PEMBUATAN PROGRAM UNTUK PENENTUAN RESTITUSI *BUNDLE ADJUSMENT* DENGAN C# *VISUAL STUDIO* 2008

Oleh

Alben S. Liu
NIM 05.25.012

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

2010

LEMBAR PENGESAHAN

Pembuatan Program Untuk Penentuan Restitusi *Bundle Adjusment* Dengan C# *Visual Studio* 2008

SKRIPSI

Dipertahankan dihadapan Majelis Penguji Sidang Skripsi

Jenjang Starata Satu (S-1)

Pada hari : Jumat

Tanggal : 20 Agustus 2010

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

Disusun oleh :

Alben S. Liu 05.25.012

Panitia Ujian Tugas Akhir

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

Anggota Penguji

Penguji I	Penguji II	Penguji III
Ir. Leo Patimena, M.Sc	Silvester Sari Sai, ST, MT	Dr. Edwin Tjahjadi, ST. M.Geom.Sc

LEMBAR PERSETUJUAN

Diajukan Sebagai Salah Satu Syarat Memperoleh Gelar Sarjana Teknik Geodesi S-1

Institut Teknologi Nasional Malang

Disusun Oleh :

Alben S. Liu 05.25.012

Meyetujui,

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

PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini :

Nama : **Alben Liu**

NIM : **05.25.012**

Program Studi : **Teknik Geodesi S-1**

Fakultas : **Fakultas Teknik Sipil Dan Perencanaan**

Menyatakan dengan sesungguhnya bahwa Skripsi saya dengan judul :

"Pembuatan Program Untuk Penentuan Restitusi *Bundle Adjusment* Dengan C# *Visual Studio* 2008 (*Studi Kasus : Jembatan Rel Kereta Api Lawang dan Fly Over Arjosari Malang*)" adalah hasil karya saya sendiri, bukan merupakan duplikat serta tidak mengutip atau menyadur dari hasil karya orang lain kecuali disebutkan sumbernya.

KATA PENGANTAR

Puji dan syukur saya panjatkan kehadirat Tuhan Yang Maha Esa karena atas Kasih dan Rahmat-Nya sehingga penyusun dapat menyelesaikan Tugas Akhir dan pembuatan programnya. Tugas ini merupakan salah satu syarat utama sebagai Mahasiswa di Jurusan Teknik Geodesi, Fakultas Teknik Sipil Dan Perencanaan, Istitut Teknologi Nasional Malang, sebagai prasyarat dalam mencapai Program Sarjana Strata Satu (S1).

Selesainya Laporan Tugas Akhir ini tidak terlepas dari bantuan, dukungan dan dorongan dari berbagai pihak. Untuk itu penyusun mengucapkan terima kasih kepada:

1. Bapak Hery Purwanto, ST, MSc. Selaku Dosen Pembimbing I.
2. Dr. Edwin Tjahjadi ST, M.Geom.SCSelaku Dosen Pembimbing II.
3. Semua Dosen yang telah membagikan ilmu slama menepuh kiliah di Jurusan Teknik Geodesi.
4. Kedua Orang Tua yang telah memberikan dukungan Doa dan materi, dengan penuh kasih dan berkat yang slalu menyertai selama proses dalam menyelesaikan Studi ini.
5. Kakak, adik, Ma Ebi, Roy, Benhard, Koko, Amida, Theresia, dan Morenho yang suda menjadi salah satu inspirasi dan pemberi smangat Buat Bapa Amm, yang juga turut mendukung lewat Doa dan semagat dalam menyelesaikan tugas ini Tuhan Yesus Memberkati.

6. Teman Teman Seperjuangan Dalam Tim Rapitmaping dan Deformasi yang selalu saja mengingatkan dan memberi semangat serta membantu menyelesaikan permasalahan yang sulit dipecahkan sendiri selama kita bersama-sama di Laboratoium Makasi banyak Tanzil, Dodik, Lia, Riri, Eno, Yusak, Roger, Ona, Agus, Gede, Desi, Akbar, Mace, Weni Makasi banyak untuk smua yang suda kalian bagikan karna atas dukungan dan kerja sama semua kita bias selesaiSmoga Persahabatan kita tetap abadi.
7. Teman teman Nekmese yang ada di malang smua untuk dukanyannya makasih banyak.
8. Mas Adri yang selalu memberi smangat dan inpirasi sambilngopi dikantin makasih mas dan smoga cepat dapat Jodoh.
9. Adry Yang selalu kusayangi dan menjadi inpirasi dan pendorong utama untuk cepat menyelesaikan Tugas Akhir ini trimakasi banyak dan smoga satu saat Tuhan Mengijinkan untuk kita bias bersama.
10. Teman-teman lain yang tak dapat disebutkan satu persatu trima kasih banyak dan semoga apa yang kalian harapkan dapat segera tercapai.

Akhir kata, semoga laporan Tugas Akhir ini dapat berguna bagi penyusun dan pembaca. Terima kasih.

