada kolom piksel*

y : *Nilai koordinat titik pertama pada baris piksel*

s : *Nilai koordinat titik kedua pada kolom piksel*

t : *Nilai koordinat titik kedua pada baris piksel*

Dengan menghitung nilai jarak dari dua buah titik piksel, kita dapat mengukur tingkat kemiripan citra. Citra dengan nilai jarak yang lebih kecil dianggap memiliki tingkat kemiripan komposisi warna yang lebih tinggi atau lebih mirip dibandingkan dengan citra yang memilki nilai jarak yang lebih besar *(Rahman, 2009)*. Selain dari itu piksel yang terdapat dekat dengan pusat dari citra memiliki bobot lebih berat/lebih besar dari bobot yang mendekati tepi gambar *(Szeliski, 2006)*.

BAB III

METODOLOGI PENELITIAN

Selain teori–teori yang berkaitan dengan penelitian, beberapa syarat juga harus dipenuhi untuk mendapatkan tujuan penelitian yang maksimal, antara lain, peralatan dan bahan penelitian, pembuatan diagram alir, dan pengolahan data. Dalam bab ini akan dibahas mengenai proses pelaksanaan penelitian mulai dari persiapan sampai pada pengolahan data yang diperoleh.

III.1 Peralatan dan Bahan Penelitian

Alat dan bahan sangat dibutuhkan dalam menunjang aktifitas penelitian, karena kesiapan dan kelengkapan alat dan bahan dapat mempengaruhi hasil dari tujuan yang ingin dicapai dalam suatu penelitian. Penjelasan alat dan bahan yang digunakan dalam penenlitian ini yaitu sebagai berikut :

III.1.1 Deskripsi Data Penelitian

Data untuk penelitian ini adalah dua buah foto stereo yang saling bertampalan.

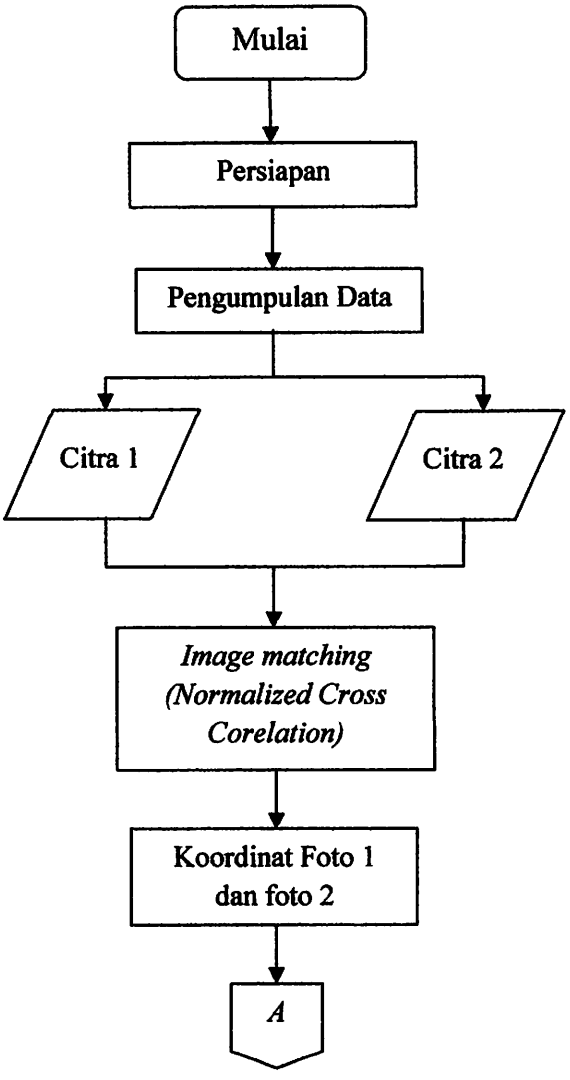
III.1.2 *Hardware* dan *Software*

1. Perangkat keras *(Hardware)*
    - Laptop DELL Vostro 1014
2. Perangkat Lunak *(Software)*
    - *Microsoft Excel* 2007
    - *Microsoft Word* 2007

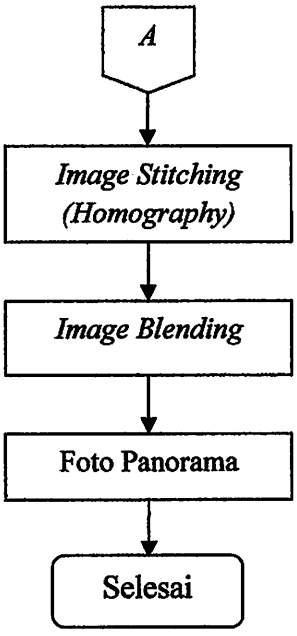
- Ms.Visual Studio (Aplikasi *C#*)

III.2 Diagram Alir Penelitian

Untuk menghasilkan desain perhitungan foto panorama, maka pelaksanaan penelitian disajikan dalan diagram alir pelaksanaan penelitian sebagai berikut :



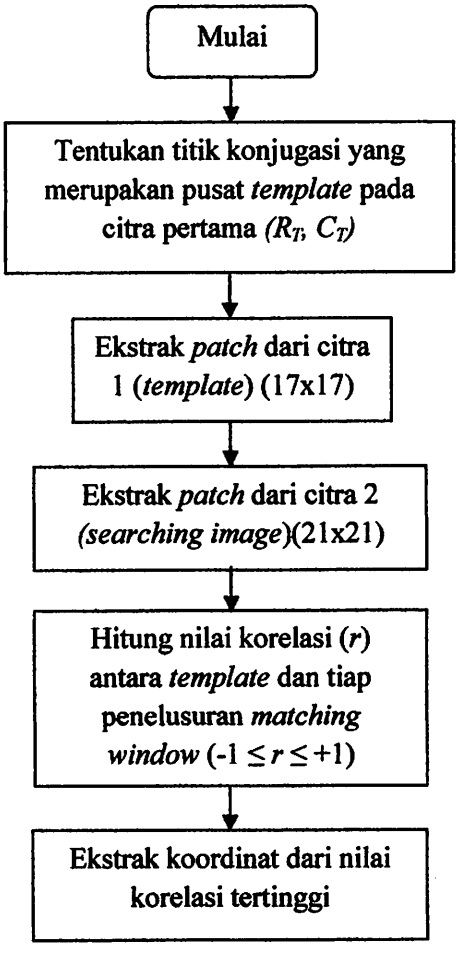
Gambar 3.1 Skema Diagram Alir Penelitian



Gambar 3.2 Skema Lanjutan Diagram Alir Penelitian

III.2.1 Diagram alir perhitungan *Image Matching* dengan teknik *Normalized Cross Correlation:*

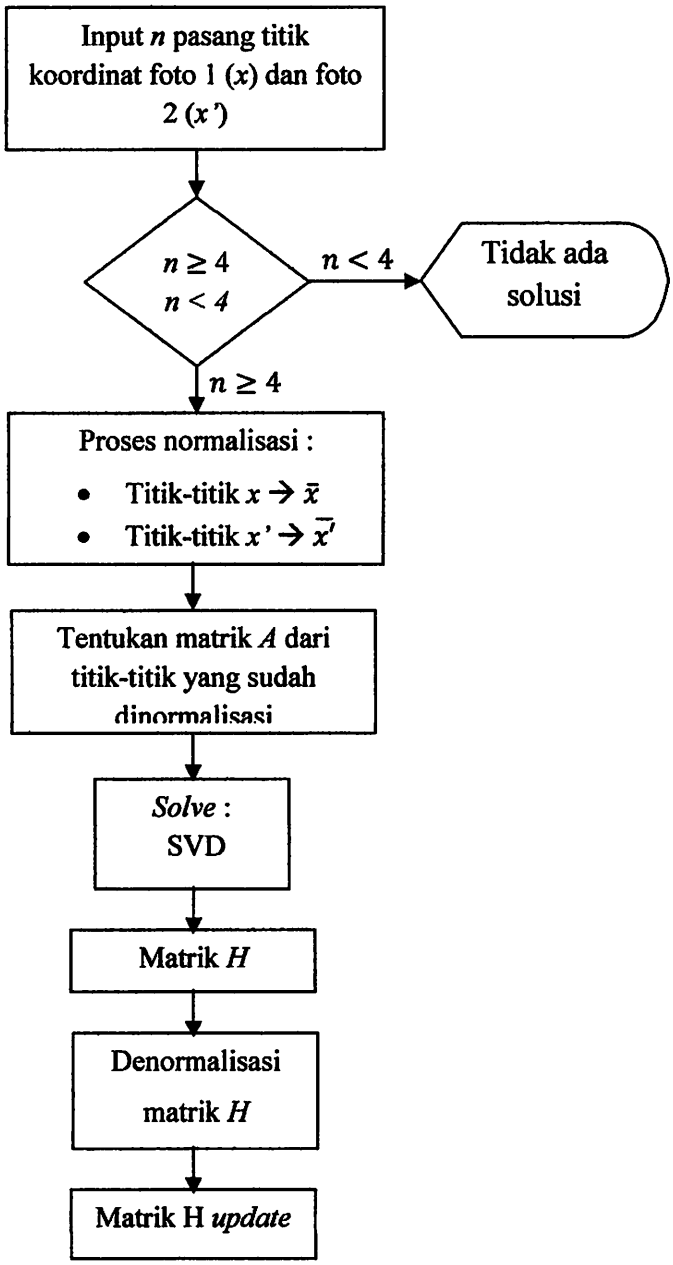
Gambar di bawah ini merupakan skema bagan alir proses *image matching* dengan proses *normalized cross correlation* untuk mengidentifikasi letak titik konjugasi :



