SKRIPSI

PEMBUATAN PAKET PROGRAM PENENTUAN KOORDINAT *OBJECT SPACE* TIGA DIMENSI DENGAN METODE INTERSEKSI

Disusun Oleh :

DESSY SAUDALE

NIM. 03.25.011

PROGRAM STUDI TEKNIK GEODESI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
INSTITUT TEKNOLOGI NASIONAL
MALANG

2010

LEMBAR PENGESAHAN

Judul Skripsi :

"Pembuatan Paket Program Penentuan Koordinat *Object Space* Tiga Dimensi Dengan Metode Interseksi"

Dipertahankan di hadapan Majelis Penguji Sidang Skripsi Jenjang Starata Satu (S-1) pada :

Hari : Sabtu

Tanggal : 21 Agustus 2010

Dan diterima untuk memenuhi salah satu persyaratan guna memperoleh gelar Sarjana Teknik.

Disusun Oleh :

Dessy Saudale

03.25.011

Panitia Ujian Tugas Akhir

Ketua	Sekretaris
Hery Purwanto, ST. M.Sc	**Silvester Sari Sai, ST. MT.**

Anggota Penguji

Penguji I	Penguji II	Penguji III
Hery Purwanto, ST. M.Sc	**Ir. Agus Darpono, MT.**	**Dr. Edwin Tjahjadi, ST. M.Geom.Sc**

JURUSAN TEKNIK GEODESI

FAKULTAS TEKNIK SIPIL DAN PERENCANAAN

INSTITUT TEKNOLOGI NASIONAL

MALANG

2010

LEMBAR PERSETUJUAN

SKRIPSI

Judul Skripsi :

"Pembuatan Paket Program Penentuan Koordinat *Object Space* Tiga Dimensi Dengan Metode Interseksi"

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh

Gelar Sarjana Teknik Geodesi S-1

Institut Teknologi Nasional Malang

Disusun Oleh :

Dessy Saudale

03.25.011

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Hery Purwanto, ST. M.Sc	**Dr. Edwin Tjahjadi, ST. M.Geom.Sc**

Mengetahui,

Ketua Jurusan Teknik Geodesi S-1

Hery Purwanto, ST. M.Sc

ABSTRAKSI

Interseksi merupakan suatu metode yang digunakan dalam penentuan koordinat *object space* tiga dimensi. Adapun pembuatan paket program dengan menyertakan metode interseksi adalah suatu kegiatan yang diharapkan dapat mempermudah proses penentuan koordinat *object space* tiga dimensi. Hal ini memungkinkan agar proses ini dapat berlangsung secara otomatis, efektif dan efisien.

Pembuatan paket program sendiri tidak terlepas dari suatu proses desain *interface*. Tujuannya adalah agar pengguna program nantinya akan lebih interaktif, efektif dan efisien dalam penggunaan program. Untuk itu dalam pembuatan paket program, *interface* program yang interaktif sangat diprioritaskan.

Adapun pembuatan paket program dapat menggunakan bahasa pemrograman C#. Dan berdasarkan paket program yang telah selesai dikerjakan, dapat dianalisa bahwa paket program sangat membantu dalam mempercepat prosesing data untuk penentuan koordinat *object space* khususnya pada pengolahan foto stereo. Manfaat lain adalah bahwa dengan paket program ini titik-titik konjugasi yang akan digunakan untuk interseksi dapat ditentukan secara otomatis dengan teknik digital. Selain itu adapun kekurangan dari paket program ini adalah nilai penyimpangan terhadap ketelitian hasil interseksi masih diatas strandar penyimpangan terhadap ketelitian yang dianggap baik yaitu masih diatas dari satu millimeter. Sangat diharapkan adanya perbaikan-perbaikan untuk meningkatkan ketelitian demi kesemprurnaan paket program ini.

Kata kunci : Intersection, ImageMatching, CrossCorrelation, LeastSquareMatching, MatrixLibrary, ComputerVision

PERNYATAAN KEASLIAN

SKRIPSI

Saya yang bertanda tangan dibawah ini.

Nama	**: Dessy Saudale**
NIM	**: 03.25.011**
Program Study	**: Teknik Geodesi S-1**
Fakultas	**: Fakultas Teknik Sipil Dan Perencanaan**

Menyatakan dengan sesungguhnya bahwa Skripsi saya dengan judul :

"Pembuatan Paket Program Penentuan Koordinat *Object Space* Tiga Dimensi Dengan Metode Interseksi"

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

Malang, 7 Juli 2011

Yang membuat pernyataan

Dessy Saudale

03.25.011

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa, karena berkah dan rahmat-Nya, penulis dapat menyelesaikan skripsi yang bejudul **"Pembuatan Paket Program Penentuan Koordinat *Object Space* Tiga Dimensi Dengan Metode Interseksi"**. Ada pun skripsi ini disusun sebagai salah satu syarat untuk meraih gelar Sarjana Teknik pada Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.

Selesainya penulisan skripsi ini tidak terlepas dari bantuan semua pihak, untuk itu pada kesempatan ini penulis ingin menyampaikan terimakasih yang sebesar-besarnya kepada :

1. Bapak Prof. Dr. Eng. Ir. Abraham Lomi, MSEE selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. A. Agus Santosa, MT. selaku Dekan Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.
3. Bapak Hery Purwanto, ST. M.Sc. selaku Ketua Jurusan Teknik Geodesi Institut Teknologi Nasional Malang, selaku dosen pembimbing I dan dosen penguji.
4. Bapak Dr. Edwin Tjahjadi, ST. MGeom.Sc. selaku Dosen Pembimbing II serta sebagai dosen penguji.
5. Bapak Agus Darpono, ST. MT. selaku dosen penguji.
6. Segenap dosen, staff pengajar dan recording Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.

7. Rekan-rekan Mahasiswa/i dan alumni Teknik Geodesi.

8. Semua pihak yang secara langsung mau pun tidak langsung turut membantu dalam proses penelitian mau pun penyusunan skripsi ini.

Penulis menyadari sepenuhnya bahwa penulisan skripsi ini masih jauh dari kesempurnaan, untuk itu penulis secara terbuka menerima kritik dan saran yang bersifat membangun. Penulis juga berharap agar hasil dari penelitian ini dapat bermanfaat baik untuk kalangan civitas akademik di jurusan teknik geodesi, untuk kalangan umum mau pun segala aspek yang menyangkut dunia pendidikan.

Malang, 7 Juli 2011

Dipersembahkan untuk semua yang terkasih, semua yang tak pernah menyerah, semua yang selalu teguh dalam segala kesulitan, semua yang selalu belajar dari setiap kegagalan dan untuk semua yang selalu ingin lebih baik di hari esok.

God bless you All.

Puji Syukur seutuhnya kepada **TUHAN** yang maha pengasih dan penyayang, yang sudah memberikan kehidupan, hikmad dan keindahan dalam segala kesenangan dan kesulitan. **Tuhan Yesus Kristus** sebagai penolong dan pemelihara yang setia dalam segala dimensi kehidupan ini. Orang tua terkasih, Bapa yang selalu mendoakan & Mama alm. yang selalu ada didalam sanubari. Saudara-saudaraku terkasih bersama keluarga kecil mereka. Jonathan, Beny... u are the best, terimakasih untuk semua keluarga di Alor, Kupang, Rote & Lombok, untuk segala motivasi dan do'anya, keponakan-keponakan genk-beach, Elsya, Beatrix, Serly, Tegar, Willyam, Robin, Yudha, Gina, Ronaldo & Day.

—

Ucapan terimakasih untuk para pendidik. Dosen pembimbing pak Edwin Tjahjadi yang sudah bersedia meluangkan waktu dan kesempatan dalam membimbing dan memberi arahan, bapak Hery Purwanto beserta seluruh pengajar di jurusan teknik geodesi ITN-Malang.

Buat teman-temanku terkasih Geodesi 03' (Danik, Dolly, Grace, Meggy, Rosa, Andy, Beno, Dony, Edy, Gandi, Indra, Irwan, Jose, Fauzan) "kita tak akan pernah terpisahkan oleh waktu dan jarak teman... karena kalian akan selalu dihati". Thank's yah, sudah menjadi bagian dari hari-hari yang menyenangkan, hari-hari dimana kita dapat saling berbagi dan memahami baik dalam suka dan duka... big family never end, teman....

Thank's buat partner programmer (brother Yusak sipp), team Rapid Mapping (Riri, Via, Weny, Rojer, Ronald & Agus), team Deformasi (Dodik, Tanzil, Eno, Lia, Akbar, Alben & Gede)... thanks full... Untuk teman-teman dari semua angkatan & semua civitas akademik di teknik geodesi ITN-Malang.

Buat teman "paling" teman... Singo... best friend always be the bast friend, thank's for journey to "Penanjakan-Bromo" akan menjadi kenangan terindah, thank's sudah bersedia menjadi pendengar yang baik disetiap keluh kesahku. Teman-teman Tercika7. Trim's sudah menjadi saudara dalam hari-hariku.

Untuk skripsi ini..., entah sudah berapa banyak kucurahkan pikiran, perhatian dan emosiku, entah sudah berapa banyak air mata yang menetes. Entah berapa banyak hari-hariku yang kuhabiskan bertemankan labtop, *sourcecode* dan buku-buku pemrograman... entah sudah berapa kali makan siangku yang ku lewatkan dan tidur malamku yang tak nyenyak hanya untuk suatu kemajuan kecil 'progresh'. Entah sudah berapa banyak tanggapan dan kritikan yang tak selamanya mengenakkan dan selalu berusaha ku pandang dari sisi positifnya.

Terkadang berasa sedang berada dalam kubangan lumpur yang membuatku berusaha untuk bernafas... terkadang berasa diatas puncak tertinggi disaat menemukan hal-hal baru dan keberhasilan-keberhasilan kecil. Entah sudah berapa banyak waktu yang kugunakan untuk menenangkan kembali pikiran ini. Butuh kesabaran, keuletan do'a dan komitmen hingga rampungnya program ini.

Meskipun demikian tak ada yang patut dibanggakan dan tak ada yang patut disesali karena semua itu hanyalah bagian dari perjalanan yang punya maksud dan makna, bersyukur aku pernah melewatinya. Membuat kita lebih bijak untuk bersedia menerima kritikan agar lebih baik, bersedia untuk suatu penolakan dan memiliki rasa syukur disetiap peristiwa yang dilewati akan menjadikan kita sebagai orang yang lebih baik dalam mengenal kehidupan ini.

Akhirnya... dahsyatnya badai pun pasti akan belalu. Banyak suka dan duka telah dijalani. Banyak peristiwa yang dapat dimaknai. Dalam perjalanan ini banyak pelajaran kehidupan yang berharga dan pastinya akan menjadi bermakna dikemudian hari.

"suatu keberhasilan dimulai dari hal-hal kecil, kegagalan hari ini adalah suatu keberhasilan yang tertunda, jadikan kegagalan hari ini sebagai cermin untuk keberhasilan di hari esok..." c-yach...

DAFTAR ISI

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

I.1. Latar Belakang Penelitian

Dalam proses pengolahan data fotogrametri dikenal suatu proses penentuan koordinat *object space* (koordinat objek pada sistem datum sembarang, sistem kartesian 3D). Penentuan koordinat ini dilakukan dengan metode *intersection*, agar proses penentuan koordinat *object space* dapat berlangsung otomatis secara digital pada pasangan foto stereo, maka diperlukan suatu perangkat lunak. Merujuk pada pentingnya perangkat lunak dalam membantu dan memberi kemudahan pengolahan data, maka penelitian ini dilakukan.

I.2. Tujuan Penelitian

Tujuan dari penelitian ini adalah menghasilkan program yang dapat digunakan untuk pengolahan pasangan foto stereo hingga menghasilkan koordinat *object space* 3D.

I.3. Manfaat Penelitian

Manfaat dari hasil penelitian ini yaitu :

1. Memberikan kemudahan pengolahan data fotogrametri
2. Penelitian ini diharapkan dapat memberikan sumbangan pemikiran terhadap sistem pengembangan pengolahan data fotogrametri secara digital

I.4. Batasan Masalah

Dalam menjaga konsistensi penelitian maka perlu adanya pembatasan skup masalah yang ditargetkan. Adapun barasan masalah penelitian ini adalah sebagai berikut.

1. Penentuan koordinat *object space* menggunakan metode *intersection.*
2. Metode penentuan koordinat *object space* mengacu pada asumsi bahwa parameter kalibrasi kamera, parameter *interior orientation*, dan parameter *exterior orientation* telah diketahui.
3. Teknik yang digunakan untuk korelasi foto adalah *area based image matching.*
4. Perangkat lunak yang dihasilkan terbatas untuk pengolahan data dua buah foto overlap yang dihasilkan dari dua buah kamera dengan posisi sejajar pada sumbu X.

I.5. Perumusan Masalah

1. Bagaimana membuat interface.
2. Bagaimana mengaplikasikan metode-metode pengolahan image seperti area based matching untuk mendukung penentuan koordinat foto.
3. Bagaimana menerapkan metode *intersection* untuk menentukan koordinat *object space.*

BAB II

DASAR TEORI

Metode interseksi merupakan Metode yang digunakan untuk menentukan koordinat objek 3 dimensi (*Mikhail et al, 2001*). Dalam bab ini akan dijelaskan mengenai hal-hal yang menjadi konsep dalam penentuan koordinat objek 3 dimensi secara otomatis. Sebelum proses interseksi berlangsung perlu dilakukan penentuan titik sekutu. Adapun teknik digital untuk menentukan koordinat titik sekutu adalah *area based matching* diantaranya adalah *cross correlation* dan *least square matching* (*Schenk, 1999*). Setelah proses penentuan titik sekutu dan proses interseksi berlangsung maka koordinat objek 3 dimensi dapat diperoleh. Titik-titik koordinat 3 dimensi yang diperoleh adalah koordinat objek pada sistem bebas (*Schenk, 1999*). Transformasi koordinat dari sistem bebas ke sistem datum dapat dilakukan jika koordinat *principle point* di ganti dengan koordinat kartesian 3 dimensi GPS pada saat foto stereo direkam. Pada pasangan foto stereo, titik-titik objek yang terlihat pada pasangan foto pertama akan terlihat juga pada pasangan foto berikutnya. Hal ini memungkinkan juga agar transformasi koordinat dapat dilakukan dari sistem pada pasangan foto pertama ke pasangan foto berikutnya. Absolut orientasi adalah metode yang digunakan untuk transformasi koordinat dari sistem pada pasangan foto pertama ke pasangan foto berikutnya. Setelah transformasi koordinat berlangsung, proses reseksi dapat dilakukan sehingga parameter orientasi luar (*exterior orientation*) dari salah satu kamera dapat diperoleh dari proses ini. Parameter orientasi luar yang diperoleh dapat dijadikan parameter awal untuk proses interseksi pada pasangan foto berikutnya. Agar semua proses yang sudah dijelaskan dapat berlangsung secara otomatis hingga menghasilkan

koordinat objek 3 dimensi maka dibutuhkan suatu program. Bahasa pemrograman C# merupakan salah satu bahasa pemrograman yang dapat digunakan dalam pembuatan program. Konsep-konsep yang telah diuraikan di atas, akan dibahas lebih lanjut pada setiap sub-bab berikut. Sub-bab pertama dalam bab ini akan menguraikan lebih lanjut mengenai konsep interseksi. Pada sub-bab kedua akan diuraikan mengenai konsep untuk menentukan koordinat titik konjugasi yaitu *area based matching*. Pada sub-bab ketiga akan diuraikan mengenai absolut orientasi. Pada sub-bab keempat akan diuraikan mengenai reseksi. Pada sub-bab kelima akan diuraikan mengenai bahasa program C#. Pada sub-bab keenam akan diuraikan mengenai perintah-perintah dasar bahasa pemrograman.

II.1. Interseksi

Interseksi berhubungan dengan penentuan posisi titik-titik pada *object space* oleh perpotongan garis dari dua atau lebih image (*Wolf & Dewitt, 2000*). Persamaan kesegarisan (*collinearity equations*) merupakan metode dasar dalam penentuan koordinat titik pada *object space*. Persamaan kesegarisan dinyatakan sebagai berikut (*Wolf & Dewitt, 2000*).

$$x_a = x_o - f\,\frac{m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)} \qquad (2.1)$$

$$y_a = y_o - f\,\frac{m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)} \qquad (2.2)$$

L, a dan A berada pada satu garis lurus, dimana:

x_a, y_a : koordinat foto titik a

X_A, Y_A, Z_A : koordinat *object space* titik A

X_L, Y_L, Z_L : koordinat *object space* stasiun pemotretan

f : panjang fokus kamera

x_o, y_o : koordinat *principle point* (dapat diketahui dari kalibrasi kamera

m : fungsi dari 3 sudut rotasi, terdiri dari ω, ϕ, κ

Persamaan (2.1) dan (2.2) merupakan persamaan garis lurus tidak linier dan terdapat sembilan unsur yang tidak diketahui, tiga sudut rotasi ω, ϕ, κ yang berhubungan dengan m, tiga koordinat stasiun pemotretan X_L, Y_L, Z_L, dan tiga koordinat titik objek X_A, Y_A, Z_A. Dapat dilinearisasi menggunakan dalil Taylor. Koordinat foto x_a, y_a merupakan konstanta tetap, x_o, y_o dan f merupakan bagian dari parameter kalibrasi. Menurut *Wolf & Dewitt* (*2000*) Persamaan garis lurus yang tidak linier dapat dilinearisasikan menggunakan dalil Taylor, persamaan (2.1) dan (2.2) dituliskan kembali sebagai berikut.

$$F = x_o - f\frac{r}{q} = x_a \tag{2.3}$$

$$G = y_o - f\frac{s}{q} = y_a \tag{2.4}$$

dimana,

$$q = m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L) \tag{2.5}$$

$$r = m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L) \tag{2.6}$$

$$s = m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L) \tag{2.7}$$

Untuk menentukan suatu koordinat *object space* titik A dengan metode *intersection*, dengan **enam elemen *exterior orientation*** ($\omega, \phi, \kappa, X_L, Y_L, Z_L$) **yang telah diketahui dan tiga elemen koordinat *object space*** (X_A, Y_A, Z_A) **yang belum diketahui** maka persamaan (2.3) dan (2.4) dapat di tuliskan dalam bentuk turunan parsial berdasarkan dalil Taylor sebagi berikut (*Wolf & Dewitt, 2000*).

$$F_0 + \left(\frac{\partial F}{\partial X_A}\right)_0 dX_A + \left(\frac{\partial F}{\partial Y_A}\right)_0 dY_A + \left(\frac{\partial F}{\partial Z_A}\right)_0 dZ_A = x_a \qquad (2.8)$$

$$G_0 + \left(\frac{\partial G}{\partial X_A}\right)_0 dX_A + \left(\frac{\partial G}{\partial Y_A}\right)_0 dY_A + \left(\frac{\partial G}{\partial Z_A}\right)_0 dZ_A = y_a \qquad (2.9)$$

F_o, G_o : fungsi dari F dan G

Istilah $(\partial F/\partial X_A)_0$ dst : turunan parsial dari fungsi F dan G dengan mempertimbangkan unsur yang belum diketahui pada pendekatan awal

x_a, y_a : koordinat foto

Menurut *Wolf & Dewitt* (*2000*) Linierisasi bentuk persamaan interseksi untuk titik A adalah dengan penyederhanaan dari persamaan (2.8) dan (2.9) termasuk nilai residual adalah sebagai berikut.

$$b_{14}\, dX_A + b_{15}\, dY_A + b_{16}\, dZ_A = J + v_{xa} \qquad (2.10)$$

$$b_{24}\, dX_A + b_{25}\, dY_A + b_{26}\, dZ_A = K + v_{ya} \qquad (2.11)$$

dimana,

$$b_{14} = \frac{f}{q^2}(rm_{31} - qm_{11}) \quad (2.12)$$

$$b_{15} = \frac{f}{q^2}(rm_{32} - qm_{12}) \quad (2.13)$$

$$b_{16} = \frac{f}{q^2}(rm_{33} - qm_{13}) \quad (2.14)$$

$$J = x_a - x_0 + f\frac{r}{q} \quad (2.15)$$

$$b_{24} = \frac{f}{q^2}(sm_{31} - qm_{21}) \quad (2.16)$$

$$b_{25} = \frac{f}{q^2}(sm_{32} - qm_{22}) \quad (2.17)$$

$$b_{26} = \frac{f}{q^2}(sm_{33} - qm_{23}) \quad (2.18)$$

$$K = y_a - y_0 + f\frac{s}{q} \quad (2.19)$$

Pada persamaan (2.10) dan (2.11) :

v_{xa}, v_{ya} : kesalahan residual dalam pengukuran

x_a, y_a : koordinat foto

dX_A, dY_A, dZ_A : nilai koreksi awal koordinat *object space* titik A

J dan K : masing-masing sama dengan $x_a - F_o$ dan y_a - G_o

b : koefisien, masing-masing sama dengan turunan parsial. Untuk memudahkan maka koefisien ini telah dijabarkan.

Nilai numerik untuk koefisien ditentukan dengan menggunakan pendekatan awal.

Persamaan (2.10) dan (2.11), dua bentuk persamaan ini dapat dituliskan untuk titik a_1 pada foto kiri dan titik a_2 pada foto kanan. Sehingga akan terdapat 4 persamaan, dan tiga elemen yang belum diketahui dX_A, dY_A, dZ_A dimana ketiga elemen yang merupakan nilai koreksi awal koordinat *object space* ini dapat dihitung dengan metode *least square*. Koreksi ini akan ditambahkan ke pendekatan awal untuk memperoleh nilai kesalahan X_A, Y_A, Z_A. Caranya dilakukan pengulangan hingga nilai koreksi menjadi sangat kecil. Dikarenakan persamaan telah dilinearisasikan dengan dalil Taylor maka pendekatan awal ditentukan untuk tiap titik yang akan dihitung koordinat *object space*nya (*Wolf & Dewitt, 2000*).