Malang, Agustus 2010
Penyusun

DAFTAR ISI

Lembar Pengesahan i

Lembar Persetujuan ii

Pernyataan Keaslian Skripsi iii

Kata Pengantar iv

Daftar Isi v

BAB I. PENDAHULUAN

1.1. Latar Belakang 1

1.2. Identifikasi Masalah 1

1.3. Rumusan Masalah 2

1.4. Tujuan Penelitian 2

1.5. Batasan Masalah 2

1.6. Metodologi Penelitian 3

1.6.1. Studi Literatur 3

1.6.2. Studi Laboratorium 3

1.7. Tinjauan Pustaka 4

BAB II. DASAR TEORI

2.1. Centroid .....

2.1.1. Stiker (*Retro reflectif Target*) 5

2.1.2. *Centroid* (Titik Tengah) 6

2.2. Relatif Orientasi 9

2.2.1. Parameter Kamera 10

2.2.2. Parameter Interinsik (Interior Orientation) 10

2.2.3. Parameter Ekstrinsik (Exterior Orientation) 11

2.2.4. Geometri Epipolar 17

2.2.5. Matrik Essensial 19

2.3. Closed Form Solution 22

2.3.1. Metode *Rampal* *22*

2.3.2. Metode Zhang 26

2.3.3. Penentuan Koordinat Kamera 27

2.3.4. Penentuan Parameter Rotasi 33

2.4. Resection *37*

2.5. Intersection *40*

2.6. Teknik Pembuatan Algoritma Dengan C# 44

BAB III. PELAKSANAAN PENELITIAN

3.1. Materi Penelitian 46

3.2. Peralatan Penelitian 46

3.2.1. Perangkat keras (*hardware*) 46

3.2.2. Perangkat lunak (*software*) 48

3.3. Lokasi Penelitian 49

3.4. Dia gram Alir Penelitian 50

3.5. Diagram Alir Pembuatan Interface dan Centroid 52

3.6. Diagram Alir Relatof Orientasi dan Interseksi Inputan Data 55

BAB IV. HASIL DAN PEMBAHASAN

4.1. Hasil 68

4.1.1. Desain Interface 68

4.1.2. Proses Penentuan Centroid 74

4.1.3. Perhitungan Relatif Orientasi 75

4.1.4. Perhitungan Interseksi 75

4.2. Pembahsan 76

4.2.1. Esensial Matriks 77

4.2.2. Parameter Ekstrinsik Kamera 79

4.2.3. Parameter Koordinat Obyek 80

BAB V. PENUTUP

5.1. Kesimpulan 82

5.2. Saran..... 82

Lampiran

BAB I

PENDAHULUAN

1.1. Latar Belakang

Perkembangan penggunaan methode *close range photogrammetry* dalam berbagai aplikasi beberapa tahun terakhir ini sangatlah pesat (Mikhail et al, 2001). Penggunaan kamera resolusi tinggi untuk merekonstruksi objek tiga dimensi menggunakan teknik fotogrametri dengan beberapa persayaratan antara lain *network design*, *camera calibration* dan *bundle adjustment* akan menghasilkan tingkat pengukuran dengan akurasi tinggi (Shirkhani et al., 2006). Dibandingkan dengan pengukuran terestris, pengukuran dan pengolahan data teknik *close range photogrammetry* lebih mudah dan cepat serta murah dalam operasionalnya. Adapula ukuran yang dihasilkan mempunyai akurasi tinggi (Wolf dan Dewitt, 2000).

Pemanfaatan perkembangan teknologi komputer yang sangat pesat terutama pada bahasa pemrograman sangat memungkinkan untuk melakukan perhitungan secara otomatis. Visual Studio 2008 merupakan slah satu bahasa yang cukup handal dan mudah untuk dioperasikan demi mewujudkan aplikasi program.

1.2. Identifikasi Masalah

Proses penentuan koodinat pendekatan *object spase* merupakan tahapan yang harus dilalui dalam *close range photogrammetry.* Permasalahan

yang akan dibahas dalam penelitian ini yaitu bagaimana merekonstruksksi proses perhitungan secara fotogrametri untuk menentukan:

1. Parameter EO (*Exterior Oriented*)
2. IO (*Intrrior Orientation*)
3. Koordinat Pendekatan *Object Sapase.*

1.3. Rumusan Masalah

Berdasarkan penjelasan pada latar belakang, dapat dirumuskan masalah dari penelitian yang dilakukan yaitu bagaimana automatisasi penentuan nilai *object spase* pendekatan menggunakan bahasa pemrograman C# Visual Studio 2008 dari algoritma yang telah tersedia.

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah membuat Program berbasis *Windows* untuk penentuan nilai pendekatan perhitungan ***Bundle Adjustment*** dari hasil pemotretan *multi* foto.

1.5. Batasan Masalah

Pada penelitian ini permasalahan dibatasi pada pembuatan *Interface*, dan *souce code* untuk penentuan *centroid, closed form* Untuk *exterior orientasi (EO), resection, intersection* menggunakan bahasa pemrograman C# Visual Studio 2008 dari algoritma yang telah dibuat.

1.6. Metodologi Penelitian

Metode yang mendasari penelitian ini agar berjalan dengan baik antara lain :

1.6.1. Studi Literatur

Studi Literatur ini digunakan sebagai dasar pembuatan program pengolahan data lapangan sampai penyajian *poin* 3 Dimensi. Langkah–langkah yang dilakukan antara lain:

1. Mempersiapkan buku referensi dan mengumpulkan parameter-parameter yang berkenaan dengan penelitian ini.
2. Mempelajari cara penggunaan bahasa pemrograman yang akan digunakan yaitu C# Visual Studio 2008.

1.6.2. Studi Laboratorium

1. Mendesain visualisasi program, mulai dari input data, proses pengolahan data, sampai dengan hasil akhir perhitungan yang diinginkan menggunakan C# Visual Studio 2008.
2. Membandingkan hasil pengolahan yang dibuat program dengan hasil pengolahannya mengunakan program perhitungan lain.
3. Hasil akhir perhitungan disajikan pada Notepad berupa angka koordinat X, Y, Z lengkap dengan ketelitiannya.

1.7. Tinjauan Pustaka

Komputer adalah perangkat yang digunakan untuk melakukan perhitungan, dan secara logis mampu membuat keputusan yang berjuta bahkan milliar kali lebih cepat daripada manusia. Komputer memproses data yang diatur oleh satu set instruksi yang disebut program komputer. Program tersebut yang menginstruksikan komputer untuk melakukan aksi yang spesifik sesuai dengan keinginan programer yang membuat (*Erick Kurniawan, S.Kom, M.Kom)*.

Microsoft mengembangkan *Framework* .NET sebagai *platform modern* untuk menghadapi tantangan dunia komputer saat ini. Kekayaan *class library* .NET dan konsep-konsep inovatifnya membuat *platform* ini "*general purpose*" namun "*powerful*" (Rachmatullah Agro, 2002).

Perkembangan teknologi komputer yang sangat pesat terutama pada bahasa pemrograman sangat diandalkan dalam perhitungan atau pengolahan data secara otomatis. Kemudahan dalam hal penggunaan, baik dari segi tampilan (*interface*), *input* data maupun dalam proses perhitungan benar-benar tidak membingungkan. Tampilan hasil memberikan informasi tentang parameter-parameter yang digunakan dalam mengolah data. C# Visual Studio 2008 sebagai salah satu bahasa pemrograman sangat cocok digunkan dalam pembuatan perangkat lunak untuk perhitungan (*Pro C# 2008 and the .NET 3.5 Platform, Fourth Edition*).

BAB II

DASAR TEORI

Fotogrametri adalah teknik untuk menentukan 3D geometri (lokasi, ukuran dan bentuk) obyek fisik dengan mengukur dan menganalisis menggunakan foto 2D. Teknik *Close Range* Fotogrammetry merupakan teknik pengukuran terhadap suatu objek dengan teknik *perekaman* dari beberapa alat sensor.

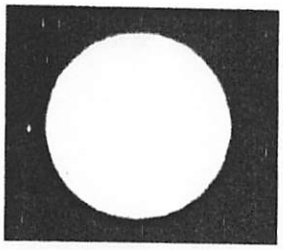
Dalam bab ini akan di jelaskan secara detail mengenai teori-teori yang mendasar dalam fotogrametri. Mulai dari penentuan koordinat foto dengan metode *centroid*, penentuan parametr rotasi posisi kamera (relatif orientasi), *resection*, dan *intersection* untuk mendapatkan poin 3D pendekatan.