Gambar 3.3 Skema perhitungan Image Matching dengan teknik Normalized Cross Correlation

III.2.2 Diagram alir menghitung matrik *Homography* dengan metode DLT :

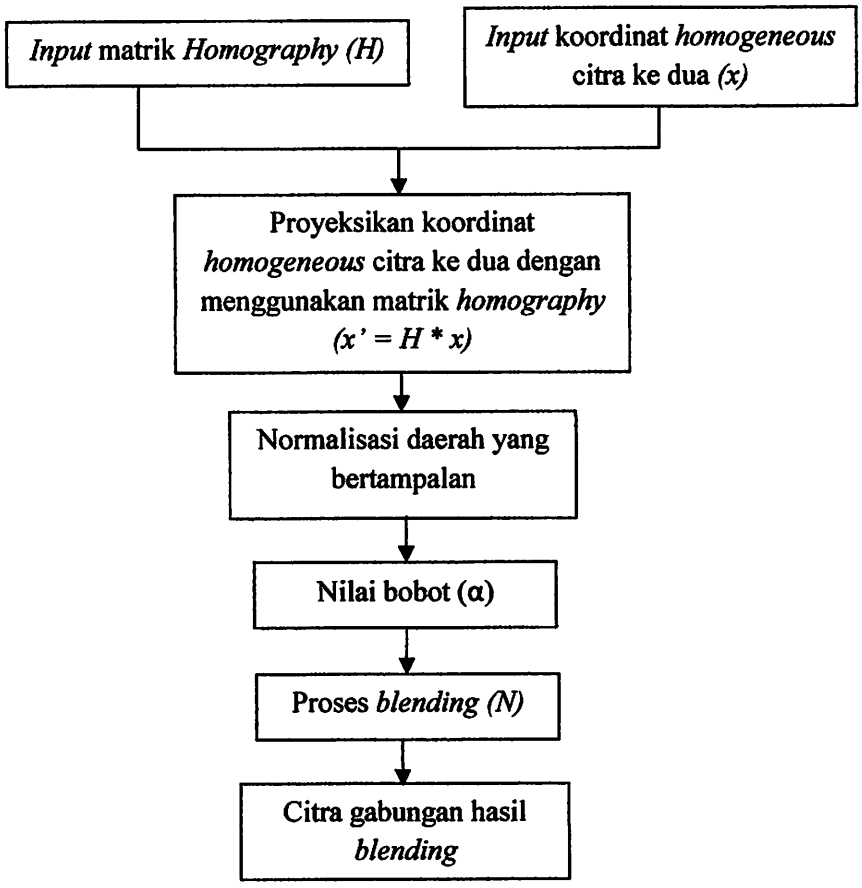
Gambar di bawah ini merupakan skema bagan alir proses perhitungan matrik *homography* (diadopsi dari *Hartley, R. & Zisserman, A. 2003)* :



Gambar 3.4 Skema perhitungan matrik Homography dengan metode DLT

III.2.3 Diagram alir perhitungan *Image Blending.*

Dibawah ini adalah skema bagan alir proses perhitungan *blending* citra dengan metode *linear gradient alpha blending* :



Gambar 3.5 Skema perhitungan Image Blending

III.3 Pengolahan Data

Desain perhitungan pembuatan foto panorama dari sepasang foto stereo melalui beberapa tahap, yaitu :

III.3.3.1 Menentukan titik-titik yang berkesesuaian *(Image Matching)*

1. Dari foto stereo yang diperoleh dari hasil perekaman objek jalan akan ditentukan titik-titik konjugasi. Titik-titik tersebut pada foto pertama akan ditentukan sebagai *template* dengan ukuran *template* 17x17 piksel, dimana pusat *template* adalah titik yang akan diidentifikasi pada foto dua.
2. Pada foto dua akan dibentuk *search window* dengan ukuran 21x21 piksel. Di dalam *search window* ini terdapat *matching window* yang berukuran sama dengan *template.*
3. Hitung nilai korelasi *(c)* antara *template* dengan setiap penelusuran *matching window* dengan menggunakan persamaan (3.1), yaitu :

$$c = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}[(A_{ij}-\bar{A})(B_{ij}-\bar{B})]}{\sqrt{\left[\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij}-\bar{A})^2\right]\left[\sum_{i=1}^{m}\sum_{j=1}^{n}(B_{ij}-\bar{B})^2\right]}} \qquad \text{Persamaan (3.1)}$$

4. Nilai korelasi tertinggi (dalam penelitian ini syarat nilai c adalah $c \geq 0.75$) dinyatakan sebagai daerah *matching window* yang paling sesuai dengan daerah *template.*

III.3.3.2 Menentukan parameter matrik *homography*

Dengan menggunakan nilai parameter koordinat piksel pada foto pertama dan foto kedua yang telah diperoleh dari proses *image matching* dapat dilakukan perhitungan nilai matrik *homography.* Adapaun proses perhitungan matrik *homography* akan dijabarkan dalam langkah-langkah sebagai berikut :

1. Sebelum menentukan koefisien matrik A, lakukan proses normalisasi pada titik-titik yang berkesesuaian pada dua buah foto secara terpisah, dengan langkah sebagai berikut,
   a. Tentukan nilai matrik translasi dari titik-titik pada foto pertama, yang direpresentasikan dengan matrik

$$T_t = \begin{pmatrix} 1 & 0 & -m_x \\ 0 & 1 & -m_y \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{Persamaan (3.2)}$$

   dimana (m_x, m_y) adalah nilai rata-rata pengukuran titik-titik pada foto pertama. Yang dihitung dengan rumus :

$$m_x = \frac{(\Sigma x)}{n} \text{ dan } m_y = \frac{(\Sigma y)}{n} \qquad \text{Persamaan (3.3)}$$

   dan n adalah jumlah titik-titik pengukuran pada foto pertama, Σx dan Σy adalah jumlah dari koordinat x dan y dari titi-titik hasil pengukuran.

   b. Kemudian hitung nilai skala dari titik-titik hasil pengukuran sehingga titik-titik tersebut memiliki rata-rata panjang sama dengan $\sqrt{2}$ dari origin, yang direpresentasikan dengan matrik

$$T_s = \begin{pmatrix} \frac{\sqrt{2}}{d} & 0 & 0 \\ 0 & \frac{\sqrt{2}}{d} & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{Persamaan (3.4)}$$

dimana *d* adalah rata-rata panjang dari pengukuran setelah nilai (m_x, m_y) di tentukan.

c. Final transformasi adalah $T=T_t*T_s$, transformasikan titik pada foto pertama dengan persamaan $\hat{x} = Tx$

d. Lakukan juga proses normalisai pada foto kedua (*x'*) dan tentukan nilai transformasi *T'* dengan melakukan hal yang sama seperti langkah a sampai c. kemudian transformasikan titik-titik pada foto kedua dengan $\widehat{x'} = T'x'$.

2. Dengan menggunakan titik-titik berkesesuaian yang sudah dinormalisasi $(\hat{x} \leftrightarrow \hat{x}')$ susun parameter matrik koefisien *A* seperti yang dijelaskan pada persamaan (2.13) yang sudah dibahas pada bab dua. Untuk lebih memudahkan maka akan ditulis kembali sebuah susunan dari matrik A sebagai berikut :

$$\begin{pmatrix} 0 & 0 & 0 & -x_1 & -y_1 & -1 & y_1'x_1 & y_1'y_1 & y_1'w_1 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1'w_1 \\ -y_1'x_1 & -y_1'y_1 & -y_1'w_1 & x_1'x_1 & x_1'y_1 & x_1'w_1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -x_n & -y_n & -1 & y_n'x_n & y_n'y_n & y_1'w_1 \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_1 & -x_1'w_1 \\ -y_n'x_n & -y_n'y_n & -y_n'w_n & x_n'x_n & x_n'y_n & x_n'w_n & 0 & 0 & 0 \end{pmatrix}$$

Persamaan (3.5)

3. Sesuai dengan persamaan (2.11) hubungan antara koefisien *A* dan matrik *homography* dituliskan dalam bentuk Ah=0. Sehingga, solusi yang di dapat untuk matrik *h* pada persamaan (3.5) dipecahkan dengan metode SVD dimana nilai dari kolom

matrik V yang berkorespondensi terhadap nilai matrik singular S terkecil atau biasanya nilai matrik h berada pada kolom terakhir dari matrik V. Dimana elemen dari matrik h adalah elemen dari matrik $\widehat{H}$.