Persamaan (2.10) dan (2.11) dapat direpresentasikan dalam bentuk matrix sebagi berikut.

$$A.X = L + V \tag{2.20}$$

dimana,

A : matriks koefisien berisi penurunan pertama persamaan terhadap parameter yang akan dicari

X : matriks berisi parameter yang dicari

L : matriks yang berisikan nilai fungsi dimana nilai koordinat foto dikurangi nilai pendekatan koordinat *object space*

V : matriks residual berisi kesalahan pengukuran

$$A = \begin{bmatrix} b_{14_a} & b_{15_a} & b_{16_a} \\ b_{24_a} & b_{25_a} & b_{26_a} \end{bmatrix} \quad X = \begin{bmatrix} dX_A \\ dY_A \\ dZ_A \end{bmatrix} \quad L = \begin{bmatrix} J_a \\ K_a \end{bmatrix} \quad V = \begin{bmatrix} V_{xa} \\ V_{ya} \end{bmatrix} \tag{2.21}$$

Jika persamaan observasi dipartisi dalam beberapa bagian, maka penyelesaian *least square* akan menjadi:

$$(A^T A)\mathrm{X} = A^T L \tag{2.22}$$

$A^T A$ merupakan matriks koefisien persamaan normal dari unit yang belum diketahui karena dalam persamaan, unit yang belum diketahui adalah matriks X maka persamaan akan ditulis sebagai berikut:

$$\mathrm{X} = (A^T A)^{-1} A^T L \tag{2.23}$$

Persamaan matrix untuk menghitung residual setelah iterasi :

$$\mathrm{V} = AX - L \tag{2.24}$$

Simpangan baku

$$\mathrm{So} = \sqrt{\frac{(V^T V)}{r}} \tag{2.25}$$

dimana,

So : simpangan baku

r : jumlah derajat kebebasan, merupakan jumlah persamaan pengamatan dikurangi dengan jumlah nilai tak dikenal, atau $r = (\mathrm{m} - \mathrm{n})$.

V : matrix residual

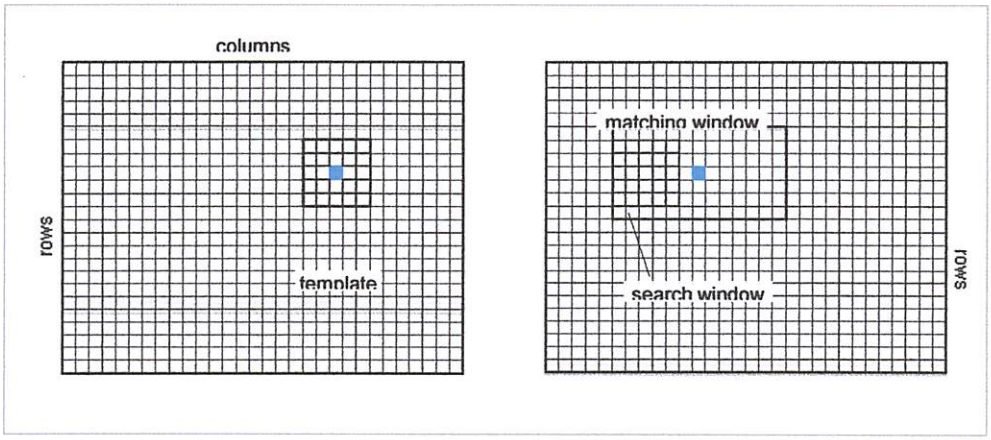
II.2. *Image Matching*

Salah satu proses mendasar dalam fotogrametri adalah mengidentifikasi titik Sekutu pada dua atau lebih foto yang overlap. Pada fotogrametri konvensional (non-digital) identifikasi titik yang bersesuaian dilakukan oleh *human operator*. *Image matching* merupakan suatu proses dalam fotogrametri digital untuk mengidentifikasikan dan mengukur titik konjugasi secara otomatis pada dua atau lebih foto yang overlap (*Schenk, 1999*).

Terdapat 3 metode yang sejauh ini banyak digunakan dalam *image matching*. Ketiga metode yang dimaksud adalah *area-based matching, feature-base matching,* dan *symbolic matching*. *Area-based matching* mendasarkan hubungan antara dua image menurut kesamaan derajat keabuan (*grey level*). Metode ini akan dibahas lebih lanjut pada sub-bab ini. Adapun teknik yang sering digunakan pada *area based matching* adalah *Normalized Cross Correlation* (NCC) dan *Least Square Matching* (LSM). *Area-Based Matching* menggunakan hubungan korelasi antara dua objek pada dua citra atau lebih. Kelemahan dari *Area-Based Matching* adalah obyek yang melalui proses *matching* belum tentu bersekutu, karena hanya mendeteksi obyek berdasarkan kesamaan nilai spektralnya, namun metode ini sangat populer dalam fotogrametri (*Schenk, 1999*).

II.2.1. Area Based Matching Dengan Teknik *Cross Correlation*

Area-based matching berhubungan dengan otomaisasi identifikasi titik konjugasi menurut derajat keabuan (*gray levels*). Teknik *area based matching* membandingkan distribusi derajat keabuan pada sebagian kecil image, yang disebut *image patch*, dengan pasangannya pada image lain.



Gambar 2.2 *template* dan *matching window*

(Gambar 2.2) menunjukkan konsep dan istilah yang digunakan dalam *area based matching*. *Template* merupakan *image patch* yang biasanya berada pada posisi fix dalam sebuah image. *Search window* merupakan 'ruang cari' dimana *image patch* (sering disebut *matching window*) dibandingkan dengan template. Perbandingan dilakukan dengan tehnik *cross-corelation* dan *least-square matching* (*Schenk, 1999*).

Cross correlation (ρ) merupakan salah satu teknik digital dalam penentuan titik sekutu. Prinsip *cross correlation* yaitu mencari pasangan titik piksel antara citra pertama dengan citra pasangan/citra kedua. Pada citra pertama ditentukan jendela sasaran (*template*) yang memuat titik piksel yang akan dicari pasangannya pada citra kedua. Pada citra kedua ditentukan daerah selidik (*search window*) yang mempunyai ukuran lebih besar dari pada daerah sasaran (*template*). Pada daerah selidik (*search window*) dibentuk pula jendela/daerah sub selidik (*matching window*) dengan ukuran yang sama dengan jendela/daerah sasaran (*template*). *Search window* ini bergerak (*moving window*) dengan *increment* 1 piksel sepanjang setiap baris dan kolom di daerah selidik (*search window*). Dihitung nilai korelasi (ρ) antara *template* dan *matching window*.

Nilai korelasi antara *template* dan *matching window* dihitung berdasarkan rumus matematis pada persamaan berikut (*schenk, 1999*) :

$$\rho = \frac{\sigma_{LR}}{\sigma_L \sigma_R} \tag{2.26}$$

Normalnya nilai ρ adalah: $-1 \leq \rho \leq +1$

$$\bar{g}_L = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} g_L(x_i, y_j)}{n \,.\, m} \tag{2.26}$$

$$\bar{g}_R = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} g_R(x_i, y_j)}{n \,.\, m} \tag{2.27}$$

$$\sigma_L = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(g_L(x_i, y_j) - \bar{g}_L\right)^2}{n \,.\, m \,-\, 1}} \tag{2.28}$$

$$\sigma_R = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(g_R(x_i, y_j) - \bar{g}_R\right)^2}{n \,.\, m \,-\, 1}} \tag{2.29}$$

$$\sigma_{LR} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\left(g_L(x_i, y_j) - \bar{g}_L\right)\left(g_R(x_i, y_j) - \bar{g}_R\right)\right)}{n \,.\, m \,-\, 1} \tag{2.30}$$

dimana,

ρ	:	koefisien korelasi
σ_L	:	standar deviasi *image patch* $_L$ (*template*)
σ_R	:	standar deviasi *image patch* $_R$ (*matching window*)
σ_{LR}	:	kofarian *image patch* $_L$ dan $_R$
$g_L(x_i, y_j)$	:	individual nilai keabuan pada *template* matrix
$\bar{g}_L$	:	rata-rata nilai keabuan pada *template* matrix
$g_R(x_i, y_j)$	:	individual nilai keabuan pada bagian *search window* matrix

$\bar{g}_R$ = rata-rata nilai keabuan pada bagian *search window* matrix

m, n = jumlah kolom dan baris pada *template* matrix

Nilai korelasi ditetapkan untuk setiap posisi baris dan kolom pada *matching window*. Nilai korelasi (ρ) berkisar antara –1 sampai dengan 1. Nilai 1 menunjukkan korelasi yang sempurna (*perfect match*), nilai 0 menunjukkan total *miss match* (tidak terdapat korelasi), serta nilai –1 menunjukkan adanya korelasi yang belawanan (*Schenk, 1999*).

II.3. Orientasi Absolut

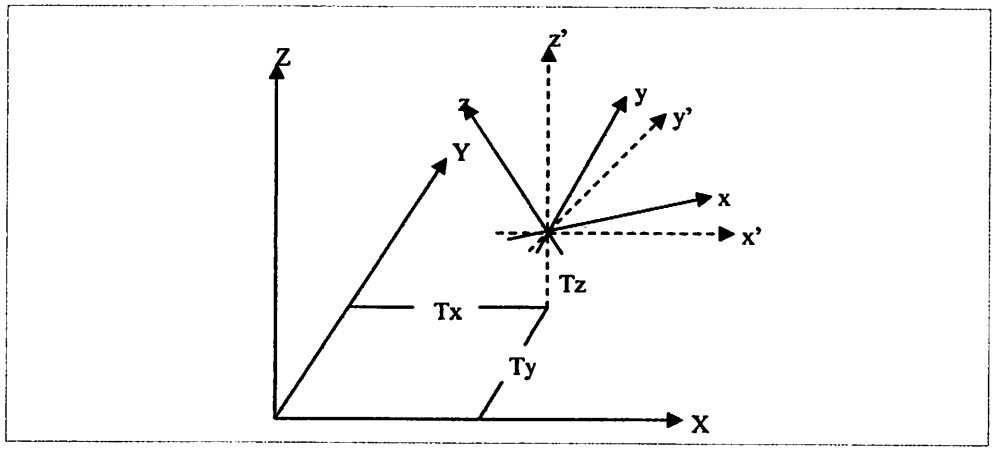
Orientasi Absolut merupakan proses dalam ilmu fotogrametri untuk menentukan nilai skala dan elemen orientasi dari model geometri dengan membangun terlebih dahulu orientasi relatif pada setiap *image (Zhang & Yao, 2008)*. Ada tujuh parameter yang akan dicari pada metode ini yaitu satu faktor skala (s), tiga faktor rotasi (ω, φ, κ) dan faktor translasi (T_x, T_y, T_z) *(Mikhail et al., 2001)*.

Metode ini akan coba digunakan untuk menentukan nilai orientasi kamera dengan menghitung terlebih dahulu nilai pendekatan untuk ketujuh parameter. Jika telah didapatkan nilai ketujuh parameter tersebut, maka nilai orientasi kamera dapat dihitung. Metode ini akan dijelaskan secara umum pada sub-bab selanjutnya.

II.3.1 Transformasi Koordinat Konform 3 Dimensi

Transformasi koordinat konform 3D meliputi pengubahan dari suatu sistem 3D ke sistem lainnya. Transformasi konform 3 dimensi atau transformasi sebangun adalah transformasi yang mempertahankan bentuk dan ukuran suatu objek. Jenis transformasi koordinat ini penting didalam fotogrametri analitik dan *automatic*. sehubungan dengan dua masalah pokok *(Wolf, 1993; Wolf & Dewit, 2000)* :

1. Untuk mengubah koordinat titik-titik dari sistem koordinat foto yang mengalami kecondongan (*tilt*) ke sistem foto tegak ekuivalennya yang sejajar dengan sistem ruang medan atau sembarang.
2. Untuk membentuk model jalur 3D dari model stereo.



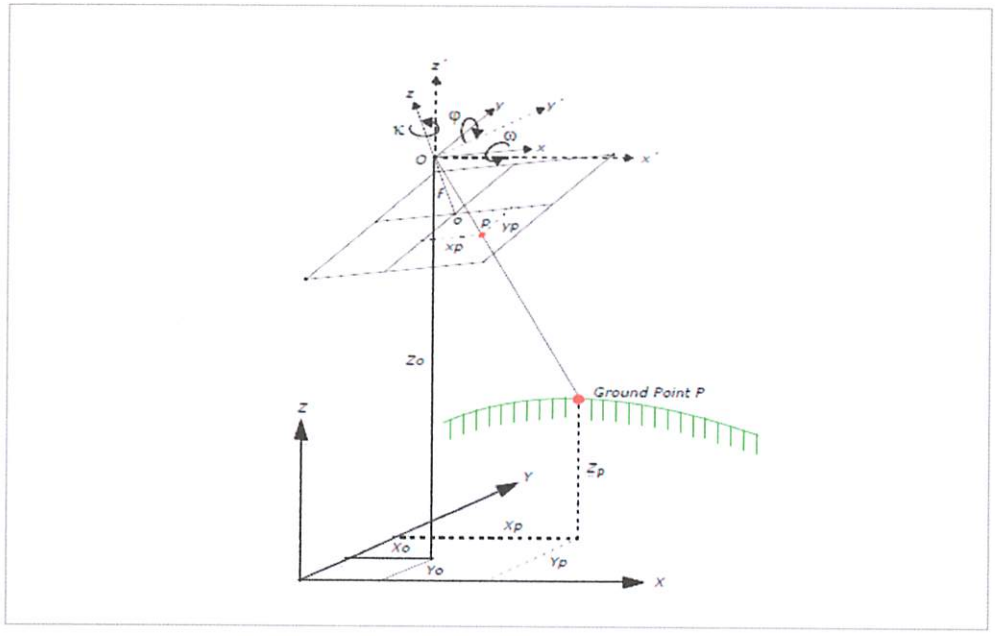
Gambar 2.4 *Sistem koordinat 3D XYZ dan xyz arah kanan*

Sesuai dengan Gambar (2.4) dilakukan pengubahan koordinat titik dari sebuah sistem xyz ke sistem XYZ. Seperti tercermin dalam gambar tersebut, dua sitem koordinat tidak sejajar. Persamaan transformasi yang perlu dapat dinyatakan sesuai dengan tujuh parameter transformasi yang terdiri dari tiga parameter sudut rotasi omega (ω), phi (φ) dan kappa (κ), sebuah faktor skala (s) dan tiga parameter translasi (T_x, T_y,

T_z). Sistem rotasi ω, φ, κ akan bernilai positif apabila arah rotasinya berlawanan terhadap arah jarum jam apabila diamati dari ujung positif sumbunya (*Wolf, 1993*).

II. 4. Reseksi

Reseksi merupakan sebuah metode dalam ilmu fotogrametri yang digunakan untuk menentukan 6 elemen orientasi luar (ω, φ, κ, X_L, Y_L, Z_L) dari sebuah foto (*Wolf, 1993;Wolf & Dewitt, 2000*). Menurut *(Mikhail, et al., 2001)*, *resection* merupakan proses untuk penentuan posisi gambar dan orientasi parameter yang berkaitan dengan sistem koordinat objek. Sedangkan menurut (*Cooper & Robson, 2001; Zheng and Wang, 1992)*, *Space resection* merupakan sebuah metode untuk mengevaluasi secara langsung pada enam elemen orientasi bagian luar (*Exterior Orientation)* diperoleh dari diukurnya koordinat foto pada *image* dengan tiga titik kontrol non kolinear yang memerlukan beberapa nilai pendekatan.



Gambar 2.4 *Elemen Orientasi Luar*

Reseksi juga termasuk determinasi posisi dan orientasi parameter secara tidak langsung, sebagai contoh, penentuan koefisien dari penggabungan gambar *polynomial* untuk menentukan posisi dan orientasi waktu yang berkaitan dengan bermacam-macam sensor atau bentuk parameter proyeksi *(Mikhail, et al., 2001)*.

Ada banyak persamaan yang dapat digunakan untuk menyelesaikan permasalahan *resection*. Dalam kasus ini, *space resection* menggunakan persamaan kolinearitи berdasarkan pada kondisi kolinearitи yang syaratnya adalah dimana koordinat pada foto, koordinat titik kontrol berada pada satu garis lurus melewati *perspective center* pada kamera (*Jue, 2008)*. Karena ada 6 parameter EO (3 untuk orientasi sudut rotasi dan 3 untuk posisi kamera) maka minimum 3 titik kontrol dibutuhkan untuk menyelesaikan sistem ini.

Jika telah diketahui minimum tiga titik kontrol pada arah X, Y, dan Z maka *space resection* dapat dihitung untuk menentukan 6 parameter posisi orientasi luar (ω, φ, κ, X_L, Y_L, Z_L) pada sebuah *image* dengan asumsi bahwa parameter *interior orientation* juga telah diketahui. *Space resection* biasanya digunakan untuk ortorektifikasi pada satu foto , dimana proses perhitungannya dilakukan untuk setiap satu foto *(Leica, 2006)*. Apabila menggunakan lebih dari 3 titik kontrol, maka teknik perhitungan *Least Square Adjusment* dapat diaplikasikan untuk menentukan nilai yang paling mungkin bagi keenam parameter tersebut.

II.5. Bahasa Program C#

C# merupakan bahasa program berorientasi objek yang dikembangkan oleh Microsoft yang menjadi bagian dari pengembangan *software* berbasis .NET (*Hughes,2005*). Adapun program C# terdiri atas kumpulan *file* dimana setiap *file*

merupakan rangkaian dari karakter *Unicode* (*Troelsen,2007*). Menurut *Hughes*(*2005*), Agar kumpulan *file* di dalam program C# dapat terorganisir dengan baik maka bahasa program C# menggunakan apa yang disebut dengan namespace. Selain namespace, dalam C# dikenal juga Class. Tujuan dari penggunaan class adalah agar semua proses dalam bahasa C# dapat terorganisir dengan baik (*Hughes,2005*). Class dapat mengandung filds, constructor, method serta operator-operator lainnya.

II.5.1. Namespace

Secara logika, *source file* dalam program C# terdiri atas suatu koleksi dari *namespace*. Namespace adalah kumpulan dari class-class yang saling berhubungan (*Troelsen,2007*), *programer* dapat menyatukan class-class yang masih memiliki hubungan dengan sebuah nama yang disebut namespace. Biasanya dalam bahasa C# terdapat lebih dari satu class. C# tidak mengizinkan penggunaan nama class yang sama dalam satu program. Menurut *Hughes*(*2005*) Tujuan utama menggunakan namespace adalah untuk menghindari konflik nama class yang sama. Konflik nama class yang sama dapat terjadi bila terdapat koleksi class yang sangat banyak.

II.5.2. Class

Dalam sebuah aplikasi setidaknya harus ada sebuah class. Class adalah kombinasi dari data (*field*) dan fungsi (*method*) yang bertugas mengolah data sehingga mencapai hasil yang diinginkan (*Jones&MacDonald 2006*). Dalam setiap class harus ada method utama yang mempunyai kendali utama. Untuk mendefinisikan sebuah tipe data baru atau class, harus terlabih dahulu mendeklarasikan kemudian baru mendefinisikan *method* dan *field*nya. *Keyword* Class digunakan untuk mendeklarasikan

sebuah class. Umumnya class yang diciptakan menggunakan *acces modifier* **public.** Menurut *Champlain&Patrick* (*2005*). fungsi umum access modifier adalah untuk membatasi atau menentukan variabel atau method apa yang dapat diakses dari dalam class atau class lain. Beberapa *access modifier* yang dapat digunakan pada C# dapat diperlihatkan pada tabel 2.1.

Tabel 2.1 : tabel *acces modifier*

Acces Modifier	Keterangan
public	Tidak ada batasan. Member dapat diakses dari method maupun dari class manapun.
privete	Member dari class X hanya dapat diakses oleh method dari dalam class X sendiri.
protected	Member dari class X hanya dapat diakses oleh method dari class X sendiri atau method dari class yang berasal dari class X.
internal	Member dari class X hanya dapat diakses oleh method dari semua class dalam kode assembly X.

Secara default bila *access modifier* tidak digunakan, maka C# akan menganggap bahwa class yang kita buat mempunyai *access modifier* **private** (*Champlain&Patrick, 2005*). Pendefinisian member dari class yang berada pada class body akan diapit oleh tanda kurung kurawal { }. Pada C# semua aktivitas program terjadi di dalam class. Sebuah class dideklarasikan menggunakan kata kunci class dengan cara berikut.

```
<modifiers>opt class <identifier>
{
      <class-members>
}
```

dimana,

<modifiers> merupaka pilihan mengenai *access modifier*

<identifier> nama dari Class yang dibentuk, harus bersifat unik

<class-members> member dari class

Contoh pendefinisian class :

```
Public class HelloWord
{
      Public static void Main()      //method utama
}
```

II.5.2.1. Fields

Field adalah variabel yang terdapat di dalam class. Field dapat dideklarasikan dengan cara berikut :

```
<modifiers>opt <type> <field-name> = <initial-value>opt
```

dimana,

<modifiers> merupaka pilihan mengenai kemampuan variabel untuk diakses

<type> menyatakan tipe variabel

<field-name> nama field yang bersifat unik

<initial-value> nilai dari pada variabel

II.5.2.2. Methods

Menurut *Brown(2006)* Method adalah anggota dari class yang menyatakan operasi atau aksi. Biasanya berisi kode-kode untuk membuat class menjadi "hidup". Method biasanya digunakan untuk membuat class dapat merespon dan bertingkahlaku

terhadap situasi tertentu. Method dapat dipanggil dari dalam class atau dari luar class tergantung pada modifier-nya. Method dapat dideklarasikan dengan cara berikut :

```
<modifiers>opt <return-type> <member-name> (<parameters>opt)
{
      <statements>opt
}
```

dimana,

`<modifiers>` merupaka pilihan mengenai kemampuan method untuk diakses

`<member-name>` nama method yang bersifat unik

`<parameters>` merupakan pilihan

`<statements>` pernyataan spesifik mengenai intruksi *computer* untuk menampilkan aksi yang dikehendaki

II.5.2.3. Constructor

Menurut *Brown(2006)* Constructor adalah method yang digunakan untuk menciptakan sebuah objek berdasarkan kerangka class dan meletakkannya pada kondisi yang tepat. Jika didefinisikan sendiri maka *constructor* harus diberi nama sama dengan nama class karena *construcor* memang ditujukan untuk menciptakan objek berdasarkan design class.