2.1. Centroid

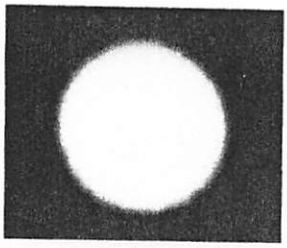
2.1.1. Stiker (*Retro reflectif Target*)

Australis menggunakan warna sasaran dalam bentuk *grayscale* untuk mengenal ukuran pasti lingkaran dalam pixsel. *Grayscale* menggunakan nilai dari 0 hingga 256 yaitu warna kelabu. Setiap pixel yang berada dalam bentuk ini mempunyai nilai kecerahan dari nilai 0 (hitam) ke 255 (putih). Prinsip ini untuk mengenali sasaran berdasarkan perbesaran nilai *grayscale* antara sasaran dengan warna latar belakangnya. Bahan *retro-reflektif* selalu digunakan dalam pekerjaan fotogrametri kerana bahan ini dapat memantulkan cahaya lebih terang dari latar belakangnya. Secara teorinya bahan ini dapat memantulkan cahaya 2000 kali lebih terang dibandingkan

target dari bahan yang lain (Clarke & Wang, 1998). Gambar 1.a menunjukkan sasaran yang menggunakan bahan *retro-reflektif* dimana Gambar 1.b menunjukkan sasaran yang menggunakan kertas putih biasa



Gambar 2.2a. Sasaran *dari retro-reflektif*

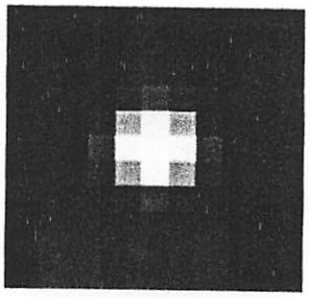


Gambar 2.2b *Sasaran dari kertas*

2.1.2. *Centroid* (Titik Tengah)

A. Defenisi *Centroid*

Secara umum centroid adalah pusat masa suatu objak (Clarke & Wang, 1998). Dalam fotogrametri terutama pada *close-range* fotogrmetri menggunakan *retro reflektif* target sebagai sasaran, *centroid* dapat didefinisikan sebagai titik tengah dari *retro reflektif* target yang dipasang pada objek yang diteliti (*Clark et al, 1993*).



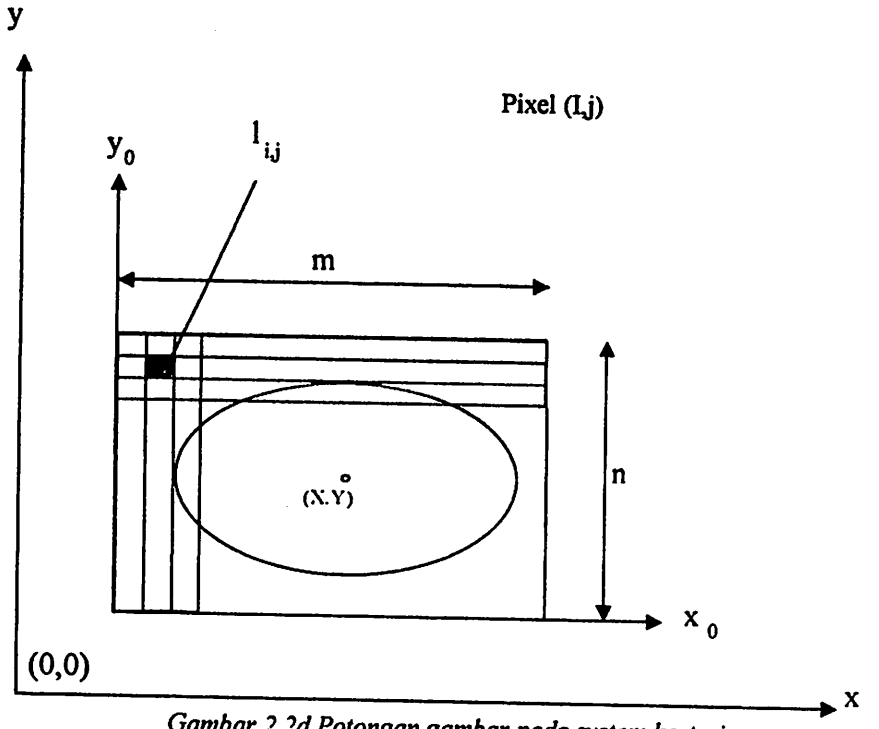
Gambar 2.2.c *Centroid* jika diperbesar

Target yang terekam pada gambar yang dihasilkan dari pemotretan setelah diperbesar tidak hanya terdiri dari satu pixel saja melainkan banyak

pixel dan target tidak memmiliki tepi yang. Oleh karena itu, titik tengah dari target harus berupa koordinat atau posisi yang dapat ditentukan dengan metode *centriod.*

B. Penentuan *Centroid*

Untuk mempermudah menentukan titik *centroid,* cara yang bisa digunakan yaitu dengan mengambil bentuk persegi dari gambar yang mencakup suatu target yang terlihat sangat dekat (Trinder, 1989). Pada gambar dibawah memperlihatkan perpotongan persegii yang diambil pada gambar dimana pada gambar tersebut terdapat data-data yaitu koordinat pixsel, kordinat lokal (x_0, y_0) dan nilai intensitas cahaya (dapat dilihat menggunakan software tertentu).



Gambar 2.2d Potongan gambar pada system kartesian

Keterangan :

(i,j) = Koordinat pixel

(xo,yo) = Koordinat lokal

(X,Y) = Koordinat Pendekatan

m = Garis yang mewakili jumlah pixel pada sumbu x

n = Garis yang mewakili jumlah pixel pada sumbu y

Untuk mendapatkan koordinat *centroid* yang mencapai ketelitian tinggi dan dapat memenuhi bentuk *Gaussian* tidaklah muda, karena terdapat beberapa faktor yang mempengaruhi kurangnya ketelitian (*Clark et al, 1993*) antara lain:

1. Ada *nois* yang biasanya terjadi kerana sedikitnya jumlah foto pada target.
2. Gelombang elektromagnetik yang dapat mereduksi arus gelap setiap 80C. Oleh karena itu dengan pendingin dapat mengurangi noise ini.
3. Metode perhitungan yang digunakan tidak tepat.

C. Metode penentuan *Centroid*

Pada dasarnya ada beberapa metode penentuan *centroid* yang digunakan *(Shortis, et al, 1994)* antara lain:

1. *Avirange of perimeter,* Metode ini sangat sederhana yaitu dengan merata-ratakan koordinat garis keliling target pada gambar yang direferensikan dari hasil praproses *thereshold.*

2. *Binary Centroid*, Pada metode ini semua menggunakan intensitas cahaya dan direferensikan dari hasil praproses *tereshold.*
3. *Gray-scale centroid*, Metode ini hampir sama dengan metode *Binary Centroid* tetapi tanpa batasan thereshold dan tidak menggunakan nilai intensitas cahaya.
4. *Squared gray-scale centoid*, Metode ini hampir sama dengan metode *Grai Scale* Centroid tetapi menggunakan nilai intensitas cahaya yang dikuadratkan.