4. Nilai matrik H final adalah,

$$H = T'^{-1}\widehat{H}T \qquad \text{Persamaan (3.6)}$$

III.3.3.3 Proses *Image Blending*

1. Proses *blending* memanfaatkan matrik *homography (H)* yang sudah didapatkan untuk dapat memproyeksikan citra ke dua *(x)* terhadap citra pertama dengan persamaan :

$$X' = H * x \qquad \text{Persamaan (3.7)}$$

2. Hitung selisih nilai jarak antara piksel yang berada pada daerah yang bertampalan dengan pusat dari dua foto tersebut dan jarak piksel yang paling dekat dengan pusat foto. Untuk menghitung jarak antara dua buah titik piksel menggunakan persamaan (2.21) dan dapat dituliskan kembali sebagai berikut :

$$d = [(x-s)^2 + (y-t)^2]^{\frac{1}{2}} \qquad \text{Persamaan (3.8)}$$

3. Normalisasi nilai tiap bobot *(α)* yang didapat dari perhitungan jarak setiap piksel dengan persamaan :

$$\bar{d} = \frac{d_1}{N} \qquad \text{Persamaan (3.9)}$$

4. Setelah nilai bobot didapat, maka nilai piksel setiap warna pada citra hasil penggabungan dapat dihitung dengan menggunakan persamaan (2.19) yang dituliskan kembali sebagai berikut :

$$N(x,y) = \alpha I(x,y) + (1-\alpha)\, C(x,y) \qquad \text{Persamaan (3.10)}$$

5. Persamaan (3.10) dilakukan pada setiap nilai *byte* citra yang bertampalan untuk menghasilkan warna baru pada citra hasil pencampuran.

III.3.3 Pembuatan Fungsi untuk Aplikasi

Untuk membuktikan proses perhitungan, maka akan dibuat sebuah aplikasi *windows-form* sederhana menggunakan pemrograman bahasa C#. Program ini akan mewakili setiap tahapan perhitungan dalam pembuatan mozaik foto panorama. Hasil akhir dari tahapan perhitungan dan pengaplikasiannya ke dalam bahasa C# adalah berupa aplikasi *windows-form* sederhana dalam pembuatan mozaik foto panorama pada sepasang foto stereo.

BAB IV

HASIL DAN PEMBAHASAN

Proses analisa atau pembahasan dari sebuah hasil yang telah dicapai selama proses pelaksanaan penelitian akan mengetahui keberhasilan dari penelitian tersebut. Pada bab ini akan disampaikan mengenai hasil dan pembahasan dari pembuatan algoritma pembuatan mosaik foto panorama pada sepasang foto stereo.

IV.1 Hasil Penelitian

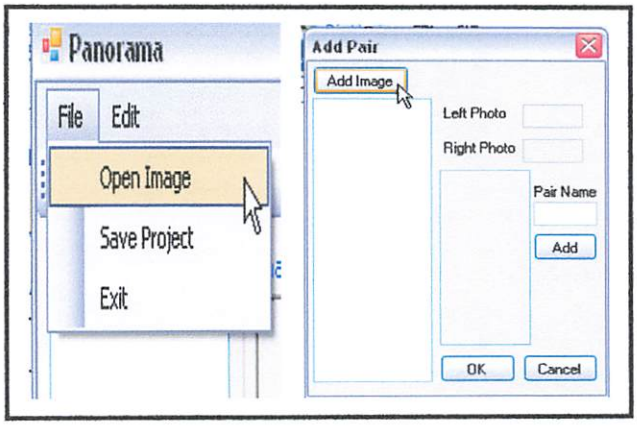
IV.1.1 Listing Kode Pemrograman dan Aplikasi

Algoritma penentuan titik konjugasi berdasarkan metode *area based* menggunakan teknik *Normalised Cross Correlation (NCC)* pada penelitian ini diadopsi dari peneliti yang telah melakukan penelitian dengan metode yang sama yaitu *Martince, N.B (2010)*. Untuk menguji kebenaran algoritma yang dibuat maka prosesnya adalah sebagai berikut:

1. Menampilkan dua buah foto stereo dalam *form*

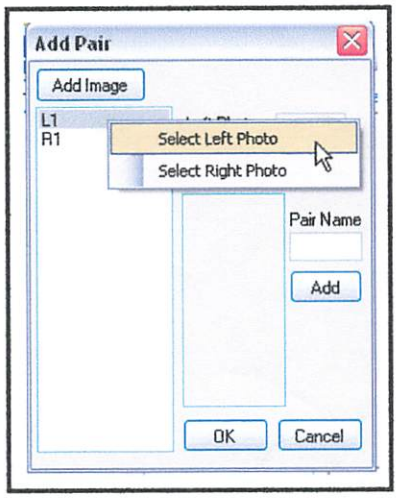
   Pada *form project* Panorama, pada menu bar pilih menu ***File → Open Image***

2. Kemudian akan muncul kotak dialog ***Add Pair***, pada *button click* ***Add Image***, *input*kan data foto dari direktori tempat penyimpanan foto.



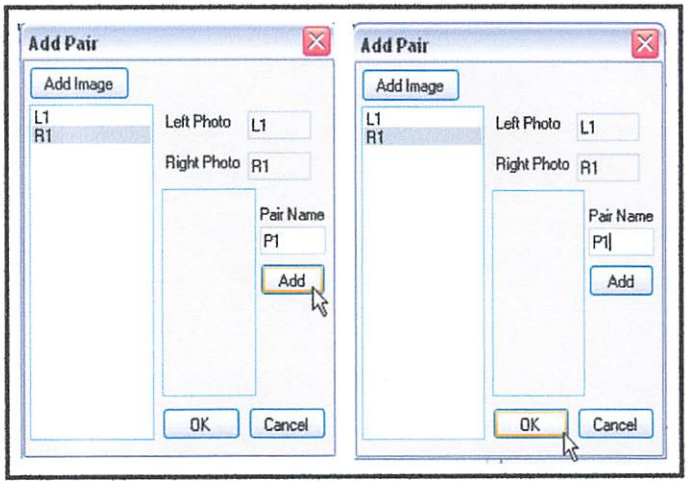
Gambar 4.1Menginputkan pasangan foto ke dalam project

3. *Input*kan foto dengan klik indeks foto kiri, kemudian ***Select Left Photo***, dan lakukan hal yang sama pada pasangan fotonya (foto kanan) dengan ***Select Right Photo***.



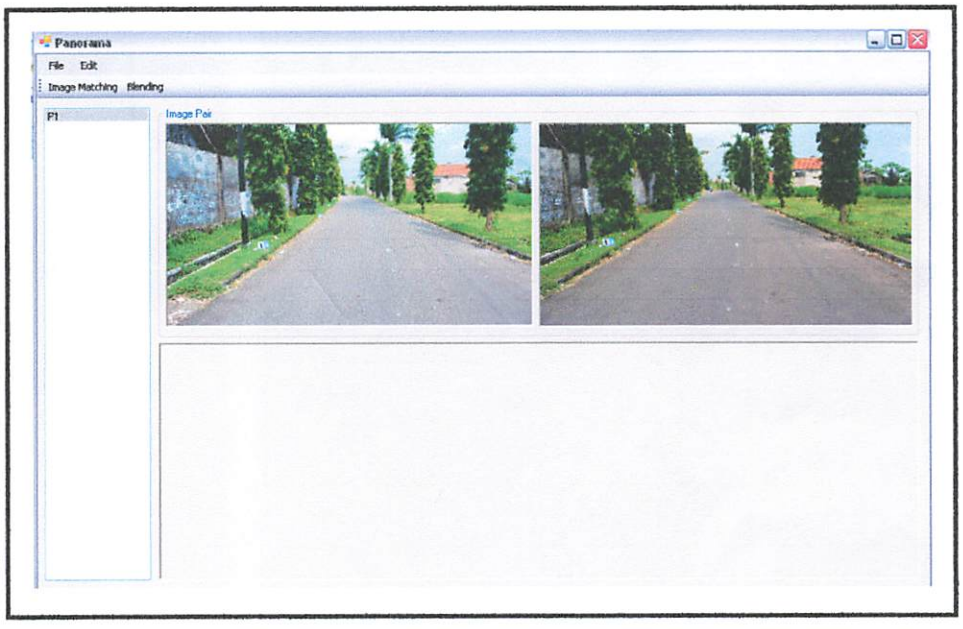
Gambar 4.2 Memasukkan file foto

4. Tuliskan nama pasangan foto yang sudah di*input* dalam *text box* ***Pair Name*** lalu klik *button* ***Add.*** Kemudian pilih **OK**.



Gambar 4.3 Menginputkan nama pasangan foto

5. *Double* klik pada pasangan foto yang ingin ditampilkan. Maka pada *form* akan ditampilkan dua foto stereo.



Gambar 4.4 Tampilan pasangan foto pada form

6. Pada *toolbar* pilih *icon* ***Image Matching***. Klik salah satu area pada foto pertama (foto kiri). Ketika salah satu titik difoto kiri diklik, maka secara otomatis sistem komputer akan merekan nilai keabuan disekitar posisi

yang dipilih dan pusat dari *template* berada pada posisi saat titik konjugasi dipilih.