II.6. Perintah-Perintah Dasar Bahasa Pemrograman

Hampir semua perintah yang akan digunakan dalam program merupakan method yang dimiliki oleh suatu class atau struktur tertentu. Karena itu, seharusnya sudah tidak ada perintah lain lagi yang diperlukan dalam pembuatan suatu program. Hanya saja kadang-kadang tidak semua method harus dikerjakan pada semua kondisi. Menurut

Penton(2005) Perintah tambahan yang dapat menyeleksi method mana yang akan dijalankan berdasarkan suatu kondisi tertentu adalah perintah `if` dan `switch`. Selain itu proses pengulangan diperlukan juga untuk melakukan perulangan pada satu atau beberapa method. Perintah tambahan yang dapat digunakan untuk melakukan proses pengulangan adalah printah `for` dan `while`.

II.6.1. Kondisi dan Operator Logika

Kondisi dan operator logika pada bahasa pemrograman diantaranya adalah perintah *if-else,* dan perintah *if* dengan beberapa perbandingan. Penjelasan lebih lanjut mengenai kondisi dan operator logika dapat dilihat pada sub-bab ini.

II.6.1.1. Perintah If – Else

Perintah if-else merupakan perintah yang paling umum digunakan untuk menyatakan kondisi. Perintah ini dapat digunakan untuk membuat kondisi dari berbagai tipe data. Format perintah if-else dapat diuraikan sebagai berikut.

```
if(kondisi)
{
      //perintah-perintah yang ada di bagian ini, akan
      //dijalankan saat kondisi bernilai true
      :
}
else
{
      //perintah-perintah yang ada di bagian ini, akan
      //dijalankan saat kondisi bernilai false
      :
}
```

Pengisian `kondisi` bagian pada perintah `if-else` dapat dilakukan dengan membandingkan suatu data dengan data yang lain. Sebuah perbandingan yang valid

akan tersusun dari dua buah data dan sebuah operator perbandingan. Perbandingan yang satu dengan yang lain harus dipisahkan oleh sebuah operator logika. Setiap perbandingan yang dilakukan harus menggunakan salah satu operator perbandingan sebagai mana yang terlihat pada tabel 2.2 dibawah.

Tabel 2.2 : tabel perbandingan

Operator	Pengecekan
==	Apakah data I sama dengan data II
!=	Apakah data I tidak sama dengan data II
<	Apakah data I lebih kecil dari data II
<=	Apakah data I lebih kecil atau sama dengan data II
>=	Apakah data I lebih besar atau sama dengan data II

II.6.1.2. Peintah if dengan Beberapa Perbandingan

Dalam suatu kasus tertentu mungkin dibutuhkan lebih dari satu proses perbandingan untuk menentukan method mana yang akan dijalankan. Untuk itu, beberapa proses perbandingan diperlukan untuk menggabungkannya dengan suatu operator logika. Format perintah kondisi diuraikan pada tabel 2.3, dan format perintah kondisi dengan beberapa perbandingan dapat dituliskan sebagai berikut.

```
if((perbandingan1) && (perbandingan2))
{
      :
}
```

Tabel 2.3 : tabel format perintah kondisi

Operator	Arti	Kegunaan
!	tidak	mengubah nilai `true` menjadi `false` dan sebaliknya

&&	atau	memberikan hasil akhir berupa nilai
		`true` bila semua hasil perbandingan bernilai `true`
		`false` bila satu saja hasil perbandingan bernilai `false`

||	tidak	memberikan hasil akhir berupa nilai :
		`false` bila semua hasil perbandingan bernilai `false`
		`true` bila semua hasil perbandingan bernilai `true`

II.6.2. Proses Pengulangan

II.6.2.1. Perintah For

Perintah `for` merupakan salah satu perintah yang disediakan C# untuk melakukan proses pengulangan terhadap satu atau beberapa perintah tertentu. Perintah `for` dapat digunakan untuk berbagai keadaan tetapi keadaan berikut merupakan keadaan yang paling cocok untuk perintah `for` :

- Bila jumlah perulangan yang diinginkan sudah diketahui atau
- Nilai awal dan akhir dari proses pengulangan sudah diketahui

Format perintah for :

```
For(nilaiAwal; kondisi; ekspresi)
{
      //perintah atau method yang dikerjakan berulang-ulang
      :
}
```

Bagian-bagian yang ada pada perintah `for` dapat dikelompokkan menjadi 4 bagian yaitu:

- Pemberian nilai awal pada suatu variabel
- Pengecekkan kondisi variabel
- Perintah atau method yang akan dikerjakan berulang-ulang

Pada umumnya perintah `for` akan membutuhkan sebuah variabel sebagai alat bantu untuk melakukan pengulangan. Mula-mula berikan nilai tertentu pada variabel ini dan letakkan proses pemberian nilai tersebut pada bagian `nilai awal`.

II.6.2.2. Perintah While

Arti kata `while` dalam bahasa indonesia adalah "selama". Format perintah `while` dapat dituliskan sebagai berikut :

```
While(kondisi)
{
      //perintah atau method yang dikerjakan berulang-ulang
      :
}
```

Bagian-bagian penyusun perintah `while` sebenarnya sama persis dengan bagian penyusun perintah `for`. Karena itu, suatu proses yang berulang sebenarnya dapat dikerjakan melalui perintah `for` maupun perintah `while`. Hanya saja pada umumnya perintah `for` lebih sering digunakan untuk melakukan proses pengulangan yang sudah diketahui banyak peluangnya, sedangkan perintah `while` tidak.

BAB III

METODOLOGI PENELITIAN

Agar suatu penelitian dapat berlangsung dengan lancar, ada beberapa hal yang perlu diperhatikan yaitu persiapan penelitian, alat dan bahan penelitian serta diagram alir dan penjelasan. Hal ini dilakukan agar proses penelitian dapat berjalan secara lancar dan efektif. Pada bab ini, akan dijelaskan secara lengkap hal-hal yang dibutuhkan untuk penelitian serta tahapan-tahapan kerja yang akan dilaksanakan dalam penelitian.

III.1. Persiapan Penelitian

Persiapan penelitian merupakan suatu proses yang sebaiknya dilakukan demi kelancaran suatu penelitian. Ada pun beberapa hal yang perlu diperhatikan dalam proses persiapan penelitian akan dibahas dalam sub-bab ini.

III.1.1. Materi Penelitian

Adapun materi yang digunakan sebagai bahan dalam penelitian ini meliputi data-data sebagai berikut.

- Foto stereo

  Foto stereo digunakan sebagai data untuk ekstraksi koordinat foto tiap titik konjugasi yang hendak diketahui koordinat *object space*nya.

- CSML (C# matrix *library*)

  CSML merupakan suatu *library* yang dibuat dalam bahasa C#. *Library* ini dapat digunakan dalam proses *coding* program khususnya yang berkaiatan dengan komputasi menggunakan matrix.

- Emgu (*computer vision library*)

  Emgu merupakan *computer vision library* yang terdiri dari fungsi-fungsi yang berhubungan dengan proses pengolahan *image*. Fungsi-fungsi pada emgu ini dapat diaplikasikan dalam proses *coding* program khususnya yang berkaiatan dengan pengolahan *image*.

- Algoritma *image matching*

  Algoritma *image mathing* secara fotogrametri dibutuhkan untuk pembuatan program dalam penelitian ini. Dimana proses ini bertujuan untuk menemukan titik konjugasi secara digital pada pasangan foto stereo.

III.1.2. Alat Penelitian

Untuk mendukung pelaksanaan kegiatan penelitian ini, maka diperlukan beberapa perangkat keras (*hardware*) dan perangkat lunak (*software*). Adapun *hardware* dan *software* pendukung penelitian dapat dijabarkan sebagai berikut.

- **Hardware**

  Spesifikasi (Central Procesing Unit atau CPU) :

  - Processor
  - RAM-DDR
  - Monitor
  - Hardisk
  - VGA
  - CD-RW

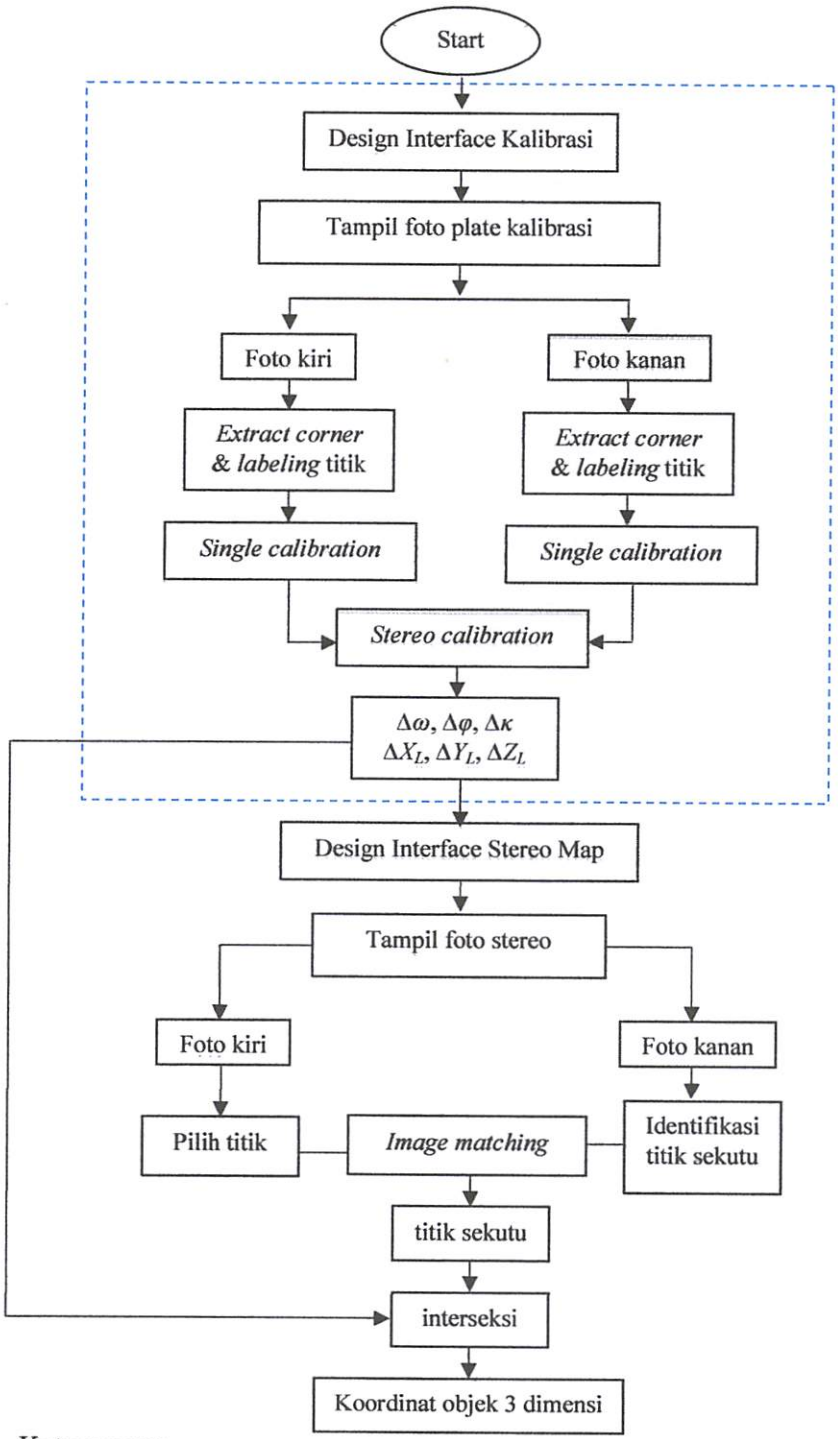
- **Software**

  - Microsoft Visual Studio 2008
  - Microsoft Excel 2007
  - Microsoft Word 2007

III.2. Diagram Alir Penelitian

Diagram alir penelitian merupakan suatu kerangka pekerjaan dalam penelitian yang disusun secara sistematis dengan tujuan agar tahapan pekerjaan dapat dilakukan secara bertahap, sistematis dan terarah. Ada pun diagram alir yang berhubungan dengan penelitian akan dipaparkan dalam sub-bab ini. Sub-bab pertama pada sub-bab ini akan ditampilkan mengenai diagram algoritma pembuatan program yang terdiri dari diagram pembuatan program utama (*interface*). Sedangkan sub-bab kedua pada sub-bab ini akan ditampilkan mengenai diagram algoritma interseksi yang terdiri dari tahapan penentuan koordinat *object space* titik dengan metode interseksi. Ada pun sub-bab ketiga pada sub-bab ini akan ditampilkan mengenai diagram algoritma *cross correlation* dimana algoritma ini akan digunakan dalam proses *coding* program khususnya dalam penentuan koordinat foto titik-titik konjugasi dengan metode *cross correlation.*

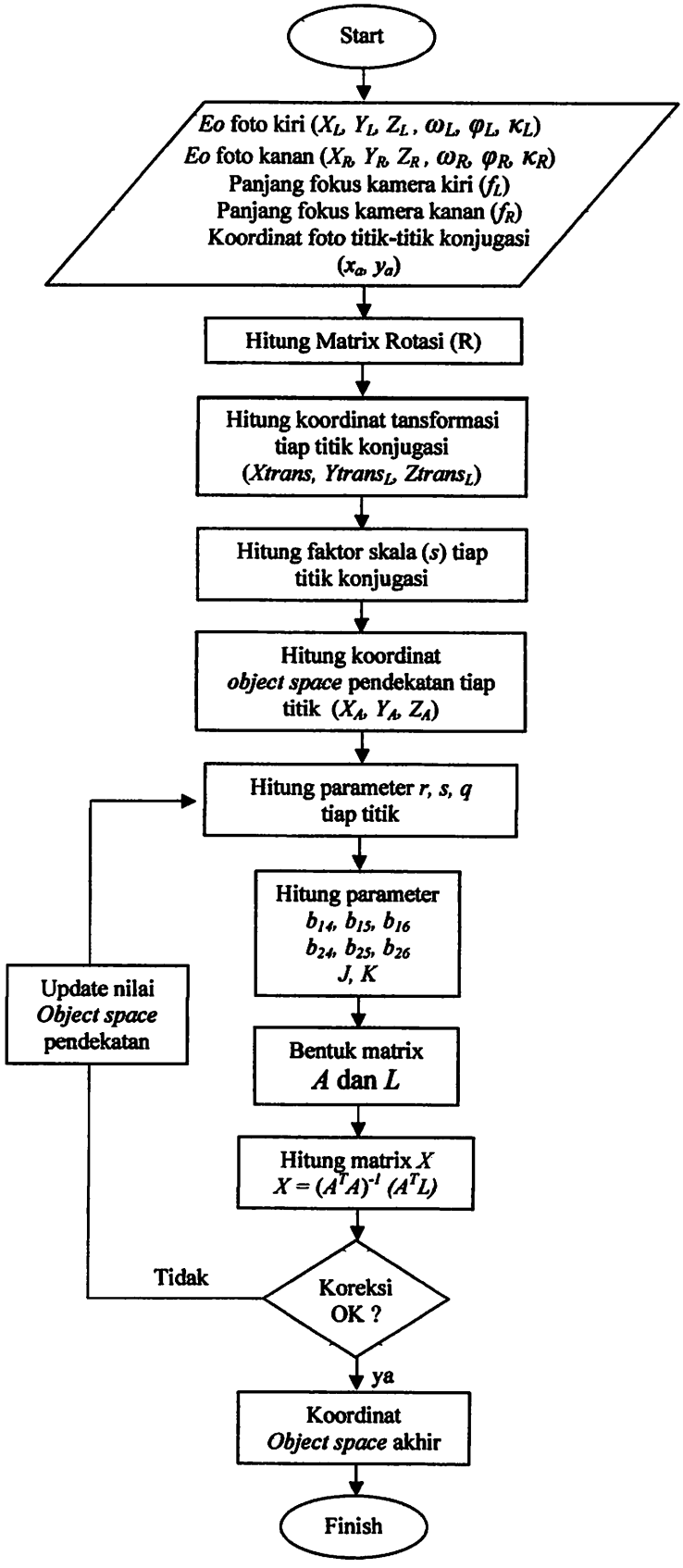
III.2.1. Algoritma Pembuatan Program



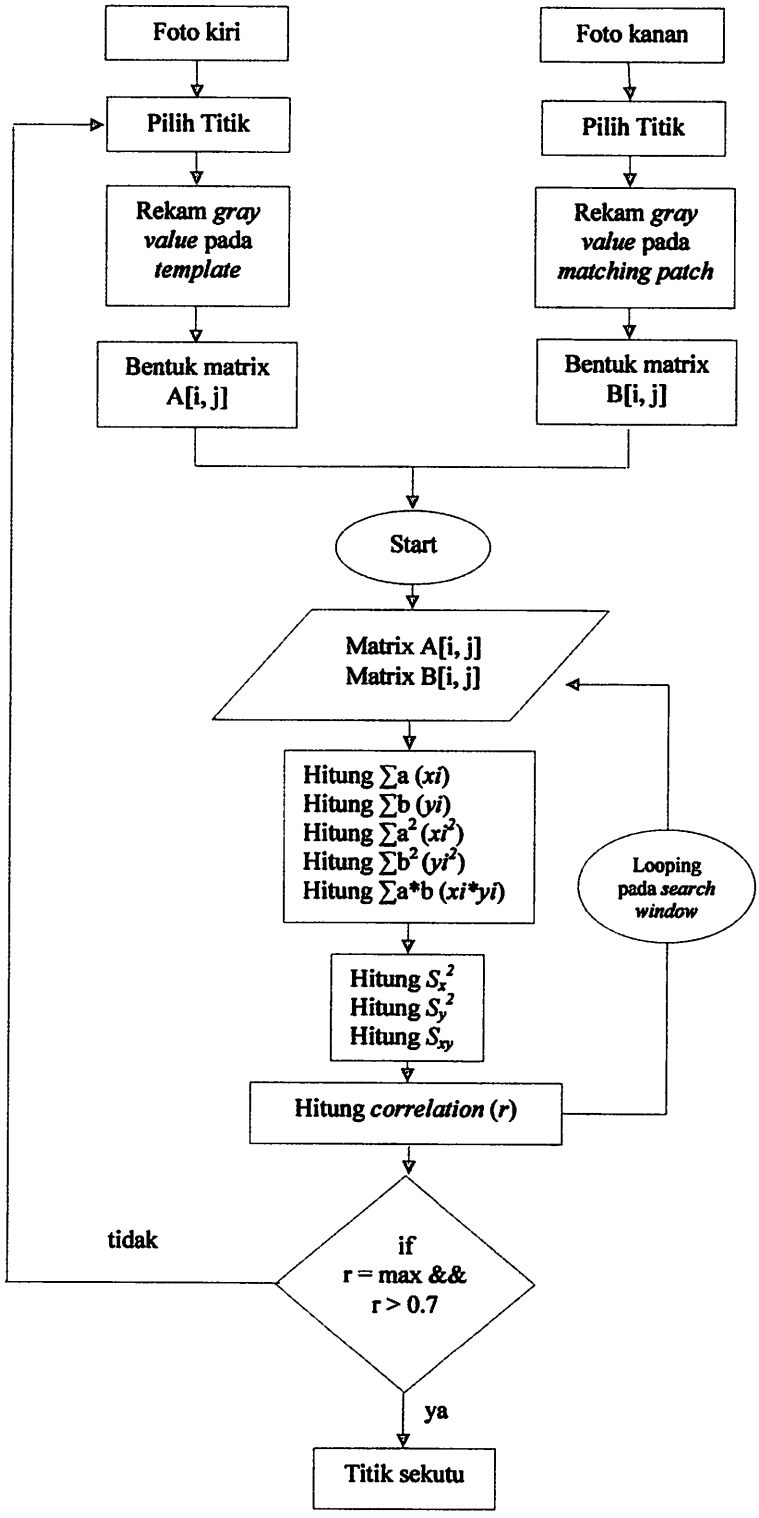
Keterangan:

Tahapan pembuatan program, dimana dalam penelitian ini diasumsikan bahwa sudah dilakukan sebelumnya.

III.2.2. Diagram Algoritma Interseksi



III.2.3. Diagram Algoritma *Cross Correlation*



III.3. Penjelasan Diagram Alir Penelitian

Penjelasan mengenai diagram alir penelitian akan dipaparkan lebih lanjut dalam sub-bab ini. Ada pun sub-bab pertama pada sub-bab ini akan dipaparkan mengenai pembuatan *interface* program yang sebelumnya sudah ditampilkan pada sub-bab III.2.1. Sedangkan pada sub-bab kedua pada sub-bab ini akan dipaparkan lebih rinci mengenai algoritma interseksi yang sebelumnya sudah ditampilkan pada sub-bab III.2.2.