2.2. Relatif Orientasi

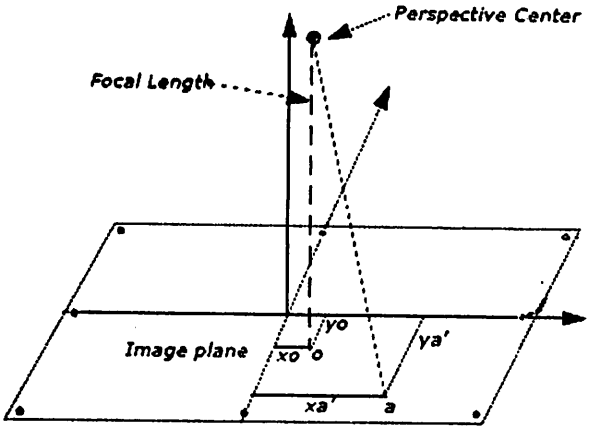
Relatif orientasi merupakan proses untuk menentukan nilai perputaran sudut rotasi dan pergeseran posisi antara dua foto(*Mikhail et al., 2001;Wolf & Dewitt, 2000*). Proses ini dilakukan dengan cara memberikan nilai posisi dan orientasi untuk foto pertama, kemudian dilakukan proses perhitungan nilai posisi dan orientasi pada foto kedua menggunakan parameter dari posisi kamera pertama dan koordinat foto dari kedua buah foto. Dalam proses relatif orientasi ini tidak menghasilkan nilai posisi dan orientasi dari foto yang sebenarnya, akan tetapi menghasilkan sebuah nilai relatif antara dua buah foto tersebut. Yaitu menetapkan beberapa parameter Eksterior orientasi (EO) ω, φ, κ, Y_L, Z_L dari foto kanan (2) dengan memperhitungkan sisa salah satunya dari pertemuan 5 berkas sinar dari koordinat obyek 3D (Xi, Yi, Zi), (*Mikhail et al., 2001;Wolf & Dewitt, 2000*) yang ada.

2.2.1 Parameter Kamera

Dalam fotogrametri maupun *computer vision*, terdapat dua parameter penting yang digunakan dalam berbagai persamaan untuk merekonstruksi obyek tiga dimensi menggunakan fotografi (*Mikhail et al., 2001;Wolf & Dewitt, 2000*). Parameter tersebut terdiri dari parameter interinsik (*Interior Orientation*) dan parameter ekstrinsik (*Exterior Orientation*). Parameter interinsik biasanya dipakai sebagai parameter untuk mendefinisikan nilai geometri kamera, sedangkan parameter ekstrinsik digunakan untuk menjelaskan hubungan geometrik posisi kamera dan obyek dalam suatu sistem koordinat referensi.

2.2.2 Parameter Interinsik (*Interior Orientation)*

Parameter interinsik merupakan parameter yang mendefinisikan geometri dari sensor kamera pada saat melakukan pengambilan foto *(Mikhail et al., 2001)*. Parameter insterinsik kamera terdiri dari tiga parameter pokok yaitu c atau f yang merupakan panjang fokus dan dua parameter *principle point* (x_0, y_0).



Gambar 2.3a Interior orientasi

Parameter *principle point* secara matematika dapat didefinisikan sebagai perpotongan garis tegak lurus yang melalui *perspective center* ke bidang foto. Panjang dari *principle point* ke *perspective center* disebut sebagai panjang fokus (*Wang, 1990*). Pada umumnya parameter ini juga dapat dikatakan sebagai parameter transformasi dimana pusat sistem koordinat terdapat pada *principle point (Geosystem, 2006a)*. Persamaan transformasi ini dapat dituliskan dalam persamaan matematika sebagai berikut (*Fraser, 2006*) :

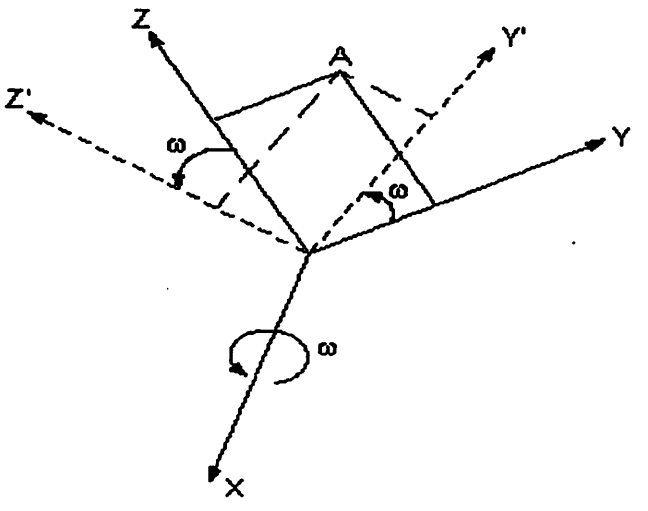
$$\begin{vmatrix} x - x_0 \\ y - y_0 \\ -f \end{vmatrix} = \begin{vmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & -f \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} \qquad (2.1)$$

2.2.3 Parameter *Ekstrinsik (Exterior Orientation)*

Parameter ekstrinsik atau *exterior orientation* merupakan parameter posisi dan orientasi sudut kamera pada saat pengambilan foto. Parameter posisi kamera didefinisikan dalam ruang tiga dimensi yang

merupakan posisi dari *perspektif center* (pusat kamera) X_o, Y_o, Z_o. Sedangkan parameter orientasi sudut berfungsi sebagai penghubung antara sistem koordinat kamera (x, y, z) dengan sistem koordinat referensi (X, Y, Z). Sesuai dengan *Elias (2007), Fraser (2006), Mikhail et al (2001), Shih (1994), Wolf & Dewitt, (2000)* orientasi sudut dapat didefinisikan dalam sistem rotasi ω, φ, κ (*omega, phi, kappa*) atau t, α, s (*tilt, azimuth, swing*).

Dari masing-masing sistem rotasi, dapat dibangun sebuah matriks rotasi yang digunakan untuk menghubungkan kedua sistem koordinat. Sebagai contoh, akan dijelaskan proses penurunan sebuah persamaan pada matriks rotasi dalam sistem rotasi *omega* untuk rotasi sumbu *x*, *phi* untuk rotasi sumbu *y* dan *kappa* sebagai rotasi sumbu *z*.