Gambar 4.5 Penentuan Titik Konjugasi

Hasil perhitungan nilai koefisien menggunakan metode *normalized cross correlation* diatas diperoleh dari susunan algoritma yang ditampilkan dalam bahasa C#. berikut ini adalah algoritmanya :

```
//construct matching patch
Debug.Assert(matching.Cols == _template.Cols && matching.Rows
== _template.Rows);
int elementNumbers = matching.Rows * matching.Cols;
//double avgGrayValMatchPatch =
matching.GetAverage().Intensity;
double sumGrayValMatchPatch = matching.GetSum().Intensity;
double sumSquaredGrayValMatchPatch = SetSumSquare(ref
matching);

double SumMult = SumMultiply(ref matching);
//Trace.WriteLine("SumMult: " + SumMult.ToString());

//From Element of Photogrammetry, Wolf page 336:
//Sx=template, Sy=matchingPatch
double SxSquared = _sumSquaredValTemplate -
((_sumGrayValTemplate * _sumGrayValTemplate) / elementNumbers);
```

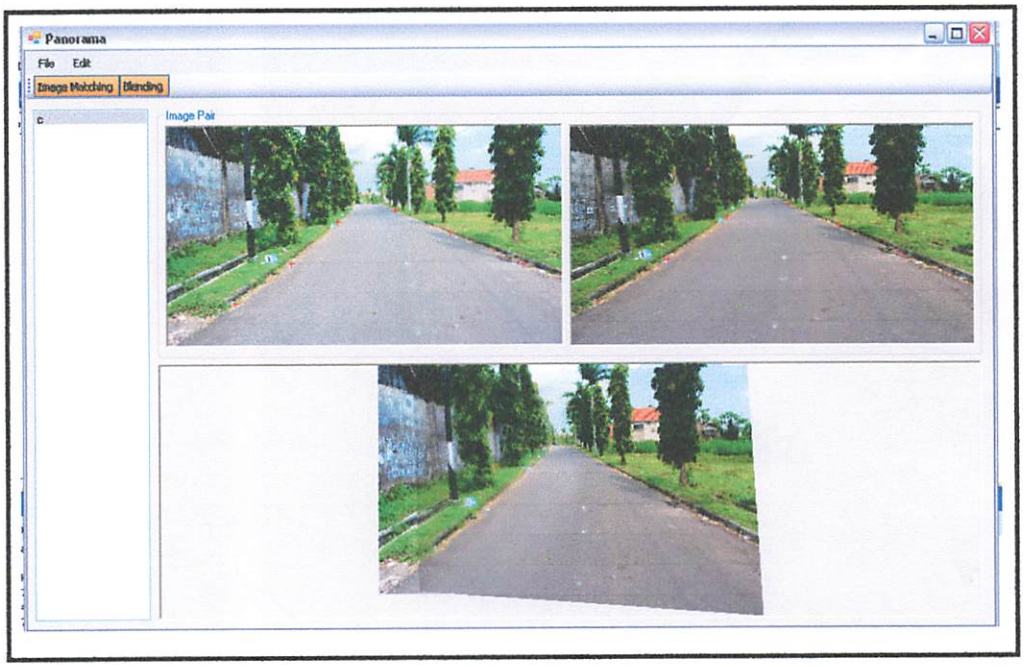
```
double SySquared = sumSquaredGrayValMatchPatch -
((sumGrayValMatchPatch * sumGrayValMatchPatch) /
elementNumbers);
double Sxy = SumMult - ((_sumGrayValTemplate *
sumGrayValMatchPatch) / elementNumbers);

//beta = Sxy / SxSquared;
//alpha = (sumGrayValMatchPatch / elementNumbers) - (beta *
(_sumGrayValTemplate / elementNumbers));
double CorrCoef = Sxy / Math.Sqrt(SxSquared * SySquared);
return CorrCoef;
```

algoritma secara lengkap dapat dilihat pada lampiran A.

7. Pada proses *image stitching* (penggabungan foto) menggunakan perhitungan *homography* ini akan membutuhkan nilai-nilai koordinat *pixel*. Nilai koordinat *pixel* ini diperoleh dari proses *normalized cross correlation.*

8. Pada *form* yang dibuat di penelitian ini penghitungan matrik *homography* dan proses *image blending*nya disatukan pada *button* ***Blending***. Setelah titik-titik konjugasi ditentukan dengan jumlah minimal 4 buah pasang titik, klik *button* ***Blending***. Maka secara otomatis pasangan foto stereo tersebut akan menyatu dan ditampilkan pada *imagebox* di *form*.



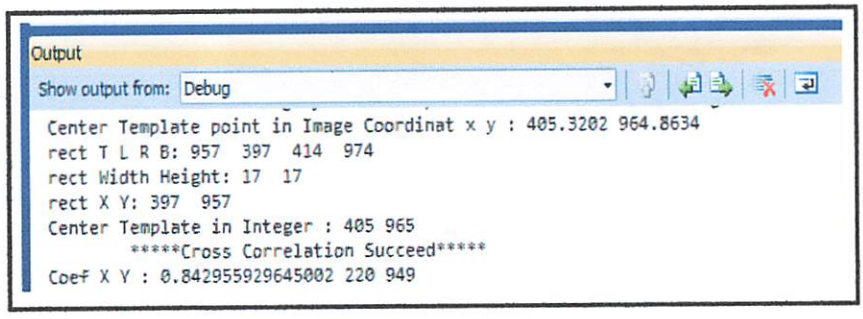
Gambar 4.6 Mosaik foto panorama

Algoritma yang digunakan dalam perhitungan matrik *homography* dan proses *image blending* ini keduanya diadopsi dari algoritma yang disusun oleh *Souza (2010)* yang dapat dilihat secara lengkap pada **Lampiran B dan Lampiran C**.

IV.1.2 Hasil Perhitungan

IV.1.2.1 Penentuan Titik Konjugasi

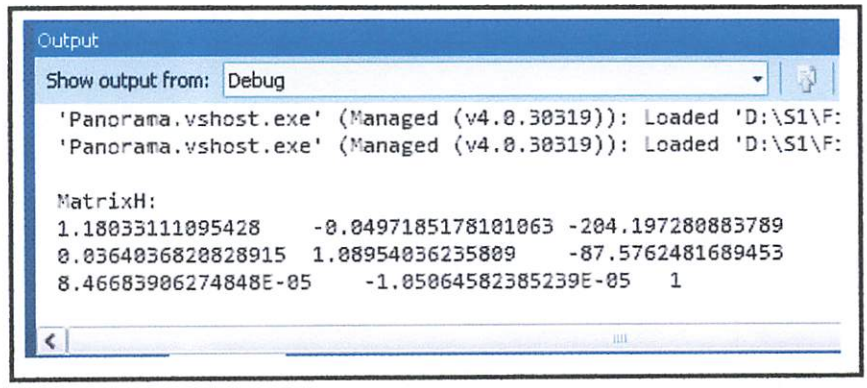
Pada saat *user* (pengguna) meng-klik suatu titik pada foto kiri maka secara otomatis sistem komputer akan membentuk *template* pada foto kiri dengan ukuran 17 x 17 *pixel*, dan pada foto kanan akan membentuk *matching window* dengan ukuran yang sama dengan *template* serta *search window* dengan ukuran 21 x 21 *pixel*. Kemudian secara otomatis akan menghitung nilai korelasi antara *template* dan setiap penelusuran *matching window* dalam *search window*. Hasil dari perhitungan korelasi antara kedua foto tersebut adalah :



Gambar 4.7 Hasil perhitungan cross correlation dengan teknik normalized cross correlation

IV.1.2.2 Parameter Matrik *Homography*

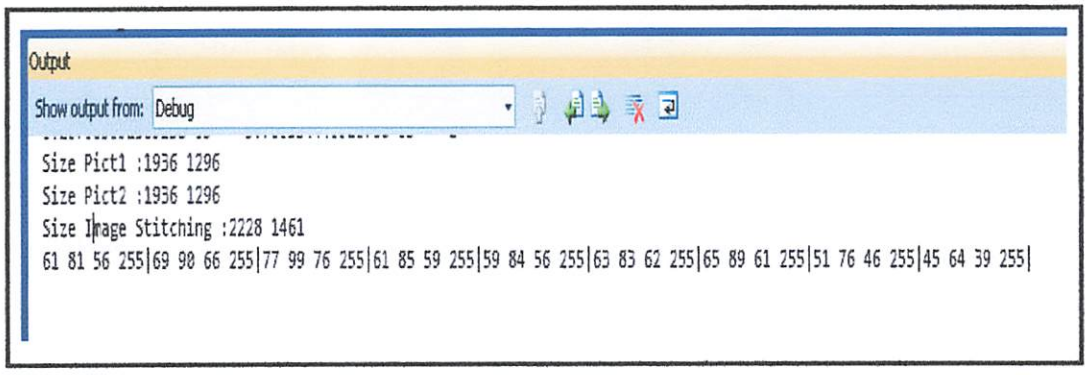
Dengan menggunakan data foto yang ditampilkan di *form* aplikasi pada Gambar 4.4 dilakukan proses perhitungan matrik *homography* dengan menggunakan algoritma yang ada di Lampiran B, maka di dapat sebuah nilai dari matrik *homography* sebagai berikut :



Gambar 4.8 Hasil perhitungan matrik homography

IV.1.2.3 Proses *Image Blending*

Dari proses *image blending* akan dihasilkan bobot setiap piksel pada daerah yang bertampalan. Nilai bobot ini akan digunakan pada perhitungan final untuk mendapatkan besarnya nilai kontribusi setiap piksel pada daerah yang bertampalan. Proses *image blending* ini dapat dilakukan dengan menggunakan persamaan 2.19- 2.21. dengan menggunakan algoritma yang ada dilampiran C, maka di dapat nilai setiap piksel *(Red (R), Green(G), Blue(B), Alpha(A))* pada daerah yang bertampalan (daerah *sample* 3x3 piksel) :



Gambar 4.9 Hasil perhitungan Nilai setiap piksel pada algoritma image blending menggunakan linear gradient alpha blending

Dari gambar 4.9 dapat diketahui nilai setiap piksel dari *sample* daerah yang bertampalan, yaitu :