III.3.1. Penjelasan Diagram Interface Program

Diagram pada sub-bab II.2.2 yang menggambarkan diagram interface program dapat dirincikan berikut ini. Sebelum interface untuk proses interseksi dibuat, perlu dibuat terlebih dahulu interface untuk proses kalibrasi. Hal ini dilakukan karena mengingat parameter output dari proses stereo kalibrasi akan dijadikan parameter awal dalam proses interseksi.

Dalam penelitian ini desain interface kalibrasi dan proses kalibrasi sendiri diasumsikan bahwa sudah dilakukan. Adapun tahapan-tahapan dalam mendesain program untuk interseksi adalah sebagai berikut.

1. Program didesain agar pasangan foto yang terdiri dari foto kiri dan foto kanan dapat tampil secara bersamaan.
2. Setelah pasangan foto tampil, proses penentuan titik sekutu dapat dilakukan. Penentuan titik sekutu dilakukan dengan teknik *image matching* yang merupakan teknik penentuan titik sekutu secara digital.
3. Jika penentuan titik sekutu telah berhasil maka proses interseksi dapat dilakukan. Proses interseksi juga memperhitungkan parameter hasil stereo kalibrasi sebagai parameter awal.
4. Proses interseksi akan menghasilkan koordinat objek tiga dimensi.

III.3.2. Penjelasan Algoritma Interseksi

Algoritma interseksi dibutuhkan dalam proses pembuatan program, khususnya penentuan koordinat *object space*. Ada pun penjelasan lebih lanjut mengenai diagram algoritma interseksi yang sebelumnya sudah ditampilkan pada sub-bab III.2.1 adalah sebagai berikut.

1. Dalam penentuan koordinat *object space* dengan metode interseksi, diperlukan data input sebagai data awal. Ada pun data input dalam proses interseksi adalah parameter orientasi luar kamera kiri dan kamera kanan, panjang fokus kamera kiri dan kamera kanan. Ada pun informasi dari parameter-parameter tersebut dapat diperoleh dari proses stereo kalibrasi yang mana dalam penelitian ini diasumsikan bahwa sudah dilakukan. Parameter lainnya yang diperlukan sebagai data awal adalah informasi koordinat foto dari titik-titik konjugasi yang hendak diketahui koordinat *object space*nya.
2. Perhitungan matrix rotasi (R) kamera kiri dan kamera kanan. Ada pun persamaan untuk menghitung matrix rotasi adalah sebagai berikut.

$$R_{\omega\varphi\kappa} = R_\kappa R_\varphi R_\omega \begin{bmatrix} \cos\varphi\cos\kappa & \sin\omega\sin\varphi\cos\kappa+\cos\omega\sin\kappa & -\cos\omega\sin\varphi\cos\kappa+\sin\omega\sin\kappa \\ -\cos\varphi\sin\kappa & -\sin\omega\sin\varphi\sin\kappa+\cos\omega\cos\kappa & \cos\omega\sin\varphi\sin+\sin\omega\cos\kappa \\ \sin\varphi & -\sin\omega\cos\varphi & \cos\omega\cos\varphi \end{bmatrix}$$

$$R = \begin{bmatrix} r_{11_a} & r_{12_a} & r_{13_a} \\ r_{21_a} & r_{22_a} & r_{23_a} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad (3.1)$$

3. Perhitungan transformasi koordinat tiap titik. Ada pun persamaan untuk menghitung nilai koordinat transformasi adalah sebagai berikut.

$$\begin{aligned} x' &= r_{11}x + r_{21}y + r_{31}z \\ y' &= r_{12}x + r_{22}y + r_{32}z \\ z' &= r_{13}x + r_{23}y + r_{33}z \end{aligned} \qquad (3.2)$$

4. Perhitungan faktor skala tiap titik. Ada pun persamaan untuk menentukan faktor skala tiap titik adalah sebagai berikut.

$$s_{right} = \frac{y'_{left}(XL_{right} - XL_{left}) - x'_{left}(YL_{right} - YL_{left})}{x'_{left}y'_{right} - x'_{right}y'_{left}} \tag{3.3}$$

$$s_{left} = \frac{y'_{right}(XL_{right} - XL_{left}) - x'_{right}(YL_{right} - YL_{left})}{x'_{right}y'_{left} - x'_{left}y'_{right}} \tag{3.4}$$

5. Perhitungan koordinat *object space* pendekatan (X_A, Y_A, Z_A). Ada pun persamaan untuk menentukan koordinat *object space* pendekatan adalah sebagai berikut.

$$\begin{aligned} X_A &= s_{left}x'_{left} + XL_{left} \\ Y_A &= s_{left}y'_{left} + YL_{left} \\ Z_A &= s_{left}z'_{left} + ZL_{left} \end{aligned} \tag{3.5}$$

Persamaan 3.5 adalah persamaan untuk foto kiri. Persamaan yang sama dapat ditulis untuk foto kanan yaitu sebagai berikut.

$$\begin{aligned} X_A &= s_{right}x'_{right} + XL_{right} \\ Y_A &= s_{right}y'_{right} + YL_{right} \\ Z_A &= s_{right}z'_{right} + ZL_{right} \end{aligned} \tag{3.6}$$

6. Perhitungan parameter *r, s, q* untuk tiap titik dengan persamaan sebagai berikut.

$$r = m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L) \tag{3.7}$$

$$s = m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L) \tag{3.8}$$

$$q = m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L) \tag{3.9}$$

7. Setelah parameter *r, s, q* diperoleh maka parameter b_{14}, b_{15}, b_{16}, b_{24}, b_{25}, b_{26} serta parameter *J* dan *K* dapat dihitung dengan persamaan sebagai berikut.

$$b_{14} = \frac{f}{q^2}(rm_{31} - qm_{11}) \quad (3.10)$$

$$b_{24} = \frac{f}{q^2}(sm_{31} - qm_{21}) \quad (3.11)$$

$$b_{15} = \frac{f}{q^2}(rm_{32} - qm_{12}) \quad (3.12)$$

$$b_{25} = \frac{f}{q^2}(sm_{32} - qm_{22}) \quad (3.13)$$

$$b_{16} = \frac{f}{q^2}(rm_{33} - qm_{13}) \quad (3.14)$$

$$b_{26} = \frac{f}{q^2}(sm_{33} - qm_{23}) \quad (3.15)$$

$$J = x_a - x_0 + f\frac{r}{q} \quad (3.16)$$

$$K = y_a - y_0 + f\frac{s}{q} \quad (3.17)$$

8. Setelah semua parameter tersebut dihitung, matrik A dan L dapat dibentuk sebagai berikut.

$$A = \begin{bmatrix} b_{14_a} & b_{15_a} & b_{16_a} \\ b_{24_a} & b_{25_a} & b_{26_a} \end{bmatrix} \qquad L = \begin{bmatrix} J_a \\ K_a \end{bmatrix} \quad (3.18)$$

9. Menghitung matrix *X*. matrix *X* merupakan matrix yang akan berisikan nilai koreksi untuk tiap koordinat titik *object space*.

$$X = (A^T A)^{-1}(A^T L) \quad (3.19)$$

10. Jika nilai koreksi tak berarti lagi (sangat kecil) maka koordinat *objek space* akhir sudah dapat ditentukan. Dan jika sebaliknya maka proses iterasi harus tetap dilakukan hingga parameter-parameter koreksi menjadi sangat kecil.

III.3.3. Penjelasan Diagram *Cross Correlation*

Penjelasan mengenai diagram algoritma *cross correlation* akan dijelaskan lebih rinci sebagai berikut.

1. Diawali dengan pemilihan titik pada foto kiri, kemudian program akan merekam nilai keabuan pada *template*, dan matrix A[i, j] dibentuk.
2. Pemilihan titik pada foto kanan, program akan merekam nilai keabuan pada *matching patch*, dan akan dibentuk matrix B[i, j].
3. Perhitungan nilai *cross correlation* (*r*) dilakukan. Perhitungan nilai *correlation* dilakukan pada *matching patch* yang bergerak didalam *search window*.
4. Titik sekutu dapat ditentukan dengan syarat bahwa posisi titik sekutu adalah posisi titik pada nilai korelasi tertinggi.

III.4. Langkah Kerja

Langkah kerja dalam penelitian akan dibahas pada sub-bab ini. Sub-bab pertama pada sub-bab ini akan dibahas mengenai persiapan data. Sedangkan sub-bab kedua pada sub-bab ini akan dibahas mengenai pembuatan program.

III.4.1. Persiapan Data

Persiapan data merupakan langkah awal sebelum pembuatan program dilakukan. Adapun data yang digunakan adalah foto stereo, contoh dari foto stereo dapat dilihat pada Gambar 3.1.

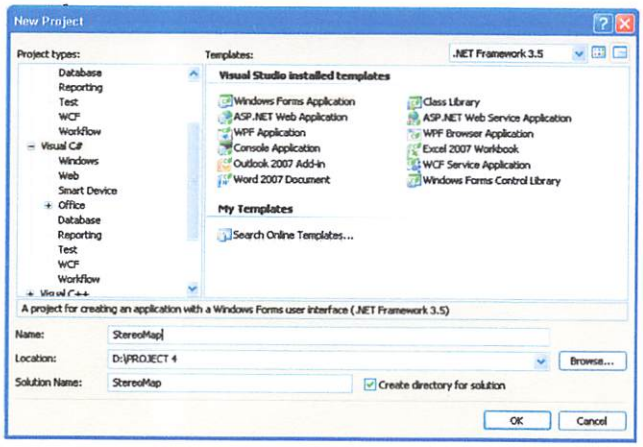
Gambar 3.1 *contoh foto stereo*

III.4.2. Pembuatan Program

Ada bebeapa hal yang perlu diketahui dalam pembuatan suatu program menggunakan Visual Studio dengan bahasa pemrograman C#. hal-hal yang perlu diketahui tersebut akan dibahas dalam sub-bab ini.

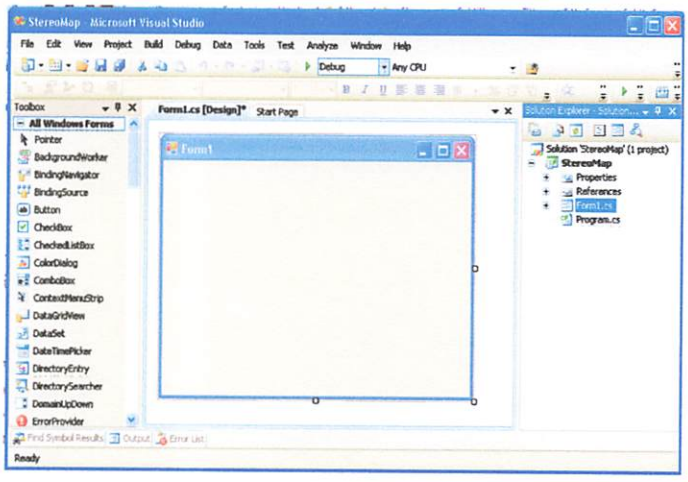
III.4.2.1. Membuat Project Baru

Membuat projek baru pada Visual Studio dengan bahasa pemrograman C#. *Project* baru pada *Microsoft Visual Studio* 2008 dapat dibuat dengan memilih menu *file* → *new* → *project*. Kotak dialog *New Project* akan muncul seperti di bawah ini.



Gambar 3.2 *kotak dialog New Project*

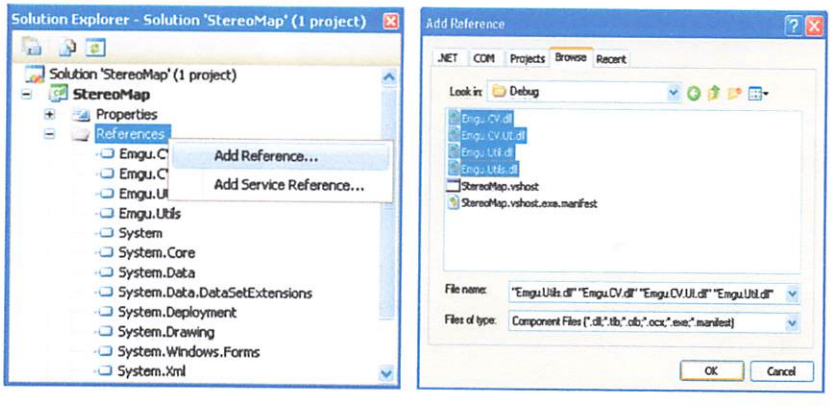
Pada kotak dialog *new project*, pilih *Windows Form Aplication*, beri nama *project*, misalnya *StereoMap*. Pilih direktori penyimpanan dan klik button OK, maka *project* baru akan terbentuk dan akan tampil *form* utama seperti pada Gambar 3.3



Gambar 3.3 *tampilan form utama program*

III.4.2.2. Menambahkan *Library*

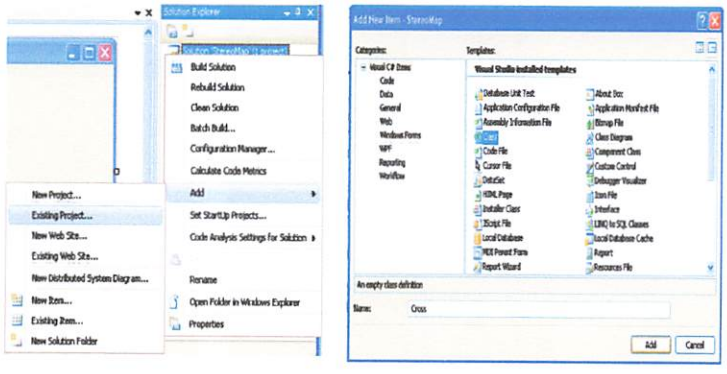
Penambahan *library* pada program bertujuan agar *library* yang sudah ada dapat digunakan kembali kedalam suatu *project* lainnya. Penambahan *library* dapat dilakukan dengan klik kanan *Reference* pada jendela *Solution Explorer*, kemudian pilih *Add Reference*. Pada jendela *Add Reference* pilih *Browse* kemudian pilih direktori penyimpanan *library*. *Library* dapat diseleksi sebelum button OK diklik, secara otomatis *library* yang pilih akan bertambah ke *Reference* project. Cara menambahkan *library* dapat dilihat pada Gambar 3.4.



Gambar 3.4 *penambahan library pada project*

III.4.2.3. Menambahkan Project

Penambahan *project* dapat berupa *project* baru atau juga *project* yang sudah ada. Berikut akan dijelaskan mengenai langkah-langkah penambahan *project* yang sudah ada. Tujuan menambahkan *project* yang sudah ada agar *project* tersebut dapat digunakan kembali dalam suatu program lainnya. Untuk menggunakan kembali suatu *projet*, pada *Solution Explorer* klik kanan pada *Solution,* kemudian pilih *Add* dan pilih *Existing Project* , seperti yang terlihat pada Gambar 3.5.



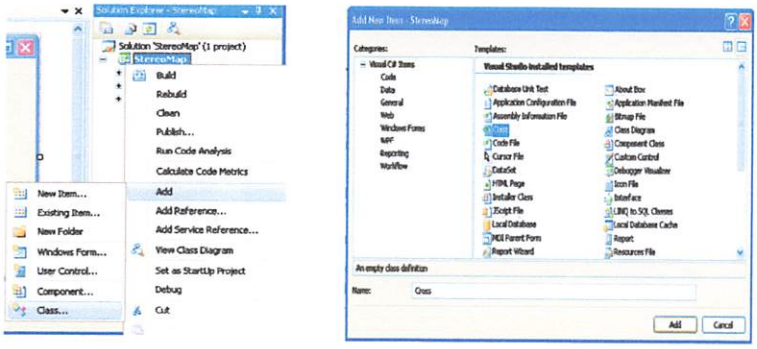
Gambar 3.5 *penambahan suatu project*

Setelah tampil jendela *Add Exsisting Project* , pilih *project* yang ingin ditambahkan, kemudian pilih button *Open*, secara otomatis *project* yang dipilih tadi akan bertambah kedalam jendela *Solution Explorer*. Dan fungsi-fungsi yang terdapat pada *project* tersebut dapat digunakan kembali sesuai kebutuhan.

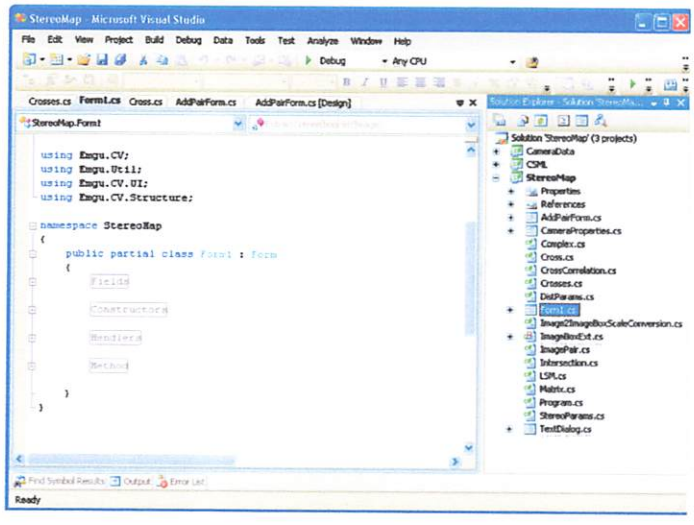
III.4.2.4. Menambahkan Class

Tujuan menggunakan class dalam suatu *project* adalah agar semua kegiatan dalam membuat program dapat terorganisir dengan baik. Seperti yang terlihat pada Gambar 3.6, pada jendela *Solution Explorer* klik kanan pada *project* kemudian

pilih *Add*. Setelah tampil jendela *Add New Items*, pilih item *class* kemudian beri nama *class* misalnya "cross", kemudian klik tombol *Add*, secara otomatis *class* tersebut akan ditambahkan ke dalam *project*. Sebuah *class* dapat berisikan *field, constructor, method,* dan *properties*. Gambar 3.7 merupakan contoh tampilan dari suatu *class*.



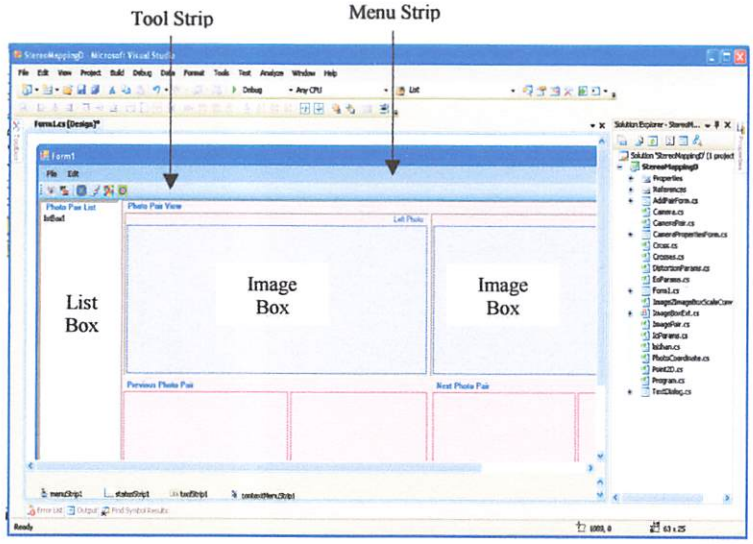
Gambar 3.6 *penambahan suatu class*



Gambar 3.7 *tampilan suatu class*

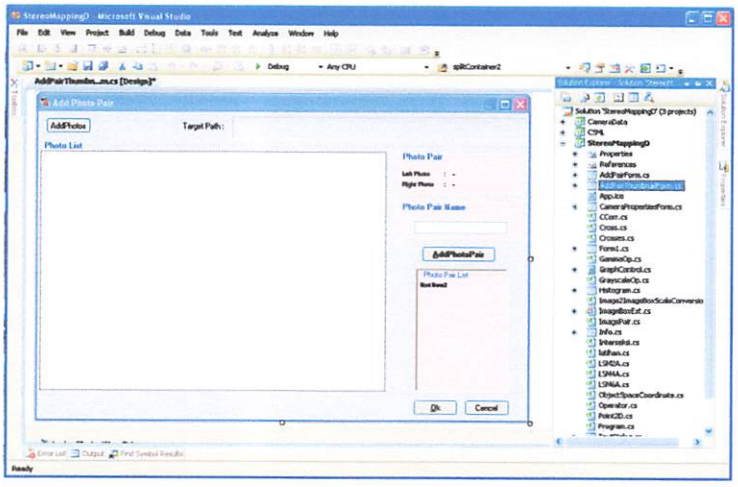
III.4.2.5. Desain Form Utama

Contoh desain *form* utama dapat dilihat pada Gambar 3.8. Desain *form* utama terdiri atas dua buah *image box* dengan tujuan agar foto stereo akan ditampilkan pada kedua *image box* tersebut. List box juga ditempatkan pada *form* utama dengan tujuan agar pada list box ini akan berisi list dari nama pasangan foto stereo. Selain itu, pada *form* utama juga ditempatkan Menu Strip dan Tool Strip.



Gambar 3.8 *contoh tampilan desain form utama*

III.4.2.6. Desain *Form* Add Pair



Gambar 3.9 *contoh tampilan desain form Add Pair*

Gambar 3.9 menampilkan contoh desain *form* Add Pair. *Form* Add Pair dibuat dengan tujuan agar pengguna dapat mengorganisir pasangan foto stereo sesuai keinginan.