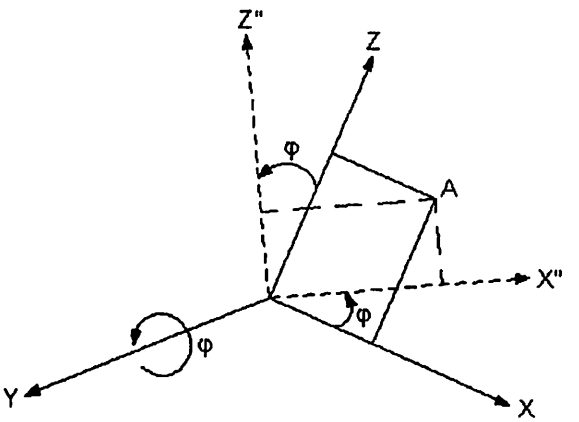
Gambar 2.7 Rotasi sumbu x

Dari gambar diatas, sumbu x positif diputar searah jarum jam, sehingga posisi titik A dalam sistem koordinat yang terotasi dapat ditulis dalam sebuah persamaan.

$$\begin{vmatrix} X' \\ Y' \\ Z' \end{vmatrix} = R_\omega \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \tag{2.2}$$

dimana parameter R_ω didefinisikan sebagai:

$$R_\omega = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{vmatrix} \tag{2.3}$$



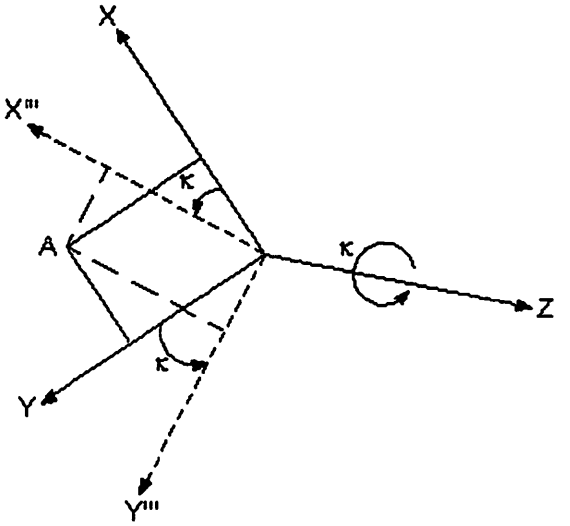
Gambar 2.8 Rotasi sumbu y

Selanjutnya, dilakukan rotasi terhadap sumbu y positif dengan arah rotasi positif searah jarum jam seperti pada *Gambar 2.8*. Dari hasil rotasi didapat nilai posisi titik A dalam sistem koordinat terotasi sebagai berikut.

$$\begin{vmatrix} X'' \\ Y'' \\ Z'' \end{vmatrix} = R_\varphi \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \tag{2.4}$$

Dengan parameter R_φ didapat dari persamaan :

$$R_\varphi = \begin{vmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{vmatrix} \tag{2.5}$$



Gambar 2.9 Rotasi sumbu z

Dan yang terakhir, sumbu positif z dirotasi positif searah jarum jam, sebagaimana diilustrasikan pada *Gambar 2.9* diatas. Sehingga posisi titik *A* pada sistem rotasi sumbu z dinyatakan dalam sebuah persamaan sebagai berikut :

$$\begin{vmatrix} X''' \\ Y''' \\ Z''' \end{vmatrix} = R_\kappa \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \tag{2.6}$$

Dimana parameter R_κ didefinisikan sebagai :

$$R_\kappa = \begin{vmatrix} \cos\kappa & \sin\kappa & 0 \\ -\sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{vmatrix} \tag{2.7}$$

Dari perkalian $R_\omega R_\varphi R_\kappa$ yang merupakan matriks rotasi dari parameter ω, φ dan κ didapat nilai matriks rotasi secara utuh sebagai berikut (*Cooper & Robson, 2001*) :

$$R_{\omega\varphi\kappa} = \begin{vmatrix} \cos\varphi\cos\kappa & \sin\omega\sin\varphi\cos\kappa + \cos\omega\sin\kappa & -\cos\omega\sin\varphi\cos\kappa + \sin\omega\sin\kappa \\ -\cos\varphi\sin\kappa & -\sin\omega\sin\varphi\sin\kappa + \cos\omega\cos\kappa & \cos\omega\sin\varphi\sin\kappa + \sin\omega\cos\kappa \\ \sin\varphi & -\sin\omega\cos\varphi & \cos\omega\cos\varphi \end{vmatrix} \qquad (2.8)$$

atau

$$R = \begin{vmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{vmatrix} \qquad (2.9)$$

Apabila nilai matriks rotasi R telah diketahui, parameter sudut rotasi berupa ω, φ dan κ dapat ditentukan pula dengan menggunakan persamaan sebagai berikut *Slabaugh (2004), Wolf (1993)*:

$$\sin\varphi = r_{31} \;\; ; \;\; \tan\omega = {}^{-r_{32}}\!/_{r_{33}} \;\; ; \;\; \tan\kappa = {}^{-r_{21}}\!/_{r_{11}} \qquad (2.10)$$

Pada dasarnya, matriks rotasi merupakan matriks ortogonal. Kondisi ini dapat dibuktikan dengan melakukan perkalian antara matriks tersebut dengan nilai transposenya sehingga akan menghasilkan sebuah matriks identitas *Stefanovic (1973), Thompson, (1959)*:

$$R^T R = RR^T = I \qquad (2.11)$$

Kondisi keortogonalan matriks rotasi dapat dibuktikan juga dengan cara melakukan *invers* terhadap matriks tersebut. Apabila nilai *invers* dari matriks tersebut sama dengan nilai transposenya, maka matriks tersebut

dapat dikatakan ortogonal sebagaimana yang dikemukakan oleh *Cooper & Robson (2001)* sebagai berikut :

$$R^{-1} = R^T \tag{2.12}$$

Matriks rotasi dapat pula dibangun dengan menggunakan sistem rotasi t, α, s (*tilt, azimuth, swing*) sebagaimana yang telah dijelaskan diatas dengan menggunakan persamaan sebagai berikut :

$$R = \begin{vmatrix} -\cos\alpha\cos s - \sin\alpha\cos t\sin s & \sin\alpha\cos s - \cos\alpha\cos t\sin s & -\sin t\sin s \\ \cos\alpha\sin s - \sin\alpha\cos t\cos s & -\sin\alpha\sin s - \cos\alpha\cos t\cos s & -\sin t\cos s \\ -\sin\alpha\sin t & -\cos\alpha\sin t & \cos t \end{vmatrix} \tag{2.13}$$

Dengan menggunakan tiga parameter *independent* yaitu a, b dan c sebagai elemen untuk membentuk matriks rotasi *Rodrigues* sebagai berikut (*Faig, 1984*) :