Nilai Setiap Piksel Dari Sample Daerah Yang Bertampalan											
Piksel (1,1)				**Piksel (1,2)**				**Piksel (1,3)**			
R	G	B	A	R	G	B	A	R	G	B	A
61	81	56	255	69	90	66	255	77	99	76	255
Piksel (2,1)				**Piksel(2,2)**				**Piksel (2,3)**			
R	G	B	A	R	G	B	A	R	G	B	A
61	85	59	255	59	84	56	255	63	83	62	255
Piksel (3,1)				**Piksel(3,2)**				**Piksel(3,3)**			
R	G	B	A	R	G	B	A	R	G	B	A
65	89	61	255	51	76	46	255	45	64	39	255

Tabel 4.1 Nilai RGBA setiap piksel

IV.2 Pembahasan

IV.2.1 Algoritma *Normalized Cross Ccorrelation*

Pada pembuatan mosaik foto panorama pada penelitian ini, diawali dengan pemilihan titik-titik konjugasi secara manual, *user* (pengguna) memilih sendiri titik-titik konjugasi pada foto yang bertampalan dengan bantuan *mouse*. Dengan teknik *normalized cross correlation* dan menggunakan data foto yang dipakai pada penelitian ini (*sample* Gambar 4.4). Nilai korelasi antara titik konjugasi di foto kiri yang memiliki koordinat piksel **405,965** adalah **0.842955929645002**

Nilai korelasi ini menunjukkan suatu nilai korelasi positif yang mendekati nilai 1 yang merupakan korelasi terbaik. Oleh karena itu koordinat titik konjugasi yang didapatkan pada foto kanan yang bernilai **220,949** dapat dianggap memiliki posisi yang sama.

IV.2.2 Algoritma Matrik *Homography*

Titik-titik konjugasi yang sudah ditentukan secara manual, kemudian digunakan dalam proses perhitungan penentuan 8 parameter *unknown* matrik *homography*. Hasil perhitunga matrik *homography* pada penelitian ini disajikan pada gambar 4.8, yaitu :

8 Paremater Matrik *Homography*			
h1	**h2**	**h3**	**h4**
1.18033	-0.04971	-204.1972	0.0364
h5	**h6**	**h7**	**h8**
1.0895	-87.5762	8.4668	-1.0506

Tabel 4.2 Nilai 8 parameter matrik Homography

Kedelapan elemen matrik *homography* ini dapat didekompisisi untuk mengetahui elemen transformasi apa saja yang ada didalamnya. Namun karena batasan penelitian ini hanya sampai batas mendapatkan matrik *homography*nya saja, proses dekompisisi matrik tersebut tidak akan dijelaskan.

Karena dalam penelitian ini titik-titik konjugasi ditentukan secara manual seturut dengan kehendak *user*. Maka bisa saja titik-titik konjugasi pada daerah yang bertampalan dipilih secara berbeda dengan jumlah dan distribusi titik yang berbeda pula. Apabila *user* memilih sekumpulan titik-titik konjugasi kemudian melakukan proses perhitungan matrik *homography*, dan dilain waktu dengan menggunakan foto yang sama memilih sekumpulan titik yang berbeda, maka

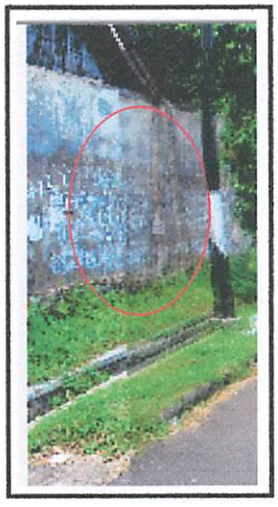
matrik *homographynya* bias saja akan berbeda dengan matrik *homography* yang terdahulu. Hal ini dikarenakan bahwa perhitungan matrik *homography* bergantung pada titik-titik koordinat antara dua buah bidang proyeksi (bidang foto). Selain itu pemilihan titik-titik yang lebih banyak dari jumlah minimal akan menghasilkan matrik *homography* yang lebih baik pula.

Kembali me*rivew* persamaan yang digunakan untuk menentukan matrik *homography* pada persamaan 2.6. Dari persamaan ini dapat dilihat nilai matrik *homography* dibentuk oleh pasangan titik yang berkesesuaian pada foto satu dan foto dua, ketika matrik *homography* tersebut sudah didapatkan maka matrik *homography* hanya akan mentransformasi titik-titik yang membentuknya. Oleh karena itu jika pemilihan titik-titik konjugasi hanya berada pada sebagian sisi dari foto yang bertampalan, matrik *homography* hanya akan mentransformasikan titik-titik yang berada pada sebagian sisi foto saja. Karena itu pemilihan distribusi titik-titik konjugasi juga akan mempengaruhi hasil dari matrik *homographyi*nya dan juga akan berpengaruh pada kualitas penyatuan kedua foto. Titik-titk yang lebih terdistribusi pada daerah foto yang bertampalan akan menghasilkan foto penggabungan yang lebih baik.

IV.2.3 Algoritma *Gradient Alpha Blending*

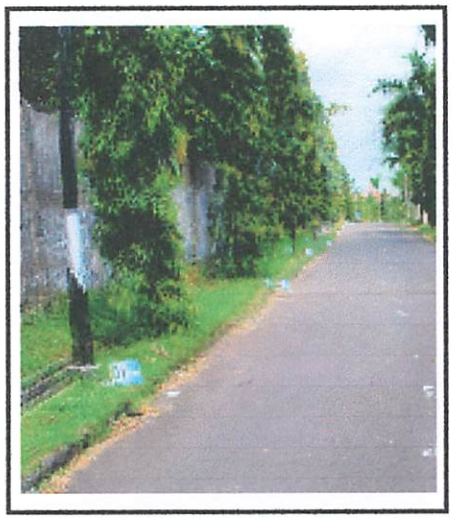
Seperti yang telah dijelakan pada bab 2 algoritma *gradient image blending* ini dipakai dengan maksud untuk meminimalisir atau bahkan menghilangkan masalah adanya perbedaan antara foto 1 dan foto 2 ketika sudah digabungkan (*visible seam*), daerah yang kabur *(blur)*, dan bayangan *(ghosting)*. Namun pada

penelitian masih terdapat adanya perbedaan tersebut ditunjukkan oleh gambar 4.10 ini,



Gambar 4.10 Seam yang masih terlihat

dan daerah yang kabur yang ditunjukkan pada gambar di bawah ini :



Gambar 4.11 Daerah blurr

Perbedaan antara foto yang digabungkan (*visible seam*) diakibatkan perbedaan intensitas cahaya, *blurring* yang terjadi karena kesalahan registrasi dan adanya bayangan (*ghosting*) karena pergerakan objek pada saat perekaman citra.

Seperti yang ditampilakan pada gambar 4.9 dengan algoritma ini juga hasil foto panorama yang dihasilkan dari foto pertama yang berukuran 1936x1296 piksel dan foto dua yang berukuran 1936x1296 adalah foto berukuran 2228x1461 piksel.

IV. 3 Kelebihan dan Kelemahan Penelitian

Dalam penelitian pembuatan mosaik foto panorama dengan metode-metode yang telah dilakukan terdapat beberapa kekurangan dan kelebihan. Di bawah ini akan diuraikan kekurangan dan kelebihan pada penelitian ini :

1. Kelebihan
   a. Penentuan titik konjugasi secara manual dianggap sudah memiliki titik yang pasti (tidak memiliki kesalahan), maka dapat dilakukan perhitungan matrik *homography* secara langsung.
2. Kelemahan
   a. Karena pemilihan titik konjugasi ditentukan secara manual maka membutuhkan waktu yang lebih lama.
   b. Ketelitian matrik *homography* tergantung pada pemilihan jumlah dan distribusi titik-titik konjugasi. Matrik *homography*nya akan semakin teliti dengan nilai titik konjugasi yang lebih banyak dan lebih terdistribusi.
   c. Proses *image blending* yang dilakukan pada setiap piksel pada foto yang bertampalan dengan metode yang sudah dilakukan terhadap penelitian ini, hanya dilakukan mulai dari pusat kedua foto. Karena itu

apabila foto yang bertampalan lebih dari 50% maka masih terdapat daerah yang belum di kenai proses *blending.*

BAB V

KESIMPULAN DAN SARAN

V.I Kesimpulan

Kesimpulan yang dapat diambil dari penelitian dengan judul "**Desain Algoritma Dalam Pembuatan Mozaik Foto Panorama pada Sepasang Foto Stereo**" ialah sebagai berikut :

1. Pada penelitian ini, hasil dari *normalized cross correlation* sudah sangat identik, hal ini dibuktikan dengan nilai korelasi setiap titik konjugasi yang didapatkan rata-rata berada di atas 0.75.
2. Titik-titik konjugasi yang lebih menyebar dan jumlah titik yang lebih banyak akan menghasilkan gambar mosaik yang lebih baik.
3. Pada proses *image blending* dengan metode *linear gradient alpha blending.* masih terlihat adanya perbedaan garis tepi antara kedua foto, dan daerah yang *blurr.*

V.I Saran

1. Bagi peneliti yang ingin melanjutkan penelitian ini, untuk mendapatkan hasil yang maksimal hendakanya menambahkan parameter kalibrasi kamera pada saat perhitungan matrik *homographynya.*
2. Untuk mendapatkan hasil dari proses *image blending* yang lebih baik, maka dianjurkan untuk mencoba menggunakan metode lain.

DAFTAR PUSTAKA

Atkinson, K.B, 2001, ***Close Range Photogrametry and Machine Vision***, Whittles Publishing. Scotland, UK.