III.4.2.7. Desain *Class* untuk *Project* Stereo Map

Agar semua kegiatan pembuatan program dapat terorganisir dengan baik maka *programmer* dapat menggunakan fasilitas *class* yang disediakan oleh Visual Studio. Berikut ini akan dijelaskan masing-masing mengenai *class-class* yang dibentuk untuk pembuatan *interface* program.

1. *Class Cross*

   Class cross mengandung fungsi-fungsi dasar untuk penggambaran tanda cross (+) dan penggambaran suatu label yang nantinya fungsi-fungsi ini dapat digunakan untuk menggambar tanda cross atau pun menggambar label pada suatu titik dengan koordinat pixel tertentu di foto.

2. *Class Crosses*

   Tujuan membentuk *class* Crosses adalah untuk menampung cross dan label. Hal ini dikarenakan pada satu *image* dapat tergambar lebih dari satu tanda cross dan lebih dari satu label.

3. *Class Image2ImageBoxScaleConversion*

   Class ini mengandung fungsi untuk penskalaan. Fungsi penskalaan perlu digunakan karena image yang tampil pada image box adalah *Stretch image* sehingga image tersebut tidak sesuai dengan ukuran image sebenarnya. Perlu adanya proses penskalaan sehingga tidak terjadi kekeliruan dalam menentukan posisi

penggambaran suatau tanda cross maupun proses koleksi posisi koordinat titik pada image.

4. *Class ImagePair*

   Class ImagePair adalah *class* yang mengandung nama file foto, baik foto kiri mau pun foto kanan. Selain itu *class* ini juga mengandung nama dari foto stereo dimana foto stereo terdiri dari foto kiri dan foto kanan.

5. *Class CrossCorrelation*

   Proses cross correlation sendiri bertujuan untuk menemukan titik sekutu pada foto yang bertampalan. Didalam *class CrossCorrelation* akan didefinisikan fungsi-fungsi untuk proses *cross correlation.* Untuk membuat fungsi *cross correlation* dapat mengacu pada algoritma *cross correlation.* Alternativ lain untuk menemukan titik sekutu adalah dengan menggunakan fungsi yang dimiliki oleh Emgu (*computer vision library*).

6. *Class Intersection*

   Dalam *class intersection* akan didefinisikan fungsi untuk proses interseksi. Alur pembuatan fungsi dapat mengacu pada algoritma interseksi.

Agar semua kegiatan pembuatan program dapat berjalan dengan baik maka untuk pengolahan image digital dapat menggunakan fungsi-fungsi yang dimiliki oleh Emgu (*computer vision librari*). Sedangkan untuk suatu proses yang berhubungan dengan komputasi matematik dapat menggunakan fungsi-fungsi yang dimiliki oleh CSML (C# matrix *library*).

BAB IV

HASIL DAN PEMBAHASAN

IV.1. Hasil Pembuatan Program

Pembuatan suatu program tidak terlepas dari pembuatan *interface* dan *coding* program. Untuk itu pada sub-bab ini akan dijelaskan mengenai hasil pembuatan program terkait pembuatan *interface* dan *coding*. Khususnya bagi c*oding* program lebih jelasnya terdapat pada lampiran.

IV.1. Pembentukan *Class*

IV.1.1 *Class Cross*

Class cross mengandung fungsi untuk penggambaran tanda cross (+) dan penggambaran label. *Class* yang disediakan oleh C# untuk menggambarkan suatu objek adalah *class Graphics*. Didalam *class Graphics* terdapat fungsi *DrawLine* dimana fungsi ini dapat digunakan untuk menggambarkan garis. Perpaduan dari dua buah garis yang saling berpotongan akan menghasilkan suatu tanda *cross*. Selain fungsi untuk menggambarkan garis, *class Graphics* juga memiliki fungsi *DrawString* untuk menggambarkan tulisan berupa huruf atau angka. Kedua fungsi ini akan digunakan didalam *class cross*. Contoh cuplikan *source code* fungsi untuk menggambarkan tanda cross dapat dilihat pada Algoritma 4.1, sedangkan *Source code* selengkapnya dapat dilihat pada Lampiran 1-1.

```
private void DrawCrossSign(Graphics g, float x, float y, Pen pen)
{
    //draw - .
    g.TranslateTransform(x, y);
    g.DrawLine(pen, 0, 0, -_radius, 0);
    g.ResetTransform();
    //draw | up
    g.TranslateTransform(x, y);
    g.DrawLine(pen, 0, 0, 0, -_radius);
    g.ResetTransform();
    //draw . -
    g.TranslateTransform(x, y);
    g.DrawLine(pen, 0, 0, _radius, 0);
    g.ResetTransform();
    //draw | down
    g.TranslateTransform(x, y);
    g.DrawLine(pen, 0, 0, 0, _radius);
    g.ResetTransform();
}
```

Algoritma 4.1

IV.1.2. *Class Crosses*

Class crosses dibuat untuk mengoleksi tiap *cross*. Pada suatu *image* tidak hanya tergambar satu tanda *cross* saja, namun terdapat beberapa tanda *cross* pada *image* tersebut. Agar tiap *cross* dapat dikoleksi kedalam suatu variabel maka pada *class crosses* didevinisikan suatu variabel yang bertipe koleksi. C# menyediakan *class System.Collections.Generic.List<T>* sebagai *class* yang dapat digunakan untuk mengoleksi. *Class* ini dapat diaplikasikan untuk kebutuhan dalam mengoleksi *cross*. *Source code* selengkapnya dari *class* ini dapat dilihat pada Lampiran 1-2.

IV.1.3. *Class Image2ImageBoxScaleConversion*

Didalam *class Image2ImageBoxScaleConversion* dibuat fungsi untuk menentukan faktor skala suatu *image* terhadap *image box* dan juga sebaliknya. Contoh cuplikan *source code* fungsi untuk menentukan faktor skala tersebut dapat dilihat pada Algoritma 4.2, sedangkan *source code* selengkapnya dapat dilihat pada Lampiran 1-3.

```
public void SetScale(int imageWidth, int imageHeight, int
                     imageBoxWidth, int imageBoxHeight)
{
    _hzScaleImage2ImageBox = (double)((double)imageBoxWidth /
            (double)imageWidth);
    _vScaleImage2ImageBox = (double)((double)imageBoxHeight /
            (double)imageHeight);
    _hzScaleImageBox2Image = 1.0 / _hzScaleImage2ImageBox;
    _vScaleImageBox2Image = 1.0 / _vScaleImage2ImageBox;
}
```

Algoritma 4.2

IV.1.4. *Class ImagePair*

Class ImagePair berisikan variabel untuk mendefinisikan nama dari foto kiri, nama dari foto kanan, nama dari pasangan foto, serta variabel dari beberapa *class* yang nantinya dapat diakses dari *class ImagePair*. *Source code* selengkapnya dari *class ImagePair* dapat dilihat pada Lampiran 1-4.

IV.1.5. *Class CrossCorrelation*

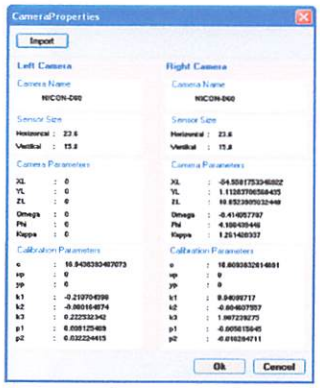
Class cross correlation mengandung fungsi *cross correlation* yang dibuat berdasarkan algoritma *cross correlation*. Ada pun cuplikan *source code* dari fungsi ini dapat dilihat pada Algoritma 4.3, sedangkan *source code* selengkapnya dapat dilihat pada Lampiran 1-5.

```
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 sumGrayValMatchPatch = matching.GetSum().Intensity;
  double sumSquaredGrayValMatchPatch = SetSumSquare(ref matching);

  double SumMult = SumMultiply(ref matching);

  //From Element of Photogrammetry, wolf page 336:
  //Sx=template,Sy=matchingPatch
  double SxSquared = _sumSquareTempValue -
     sumGrayTempValue * _sumGrayTempValue) / elementNumbers);
  double SySquared = sumSquaredGrayValMatchPatch -
      ((sumGrayValMatchPatch * sumGrayValMatchPatch) /
```

Algoritma 4.3

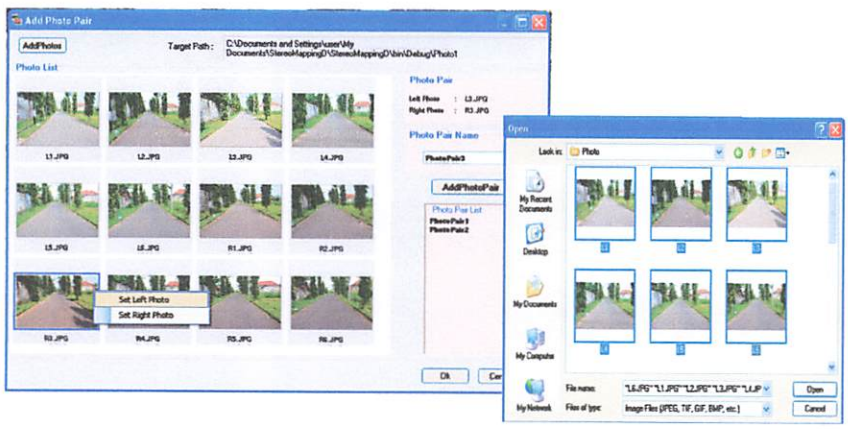
IV.2. Hasil Desain Form Camera Properties

Form Camera Properies adalah suatu *form* yang dibuat dengan tujuan agar pengguna dapat melakukan pengaturan terhadap semua parameter kamera yang diasumsikan sudah diketahui, dalam hal ini parameter-parameter tersebut diperoleh dari proses *stereo calibration.* Adapun tampilan dari *form* Camera Properties dapat dilihat pada cuplikan Gambar 4.4. *Source code* pada *form* ini selengkapnya dapat dilihat pada Lampiran 1-6.



Gambar 4.1 tampillan desain form Camera Propeties

IV.3. Hasil Desain Form Add Pair



Gambar 4.2 *tampilan hasil desain form Add Pair*

Gambar 4.2. menampilkan desain *form* Add-Pair. Melalui *form* Add Pair, pengguna dapat melakukan *input* foto dan dapat mengorganisir foto sesuai kebutuhan. Pada *form* AddrPair, jika button *AddImage* dipilih maka akan tampil jendela *Open File Dialog*. Dari jendela *Open File Dialog* ini dapat dipilih foto-foto yang ingin ditambahkan kedalam *project* . Jika button *Open* diklik maka secara otomatis nama dari foto-foto yang dipilih akan masuk ke dalam *list box* yang berada pada *form* Add-Pair. Penentuan *image* sebagai *image* kiri mau pun *image* kanan serta pemberian nama terhadap pasangan foto dapat dilakukan pada *form* ini.

IV.3. Hasil Desain Form StereoMap

Gambar 4.3 tampillan desain form StereoMap

Form Stereo Map seperti pada cuplikan Gambar 4.4. merupakan *form* yang didesain sebagai *form* utama dimana *form* ini didesain agar foto stereo dapat tampil secara bersamaan pada *image box* pada saat nama pasangan foto yang berada pada *list box* dipilih. Semua proses yang menyangkut aplikasi pengolahan *image*, penentuan titik konjugasi maupun intruksi untuk penentuan koordinat *object space* dengan metode

interseksi terjadi pada *form* ini. Selain itu informasi mengenai koordinat pixel, koordinat foto dan koordinat *object space* tiga dimensi tiap titik dapat dilihat pada *form* ini.

IV.4. Hasil Interseksi

Hasil dari proses interseksi adalah koordinat *object space* tiga dimensi. Dengan program yang dihasilkan proses ini dapat berlangsung secara otomatis pada semua titik yang berada pada pasangan foto pair. Seperti cuplikan gambar 4.3 menunjukkan bahwa titik-titik koordinat *object space* tiga dimensi diperoleh secara otomatis dan dapat dilihat secara langsung untuk semua foto pair.

Ada pun hasil interseksi memiliki tingkat ketelitian tertentu. Pada penelitian ini penyimpangan terhadap ketelitian masih diatas standar ketelitian yang baik yaitu masih di atas dari satu millimeter. Sehingga dapat dianalisa bahwa hal ini dapat disebabkan oleh beberapa faktor, diantaranya adalah belum dikoreksinya koordinat foto tiap titik terhadap parameter distorsi sebelum proses interseksi itu berlangsung. Agar penyimpangan terhadap ketelitian dapat lebih kecil maka faktor ini perlu diperhatikan.

IV.4. Hasil Validasi

Pada penelitian ini, validasi dilakukan dengan sangat sederhana. Validasi dilakukan dengan cara melihat pola penyebaran titik-titik hasil interseksi. Pada kenyataannya dengan menggunakan data foto papan kalibrasi pola penyebaran titik-titik koordinat *object space* hasil interseksi sudah sesuai dengan kondisi nyata. Perbandingan dapat dilihat dari posisi suatu titik koordinat *object space* terhadap titik-titik *object space* lainnya.

BAB V

KESIMPULAN DAN SARAN

V.1. Kesimpulan

Kesimpulan dari hasil penelitian yang dilakukan adalah :

- Penentuan koordinat titik objek tiga dimensi dapat dilakukan dengan proses interseksi. Ada pun proses interseksi ini dapat berlangsung jika parameter-parameter awal untuk proses perhitungan ini sudah diketahui.
- Parameter-parameter yang harus diketahui sebelum proses interseksi berlangsung diantaranya adalah parameter *interior orientation*, parameter *eksterior orientation* kamera kanan terhadap kamera kiri yang dapat diperoleh dari proses stereo kalibrasi, dalam hal ini diasumsikan bahwa sudah diketahui serta informasi titik-titik sekutu yang dapat diperoleh secara digital dari proses *area based matching*
- Mengenai pembuatan program, interface utama adalah suatu kerangka yang dibutuhkan agar proses pembuatan program dapat berlangsung dengan efektif dan efisien. Untuk itu, kerangka pemikiran secara sistematis dan menyeluruh sangat diperlukan.

V.2. Saran

- Untuk meningkatkan ketelitian sebaiknya adanya pemberian koreksi titik-titik koordinat foto terhadap parameter distorsi dimana parameter ini ikut mempengaruhi dalam mengurangi penyimpangan terhadap ketelitian.

- Agar proses pembuatan program dapat berlangsung dengan lancar, maka hal-hal yang mendukung proses pembuatan program diantaranya persiapan algoritma dan penguasaan materi sudah harus disiapkan.

DAFTAR PUSTAKA

Atkinson, K.B. (2001) Close Range Photogrammetry and Machine Vision (*Whittles Publishing Services*)

Bradski G. & Kaehler A. (2008) Learning OpenCV Computer Vision with the OpenCV Library (*O'Reilly Media, Inc*)

Brown, E. (2006) Windows Form in Action (*Manning Publication*)

Champian, M.d. & Patrick B.G. (2005) C# 2.0: Practical Guide for Programmers (*Morgan Kaufman Publishers*)

Church, E. (1950) Illustrative Solutions of Analytic Problems in Aerial Photogrammetry (*Ohio State University Research Foundation*)

Church, E. (1945) Revised Geometry of the Aerial Photograph (*Syracuse University Press*)

Cipolla, R., Battiato, S. & Farinella, G.M. (2010) Computer Vision Detection, Recognition & Reconstruction (*Springer-Verlag Berlin Heidelberg*)

Cyganek, B. & Siebert J.P. (2009) An Introduction to 3D Computer Vision Techniques and Algorithms (*A John Wiley & Sons, Ltd, Publication*)

Ebner, H., Fritsch, D. & Heipke, C. (1991) Digital Photogrammetric System (*Wichmann*)

Ebner, H., Heipke. C. (1988) Integration of Digital Image Matching and Object Surface Reconstruction (*In International Archives of Photogrammetry and Remote Sensing: International Society for Photogrammetry and Remote Sensing*)

Forstner, W. (1982) On the Geometric Precision of Digital Correlation (*In International Archives of Photogrammetry and Remote Sensing: International Society for Photogrammetry and Remote Sensing*)

Fryer, J.G. & Shortis, M.R. (1994) Close Range Techniques and Machine Vision (*International Archive of Photogrammetry and Remote Sensing*)

Grun, A. (1994) Digital Closerange Photogrammetry : Progress Through Automation (*Internarional Archives of Photogrammetry and Remote Senring, 30(5): 122-135*)

Hannah, M. J. (1989) A System for Digital Stereo Image Matching (*Photogrammetric Engineering and Remote Sensing*)

Heipke, C. (1992) A Global Approach for Least_Square Image Matching and Surface Reconstruction in Object Space (*Photogrammetric Engineering and Remote Sensing*)

Heipke, C. (1997) Automation of Interior, Relative, and Absolut Orientation (*Journal of Photogrammetry and Remote Sensing*)

Helava, U. V. (1988) Object-Space Least-Squares Correlation (*Photogrammetric Engineering and Remote Sensing*)

Jue, L. (2008) Research on close range photogrammetry with big rotation angle. *IAPRS*, 37

Kasser, M. & Egles Y. (2002) Digital Photogrammetry (*Taylor & Francis*)

Karara, H.M. (1989) Non-Topographic Photogrammetry : Second edition, (*American Society For Photogrammetry and Remote Sensing*)

Kingsley_Hughes, A. (2005) C# 2005 programmer's Reference (*Wiley Publishing Inc.*)

MacDonald, M., Jones, A. & Rajan, R. (2006) Visual C# 2005 Recipes : A Problem-Solution Approach (*Apress*)

Mikhail, E.M., Bethel, J.S. & Mcglone, J.C. (2001) Introduction to modern photogrammetry (*United States of America, John Willey & Sons*).

Salehi, S. (2006) ImageMagick Tricks, Web Image Effects from the Command Line and PHP (*Packt Publishing*)

Schenk, T. (1999) Digital Photogrammetry : Volume I (*TerraScience, USA*)

Schenk, T. (1999) Matching Surfaces (*Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, Columbus*)

Sharp, J. (2008) Visual C# 2008 Step by Step (*Microsoft Press*)

Watson, K., Nagel, C., Pedersen, J.H., Reid, J.D., Skinner, M. & White, E. (2008) Beginning Microsoft Visual C# 2008 (*Wiley Publishing, Inc., Indianapolis, Indiana*)

Wolf, P.R. & Dewit, B.A. (2000) Element of photogrammetry with applications in gis : Third edition, (*Madison, McGraw-Hill Companies*).

Wolf, P.R. (1993) Elemen fotogrametri dengan interpretasi foto udara dan penginderaan jauh : Edisi kedua, (D. *Gunadi, M. S. Drs. Totok Gunawan & D. Zuharnen, Trans.*) (*Bulaksumur, Yogyakarta, Gadjah Mada University Press*)

Zhang, C. & Yao, W. (2008) The comparison of 3d anlysis between photogrammetry and computer vision. IAPRS, 37

Lampiran 1-1

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

using StereoMappingD.CameraData;


namespace StereoMappingD
{
    public class Cross
    {
        #region Fields
        //centre point of the cross
        //ALL DONE IN IMAGE COORDINATES!!!!!