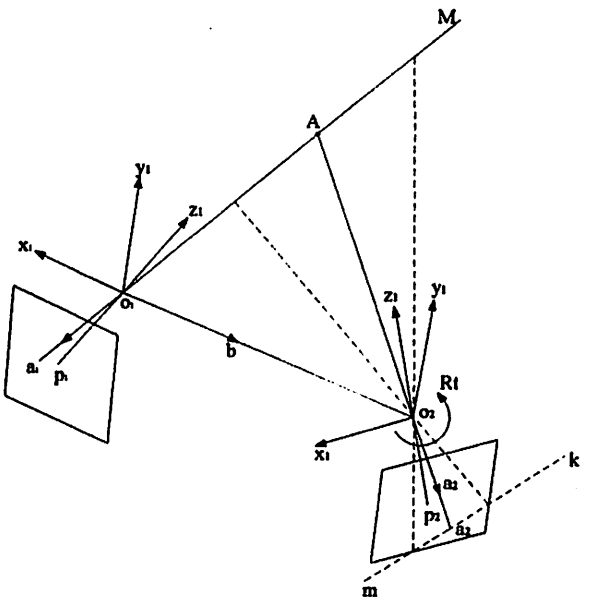
$$R = \Delta^{-1} \begin{vmatrix} 1 + \tfrac{1}{4}(a^2 - b^2 - c^2) & -c + \tfrac{1}{2}ab & b + \tfrac{1}{2}ac \\ c + \tfrac{1}{2}ba & 1 + \tfrac{1}{4}(-a^2 + b^2 - c^2) & -a + \tfrac{1}{2}bc \\ -b + \tfrac{1}{2}ca & a + \tfrac{1}{2}cb & 1 + \tfrac{1}{4}(-a^2 - b^2 + c^2) \end{vmatrix} \tag{2.14}$$

dimana $\Delta = 1 + \tfrac{1}{4}(a^2 + b^2 + c^2)$.

Masih terdapat dua persamaan lainnya untuk membentuk suatu matriks rotasi yaitu matriks *Pope-Hinsken* yang telah di apalikasikan oleh *Zheng & Wang (1992)* dalam proses reseksi dan matriks *Quaternion* yang pertama kali dikenalkan oleh *Hamilton (1999,)* dan telah banyak digunakan dalam aplikasi *computer vision* maupun fotogrametri.

2.2.4 Geometri *Epipolar*

Target A dengan koordinat a_1 pada kamera dengan *perspective center* pada O_1 dan a_2 pada kamera O_2 sesuai dengan *Gambar 2.12*. Bidang epipolar ditunjukkan dengan arsiran. Bidang epipolar tersebut memotong bidang foto pada kamera O_2 melalui garis epipolar km, dimana k merupakan titik proyeksi dari titik pusat kamera O_1 yang melalui pusat kamera O_2 dan M merupakan titik sepanjang vektor λs. Dengan mengasumsikan parameter ekstrinsik dari pusat kamera O_2 telah ditentukan dan memiliki nilai relatif terhadap sumbu koordinat (x_1, y_1, z_1) menggunakan prosedur relatif orientasi. Hal ini memungkinkan untuk menentukan sebuah persamaan garis km relatif terhadap sumbu koordinat (x_2, y_2, z_2). Sehingga, dengan memberikan nilai posisi koordinat homogen A pada kamera O_1, maka koordinat *homogen* A pada kamera O_2 akan berada pada sepanjang garis epipolar.



Gambar 2.12 Bidang Epipolar dan Garis Epipolar

Nilai koordinat k dapat ditentukan dengan menggunakan persamaan kolinear sebagai berikut (*Cooper & Robson, 2001*):

$$x_k = -f \frac{r_{11}(X_{L2} - X_{L1}) + r_{12}(Y_{L2} - Y_{L1}) + r_{13}(Z_{L2} - Z_{L1})}{r_{31}(X_{L2} - X_{L1}) + r_{32}(Y_{L2} - Y_{L1}) + r_{33}(Z_{L2} - Z_{L1})} \qquad (2.15)$$

$$y_k = -f \frac{r_{21}(X_{L2} - X_{L1}) + r_{22}(Y_{L2} - Y_{L1}) + r_{23}(Z_{L2} - Z_{L1})}{r_{31}(X_{L2} - X_{L1}) + r_{32}(Y_{L2} - Y_{L1}) + r_{33}(Z_{L2} - Z_{L1})} \qquad (2.16)$$

Untuk mengevaluasi nilai koordinat k. Titik M dapat didefinisikan dengan menentukan sebuah nilai skala positif λ. Dengan demikian, posisi vektor M relatif terhadap sumbu koordinat (x_1, y_1, z_1) adalah λs, dimana $s = {-a_1}/{|a_1|}$. Sehingga koordinat m dapat ditentukan dengan persamaan:

$$x_m = -f_2 \frac{r_{11}(\Delta X_L - \lambda s_x) + r_{12}(\Delta Y_L - \lambda s_y) + r_{13}(\Delta Z_L - \lambda s_z)}{r_{31}(\Delta X_L - \lambda s_x) + r_{32}(\Delta Y_L - \lambda s_y) + r_{33}(\Delta Z_L - \lambda s_z)} \qquad (2.17)$$

$$x_m = -f_2 \frac{r_{21}(\Delta X_L - \lambda s_x) + r_{22}(\Delta Y_L - \lambda s_y) + r_{23}(\Delta Z_L - \lambda s_z)}{r_{31}(\Delta X_L - \lambda s_x) + r_{32}(\Delta Y_L - \lambda s_y) + r_{33}(\Delta Z_L - \lambda s_z)} \qquad (2.18)$$

Nilai koordinat x dan y untuk titik k dan m merupakan definisi dari vektor posisi k dan m dalam sistem koordinat dua dimensi relatif terhadap nilai pusat foto (*principle point*). Persamaan vektor untuk garis yang melalui titik m dan k ialah:

$$r = k + v(m - k) \qquad (2.19)$$

Dimana, v merupakan variabel skala. Persamaan untuk garis epipolar pada kamera O_1 dapat ditentukan dengan jalan yang sama seperti yang telah dijelaskan diatas.

2.2.5 Matriks *Essensial*

Essensial matriks merupakan sebuah matriks spesial dari persamaan koplanar untuk dua buah foto yang diketahui parameter interior orientasinya. Matriks ini pertama kali diperkenalkan oleh *Longuet-Higgins (1981)* pada komunitas *computer vision* dan *Stefanovic (1973)* pada komunitas fotogrametri.