Belo, 2009. ***An Introduction to Robotic Vision***. http://web.me.com/felipebelo/page9/page12/assets/Robotic%20Vision%20Handout.pdf [online access 18 Juli 2011].

Brown, M., and Lowe, D., 2003. ***Recognising Panoramas***. In: Proc.Int.Conf.Computer Vision, Vol.2, pp.1218-1225.

Estrada. F. J et all, 2004., ***Planar Homographies***. Catatan kuliah, http://www.cs.utoronto.ca/~strider/vis-notes/tutHomography04.pdf [online access.

Gonzalez, R. C. and Woods, R. E. 2008. ***Digital Image Processing***, *3rd ed.*, Prentice Hall, Upper Saddle River, NJ.

Hartley, R. & Zisserman, A. 2003. ***Multiple View Geometry in Computer Vision 2nd Edition***. Cambridge University Press. Cambridge.

Heuel. S, 2005. ***Uncertain Projective Geometry***, Springer, United State of America.

Huang, F. Klette, R. Scheibe, K. 2008. ***Panoramic Imaging Sensor-Line Cameras and Laser Range-Finder***. A John Wiley and Son Ltd. Inggris.

Karara, H. M. 1989. ***Non-Topographic Photogrammetry : Second Edition***, American Society for Photogrammetry and Remote Sensing.

Kriegman.D, 2007. ***Homography Estimation,*** *CSE 252A. http://cseweb.ucsd.edu/classes/wi07/cse252a/homography_estimation/homography_estimation.pdf, Catatan Kuliah.[online access* 18 Juli 2011].

Kwon. Y.H., 1998, ***DLT Method.*** http://www.kwon3d.com/theory/dlt/dlt.html [online access 18 Juli 2011].

Mikhail, E., Bethel, J., and McGlone, J., 2001. ***Introduction to Modern Photogrammetry.*** John Wiley & Sons Inc., New York.

Novianti. M, 2010, ***Studi Penentuan Titik Konjugasi Pada Foto Yang Bertampalan (Image Matching) Menggunakan Metode Area-Based Matching Pada Foto Stereo***, Jurusan Teknik Geodesi, Institut Teknologi Nasional, Malang.

Pontinen. P, 2000, ***On The Creation of Panoramic Images From Image Sequences. International*** Archives of Photogrametry and Remote Sensing. Vol.XXXIII, Part B5. Amsterdam.

Porter.T and Duff. T, 1984, ***Compositing Digital Images***. Proceeding of ACM SIGGRAPH 84, ACM Computer Graphic., pp.253-259.

Rankov. V et all., 2005, ***An Algorithm for Image Stitching and Blending***. Proceeding of SPIE volume 570.

Rahman. Arif., 2009. ***Sistem Temu Balik Citra Menggunakan Jarak Histogram dalam Model Warna YIQ.*** Seminar Nasional Aplikasi Teknologi Informasi. Yogyakarta.

Remondino. F. 2002, 3D ***Reconstruction of Articulated Objects From Uncalibrated Images***, Three-Dimensional Image Capture and Aplication V, SPIE Electronic Imaging, Proc,of SPIE 4661, San Jose, USA.

Schenk, T., 1999. ***Digital Photogrammetry,*** TerraScience, Ohio, USA.

Shum.H.Y and Szeliski.R,_____, ***Panoramic Image Mosaic***, Technical Report Microsoft Research, http://www.research.microsoft.com/ (diakses tanggal 18 Agustus 2011).

Szeliski. R., 2006. ***Image Alignment and Stitching : A Tutorial.*** Microsoft Research, USA.

Szeliski. R., 2011., ***Computer Vision Algorithms and Applications***. Eds : David Gries and Fred B.Schneider, Springer, New York.

Sofwan. L. M, 2002, ***Uji Ketelitian Teknik Pembuatan Mosaik Foto Udara Dengan Menggunakan Perangkat Lunak PCI Orthoengine V6.3 (Sk : Likupang, Sulawesi Utara)***, Jurusan Teknik Geodesi, Institut Teknologi Nasional, Malang.

Souza. Cesar., 2009-2010., http://www.codeproject.com/KB/recipes/automatic_ panoramas.aspx (accord.net, *online access* 7 Juli 2011).

Wikipedia, 2011.,***Normalization (Image Processing)*** http://en.wikipedia.org/wiki/Normalization_%28image_processing%29 (Diakses tanggal 13 Oktober 2011).

Wolf , P.R., and Dewitt, B. A., 2000. ***Element of Photogrammetry with Application in GIS 3rd***, McGraw-Hill Higher Education.pp 334-341.New York.

LAMPIRAN

LAMPIRAN A

ALGORITMA *NORMALISED CROSS CORRELATION*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.IO;
using System.Drawing;

using Emgu.CV;
using Emgu.Util;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;

using Accord.Imaging;

namespace Panorama
{
    public class CCorr
    {
        #region Fields
        double _correlationCoeficient;
        PointH _matchedPoint;
        PointH centreTempl;
        Image<Gray, Byte> _template;
        double _sumGrayValTemplate;
        double _sumSquaredValTemplate;
        //PointF[] Points2;
        #endregion

        #region Constructors
        public CCorr()
        {
            _correlationCoeficient = 0.0;
            _matchedPoint = new PointH();
            _sumGrayValTemplate = 0.0;
            _sumSquaredValTemplate = 0.0;

        }

        public CCorr(ref Image<Gray, Byte> source, ref Image<Gray, Byte>
        matchingImg, PointH centerTemplate)
            : this()
        {
       //Trace.WriteLine("\t\t------------------- Point at LeftPhoto ------
       ---------------------------------");
            //Perform EMGU template matching:
            //1. create template patch based on its centerTemplate point
            Rectangle rectTemplate = new Rectangle((int)((centerTemplate.X+
        0.5) - 10), (int)((centerTemplate.Y + 0.5) - 10), 21, 21);
            Trace.WriteLine("rect T L R B: " + rectTemplate.Top.ToString() +
        "   " + rectTemplate.Left.ToString() + "   " +
```

```
                    rectTemplate.Right.ToString() + "   " +
    rectTemplate.Bottom.ToString());
    Trace.WriteLine("rect Width Height: " +
    rectTemplate.Width.ToString() + "   " +
    rectTemplate.Height.ToString());
    Trace.WriteLine("rect X Y: " + rectTemplate.X.ToString() + "   "
    + rectTemplate.Y.ToString());

    //centre template in integer
      centreTempl = new PointH((int)(centerTemplate.X + 0.5),
      (int)(centerTemplate.Y + 0.5));
    Trace.WriteLine("Center Template in Integer : " +
    centreTempl.X.ToString() + " " + centreTempl.Y.ToString());
    _template = source.Copy(rectTemplate);
    _template.ROI.Offset(rectTemplate.X, rectTemplate.Y);
    _sumGrayValTemplate = _template.GetSum().Intensity;
    _sumSquaredValTemplate = SetSumSquare(ref _template);

    //2.Perform CrossCorrelation/ Template Matching using Emgu
      Image<Gray, float> temp = matchingImg.MatchTemplate(_template,
      TM_TYPE.CV_TM_CCOEFF_NORMED);
    double minVal = 0.0;
    double maxVal = 0.0;
    Point minLoc = new Point();
    Point maxLoc = new Point();
      CvInvoke.cvMinMaxLoc(temp, ref minVal, ref maxVal, ref minLoc,
      ref maxLoc, IntPtr.Zero);
    //Result:
    _matchedPoint.X = maxLoc.X + 10;
    _matchedPoint.Y = maxLoc.Y + 10;
    _matchedPoint.W = 1;

    //Perform Photogrammetric CrossCorrelation
  //  PhotogrammetricNormalisedCrossCorrelation(ref matchingImg);

}
#endregion

#region Properties
public double CorrelationCoeficient
{
    get
    {
        return _correlationCoeficient;
    }
}

public PointH MatchedPoint
{
    get
    {
        return _matchedPoint;
    }
}

public PointH CenterTemplate
{
    get
    {
```

```
        return centreTempl;
    }
}

#endregion

#region Methods
 public bool PhotogrammetricNormalisedCrossCorrelation(ref
Image<Gray, Byte> matchingImg)
{
    //create a searching window which size of 15x15
    PointH initialMatchedPt = new PointH ();
    initialMatchedPt.X = _matchedPoint.X;
    initialMatchedPt.Y = _matchedPoint.Y;
    initialMatchedPt.W = 1;
    double maxCorrVal = 0.750;
    bool result = false;
for (float row = initialMatchedPt.Y - 8; row <= initialMatchedPt.Y
+ 8; row++)
    {
                for (float col = initialMatchedPt.X - 8; col <=
initialMatchedPt.X + 8; col++)
        {
                    Image<Gray, Byte> matchingPatch =
SetMatchingWindow(ref matchingImg, col, row); //create matching
patch
            double temp = GetCorrelationCoef(ref matchingPatch);
            if (temp >= maxCorrVal)
            {
                _matchedPoint.X = col;
                _matchedPoint.Y = row;
                maxCorrVal = temp;
                result = true;
            }
        }
    }
    _correlationCoeficient = maxCorrVal;
    return result;
}

private double SumMultiply(ref Image<Gray, Byte> img)
{
    double sum = 0.0;
  Debug.Assert(img.Cols == _template.Cols && img.Rows == img.Rows);
    for (int y = 0; y < img.Rows; y++)
    {
        for (int x = 0; x < img.Cols; x++)
        {
            sum += _template[y, x].Intensity * img[y, x].Intensity;
        }
    }
    return sum;
}

private double SetSumSquare(ref Image<Gray, Byte> img)
{
    int row = img.Rows;
    int col = img.Cols;
    double sumSquared = 0;
```