        //Label stuff
        private String _label;

        private PointF _position;

        protected Rectangle _boundRect;
        private const int _radius = 3;

        protected bool _highlighted;
        Color _highlightedColor;


        //photo coordinate variable
        private double _x;
        private double _y;

        private int _imageW;
        private int _imageH;

        #endregion Fields

        #region Constructors

        //set
        public Cross(PointF centre, String label, Camera camera)
        {
            _position = centre;
            _highlighted = false;
            _highlightedColor = Color.Green;

            //Label stuff
            _label = label;

            this._boundRect = new Rectangle(
                (int)(_position.X + 0.5f) - _radius,
                (int)(_position.Y + 0.5f) - _radius,
                (int)(18 * _radius),
                (int)(18 * _radius));
            if (_boundRect.Width < 2) _boundRect.Width = 2;
            if (_boundRect.Height < 2) _boundRect.Height = 2;

            //_imageW = imageSizeW;
            //_imageH = imageSizeH;

            Pixel2PhotoCoord(centre, camera,_imageW,_imageH);
        }
```

```
        //set when read project
        public Cross(PointF centre, String label, Camera camera, int imageSizeW, int
    imageSizeH)
        {
            _position = centre;
            _highlighted = false;
            _highlightedColor = Color.Green;

            //Label stuff
            _label = label;

            this._boundRect = new Rectangle(
                (int)(_position.X + 0.5f) - _radius,
                (int)(_position.Y + 0.5f) - _radius,
                (int)(18 * _radius),
                (int)(18 * _radius));
            if (_boundRect.Width < 2) _boundRect.Width = 2;
            if (_boundRect.Height < 2) _boundRect.Height = 2;

            _imageW = imageSizeW;
            _imageH = imageSizeH;

            Pixel2PhotoCoord(centre, camera, _imageW, _imageH);
        }



        public Cross(int imageW, int ImageH)
        {
            _imageW = imageW;
            _imageH = ImageH;
        }


        #endregion Constructors

        #region Properties
        public String LabelName
        {
            get { return _label; }
            set { _label = value; }
        }

        public PointF PixelCoordinate
        {
            get { return _position; }
            set { _position = value; }
        }

        public double xPhoto
        {
            get { return _x; }
            set
            {
                _x = value;
            }
        }

        public double yPhoto
        {
            get { return _y; }
            set
            {
                _y = value;
            }
        }
```

```
        public Rectangle BoundingRectangle
        {
            get
            {
                return Rectangle.Inflate(_boundRect, _radius, _radius);
            }
            set
            {
                _boundRect = value;
            }
        }

        public bool Highlighted
        {
            get { return _highlighted; }
            set { _highlighted = value; }
        }

        #endregion Properties

        #region Methods
        public bool Hit(Point p)
        {
            return _boundRect.Contains(p);
        }

        //draw cross sign from extracted corner on imageBox
        public void Draw(Graphics g, double xScale, double yScale, Pen pen, Font font,
    SolidBrush brush)
        {
            pen.Color = _highlighted ? _highlightedColor : Color.Red;
            brush.Color = _highlighted ? _highlightedColor : Color.Red;
            float x = _position.X * (float)xScale;
            float y = _position.Y * (float)yScale;

            DrawCrossSign(g, x, y, pen);

            //Draw Text
            DrawLabel(g, x, y, pen, font, brush);
        }

        private void DrawCrossSign(Graphics g, float x, float y, Pen pen)
        {
            //draw - .
            g.TranslateTransform(x, y);
            g.DrawLine(pen, 0, 0, -_radius, 0);
            g.ResetTransform();
            //draw | up
            g.TranslateTransform(x, y);
            g.DrawLine(pen, 0, 0, 0, -_radius);
            g.ResetTransform();
            //draw . -
            g.TranslateTransform(x, y);
            g.DrawLine(pen, 0, 0, _radius, 0);
            g.ResetTransform();
            //draw | down
            g.TranslateTransform(x, y);
            g.DrawLine(pen, 0, 0, 0, _radius);
            g.ResetTransform();
        }

        private void DrawLabel(Graphics g, float x, float y, Pen pen, Font font,
    SolidBrush brush)
        {
            g.TranslateTransform(x, y);
```

```
            String temp = String.Empty;
            if (_label == "xx")
                g.DrawString(temp, font, brush, 0.0f, 0.0f);
            else
                g.DrawString(_label, font, brush, 0.0f, 0.0f);
            g.ResetTransform();
            //Trace.WriteLine("Draw Label");
        }


        public void SetImgParam(int imageW, int imageH)
        {
            _imageW = imageW;
            _imageH = imageH;
        }

        public void Pixel2PhotoCoord(PointF pt, Camera camera, int imageSizeW, int
    imageSizeH)
        {
            _x = (((double)pt.X) - ((imageSizeW / 2) - 0.5)) * (camera.IoParams.SensorHz
    / imageSizeW);
            _y = ((((double)pt.Y) - ((imageSizeH / 2) - 0.5)) * (camera.IoParams.SensorVt
     / imageSizeH)) * -1;
        }

        public override string ToString()
        {
            string text = String.Format("{0} {1}", _x, _y);
            return text;
        }


        #endregion Methods

    }
}
```

Lampiran 1-2

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


namespace StereoMappingD
{
    public class Crosses
    {

        #region Fields
        private Pen _pen;

        //Label stuff
        private SolidBrush _brush;
        private Font _font;

        private List<Cross> _crossPoints;

        private double _hzScaleImage2ImageBox;
        private double _vScaleImage2ImageBox;
        private double _hzScaleImageBox2Image;
        private double _vScaleImageBox2Image;

        #endregion Fields


        #region Constructors
        public Crosses()
        {
            //Label stuff
            _brush = new SolidBrush(Color.Red);
            _font = new Font("SanSerif", 7);

            _pen = new Pen(Color.Red, 1.0f);
            //_highlightedColor = Color.Green;

            _crossPoints = new List<Cross>();
        }

        #endregion Constructors


        #region Properties
        public List<Cross> CrossPoints
        {
            get { return _crossPoints; }
            set { _crossPoints = value; }
        }

        public double HzScaleImage2ImageBox
        {
            get { return _hzScaleImage2ImageBox; }
        }

        public double VScaleImage2ImageBox
        {
            get { return _vScaleImage2ImageBox; }
        }

        public double HzScaleImageBox2Image
        {
            get { return _hzScaleImageBox2Image; }
```

```
        }

        public double VScaleImageBox2Image
        {
            get { return _vScaleImageBox2Image; }
        }

        #endregion Properties


        #region Methods
        //Draw cross on ImageBox
        public void DrawCrossOnImageBox(Graphics g)
        {
            //draw crosses on left imageBox
            //_pen.Color = _highlighted ? _highlightedColor : Color.Red;
            foreach (Cross cross in _crossPoints)
                cross.Draw(g, _hzScaleImage2ImageBox, _vScaleImage2ImageBox, _pen, _font,
    _brush);

        }

        public void SetScaleForDisplay(double hzScaleI2IB, double vScaleI2IB, double
    hzScaleIB2I, double vScaleIB2I)
        {
            _hzScaleImage2ImageBox = hzScaleI2IB;
            _vScaleImage2ImageBox = vScaleI2IB;
            _hzScaleImageBox2Image = hzScaleIB2I;
            _vScaleImageBox2Image = vScaleIB2I;
        }

        public Cross HitCross(Point p)
        {
            foreach (Cross cross in _crossPoints)
            {
                if (cross.Hit(p))
                    return cross;
            }

            return null;
        }

        public Cross Delete(Cross element)
        {
            _crossPoints.Remove(element);

            return element;
        }


        #endregion Methods

    }
}
```

Lampiran 1-3

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace StereoMappingD
{
    public class Image2ImageBoxScaleConversion
    {

        #region Fields
        private double _hzScaleImage2ImageBox;
        private double _vScaleImage2ImageBox;
        private double _hzScaleImageBox2Image;
        private double _vScaleImageBox2Image;

        #endregion Fields

        #region Constructors
        public Image2ImageBoxScaleConversion()
        {
            _hzScaleImage2ImageBox = 0.0;
            _vScaleImage2ImageBox = 0.0;
            _hzScaleImageBox2Image = 0.0;
            _vScaleImageBox2Image = 0.0;
        }

        #endregion Constructors

        #region Properties
        public double HorisontalScaleImage2ImageBox
        {
            get { return _hzScaleImage2ImageBox; }
        }

        public double VerticalScaleImage2ImageBox
        {
            get { return _vScaleImage2ImageBox; }
        }

        public double HorisontalScaleImageBox2Image
        {
            get { return _hzScaleImageBox2Image; }
        }

        public double VerticalScaleImageBox2Image
        {
            get { return _vScaleImageBox2Image; }
        }
        #endregion Properties

        #region Methods
        public void SetScale(int imageWidth, int imageHeight, int imageBoxWidth, int
imageBoxHeight)
        {
            _hzScaleImage2ImageBox = (double)((double)imageBoxWidth / (double)imageWidth)
;
            _vScaleImage2ImageBox = (double)((double)imageBoxHeight / (double)
imageHeight);
            _hzScaleImageBox2Image = 1.0 / _hzScaleImage2ImageBox;
            _vScaleImageBox2Image = 1.0 / _vScaleImage2ImageBox;
        }

        #endregion Methods
    }
```

Lampiran 1-4

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

using StereoMappingD.CameraData;


namespace StereoMappingD
{
    public class ImagePair
    {

        #region Fields
        String _leftPhoto;
        String _rightPhoto;
        String _pairName;

        Crosses _crossesLeftPhoto;
        Crosses _crossesRightPhoto;

        Interseksi _intersection;
        ObjectSpaceCoordinate _3DobjSpaceCoord;

        private ImageSize _imageSizeLeft;
        private ImageSize _imageSizeRight;

        #endregion Fields


        #region Constructors
        public ImagePair()
        {
            _leftPhoto = String.Empty;
            _rightPhoto = String.Empty;
            _pairName = String.Empty;

            _crossesLeftPhoto = new Crosses();
            _crossesRightPhoto = new Crosses();


            _intersection = new Interseksi();
            _3DobjSpaceCoord = new ObjectSpaceCoordinate();

            _imageSizeLeft = new ImageSize();
            _imageSizeRight = new ImageSize();
        }

        //to set on AddPairThumbnailForm, with image size
        public ImagePair(String leftPhoto, String rightPhoto, String pairName, int
    imageSizeLeftW, int imageSizeLeftH, int imageSizeRightW, int imageSizeRightH)
        {
            _leftPhoto = leftPhoto;
            _rightPhoto = rightPhoto;
            _pairName = pairName;

            _crossesLeftPhoto = new Crosses();
            _crossesRightPhoto = new Crosses();

            _intersection = new Interseksi();
            _3DobjSpaceCoord = new ObjectSpaceCoordinate();

            _imageSizeLeft = new ImageSize();
            _imageSizeRight = new ImageSize();
```

```
            SetImageSize(imageSizeLeftW, imageSizeLeftH, imageSizeRightW,
    imageSizeRightH);
        }


        //to set on AddPairForm without image size
        public ImagePair(String leftPhoto, String rightPhoto, String pairName)
        {
            _leftPhoto = leftPhoto;
            _rightPhoto = rightPhoto;
            _pairName = pairName;

            _crossesLeftPhoto = new Crosses();
            _crossesRightPhoto = new Crosses();

            _intersection = new Interseksi();
            _3DobjSpaceCoord = new ObjectSpaceCoordinate();

        }

        #endregion Constructors


        #region Properties
        public String LeftPhoto
        {
            get { return _leftPhoto; }
            set { _leftPhoto = value; }
        }

        public String RightPhoto
        {
            get { return _rightPhoto; }
            set { _rightPhoto = value; }
        }

        public String PhotoPairName
        {
            get { return _pairName; }
            set { _pairName = value; }
        }

        public Crosses CrossesLeftPhoto
        {
            get { return _crossesLeftPhoto; }
        }

        public Crosses CrossesRightPhoto
        {
            get { return _crossesRightPhoto; }
        }

        public Interseksi Intersection
        {
            get { return _intersection; }
        }

        public ObjectSpaceCoordinate ObjectSpaceCoord
        {
            get { return _3DobjSpaceCoord; }
            set { _3DobjSpaceCoord = value; }
        }

        public ImageSize ImageSizeLeft
        {
            get { return _imageSizeLeft; }
```

```
            set { _imageSizeLeft = value; }
        }

        public ImageSize ImageSizeRight
        {
            get { return _imageSizeRight; }
            set { _imageSizeRight = value; }
        }

        #endregion Properties


        #region Methods
        public Cross GetCross(bool activeLeftImageBox, Point p)
        {
            if (activeLeftImageBox)//leftimage
                return _crossesLeftPhoto.HitCross(p);

            else
                return _crossesRightPhoto.HitCross(p);

        }

        public void SetImageSize(int imageSizeLeftW, int imageSizeLeftH, int
    imageSizeRightW, int imageSizeRightH)
        {
            _imageSizeLeft.ImageSizeW = imageSizeLeftW;
            _imageSizeLeft.ImageSizeH = imageSizeLeftH;
            _imageSizeRight.ImageSizeW = imageSizeRightW;
            _imageSizeRight.ImageSizeH = imageSizeRightH;
        }


        #endregion Methods

    }
}
```

Lampiran 1-5

```
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;

namespace StereoMappingD
{
    public class CCorr
    {
        #region Fields
        double _correlationCoeficient;
        Point _matchedPoint;
        Image<Gray, Byte> _template;
        Image<Gray, Byte> _cropMatchImg;     //cropp image for manualy image matching

        double _sumGrayValTemplate;
        double _sumSquaredValTemplate;
        #endregion


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

        public CCorr(ref Image<Gray, Byte> source, ref Image<Gray, Byte> matchingImg,
    Point2D centerTemplate)
            :this()
        {
            Trace.WriteLine("\t\t-------------------- CCorr constructor is called -------
    --------------------------------");
            //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
            Point centreTempl = new Point((int)(centerTemplate.X + 0.5), (int)
    (centerTemplate.Y + 0.5));
            Trace.WriteLine("centreTempl: " + centreTempl.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;
        _template.Save(@"D:\PENULISAN T U G A S AKHIR\PROGRAM Kalibrasi & Interseksi\
RapidMap\StereoMappingD\StereoMappingD\bin\Debug\template.jpg");
        //Perform Photogrammetric CrossCorrelation
        //PhotogrammetricNormalisedCrossCorrelation(ref matchingImg);

    }
    #endregion


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

    public Point MatchedPoint
    {
        get
        {
            return _matchedPoint;
        }
    }


    #endregion


    #region Methods
    public bool PhotogrammetricNormalisedCrossCorrelation(ref  Image<Gray, Byte>
matchingImg)
    {
        //create a searching window which size of 15 x 15
        Point initialMatchedPt = new Point();
        initialMatchedPt.X = _matchedPoint.X;
        initialMatchedPt.Y = _matchedPoint.Y;
        double maxCorrVal = 0.680;
        bool result = false;
        for (int row = initialMatchedPt.Y - 7; row <= initialMatchedPt.Y + 7; row++)
        {
            for (int col = initialMatchedPt.X - 7; col <= initialMatchedPt.X + 7; 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;
    }



    //call on
    public bool PhotogrammetricNormalisedCrossCorrelationManualy(Point
approxsimationMatchedPoint, ref  Image<Gray, Byte> matchingImg)
    {
        //create a searching window which size of 31 x 31
        Point initialMatchedPt = new Point();
        initialMatchedPt.X = approxsimationMatchedPoint.X;
        initialMatchedPt.Y = approxsimationMatchedPoint.Y;
        double maxCorrVal = 0.680;
        bool result = false;
        for (int row = initialMatchedPt.Y - 15; row <= initialMatchedPt.Y + 15; row+
+)
        {
            for (int col = initialMatchedPt.X - 15; col <= initialMatchedPt.X + 15;
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, int x, 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("");
        }
    }


    // crop image for manualy image matching
    public Image<Gray, Byte> setMatchingImageManualy(Point approxsimationMatchedPoint
```

```
    ,ref Image<Gray, Byte> sourceMathingImg)
        {
            // create a matching image for manualy image matching which size of 31 x 31
            Rectangle rectMatchingImg = new Rectangle((int)((approxsimationMatchedPoint.X
     + 0.5) - 15), (int)((approxsimationMatchedPoint.Y + 0.5) - 15), 31, 31);
            //centre mathingImg in integer
            Point centreMatchingImage = new Point((int)(approxsimationMatchedPoint.X + 0.
    5), (int)(approxsimationMatchedPoint.Y + 0.5));
            Trace.WriteLine("centreMatchingImage: " + centreMatchingImage.ToString());

            _cropMatchImg = sourceMathingImg.Copy(rectMatchingImg);
            _cropMatchImg.ROI.Offset(rectMatchingImg.X, rectMatchingImg.Y);
            _cropMatchImg.Save(@"D:\PENULISAN T U G A S AKHIR\PROGRAM Kalibrasi &
    Interseksi\RapidMap\StereoMappingD\StereoMappingD\bin\Debug\matchingImgCropp.jpg");
            return _cropMatchImg;
        }

        #endregion

    }
}
```

Lampiran 1-6

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Diagnostics;

namespace StereoMappingD
{
    public partial class CameraPropertiesForm : Form
    {

        #region Fields
        string _camNameLeft;
        double _sensorVtLeft;
        double _sensorHzLeft;
        double _cLeft;
        double _xpLeft;
        double _ypLeft;
        double _k1Left;
        double _k2Left;
        double _k3Left;
        double _p1Left;
        double _p2left;
        //Eo left
        double _XLleft;
        double _YLleft;
        double _ZLleft;
        double _omegaLeft;
        double _phiLeft;
        double _kappaLeft;

        string _camNameRight;
        double _sensorrVtRight;
        double _sensorHzRight;
        double _cRight;
        double _xpRight;
        double _ypRight;
        double _k1Right;
        double _k2Right;
        double _k3Right;
        double _p1Right;
        double _p2Right;
        //Eo right
        double _XLright;
        double _YLright;
        double _ZLright;
        double _omegaRight;
        double _phiRight;
        double _kappaRight;

        #endregion Fields



        #region Constructors
        public CameraPropertiesForm()
        {
            InitializeComponent();
            _camNameLeft = " ";
            _sensorVtLeft = 0.0;
            _sensorHzLeft = 0.0;
```

```
            _cLeft = 0.0;
            _xpLeft = 0.0;
            _ypLeft = 0.0;
            _k1Left = 0.0;
            _k2Left = 0.0;
            _k3Left = 0.0;
            _p1Left = 0.0;
            _p2left = 0.0;
            _XLleft = 0.0;
            _YLleft = 0.0;
            _ZLleft = 0.0;
            _omegaLeft = 0.0;
            _phiLeft = 0.0;
            _kappaLeft = 0.0;

            _camNameRight = " ";
            _sensorrVtRight = 0.0;
            _sensorHzRight = 0.0;
            _cRight = 0.0;
            _xpRight = 0.0;
            _ypRight = 0.0;
            _k1Right = 0.0;
            _k2Right = 0.0;
            _k3Right = 0.0;
            _p1Right = 0.0;
            _p2Right = 0.0;
            _omegaRight = 0.0;
            _phiRight = 0.0;
            _kappaRight = 0.0;
        }

        public CameraPropertiesForm(double omegaR, double phiR, double kappaR)
        {
            _omegaRight = omegaR;
            _phiRight = phiR;
            _kappaRight = kappaR;
        }

        #endregion Constructors


        #region Properties
        public string CameraNameLeft
        {
            get { return _camNameLeft; }
            set { _camNameLeft = value; }
        }
        public double SensorVtLeft
        {
            get { return _sensorVtLeft; }
            set { _sensorVtLeft = value; }
        }
        public double SensorHzLeft
        {
            get { return _sensorHzLeft; }
            set { _sensorHzLeft = value; }
        }
        public double CLeft
        {
            get { return _cLeft; }
            set { _cLeft = value; }
        }
        public double XpLeft
        {
            get { return _xpLeft; }
```

```
            set { _xpLeft = value; }
        }
        public double YpLeft
        {
            get { return _ypLeft; }
            set { _ypLeft = value; }
        }
        public double K1Left
        {
            get { return _k1Left; }
            set { _k1Left = value; }
        }
        public double K2Left
        {
            get { return _k2Left; }
            set { _k2Left = value; }
        }
        public double K3Left
        {
            get { return _k3Left; }
            set { _k3Left = value; }
        }
        public double P1Left
        {
            get { return _p1Left; }
            set { _p1Left = value; }
        }
        public double P2Left
        {
            get { return _p2left; }
            set { _p2left = value; }
        }
        public double XLLeft
        {
            get { return _XLleft; }
            set { _XLleft = value; }
        }
        public double YLLeft
        {
            get { return _YLleft; }
            set { _YLleft = value; }
        }
        public double ZLLeft
        {
            get { return _ZLleft; }
            set { _ZLleft = value; }
        }
        public double OmegaLeft
        {
            get { return _omegaLeft; }
            set { _omegaLeft = value; }
        }
        public double PhiLeft
        {
            get { return _phiLeft; }
            set { _phiLeft = value; }
        }
        public double KappaLeft
        {
            get { return _kappaLeft; }
            set { _kappaLeft = value; }
        }

        public string CameraNameRight
        {
            get { return _camNameRight; }
```

```
        set { _camNameRight = value; }
    }
    public double SensorVtRight
    {
        get { return _sensorrVtRight; }
        set { _sensorrVtRight = value; }
    }
    public double SensorHzRight
    {
        get { return _sensorHzRight; }
        set { _sensorHzRight = value; }
    }
    public double CRight
    {
        get { return _cRight; }
        set { _cRight = value; }
    }
    public double XpRight
    {
        get { return _xpRight; }
        set { _xpRight = value; }
    }
    public double YpRight
    {
        get { return _ypRight; }
        set { _ypRight = value; }
    }
    public double K1Right
    {
        get { return _k1Right; }
        set { _k1Right = value; }
    }
    public double K2Right
    {
        get { return _k2Right; }
        set { _k2Right = value; }
    }
    public double K3Right
    {
        get { return _k3Right; }
        set { _k3Right = value; }
    }
    public double P1Right
    {
        get { return _p1Right; }
        set { _p1Right = value; }
    }
    public double P2Right
    {
        get { return _p2Right; }
        set { _p2Right = value; }
    }
    public double XLRight
    {
        get { return _XLright; }
        set { _XLright = value; }
    }
    public double YLRight
    {
        get { return _YLright; }
        set { _YLright = value; }
    }
    public double ZLRight
    {
        get { return _ZLright; }
        set { _ZLright = value; }
```

```
        }
        public double OmegaRight
        {
            get { return _omegaRight; }
            set { _omegaRight = value; }
        }
        public double PhiRight
        {
            get { return _phiRight; }
            set { _phiRight = value; }
        }
        public double KappaRight
        {
            get { return _kappaRight; }
            set { _kappaRight = value; }
        }

        #endregion Properties



        #region Hendler
        private void importButton_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog dlg = new OpenFileDialog())
            {
                dlg.Title = "Camera Properties";
                dlg.Multiselect = false;
                dlg.DefaultExt = "txt";
                dlg.Filter = "Album files (*.txt)|*.txt|"
                    + "All files|*.*";
                dlg.InitialDirectory = Environment.CurrentDirectory;
                dlg.RestoreDirectory = true;

                if (dlg.ShowDialog() == DialogResult.OK)
                {
                    string sr = null;
                    sr = dlg.FileName;
                    string[] readData = File.ReadAllLines(sr);
                    Trace.WriteLine(readData.Count<string>());

                    for (int i = 0; i < readData.Count(); i++)
                    {
                        if (i == 0)
                        {
                            String[] firstLineData = readData[0].Split(new char[] { ' ', ↙
'\t' });
                            if (firstLineData.Length == 1)
                            {
                                _camNameLeft = (firstLineData[0]);
                            }
                            cameraNameLeftLabel.Text = CameraNameLeft.ToString();
                        }

                        else if (i == 1)
                        {
                            String[] secondLineData = readData[1].Split(new char[] { ' ',↙
'\t' });
                            if (secondLineData.Length == 2)
                            {
                                _sensorHzLeft = double.Parse(secondLineData[0]);
                                _sensorVtLeft = double.Parse(secondLineData[1]);
                            }
                            sensorHzLeftLabel.Text = SensorHzLeft.ToString();
                            sensorVtLeftLabel.Text = SensorVtLeft.ToString();
                        }
```