Dikarenakan matriks essensial merupakan matriks spesial dari kondisi koplanar, maka persamaan untuk matriks essensial dapat dibentuk dari persamaan koplanar seperti pada *Persamaan 2.19* dengan sedikit modifikasi terhadap tiap elemennya sebagai berikut :

$$\begin{pmatrix} x_1 & y_1 & -f \end{pmatrix} R_1 \begin{pmatrix} 0 & b_Z & -b_Y \\ -b_Z & 0 & b_X \\ b_Y & -b_X & 0 \end{pmatrix} R_2 \begin{pmatrix} x_2 \\ y_2 \\ -f \end{pmatrix} = 0 \qquad (2.20)$$

Dimana x_1, y_1, x_2, y_2 merupakan koordinat pada sistem foto, R_1, R_2 merupakan nilai parameter matriks rotasi yang dibentuk dari sudut rotasi ω, φ, κ dan b_X, b_Y, b_Z merupakan selisih posisi antara dua buah posisi kamera. Untuk menyingkat persamaan diatas dapat ditulis kembali dalam sebuah persamaan baru (*Fraser, 2006;, Shan, 1999*).

$$\begin{pmatrix} x_1 & y_1 & -f \end{pmatrix} E \begin{pmatrix} x_2 \\ y_2 \\ -f \end{pmatrix} = 0 \qquad (2.21)$$

atau

$$x_1 E x_2 = 0 \qquad (2.22)$$

Dimana E merupakan matriks essensial yang berhubungan dengan parameter basis dan rotasi foto, x_1, x_2 merupakan koordinat yang terekam pada tiap foto.

Persamaan (2.21) dan *(2.22)* dapat diselesaikan dengan sebuah persamaan *linear homogenous* untuk menentukan enam parameter yang tidak diketahui pada matriks essensial sebagai berikut (*Fraser, 2006b, Stefanovic, 1973*) :

$$Ae = 0 \qquad (2.23)$$

Dimana A adalah matriks koefisien dan e merupakan matriks parameter yang tidak diketahui, sehingga keduanya dapat direpresentasikan dalam bentuk matriks sebagai berikut (*Fraser, 2006b; Pan et al., 1995a, Remondino & El-Hakim, 2006, Stefanovic, 1973, Torr, 2002*) :

$$A = \begin{vmatrix} x_1x_2 & y_1x_2 & -f_1x_2 & x_1y_2 & y_1y_2 & -f_1y_2 & x_1(-f)_2 & y_1(-f)_2 & -f_1(-f)_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1n}x_{2n} & y_{1n}x_{2n} & -f_{1n}x_{2n} & x_{1n}y_{2n} & y_{1n}y_{2n} & -f_{1n}y_{2n} & x_{1n}(-f)_{2n} & y_{1n}(-f)_{2n} & -f_{1n}(-f)_{2n} \end{vmatrix} \qquad (2.24)$$

$$e = \begin{vmatrix} e_{11} & e_{12} & e_{13} & e_{21} & e_{22} & e_{23} & e_{31} & e_{32} & e_{33} \end{vmatrix}^T \qquad (2.25)$$

Selama sistem *Persamaan (2.23)* merupakan persamaan *homogenous*, maka solusi yang akan dihasilkan tidak unik, sehingga permasalahan yang timbul adalah bagaimana menentukan rasio dari sembilan parameter yang tidak diketahui untuk mendapatkan solusi yang unik dari persamaan *homogenous* diatas. Akan tetapi, selama matriks pada *Persamaan (2.24)* memiliki rank lebih kecil dari 9 dapat diasumsikan menjadi 8, dapat ditentukan sebuah rasio perbandingan dari parameter yang tidak diketahui dengan cara sebagai berikut :

$$e_{11} : e_{12} : \cdots\cdots\cdots : e_{32} : e_{33} = D_1 : D_2 : \cdots\cdots\cdots : D_8 : D_9 \qquad (2.26)$$

Dimana D merupakan nilai determinan dari sub-matriks A, dengan nilai subskrip mengindikasikan nilai kolom yang tidak digunakan dalam proses perhitungan determinan.

Apabila nilai rasio tiap element matriks e diketahui, pemecahan solusinya dapat dilakukan dengan menggunakan metode *least square* dengan menentapkan nilai kolom yang memiliki rasio terbesar sebagai matriks observasi.

Selain cara diatas, pemecahan lain dapat pula dilakukan untuk mendapatkan solusi yang unik untuk sistem persamaan *linear homogenous*. Cara tersebut dapat dilakukan dengan menerapkan metode *Singular Value Decompotion* (*SVD*) (*Stewenius et al. 2006; Torr, 2002; Triggs, 2000*) untuk memecah matriks A menjadi tiga buah matriks singular yaitu U,S,V. Adapun persamaan tersebut sebagai berikut :

$$\underset{m\times n}{A} = \underset{m\times m}{U}\ \underset{m\times n}{S}\ \underset{n\times n}{V^T} \qquad (2.27)$$

Dari persamaan diatas, solusi yang didapat untuk parameter matriks e pada *Persamaan (2.25)* merupakan nilai dari kolom matriks V yang berkorespondesi terhadap nilai matriks singular S terkecil dan nilai yang dihasilkan ialah sebuah matriks dengan skala sembarang (*up to scale)*. Pada kasus normal nilai parameter matriks e biasanya berada pada kolom terakhir dari matriks V.

Setelah keseluruhan parameter penyusun matriks essential telah ditentukan, selanjutnya adalah memecah matriks tersebut kedalam nilai

rotasi dan translasi. Proses dekomposi ini dapat dilakukan dengan berbagai cara, seperti yang telah banyak dikemukakan oleh para ahli seperti *Harley (1992); Horn, (1990)* dan *Pan et al. (1995°)* dan akan dibahas kembali pada sub-bab dibawah ini.

2.3. Closed Form Solution

2.4.1 Metode *Rampal*

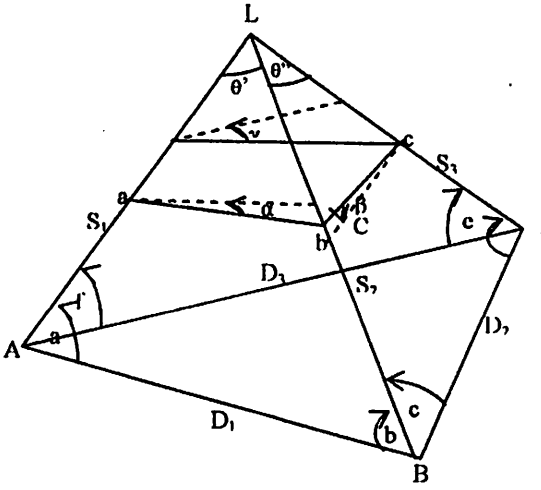
Prinsip dari metode yang dikembangkan oleh *(Rampal, 1979)* yaitu dengan merubah persamaan non linier menjadi persamaan linier dari kondisi kolinieritas yang dilakukan dengan cara proses iterasi dengan asumsi adanya nilai pendekatan. Maka metode *Rampal* hanya dapat menghasilkan parameter sudut rotasi yaitu *omega* (ω), *phi* (φ), dan *kappa* (κ) *(Rampal, 1979). Rampal* (1979) menyatakan bahwa tidak memerlukan nilai pendekatan parameter akan dapat dihasilkan koordinat kamera. Dengan metode *Church's,* nilai koordinat sebenarnya dapat dianalisa tingkat konvergen dan kebenarannya, agar diperoleh solusi yang unik *(Rampal, 1979).*