```
    for (int y = 0; y < row; y++)
    {
        for (int x = 0; x < col; x++)
        {
            Emgu.CV.Structure.Gray g = img[y, x];
            sumSquared += (g.Intensity * g.Intensity);
        }
    }

    return sumSquared;
}

private Image<Gray, Byte> SetMatchingWindow(ref Image<Gray, Byte>
matchingImg, float x, float y)
{
    int X;
    int Y;
    X = (int)x;
    Y = (int)y;
    Rectangle rect = new Rectangle(X - 10, Y - 10, 21, 21);
    Image<Gray, Byte> matchingPatch = matchingImg.Copy(rect);
    return matchingPatch;
}

private double GetCorrelationCoef(ref Image<Gray, Byte> matching/*,
out double alpha, out double beta*/)
{
    //construct matching patch
Debug.Assert(matching.Cols == _template.Cols && matching.Rows ==
_template.Rows);
    int elementNumbers = matching.Rows * matching.Cols;
//double avgGrayValMatchPatch = matching.GetAverage().Intensity;
    double sumGrayValMatchPatch = matching.GetSum().Intensity;
double sumSquaredGrayValMatchPatch = SetSumSquare(ref matching);

    double SumMult = SumMultiply(ref matching);
    //Trace.WriteLine("SumMult: " + SumMult.ToString());

    //From Element of Photogrammetry, Wolf page 336:
    //Sx=template, Sy=matchingPatch
double SxSquared = _sumSquaredValTemplate - ((_sumGrayValTemplate *
_sumGrayValTemplate) / elementNumbers);
    double SySquared = sumSquaredGrayValMatchPatch -
((sumGrayValMatchPatch * sumGrayValMatchPatch) / elementNumbers);
    double Sxy = SumMult - ((_sumGrayValTemplate *
sumGrayValMatchPatch) / elementNumbers);

    //beta = Sxy / SxSquared;
//alpha = (sumGrayValMatchPatch / elementNumbers) - (beta *
(_sumGrayValTemplate / elementNumbers));
    double CorrCoef = Sxy / Math.Sqrt(SxSquared * SySquared);
    return CorrCoef;

}

//private void PrintPatch(ref Image<Gray, Byte> img)
//{
//    Trace.WriteLine("Patch gray value:");
//    int row = img.Rows;
```

```
            //    int col = img.Cols;
            //    for (int y = 0; y < row; y++)
            //    {
            //        for (int x = 0; x < col; x++)
            //        {
            //            Trace.Write(img[y, x].Intensity.ToString() + "  ");
            //        }
            //        Trace.WriteLine("");
            //    }
            //}
            #endregion

        }
    }
```

LAMPIRAN B

ALGORITMA PERHITUNGAN MATRIK *HOMOGRAPHY*

```
/// <summary>
///    Creates an homography matrix matching points
///    from a set of points to another.
/// </summary>
      public static MatrixH Homography(PointF[] points1, PointF[]
      points2)
      {
     // Initial argument checkings
     if (points1.Length != points2.Length)
       throw new ArgumentException("The number of points should be
       equal.");

     if (points1.Length < 4)
       throw new ArgumentException("At least four points are required
       to fit an homography");


     int N = points1.Length;

     MatrixH T1, T2; // Normalize input points
     points1 = Tools.Normalize(points1, out T1);
     points2 = Tools.Normalize(points2, out T2);

     // Create the matrix A
     double[,] A = new double[3 * N, 9];
     for (int i = 0; i < N; i++)
     {
         PointF X = points1[i];
         double x = points2[i].X;
         double y = points2[i].Y;
         int r = 3 * i;

         A[r, 0] = 0;
         A[r, 1] = 0;
         A[r, 2] = 0;
         A[r, 3] = -X.X;
         A[r, 4] = -X.Y;
         A[r, 5] = -1;
         A[r, 6] = y * X.X;
         A[r, 7] = y * X.Y;
         A[r, 8] = y;

         r++;
         A[r, 0] = X.X;
         A[r, 1] = X.Y;
         A[r, 2] = 1;
         A[r, 3] = 0;
         A[r, 4] = 0;
         A[r, 5] = 0;
         A[r, 6] = -x * X.X;
         A[r, 7] = -x * X.Y;
         A[r, 8] = -x;
```

```
            r++;
            A[r, 0] = -y * X.X;
            A[r, 1] = -y * X.Y;
            A[r, 2] = -y;
            A[r, 3] = x * X.X;
            A[r, 4] = x * X.Y;
            A[r, 5] = x;
            A[r, 6] = 0;
            A[r, 7] = 0;
            A[r, 8] = 0;
        }


        // Create the singular value decomposition
        SingularValueDecomposition svd = new
        SingularValueDecomposition(A, false, true);
        double[,] V = svd.RightSingularVectors;


        // Extract the homography matrix
        MatrixH H = new MatrixH((float)V[0, 8], (float)V[1, 8],
        (float)V[2, 8],
        (float)V[3, 8], (float)V[4, 8], (float)V[5, 8],
        (float)V[6, 8], (float)V[7, 8], (float)V[8, 8]);

        // Denormalize
        H = T2.Inverse().Multiply(H.Multiply(T1));

        return H;
    }



    /// <summary>
    ///   Normalizes a set of homogeneous points so that the origin is
    located
    ///   at the centroid and the mean distance to the origin is
    sqrt(2).
    /// </summary>
    public static PointH[] Normalize(this PointH[] points, out MatrixH T)
    {
        float n = points.Length;
        float xmean = 0, ymean = 0;
        for (int i = 0; i < points.Length; i++)
        {
            points[i].X = points[i].X / points[i].W;//mx
            points[i].Y = points[i].Y / points[i].W;//my
            points[i].W = 1;

            xmean += points[i].X;
            ymean += points[i].Y;
        }
        xmean /= n; ymean /= n;
        Trace.WriteLine("mean : " + xmean.ToString() + " " +
        ymean.ToString());
```

```
        float scale = 0;
        for (int i = 0; i < points.Length; i++)
        {
            float x = points[i].X - xmean;
            float y = points[i].Y - ymean;

            scale += (float)System.Math.Sqrt(x * x + y * y);
        }

        scale = (float)(SQRT2 * n / scale);
        Trace.WriteLine("Scale : "+scale.ToString());

        T = new MatrixH
            (
                scale, 0, -scale * xmean,
                0, scale, -scale * ymean,
                0, 0, 1
            );

        return T.TransformPoints(points);
    }
```

LAMPIRAN C

ALGORITMA PROSES *IMAGE BLENDING*

```
// Accord Imaging Library
// Accord.NET framework
// http://www.crsouza.com
//
// Copyright © César Souza, 2009-2010
// cesarsouza at gmail.com
//

namespace Accord.Imaging.Filters
{
    using System.Collections.Generic;
    using System.Drawing;
    using System.Diagnostics;
    using System.Drawing.Imaging;
    using AForge.Imaging;
    using System;
    using Matrix = Accord.Math.Matrix;


    /// <summary>
    ///   Linear Gradient Blending filter.
    /// </summary>
    /// <remarks>
    /// <para>
    ///   The blending filter is able to blend two images using a
        homography matrix.
    ///   A linear alpha gradient is used to smooth out differences between
        the two
    ///   images, effectively blending them in two images. The gradient is
        computed
    ///   considering the distance between the centers of the two
        images.</para>
    /// <para>
    ///   The first image should be passed at the moment of creation of the
        Blending
    ///   filter as the overlay image. A second image may be projected on
        top of the
    ///   overlay image by calling the Apply method and passing the second
        image as
    ///   argument.</para>
    /// <para>
    ///   Currently the filter always produces 32bpp images, disregarding
        the format
    ///   of source images. The alpha layer is used as an intermediate mask
        in the
    ///   blending process.</para>
    /// </remarks>
    public class Blend : AForge.Imaging.Filters.BaseTransformationFilter
    {
        private MatrixH homography;
        private Bitmap overlayImage;
```

```
private Point offset;
private Point center;
private Size imageSize;
private Color fillColor = Color.FromArgb(0, Color.Black);
private Dictionary<PixelFormat, PixelFormat> formatTranslations =
new Dictionary<PixelFormat, PixelFormat>();


/// <summary>
/// Format translations dictionary.
/// </summary>
public override Dictionary<PixelFormat, PixelFormat>
FormatTranslations
{
    get { return formatTranslations; }
}

/// <summary>
///   Gets or sets the Homography matrix used to map a image passed
    to
///   the filter to the overlay image specified at filter creation.
/// </summary>
public MatrixH Homography
{
    get { return homography; }
    set { homography = value; }
}

/// <summary>
///   Gets or sets the filling color used to fill blank spaces.
/// </summary>
/// <remarks>
///   The filling color will only be visible after the image is
    converted
///   to 24bpp. The alpha channel will be used internally by the
    filter.
/// </remarks>
public Color FillColor
{
    get { return fillColor; }
    set { fillColor = value; }
}

/// <summary>
///   Constructs a new Blend filter.
/// </summary>
/// <param name="homography">The homography matrix mapping a second
    image to the overlay image.</param>
/// <param name="overlayImage">The overlay image (also called the
    anchor).</param>
public Blend(double[,] homography, Bitmap overlayImage)
    : this(new MatrixH(homography), overlayImage)
{
}