```
            else if (i == 2)
            {
                String[] thirdLineData = readData[2].Split(new char[] { ' ', ↙
'\t' });
                if (thirdLineData.Length == 6)
                {
                    _XLleft = double.Parse(thirdLineData[0]);
                    _YLleft = double.Parse(thirdLineData[1]);
                    _ZLleft = double.Parse(thirdLineData[2]);
                    _omegaLeft = double.Parse(thirdLineData[3]);
                    _phiLeft = double.Parse(thirdLineData[4]);
                    _kappaLeft = double.Parse(thirdLineData[5]);
                }

                xlLeftlabel.Text = XLLeft.ToString();
                ylLeftlabel.Text = YLLeft.ToString();
                zlLeftLabel.Text = ZLLeft.ToString();
                omegaLeftLabel.Text = OmegaLeft.ToString();
                phiLeftLabel.Text = PhiLeft.ToString();
                kappaLeftLabel.Text = KappaLeft.ToString();
            }

            else if (i == 3)
            {
                String[] forthLineData = readData[3].Split(new char[] { ' ', ↙
'\t' });
                if (forthLineData.Length == 3)
                {
                    _cLeft = double.Parse(forthLineData[0]);
                    _xpLeft = double.Parse(forthLineData[1]);
                    _ypLeft = double.Parse(forthLineData[2]); ;
                }

                cLeftLabel.Text = CLeft.ToString();
                xpLeftLabel.Text = XpLeft.ToString();
                ypLeftLabel.Text = YpLeft.ToString();
            }

            else if (i == 4)
            {
                String[] fifthLineData = readData[4].Split(new char[] { ' ', ↙
'\t' });
                if (fifthLineData.Length == 5)
                {
                    _k1Left = double.Parse(fifthLineData[0]);
                    _k2Left = double.Parse(fifthLineData[1]);
                    _k3Left = double.Parse(fifthLineData[2]);
                    _p1Left = double.Parse(fifthLineData[3]);
                    _p2left = double.Parse(fifthLineData[4]);
                }

                k1LeftLabel.Text = K1Left.ToString();
                k2LeftLabel.Text = K2Left.ToString();
                k3LeftLabel.Text = K3Left.ToString();
                p1LeftLabel.Text = P1Left.ToString();
                p2LeftLabel.Text = P2Left.ToString();
            }

            else if (i == 5)
            {
                String[] sixthLineData = readData[5].Split(new char[] { ' ', ↙
'\t' });
                if (sixthLineData.Length == 1)
                {
                    _camNameRight = (sixthLineData[0]);
```

```
                }
                cameraNameRightLabel.Text = CameraNameRight.ToString();
            }

            else if (i == 6)
            {
                String[] seventhLineData = readData[6].Split(new char[] { ' '↙
, '\t' });
                if (seventhLineData.Length == 2)
                {
                    _sensorHzRight = double.Parse(seventhLineData[0]);
                    _sensorrVtRight = double.Parse(seventhLineData[1]);
                }
                sensorHzRightLabel.Text = SensorHzRight.ToString();
                sensorVtRightLabel.Text = SensorVtRight.ToString();
            }

            else if (i == 7)
            {
                String[] eighthLineData = readData[7].Split(new char[] { ' ',↙
'\t' });
                if (eighthLineData.Length == 6)
                {
                    _XLright = double.Parse(eighthLineData[0]);
                    _YLright = double.Parse(eighthLineData[1]);
                    _ZLright = double.Parse(eighthLineData[2]);
                    _omegaRight = double.Parse(eighthLineData[3]);
                    _phiRight = double.Parse(eighthLineData[4]);
                    _kappaRight = double.Parse(eighthLineData[5]);

                }
                xlRightLabel.Text = XLRight.ToString();
                ylRightLabel.Text = YLRight.ToString();
                zlRightLabel.Text = ZLRight.ToString();
                omegaRightLabel.Text = OmegaRight.ToString();
                phiRightLabel.Text = PhiRight.ToString();
                kappaRightLabel.Text = KappaRight.ToString();
            }

            else if (i == 8)
            {
                String[] ninethLineData = readData[8].Split(new char[] { ' ',↙
'\t' });
                if (ninethLineData.Length == 3)
                {
                    _cRight = double.Parse(ninethLineData[0]);
                    _xpRight = double.Parse(ninethLineData[1]);
                    _ypRight = double.Parse(ninethLineData[2]);
                }
                cRightLabel.Text = CRight.ToString();
                xpRightLabel.Text = XpRight.ToString();
                ypRightLabel.Text = YpRight.ToString();
            }

            else if (i == 9)
            {
                String[] tenthLineData = readData[9].Split(new char[] { ' ', ↙
'\t' });
                if (tenthLineData.Length == 5)
                {
                    _k1Right = double.Parse(tenthLineData[0]);
                    _k2Right = double.Parse(tenthLineData[1]);
                    _k3Right = double.Parse(tenthLineData[2]);
                    _p1Right = double.Parse(tenthLineData[3]);
                    _p2Right = double.Parse(tenthLineData[4]);
                }
```

```
                            k1RightLabel.Text = K1Right.ToString();
                            k2RightLabel.Text = K2Right.ToString();
                            k3RightLabel.Text = K3Right.ToString();
                            p1RightLabel.Text = P1Right.ToString();
                            p2RightLabel.Text = P2Right.ToString();
                        }
                    }
                    //string file = sr.ReadToEnd();
                    //string[] files = file.Split(new char[] { '\n' });
                    //Trace.WriteLine(files.Count<string>());

                }
            }
            Trace.WriteLine("test " + OmegaRight.ToString());
        }

        #endregion Hendler


    }
}
```

Lampiran 1-7

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using System.IO;

namespace StereoMappingD
{
    public partial class AddPairThumbnailForm : Form
    {

        #region Events & Delegates
        private delegate void AddThumbnailDelegate(ThumbnailView1 thumbView);
        #endregion Events & Delegates


        #region Data
        private bool m_isActive = false;
        private Cursor m_handCursor = null;                    //hand cursor for dragging highlighted picture area
        private ThumbnailController m_controller;
        private ThumbnailView1 m_currentlyActiveThumbnail;    //get focus only one at a time

        delegate void DelegateAddImage(string imageFilename);
        private DelegateAddImage m_AddImageDelegate;

        string m_pairName;
        string m_leftPhoto;
        string m_rightPhoto;


        String _pairName;
        List<ImagePair> _photoPair;

        Image _image;
        Image _imageLeft;
        Image _imageRight;
        #endregion Data


        #region Properties

        public string PairName
        {
            get { return m_pairName; }
            set { m_pairName = value; }
        }

        public string LeftPhotoImageLocation
        {
            get { return m_leftPhoto; }
            set { m_leftPhoto = value; }
        }

        public string RightPhotoLocation
        {
            get { return m_rightPhoto; }
            set { m_rightPhoto = value; }
        }
```

```
        private int ImageSize
        {
            get
            {
                return (120);//* (this.trackBarSize.Value + 1));
            }
        }

        public Cursor HandCursor
        {
            get { return m_handCursor; }
            set { m_handCursor = value; }
        }

        public FlowLayoutPanel FlowPanelView
        {
            get { return this.flowLayoutPanel1; }
        }

        public bool IsThumnailActive
        {
            get { return m_isActive; }
            set
            {
                m_isActive = value;
                Invalidate();
            }
        }

        public List<ImagePair> PhotoPairList
        {
            get { return _photoPair; }
        }

        #endregion Properties


        #region Constructors
        public AddPairThumbnailForm()
        {
            InitializeComponent();
            _photoPair = new List<ImagePair>();

            m_AddImageDelegate = new DelegateAddImage(this.AddImage);

            m_controller = new ThumbnailController();
            m_controller.OnStart += new ThumbnailControllerEventHandler
(m_Controller_OnStart);
            m_controller.OnAdd += new ThumbnailControllerEventHandler(m_Controller_OnAdd)
;
            m_controller.OnEnd += new ThumbnailControllerEventHandler(m_Controller_OnEnd)
;
        }


        #endregion Constructors


        #region Methods
        private void AddImage(string imageFilename)
        {

            if (this.InvokeRequired)
            {
```

```
            this.Invoke(m_AddImageDelegate, imageFilename);
        }
        else
        {
            _image = new Bitmap(imageFilename);
            int size = ImageSize;

            ThumbnailView1 thumbnailView = new ThumbnailView1();
            thumbnailView.Dock = DockStyle.Bottom;
            thumbnailView.LoadImage(imageFilename, 256, 256);


            thumbnailView.Width = size;
            thumbnailView.Height = size;

            thumbnailView.MouseDown += new MouseEventHandler(thumbView_MouseDown);
            thumbnailView.MouseUp += new MouseEventHandler(thumbView_MouseUp);
            thumbnailView.ContextMenuStrip = this.tumbnailContextMenuStrip;
            FlowPanelView.Controls.Add(thumbnailView);

            this.flowLayoutPanel1.Controls.Add(thumbnailView);

        }
    }

    void thumbView_MouseUp(object sender, MouseEventArgs e)
    {
        MouseButtons button = e.Button;
        switch (button)
        {
            case MouseButtons.Right:
                Trace.WriteLine("Right button is up");
                break;
            case MouseButtons.Left:
                break;
        }
    }

    void thumbView_MouseDown(object sender, MouseEventArgs e)
    {
        if (m_currentlyActiveThumbnail != null)
        {
            m_currentlyActiveThumbnail.IsThumbnailActive = false;
            m_currentlyActiveThumbnail = null;
        }

        m_currentlyActiveThumbnail = (ThumbnailView1)sender;
        m_currentlyActiveThumbnail.IsThumbnailActive = true;

        MouseButtons button = e.Button;
        switch (button)
        {
            case MouseButtons.Left:
                Trace.WriteLine("Left button is pressed");
                break;
            case MouseButtons.Right:
                Trace.WriteLine("Right button is pressed");
                break;
        }
    }


    private void m_Controller_OnStart(object sender, ThumbnailControllerEventArgs e)
    {

    }
```

```
        private void m_Controller_OnAdd(object sender, ThumbnailControllerEventArgs e)
        {
            this.AddImage(e.ImageFilename);
            this.Invalidate();
        }

        private void m_Controller_OnEnd(object sender, ThumbnailControllerEventArgs e)
        {
            if (this.InvokeRequired)
            {
                this.Invoke(new ThumbnailControllerEventHandler(m_Controller_OnEnd),
    sender, e);
            }
            else
            {

            }
        }



        #endregion Methods



        #region Hendler
        private void CenselButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void addPhotosButton_Click(object sender, EventArgs e)
        {
            FolderBrowserDialog dlg = new FolderBrowserDialog();
            dlg.Description = @"Choose folder path";
            if (dlg.ShowDialog() == DialogResult.OK)
            {
                //this.flowLayoutPanelMain.Controls.Clear();
                m_controller.AddFolder(dlg.SelectedPath);

                this.labelTargetPathValue.Text = dlg.SelectedPath;
            }
        }

        private void displayLeftImageToolStripMenuItem_Click(object sender, EventArgs e)
        {

            m_leftPhoto = m_currentlyActiveThumbnail.ImageLocation;
            Trace.WriteLine(m_leftPhoto);
            _imageLeft = new Bitmap(m_leftPhoto);
            this.leftImageLabel.Text = Path.GetFileName(m_leftPhoto);

        }

        private void displayRightImageToolStripMenuItem_Click(object sender, EventArgs e)
        {
            m_rightPhoto = m_currentlyActiveThumbnail.ImageLocation;
            Trace.WriteLine(m_rightPhoto);
            _imageRight = new Bitmap(m_rightPhoto);
            this.rightImageLabel.Text = Path.GetFileName(m_rightPhoto);
        }

        private void addPairButton_Click(object sender, EventArgs e)
        {
```

```
            _pairName = pairNameTextBox.Text;

            if (!String.IsNullOrEmpty(_pairName))
            {
                ImagePair pair = new ImagePair(m_leftPhoto, m_rightPhoto, _pairName,
    _imageLeft.Width,_imageLeft.Height,_imageRight.Width,_imageRight.Height);
                _photoPair.Add(pair);
                listBox2.Items.Add(pair.PhotoPairName);
                pairNameTextBox.Clear();
            }
            else
                return;
        }

        #endregion Hendler


        public delegate void ThumbnailImageEventHandler(object sender,
    ThumbnailImageEventArgs e);

        public class ThumbnailImageEventArgs : EventArgs
        {
            public ThumbnailImageEventArgs(int size)
            {
                this.Size = size;
            }
            public int Size;
        }




    }
}
```

Lampiran 1-8

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

using StereoMappingD.CameraData;


namespace StereoMappingD
{
    public class Interseksi
    {

        #region Fields
        private int _pointCountLeft;
        private int _pointCountRight;

        private double _focallenghtLeft;
        private double _focallenghtRight;

        private double _xpLeft;
        private double _ypLeft;
        private double _xpRight;
        private double _ypRight;

        private string _pointIDos;
        private string _scriptPoint;
        private string _scriptText;

        //Eo left camera
        private double _XL;
        private double _YL;
        private double _ZL;
        private double _omega_left;
        private double _phi_left;
        private double _kappa_left;

        //Eo right camera
        private double _XR;
        private double _YR;
        private double _ZR;
        private double _omega_right;
        private double _phi_right;
        private double _kappa_right;

        private ObjectSpaceCoordinate _oSc;
        private List<ObjectSpaceCoordinate> _objectSpaceCoordinateList;

        #endregion Fields



        #region Constructors
        public Interseksi()
        {
            _focallenghtLeft = 0.0;
            _focallenghtRight = 0.0;
            _xpLeft = 0.0;
            _ypLeft = 0.0;
            _xpRight = 0.0;
            _ypRight = 0.0;


            //Eo left camera
```

```
            _XL = 0.0;
            _YL = 0.0;
            _ZL = 0.0;
            _omega_left = 0.0;
            _phi_left = 0.0;
            _kappa_left = 0.0;

            //Eo right camera
            _XR = 0.0;
            _YR = 0.0;
            _ZR = 0.0;
            _omega_right = 0.0;
            _phi_right = 0.0;
            _kappa_right = 0.0;

            _oSc = new ObjectSpaceCoordinate();
            _objectSpaceCoordinateList = new List<ObjectSpaceCoordinate>();
        }

        #endregion Constructors



        #region Properties
        public List<ObjectSpaceCoordinate> ObjectSpaceCoord
        {
            get { return _objectSpaceCoordinateList; }
            set { _objectSpaceCoordinateList = value; }
        }

        #endregion Properties



        #region Methods


        double[] XYZ_obj;


        public List<ObjectSpaceCoordinate> ObjectSpaceCoordinateCount(List<Cross>
    photoCoordinateListLeft, List<Cross> photoCoordinateListRight,
            CameraPair cameraPair)
        {
            _focallenghtLeft = cameraPair.CameraLeft.IoParams.C * -1;
            _focallenghtRight = cameraPair.CameraRight.IoParams.C * -1;
            _xpLeft = cameraPair.CameraLeft.IoParams.Xp;
            _ypLeft = cameraPair.CameraLeft.IoParams.Yp;
            _xpRight = cameraPair.CameraRight.IoParams.Xp;
            _ypRight = cameraPair.CameraRight.IoParams.Yp;


            //Eo left camera
            _XL = cameraPair.CameraLeft.EoParams.XL;
            _YL = cameraPair.CameraLeft.EoParams.YL;
            _ZL = cameraPair.CameraLeft.EoParams.ZL;
            _omega_left = cameraPair.CameraLeft.EoParams.Omega;
            _phi_left = cameraPair.CameraLeft.EoParams.Phi;
            _kappa_left = cameraPair.CameraLeft.EoParams.Kappa;

            //Eo right camera
            _XR = cameraPair.CameraRight.EoParams.XL;
            _YR = cameraPair.CameraRight.EoParams.YL;
            _ZR = cameraPair.CameraRight.EoParams.ZL;
            _omega_right = cameraPair.CameraRight.EoParams.Omega;
            _phi_right = cameraPair.CameraRight.EoParams.Phi;
```

```
            _kappa_right = cameraPair.CameraRight.EoParams.Kappa;


            _pointCountLeft = photoCoordinateListLeft.Count;
            _pointCountRight = photoCoordinateListRight.Count;

            //Compute Rotation Matrix
            double[,] M_left = new double[3, 3];
            double[,] M_right = new double[3, 3];

            M_left = RotationMatrix(_omega_left, _phi_left, _kappa_left);
            M_right = RotationMatrix(_omega_right, _phi_right, _kappa_right);
            PrintMatrixDebug(ref M_left, "M_left");
            PrintMatrixDebug(ref M_right, "M_right");


            //Compute coordinate transformation
            double[,] XYZtransLeft = new double[_pointCountLeft, 3];
            double[,] XYZtransRight = new double[_pointCountRight, 3];

            XYZtransLeft = CoordinateTransformationLeft(ref M_left,
photoCoordinateListLeft, _pointCountLeft, _focallenghtLeft);
            XYZtransRight = CoordinateTransformationRight(ref M_right,
photoCoordinateListRight, _pointCountRight, _focallenghtRight);

            PrintMatrixDebug(ref XYZtransLeft, "xyz_trans_left");
            PrintMatrixDebug(ref XYZtransRight, "xyz_trans_right");


            //Compute scale factor
            double[] scale = new double[_pointCountLeft];

            scale = ScaleFactor(ref XYZtransLeft, ref XYZtransRight, _pointCountLeft, _XL
, _YL, _XR, _YR);
            PrintVectorDebug(ref scale, "scale_factor");


            //Compute approximation value of object space point
            double[,] threeDobject = new double[_pointCountLeft, 3];

            threeDobject = Approximation3DobjectSpace(ref scale, ref XYZtransRight,
_pointCountLeft, _XR, _YR, _ZR);
            PrintMatrixDebug(ref threeDobject, "3D_Point_Approximation_ObjectSpace");


            //Compute least square
            for (int i = 0; i < _pointCountLeft; i++)
            {
                XYZ_obj = new double[3] { threeDobject[i, 0], threeDobject[i, 1],
threeDobject[i, 2] };
                PrintVectorDebug(ref XYZ_obj, "3D_Point_Approximation_ObjectSpace");

                double[] xy_photo = new double[4] { photoCoordinateListLeft[i].xPhoto,
photoCoordinateListLeft[i].yPhoto,
                    photoCoordinateListRight[i].xPhoto,photoCoordinateListRight[i].
yPhoto};

                //PrintVectorDebug(ref xy_photo, "xy_photo");

                for (int iter = 0; iter < 25; iter++)
                {
                    System.Diagnostics.Trace.WriteLine("");
                    System.Diagnostics.Trace.WriteLine("");
                    System.Diagnostics.Trace.WriteLine("Iterasi" + iter.ToString());

                    //Compute _r_s_q
```

```
                double[,] rsq_left = new double[1, 3];
                double[,] rsq_right = new double[1, 3];

                rsq_left = rsqLeft(ref M_left, ref XYZ_obj, _pointCountLeft, _XL, _YL
, _ZL);
                rsq_right = rsqRight(ref M_right, ref XYZ_obj, _pointCountRight, _XR,
 _YR, _ZR);
                PrintMatrixDebug(ref rsq_left, "rsq_left");
                PrintMatrixDebug(ref rsq_right, "rsq_right");

                //Compose Matrix A(coefisien matrix)
                double[,] A = new double[4, 3];

                A = ComposeMatrixA(ref rsq_left, ref M_left, ref rsq_right, ref
M_right, _focallenghtLeft, _focallenghtRight);
                PrintMatrixDebug(ref A, "A");

                //Compose Matrix L(observation matrix)
                double[,] L = new double[4, 1];
                L = ComposeMatrixL(ref xy_photo, ref rsq_left, ref rsq_right,
_focallenghtLeft, _focallenghtRight);
                PrintMatrixDebug(ref L, "L");

                //Convert double matrix to CSML
                CSML.Matrix A_csml = new CSML.Matrix(A);
                CSML.Matrix L_csml = new CSML.Matrix(L);

                //Least Square process for coefisien and observation equals
                CSML.Matrix x = CSML.Matrix.Solve(A_csml.Transpose() * A_csml, A_csml
.Transpose() * L_csml);
                System.Diagnostics.Trace.WriteLine("x :\n" + x.ToString());

                //Convert CSML to double array
                double[,] xx = CSML2Array(ref x);

                //Give corection value for each point coordinate
                XYZ_obj[0] = XYZ_obj[0] + xx[0, 0];
                XYZ_obj[1] = XYZ_obj[1] + xx[1, 0];
                XYZ_obj[2] = XYZ_obj[2] + xx[2, 0];

                PrintVectorDebug(ref XYZ_obj, "XYZ_obj After iteration");
                if (x.Norm() < 1.0e-10)
                    break;
            }

            _pointIDos = photoCoordinateListLeft[i].LabelName;

            _scriptPoint = ("Point " + XYZ_obj[0].ToString() + "," + XYZ_obj[1].
ToString() + "," + XYZ_obj[2].ToString() + " "); //== to copy-paste on A-Cad command
line

            _scriptText = ("Text " + XYZ_obj[0].ToString() + "," + XYZ_obj[1].
ToString() + "," + XYZ_obj[2] + " 1.5 0 " + _pointIDos.ToString() + " "); //==>to
save on notePad and than run script on A_Cad

            _oSc = new ObjectSpaceCoordinate(_pointIDos, XYZ_obj[0], XYZ_obj[1],
XYZ_obj[2], _scriptPoint, _scriptText);
            _objectSpaceCoordinateList.Add(_oSc);


        }

        return _objectSpaceCoordinateList;
    }
```

```
    public double[,] RotationMatrix(double Omega, double Phi, double Kappa)
    {
        double[,] M = new double[3, 3];
        double a = Math.PI / 180;

        M[0, 0] = Math.Cos(Phi * a) * Math.Cos(Kappa * a);
        M[0, 1] = Math.Sin(Omega * a) * Math.Sin(Phi * a) * Math.Cos(Kappa * a) +
Math.Cos(Omega * a) * Math.Sin(Kappa * a);
        M[0, 2] = Math.Sin(Omega * a) * Math.Sin(Kappa * a) - Math.Cos(Omega * a) *
Math.Sin(Phi * a) * Math.Cos(Kappa * a);

        M[1, 0] = -Math.Cos(Phi * a) * Math.Sin(Kappa * a);
        M[1, 1] = Math.Cos(Omega * a) * Math.Cos(Kappa * a) - Math.Sin(Omega * a) *
Math.Sin(Phi * a) * Math.Sin(Kappa * a);
        M[1, 2] = Math.Sin(Omega * a) * Math.Cos(Kappa * a) + Math.Cos(Omega * a) *
Math.Sin(Phi * a) * Math.Sin(Kappa * a);

        M[2, 0] = Math.Sin(Phi * a);
        M[2, 1] = -Math.Sin(Omega * a) * Math.Cos(Phi * a);
        M[2, 2] = Math.Cos(Omega * a) * Math.Cos(Phi * a);

        return M;
    }

    public double[,] CoordinateTransformationLeft(ref double[,] M_left, List<Cross>
PhotoCoordListLeft, int PointCount, double FocalLenghtLeft)
    {
        double[,] XYZtransLeft = new double[PointCount, 3];
        for (int i = 0; i < PointCount; i++)
        {
            XYZtransLeft[i, 0] = PhotoCoordListLeft[i].xPhoto * M_left[0, 0] +
PhotoCoordListLeft[i].yPhoto * M_left[1, 0] + FocalLenghtLeft * M_left[2, 0];
            XYZtransLeft[i, 1] = PhotoCoordListLeft[i].xPhoto * M_left[0, 1] +
PhotoCoordListLeft[i].yPhoto * M_left[1, 1] + FocalLenghtLeft * M_left[2, 1];
            XYZtransLeft[i, 2] = PhotoCoordListLeft[i].xPhoto * M_left[0, 2] +
PhotoCoordListLeft[i].yPhoto * M_left[1, 2] + FocalLenghtLeft * M_left[2, 2];
        }
        return XYZtransLeft;
    }

    public double[,] CoordinateTransformationRight(ref double[,] M_right, List<Cross>
 PhotoCoordListRight, int PointCount, double FocalLenghtRight)
    {
        double[,] XYZtransRight = new double[PointCount, 3];
        for (int i = 0; i < PointCount; i++)
        {
            XYZtransRight[i, 0] = PhotoCoordListRight[i].xPhoto * M_right[0, 0] +
PhotoCoordListRight[i].yPhoto * M_right[1, 0] + FocalLenghtRight * M_right[2, 0];
            XYZtransRight[i, 1] = PhotoCoordListRight[i].xPhoto * M_right[0, 1] +
PhotoCoordListRight[i].yPhoto * M_right[1, 1] + FocalLenghtRight * M_right[2, 1];
            XYZtransRight[i, 2] = PhotoCoordListRight[i].xPhoto * M_right[0, 2] +
PhotoCoordListRight[i].yPhoto * M_right[1, 2] + FocalLenghtRight * M_right[2, 2];
        }
        return XYZtransRight;
    }

    public double[] ScaleFactor(ref double[,] XYZtransLeft, ref double[,]
XYZtransRight, int PointCount, double X_camLeft, double Y_camLeft, double X_camRight,
 double Y_camRight)
    {
        double[] scale = new double[PointCount];
        for (int j = 0; j < PointCount; j++)
        {
            scale[j] = (XYZtransLeft[j, 1] * (X_camRight - X_camLeft) - XYZtransLeft
[j, 0] * (Y_camRight - Y_camLeft)) / (XYZtransLeft[j, 0] * XYZtransRight[j, 1] -
XYZtransRight[j, 0] * XYZtransLeft[j, 1]);
```

```
        }
        return scale;
    }

    public double[,] Approximation3DobjectSpace(ref double[] scale, ref double[,] XYZtransRight, int PointCount, double X_camRight, double Y_camRight, double Z_camRight)
    {
        double[,] threeDobject = new double[PointCount, 3];
        for (int k = 0; k < PointCount; k++)
        {
            threeDobject[k, 0] = scale[k] * XYZtransRight[k, 0] + X_camRight;
            threeDobject[k, 1] = scale[k] * XYZtransRight[k, 1] + Y_camRight;
            threeDobject[k, 2] = scale[k] * XYZtransRight[k, 2] + Z_camRight;
        }
        return threeDobject;
    }

    public double[,] rsqLeft(ref double[,] M_left, ref double[] XYZ_objApp, int PointCount, double X_camLeft, double Y_camLeft, double Z_camLeft)
    {
        double[,] rsq_left = new double[1, 3];
        for (int k = 0; k < PointCount; k++)
        {
            rsq_left[0, 0] = M_left[0, 0] * (XYZ_objApp[0] - X_camLeft) + M_left[0, 1] * (XYZ_objApp[1] - Y_camLeft) + M_left[0, 2] * (XYZ_objApp[2] - Z_camLeft);
            rsq_left[0, 1] = M_left[1, 0] * (XYZ_objApp[0] - X_camLeft) + M_left[1, 1] * (XYZ_objApp[1] - Y_camLeft) + M_left[1, 2] * (XYZ_objApp[2] - Z_camLeft);
            rsq_left[0, 2] = M_left[2, 0] * (XYZ_objApp[0] - X_camLeft) + M_left[2, 1] * (XYZ_objApp[1] - Y_camLeft) + M_left[2, 2] * (XYZ_objApp[2] - Z_camLeft);
        }
        return rsq_left;
    }

    public double[,] rsqRight(ref double[,] M_right, ref double[] XYZ_objApp, int PointCount, double X_camRight, double Y_camRight, double Z_camRight)
    {
        double[,] rsq_right = new double[1, 3];
        for (int k = 0; k < PointCount; k++)
        {
            rsq_right[0, 0] = M_right[0, 0] * (XYZ_objApp[0] - X_camRight) + M_right[0, 1] * (XYZ_objApp[1] - Y_camRight) + M_right[0, 2] * (XYZ_objApp[2] - Z_camRight);
            rsq_right[0, 1] = M_right[1, 0] * (XYZ_objApp[0] - X_camRight) + M_right[1, 1] * (XYZ_objApp[1] - Y_camRight) + M_right[1, 2] * (XYZ_objApp[2] - Z_camRight);
            rsq_right[0, 2] = M_right[2, 0] * (XYZ_objApp[0] - X_camRight) + M_right[2, 1] * (XYZ_objApp[1] - Y_camRight) + M_right[2, 2] * (XYZ_objApp[2] - Z_camRight);
        }
        return rsq_right;
    }

    public double[,] ComposeMatrixA(ref double[,] rsq_left, ref double[,] M_left, ref double[,] rsq_right, ref double[,] M_right, double FocalLenghtLeft, double FocalLenghtRight)
    {
        double[,] A = new double[4, 3];
        A[0, 0] = -1.0 * FocalLenghtLeft / Math.Pow(rsq_left[0, 2], 2) * (rsq_left[0, 0] * M_left[2, 0] - rsq_left[0, 2] * M_left[0, 0]);
        A[0, 1] = -1.0 * FocalLenghtLeft / Math.Pow(rsq_left[0, 2], 2) * (rsq_left[0, 0] * M_left[2, 1] - rsq_left[0, 2] * M_left[0, 1]);
        A[0, 2] = -1.0 * FocalLenghtLeft / Math.Pow(rsq_left[0, 2], 2) * (rsq_left[0, 0] * M_left[2, 2] - rsq_left[0, 2] * M_left[0, 2]);

        A[1, 0] = -1.0 * FocalLenghtLeft / Math.Pow(rsq_left[0, 2], 2) * (rsq_left[0, 1] * M_left[2, 0] - rsq_left[0, 2] * M_left[1, 0]);
        A[1, 1] = -1.0 * FocalLenghtLeft / Math.Pow(rsq_left[0, 2], 2) * (rsq_left[0, 1] * M_left[2, 1] - rsq_left[0, 2] * M_left[1, 1]);
```

```
        A[1, 2] = -1.0 * FocalLenghtLeft / Math.Pow(rsq_left[0, 2], 2) * (rsq_left[0, 1] * M_left[2, 2] - rsq_left[0, 2] * M_left[1, 2]);

        A[2, 0] = -1.0 * FocalLenghtRight / Math.Pow(rsq_right[0, 2], 2) * (rsq_right[0, 0] * M_right[2, 0] - rsq_right[0, 2] * M_right[0, 0]);
        A[2, 1] = -1.0 * FocalLenghtRight / Math.Pow(rsq_right[0, 2], 2) * (rsq_right[0, 0] * M_right[2, 1] - rsq_right[0, 2] * M_right[0, 1]);
        A[2, 2] = -1.0 * FocalLenghtRight / Math.Pow(rsq_right[0, 2], 2) * (rsq_right[0, 0] * M_right[2, 2] - rsq_right[0, 2] * M_right[0, 2]);

        A[3, 0] = -1.0 * FocalLenghtRight / Math.Pow(rsq_right[0, 2], 2) * (rsq_right[0, 1] * M_right[2, 0] - rsq_right[0, 2] * M_right[1, 0]);
        A[3, 1] = -1.0 * FocalLenghtRight / Math.Pow(rsq_right[0, 2], 2) * (rsq_right[0, 1] * M_right[2, 1] - rsq_right[0, 2] * M_right[1, 1]);
        A[3, 2] = -1.0 * FocalLenghtRight / Math.Pow(rsq_right[0, 2], 2) * (rsq_right[0, 1] * M_right[2, 2] - rsq_right[0, 2] * M_right[1, 2]);
        return A;
    }

    public double[,] ComposeMatrixL(ref double[] xy_photo, ref double[,] rsq_left, ref double[,] rsq_right, double FocalLenghtLeft, double FocalLenghtRight)
    {
        double[,] L = new double[4, 1];
        Trace.WriteLine("test : " + _xpLeft.ToString() + " " + _ypLeft.ToString() + " " + _xpRight.ToString() + " " +_ypRight.ToString());
        L[0, 0] = (xy_photo[0] -_xpLeft) + (-1.0 * FocalLenghtLeft) * rsq_left[0, 0] / rsq_left[0, 2];
        L[1, 0] = (xy_photo[1] - _ypLeft) + (-1.0 * FocalLenghtLeft) * rsq_left[0, 1] / rsq_left[0, 2];
        L[2, 0] = (xy_photo[2] - _xpRight) + (-1.0 * FocalLenghtRight) * rsq_right[0, 0] / rsq_right[0, 2];
        L[3, 0] = (xy_photo[3] - _ypRight) + (-1.0 * FocalLenghtRight) * rsq_right[0, 1] / rsq_right[0, 2];
        return L;
    }



    public void PrintMatrixDebug(ref double[,] a, string name)
    {
        int lower1 = a.GetLowerBound(0);
        int upper1 = a.GetUpperBound(0);
        int lower2 = a.GetLowerBound(1);
        int upper2 = a.GetUpperBound(1);
        int length = a.Length;

        System.Diagnostics.Trace.WriteLine("\n" + name + ":");
        for (int i = lower1; i <= upper1; i++)
        {
            for (int j = lower2; j <= upper2; j++)
            {
                System.Diagnostics.Trace.Write(a[i, j].ToString() + "\t");
            }
            System.Diagnostics.Trace.WriteLine("");
        }
    }

    public void PrintVectorDebug(ref double[] v, string name)
    {
        int lower = v.GetLowerBound(0);
        int upper = v.GetUpperBound(0);
        int length = v.Length;

        System.Diagnostics.Trace.WriteLine("\n" + name + ":");
        for (int i = lower; i <= upper; i++)
        {
```

Lampiran 1-9

```
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;

namespace StereoMappingD
{
    /// <summary>
    /// LSM Implementation according to Toni Schenk
    /// Using 6 affine Parameters: shift: dx dy; Scale: x y. Parameters: a0,a1,a2; b0, b1
    ,b2
    /// x = a0 + a1 * i + a2 * j    ;  i = col in pixel
    /// y = b0 + b1 * i + b2 * j    ;  j = row in pixel
    /// Steps:
    /// 1. Select the centre of one template
    /// 2. Determine approximate location for the mathing patch
    /// 3. Start first iteration with mathing patch at the approximate location
    /// 4. Transform matching patch and determine the gray value for the tessellation
    (resampling)
    /// 5. Repeat Adustment and resampling sequence until it is converged
    /// </summary>
    public class LSM6A
    {
        #region Fields
        /// <summary>
        /// Template Image
        /// </summary>
        Image<Gray, Byte> _templateImg;

        /// <summary>
        /// Matching Image
        /// </summary>
        Image<Gray, Byte> _matchingImg;

        /// <summary>
        /// Coordinates of the centre of a selected template patch, in pixel
        /// </summary>
        Point2D _centreTemplate;

        /// <summary>
        /// Coordinate of the centre of the matching patch, in pixel. It is resulted from
     CrossCorrelation
        /// </summary>
        Point2D _matchedPoint;          //in pixel

        /// <summary>
        /// Coordinate of the final matching patch, in pixel. It is from the LSM
        /// </summary>
        Point2D _correctedMatchedPt;    //final result

        /// <summary>
        /// Affine Parameters: Shift x
        /// </summary>
        double _a0;

        /// <summary>
        /// Affine Parameters: Shift y
```

```
        /// </summary>
        double _b0;

        /// <summary>
        /// Affine Parameters: Scale x
        /// </summary>
        double _a1;

        /// <summary>
        /// Affine Parameters: Scale y
        /// </summary>
        double _b2;

        /// <summary>
        /// Affine Parameters: Shear y
        /// </summary>
        double _a2;

        /// <summary>
        /// Affine Parameters: Shear x
        /// </summary>
        double _b1;

        /// <summary>
        /// constant: gray value different T(i,j) - m(i,j)
        /// </summary>
        CSML.Matrix _b;

        /// <summary>
        /// Indicator whether the LSM iteration can converge
        /// </summary>
        bool _lsmConverge;



        #endregion Fields

        #region Constructors
        public LSM6A()
        {
            _centreTemplate = new Point2D();
            _matchedPoint = new Point2D();
            _correctedMatchedPt = new Point2D();

            _a0 = 0.0;
            _a1 = 0.0;
            _a2 = 0.0;
            _b0 = 0.0;
            _b1 = 0.0;
            _b2 = 0.0;
            _lsmConverge = false;
        }

        public LSM6A(ref Image<Gray, Byte> source, ref Image<Gray, Byte> matchingImg,
             Point2D centerTemplate, Point centreMatchingPatch)
            : this()
        {
            Trace.WriteLine("\t\t------------------- Constructor LSM6A() is called ------↙
------");
            _templateImg = source;
            _matchingImg = matchingImg;
            _centreTemplate.X = (double)((int)(centerTemplate.X + 0.5));
            _centreTemplate.Y = (double)((int)(centerTemplate.Y + 0.5));
            _matchedPoint.X = (double)centreMatchingPatch.X;
            _matchedPoint.Y = (double)centreMatchingPatch.Y;
```

```
            _correctedMatchedPt.X = (double)centreMatchingPatch.X;
            _correctedMatchedPt.Y = (double)centreMatchingPatch.Y;

            double shiftX = _centreTemplate.X - _matchedPoint.X;
            double shiftY = _centreTemplate.Y - _matchedPoint.Y;
            Trace.WriteLine("Shift X Y: " + shiftX.ToString() + "   " + shiftY.ToString())↙
;
        }

        #endregion Constructors

        #region Properties
        public Point2D CorrectedMatchedPt
        {
            get { return _correctedMatchedPt; }
            set { _correctedMatchedPt = value; }
        }

        #endregion Properties

        #region Private Methods
        private Point2D Affine6Params(double x, double y)
        {
            Point2D pt = new Point2D();
            pt.X = _a0 + _a1 * x + _a2 * y;
            pt.Y = _b0 + _b1 * x + _b2 * y;
            return pt;
        }

        private void ComposeA_b6Params1stIteration(ref CSML.Matrix a, ref CSML.Matrix b)
        {
            //Template Patch 17x17
            int rowStartTemplatePatch = (int)_centreTemplate.Y - 8;
            int rowEndTemplatePatch = (int)_centreTemplate.Y + 8;
            int colStartTemplatePatch = (int)_centreTemplate.X - 8;
            int colEndTemplatePatch = (int)_centreTemplate.X + 8;

            //Matching Patch 17x17
            int rowStartMatchingPatch = (int)_matchedPoint.Y - 8;
            int rowEndMatchingPatch = (int)_matchedPoint.Y + 8;
            int colStartMatchingPatch = (int)_matchedPoint.X - 8;
            int colEndMatchingPatch = (int)_matchedPoint.X + 8;

            int rowNumbers = 17 * 17;
            CSML.Matrix A = new CSML.Matrix(rowNumbers, 6);
            _b = new CSML.Matrix(rowNumbers, 1);
            int index = 1;

            for (int rowIndex = 0; rowIndex < 17; rowIndex++)
            {
                for (int colIndex = 0; colIndex < 17; colIndex++)
                {
                    //Resample matching patch onto template image == original matching    ↙
patch
                    double xMPatch = (double)(colIndex + colStartMatchingPatch);
                    double yMPatch = (double)(rowIndex + rowStartMatchingPatch);

                    double xTPatch = (double)(colIndex + colStartTemplatePatch);
                    double yTPatch = (double)(rowIndex + rowStartTemplatePatch);

                    //Compute Gradient of A on the matching patch
                    //m0(x,y)=m(i,j)
                    double dx = (_matchingImg[(int)yMPatch, (int)(xMPatch + 1)].Intensity↙
-
                                 _matchingImg[(int)yMPatch, (int)(xMPatch - 1)].Intensity)↙
/ 2.0;     //((x+1)(x-1))/2
```

```
                double xy = _matchingImg[(int)yMPatch, (int)xMPatch].Intensity;
            //g(x,y)
                double dy = (_matchingImg[(int)(yMPatch + 1), (int)xMPatch].Intensity
-
                            _matchingImg[(int)(yMPatch - 1), (int)xMPatch].Intensity)
/ 2.0;    //((y+1)-(y-1))/2

                //Compute A
                A[index, 1].Re = dx;            //a0: dx
                A[index, 2].Re = dx * xMPatch;  //a1: dx * x
                A[index, 3].Re = dx * yMPatch;  //a2: dx * y
                A[index, 4].Re = dy;            //b0: dy
                A[index, 5].Re = dy * xMPatch;  //b1: dy * x
                A[index, 6].Re = dy * yMPatch;  //b2: dy * y

                //Compute b
                _b[index, 1].Re = _templateImg[(int)yTPatch, (int)xTPatch].Intensity
- xy;
                index++;
            }
        }
        CSML.Matrix AT = A.Transpose();
        a = AT * A;
        b = AT * _b;

    }
    #endregion Private Methods

    #region Public Methods
    public bool FindLSMConjugatePoint()
    {
        CSML.Matrix aTa = new CSML.Matrix();
        CSML.Matrix aTb = new CSML.Matrix();
        CSML.Matrix solution = new CSML.Matrix();

        ComposeA_b6Params1stIteration(ref aTa, ref aTb);
        bool aTaCond = aTa.IsSymmetricPositiveDefinite();
        double aTaDet = aTa.Determinant().Re;
        Trace.WriteLine("Det aTa: \t" + aTaDet.ToString());
        Trace.WriteLine("Is aTa symmetric positif definite: \t" + aTaCond.ToString())
;
        //Trace.WriteLine("aTa: \n" + aTa.ToString());
        //Trace.WriteLine("ATb : \n" + aTb.ToString());
        //Trace.WriteLine("aTa Inverse: \n" + aTa.Inverse().ToString());

        int iteration = 2;
        const double DELTA = 0.02;    //pixel
        if (aTaCond && aTaDet > 0.0)
        {
            solution = CSML.Matrix.Solve(aTa, aTb);
            _correctedMatchedPt.X += solution[1, 1].Re + solution[2, 1].Re *
_correctedMatchedPt.X +
                                     solution[3, 1].Re * _correctedMatchedPt.Y;
            _correctedMatchedPt.Y += solution[4, 1].Re + solution[5, 1].Re *
_correctedMatchedPt.X +
                                     solution[6, 1].Re * _correctedMatchedPt.Y;
            _lsmConverge = Threshold(solution, DELTA);
            Trace.WriteLine("Iteration 1: " + "\n" + solution.ToString());
            Trace.WriteLine("_correctedMatchedPt X Y: " + _correctedMatchedPt.X.
ToString() + "  " + _correctedMatchedPt.Y.ToString());
        }

        return _lsmConverge;
    }

    private bool Threshold(CSML.Matrix m, double threshold)
```

```
        {
            if (Math.Abs(m[1, 1].Re) <= threshold && Math.Abs(m[2, 1].Re) <= threshold &&
                Math.Abs(m[3, 1].Re) <= threshold && Math.Abs(m[4, 1].Re) <= threshold &&
                Math.Abs(m[5, 1].Re) <= threshold && Math.Abs(m[6, 1].Re) <= threshold)
                return false;
            else
                return true;
        }

        #endregion Public Methods
}
}
```