/// <summary>
///   Constructs a new Blend filter.
/// </summary>
```

```
/// <param name="homography">The homography matrix mapping a second
    image to the overlay image.</param>
/// <param name="overlayImage">The overlay image (also called the
    anchor).</param>
public Blend(MatrixH homography, Bitmap overlayImage)
{
    this.homography = homography;
    this.overlayImage = overlayImage;
    formatTranslations[PixelFormat.Format8bppIndexed] =
    PixelFormat.Format32bppArgb;
    formatTranslations[PixelFormat.Format24bppRgb] =
    PixelFormat.Format32bppArgb;
    formatTranslations[PixelFormat.Format32bppArgb] =
    PixelFormat.Format32bppArgb;
}

/// <summary>
///   Computes the new image size.
/// </summary>
protected override Size CalculateNewImageSize(UnmanagedImage
    sourceData)
{
    // Calculate source size
    float w = sourceData.Width;
    float h = sourceData.Height;

    // Get the four corners and the center of the image
    PointF[] corners =
    {
        new PointF(0, 0),
        new PointF(w, 0),
        new PointF(0, h),
        new PointF(w, h),
        new PointF(w / 2f, h / 2f)
    };

    // Project those points
    corners = homography.Inverse().TransformPoints(corners);

    // Recalculate image size
    float[] px = { corners[0].X, corners[1].X, corners[2].X,
    corners[3].X };
    float[] py = { corners[0].Y, corners[1].Y, corners[2].Y,
    corners[3].Y };

    float maxX = Matrix.Max(px);
    float minX = Matrix.Min(px);
    float newWidth = Math.Max(maxX, overlayImage.Width) -
    Math.Min(0, minX);

    float maxY = Accord.Math.Matrix.Max(py);
    float minY = Accord.Math.Matrix.Min(py);
    float newHeight = Math.Max(maxY, overlayImage.Height) -
    Math.Min(0, minY);


    // Store overlay image size
    this.imageSize = new Size((int)Math.Round(maxX-minX),
    (int)Math.Round(maxY-minY));
```

```
        // Store image center
        this.center = Point.Round(corners[4]);

        // Calculate and store image offset
        int offsetX = 0, offsetY = 0;
        if (minX < 0) offsetX = (int)Math.Round(minX);
        if (minY < 0) offsetY = (int)Math.Round(minY);

        this.offset = new Point(offsetX, offsetY);

        // Return the final image size
        return new Size((int)Math.Ceiling(newWidth), (int)Math.Ceiling(newHeight));

    }


    /// <summary>
    ///   Process the image filter.
    /// </summary>
    protected override void ProcessFilter(UnmanagedImage sourceData, UnmanagedImage
        destinationData)
    {

        // Locks the overlay image
        BitmapData overlayData = overlayImage.LockBits(new Rectangle(0, 0,
        overlayImage.Width, overlayImage.Height),ImageLockMode.ReadOnly,
        overlayImage.PixelFormat);


        // get source image size
        int width = sourceData.Width;
        int height = sourceData.Height;

        // get destination image size
        int newWidth = destinationData.Width;
        int newHeight = destinationData.Height;

        int srcPixelSize =
        System.Drawing.Image.GetPixelFormatSize(sourceData.PixelFormat) / 8;
        int orgPixelSize =
        System.Drawing.Image.GetPixelFormatSize(overlayData.PixelFormat) / 8;

        int srcStride = sourceData.Stride;
        int dstOffset = destinationData.Stride - newWidth * 4; // destination always
        32bpp argb


        // Get center of first image
        Point center1 = new Point((int)(overlayImage.Width /
        2.0),(int)(overlayImage.Height / 2.0));

        // Get center of second image
        Point center2 = this.center;



        // Compute maximum center distances
        double dmax1 = Math.Min(
```

```
    distance(center1.X, center1.Y, center1.X + overlayImage.Width / 2.0,
center1.Y), distance(center1.X, center1.Y, center1.X, center1.X +
overlayImage.Height / 2.0));

double dmax2 = Math.Min(
    distance(center2.X, center2.Y, center2.X + imageSize.Width / 2.0,
center2.Y), distance(center2.X, center2.Y, center2.X, center2.Y +
imageSize.Height / 2.0));


double dmax = -System.Math.Abs(dmax2 - dmax1);

// fill values
byte fillR = fillColor.R;
byte fillG = fillColor.G;
byte fillB = fillColor.B;
byte fillA = 0;//fillColor.A;

// Retrieve homography matrix as float array
float[,] H = (float[,])homography;


// do the job
unsafe
{
    byte* org = (byte*)overlayData.Scan0.ToPointer();
    byte* src = (byte*)sourceData.ImageData.ToPointer();
    byte* dst = (byte*)destinationData.ImageData.ToPointer();

    // destination pixel's coordinate relative to image center
    double cx, cy;

    // destination pixel's homogenous coordinate
    double hx, hy, hw;

    // source pixel's coordinates
    int ox, oy;



    // Copy the overlay image
    for (int y = 0; y < newHeight; y++)
    {
        for (int x = 0; x < newWidth; x++, dst += 4)
        {
            ox = (int)(x + offset.X);
            oy = (int)(y + offset.Y);

            // validate source pixel's coordinates
              if ((ox < 0) || (oy < 0) || (ox >= overlayData.Width) || (oy
              >= overlayData.Height))
            {
                // fill destination image with filler
                dst[0] = fillB;
                dst[1] = fillG;
                dst[2] = fillR;
                dst[3] = fillA;
            }
            else
```

```
        {
            int c = oy * overlayData.Stride + ox * orgPixelSize;

            // fill destination image with pixel from original image
            dst[0] = org[c];

            if (orgPixelSize >= 3)
            {
                // 24/32 bpp
                dst[1] = org[c + 1];
                dst[2] = org[c + 2];
                dst[3] = (orgPixelSize == 4)
                    ? org[c + 3] // 32 bpp
                    : (byte)255; // 24 bpp
            }
            else
            {
                // 8 bpp
                dst[1] = org[c];
                dst[2] = org[c];
                dst[3] = 255;
            }
        }
    }
    dst += dstOffset;
}

org = (byte*)overlayData.Scan0.ToPointer();
src = (byte*)sourceData.ImageData.ToPointer();
dst = (byte*)destinationData.ImageData.ToPointer();

// Project and blend the second image
for (int y = 0; y < newHeight; y++)
{
    for (int x = 0; x < newWidth; x++, dst += 4)
    {
        cx = x + offset.X;
        cy = y + offset.Y;

        // projection using homogenous coordinates
        hw = H[2, 0] * cx + H[2, 1] * cy + H[2, 2];
        hx = (H[0, 0] * cx + H[0, 1] * cy + H[0, 2]) / hw;
        hy = (H[1, 0] * cx + H[1, 1] * cy + H[1, 2]) / hw;

        // coordinate of the nearest point
        ox = (int)(hx);
        oy = (int)(hy);

        // validate source pixel's coordinates
        if ((ox >= 0) && (oy >= 0) && (ox < width) && (oy < height))
        {
            int c = oy * srcStride + ox * srcPixelSize;

            // fill destination image with pixel from source image
            if (dst[3] > 0)
            {
                // there is a pixel from the other image here, blend
                double d1 = distance(ox, oy, center1.X, center1.Y);
                double d2 = distance(ox, oy, center2.X, center2.Y);
```

```
                double f = Accord.Math.Tools.Scale(0, dmax, 0, 1, d1 -
                d2);

                if (f < 0) f = 0;
                if (f > 1) f = 1;
                double f2 = (1.0 - f);

                dst[0] = (byte)(src[c] * f2 + dst[0] * f);

                if (srcPixelSize >= 3)
                {
                    // 24/32 bpp
                    dst[1] = (byte)(src[c + 1] * f2 + dst[1] * f);
                    dst[2] = (byte)(src[c + 2] * f2 + dst[2] * f);
                    dst[3] = (srcPixelSize == 4)
                        ? (byte)(src[c + 3] * f2 + dst[3] * f) // 32 bpp
                        : (byte)255;                           // 24 bpp
                }
                else
                {
                    // 8 bpp
                    dst[1] = (byte)(src[c] * f2 + dst[1] * f);
                    dst[2] = (byte)(src[c] * f2 + dst[2] * f);
                    dst[3] = (byte)255;
                }
            }
            else
            {
                // just copy the source into the destination image
                dst[0] = (byte)src[c];

                if (srcPixelSize >= 3)
                {
                    // 24/32bpp
                    dst[1] = (byte)src[c + 1];
                    dst[2] = (byte)src[c + 2];
                    dst[3] = (srcPixelSize == 4)
                        ? (byte)src[c + 3] // 32bpp
                        : (byte)255;       // 24bpp
                }
                else
                {
                    // 8bpp
                    dst[1] = (byte)src[c];
                    dst[2] = (byte)src[c];
                    dst[3] = (byte)255;
                }
                        }
                    }
                }
                dst += dstOffset;

            }
        }

        overlayImage.UnlockBits(overlayData);
    }

    /// <summary>
```

```
        ///   Computes a distance metric used to compute the blending mask
        /// </summary>
        private static double distance(double x1, double y1, double x2,
double y2)
        {
            // Euclidean distance
            double u = (x1 - x2);
            double v = (y1 - y2);
            return Math.Sqrt(u * u + v * v);
        }
    }
}
```