Persamaan dasar dari metode *Rampal* yaitu menggunakan persamaan kesegarisan (*Persamaan* 2.28), dari persamaan tersebut maka dilakukan proses linierisasi dengan asumsi ω = φ =κ = 0 (*Persamaan* 2.28) (Rampal, 1979).

$$x - x_0 = c\frac{(X - X_L)}{(Z - Z_L)}$$
$$y - y_0 = c\frac{(Y - Y_L)}{(Z - Z_L)} \qquad (2.28)$$

x,y merupakan koordinat foto, x_o,y_o merupakan koordinat titik tengah foto (*principal point*), X,Y,Z merupakan koordinat objek dan X_L,Y_L,Z_L merupakan koordinat kamera. Dari *Persamaan* (2.28) maka akan diperoleh koordinat kamera (X_L,Y_L,Z_L). Secara normal proses iterasi akan dilakukan sebanyak 3 atau 4 kali iterasi untuk mendapatkan solusi yang unik dan konvergen *(Rampal, 1979)*.

Pada proses perhitungannya *(Rampal, 1979)* menggunakan prinsip rumus segitiga dengan model piramid (lihat pada *Gambar* 2.11).



Gambar2.11 Geometri sudut foto

S_1,S_2, dan S_3 merupakan jarak kaki (*leg*) antara koordinat kamera dan koordinat objek, D_1,D_2 dan D_3 merupakan jarak koordinat objek, a,b,c,d,e, dan f merupakan parameter sudut dalam geometri piramid sudut foto, α,β

dan γ merupakan sudut kesendengan dari foto, θ', θ" merupakan sudut yang dibentuk antara koordinat kamera dan koordinat objek.

Dari *Gambar* (2.11) dapat terbentuk bidang segitiga, sehingga dapat ditulis kondisi persamaan *(Rampal, 1979)*:

$$\begin{aligned} S_1 &= \frac{D_1}{\sin\theta'}\sin(b'+\alpha) = \frac{D_3}{\sin\theta'''}\sin(e'-\gamma) \\ S_2 &= \frac{D_1}{\sin\theta'}\sin(a'-\alpha) = \frac{D_2}{\sin\theta''}\sin(d'+\beta) \\ S_3 &= \frac{D_2}{\sin\theta''}\sin(c'-\beta) = \frac{D_3}{\sin\theta'''}\sin(f'+\gamma) \end{aligned} \qquad (2.29)$$

dimana, α, β, dan γ adalah nilai kecil. *Persamaan* (2.29) merupakan persamaan non linier, maka harus dirubah menjadi persamaan linier menjadi *(Rampal, 1979)*:

$$\begin{aligned} a_1\alpha + b_1\beta &= C_1 \\ a_2\alpha + b_2\gamma &= C_2 \\ a_3\beta + b_3\gamma &= C_3 \end{aligned} \qquad (2.30)$$

dimana, diberikan persamaan :

$$\alpha = \frac{a_3 b_2 C_1 + b_3 C_2 b_1 - b_1 b_2 C_3}{a_1 a_3 b_2 + a_2 b_1 b_3} \qquad (2.31)$$

Diketahui bahwa nilai koefisien yaitu :

$$\begin{aligned} a_1 &= D_1 \cos a' \sin\theta'' \\ b_1 &= D_2 \cos d' \sin\theta' \\ C_1 &= D_1 \sin a' \sin\theta'' - D_2 \sin d' \sin\theta' \\ a_2 &= D_1 \cos b' \sin\theta'' \\ b_2 &= D_3 \cos e' \sin\theta' \\ C_2 &= D_3 \sin c'\theta' - D_1 \sin b' \sin\theta'' \\ a_3 &= D_2 \cos c' \sin\theta''' \\ b_3 &= D_3 \cos f' \sin\theta'' \\ C_3 &= D_2 \sin c'\theta''' - D_3 \sin f' \sin\theta'' \end{aligned} \qquad (2.32)$$

Dengan diketahui persamaan untuk θ' yaitu :

$$\cos\theta' = \frac{s_1^2 + s_2^2 - d^2}{2s_1 s_2} \qquad (2.33)$$

Dimana,

$$\begin{aligned} s_1^2 &= (x_1 - x_0)^2 + (y_1 - y_0)^2 + C^2 \\ s_2^2 &= (x_2 - x_0)^2 + (y_2 - y_0)^2 + C^2 \\ d^2 &= (x_2 - x_1)^2 + (y_2 - y_1)^2 \end{aligned} \qquad (2.34)$$

Dengan cara yang sama, maka diperoleh :

$$\begin{aligned} \cos a' &= \frac{s_1^2 + d^2 - s_2^2}{2s_1 d} \\ \cos b' &= \frac{s_2^2 + d^2 - s_1^2}{2s2d} \end{aligned} \qquad (2.35)$$

Dari *Persamaan* (2.29) dan (2.32) diperoleh persamaan baru untuk menentukan α, β, dan γ yang mungkin dengan menggunakan data koordinat foto dan objek. Dari hasil sudut sebelumnya memungkinkan untuk menyelesaikan S_1, S_2 dan S_3 dengan persamaan sederhana (*Persamaan* 2.36).

$$\frac{S_1}{\sin(a' - \alpha)} = \frac{D_1}{\sin\theta'} \qquad (2.36)$$

Dengan menggunakan cara yang sama, dapat ditentukan nilai S_2 dan S_3. Jika S_1 dan S_2 maka *Persamaan* (2.37) dapat dirubah kedalam bentuk persamaan linier.

$$\frac{1 + D_3 + D_5}{D_6} = \frac{\Delta X\Delta X_2 + \Delta Y\Delta Y_2 + \Delta Z\Delta Z_2 + DS_2}{\Delta X\Delta X_1 + \Delta Y\Delta Y_1 + \Delta Z\Delta Z_1 + DS_1} = \cot\left(\frac{a' - \alpha}{2}\right)\cot\left(\frac{b' + \alpha}{2}\right) = k \qquad (2.37)$$

dengan menggunakan persamaan (2.37) maka :

$$\Delta X(X_2 - X_0) + \Delta Y(Y_2 - Y_0) + \Delta Z(Z_2 - Z_0) = D(kS_1 - S_2) \qquad (2.38)$$

dimana,

$$X_2 = X_1 + \Delta X \quad Y_2 = Y_1 + \Delta Y \quad Z_2 = Z_1 + \Delta Z \qquad (2.39)$$

Sehingga dapat diperleh persamaan baru untuk memperoleh koordinat posisi kamera, yaitu:

$$\Delta XX_0 + \Delta YY_0 + \Delta ZZ_0 = \Delta XX_1 + \Delta YY_1 + \Delta ZZ_1 - \frac{D_1(D_1 + D_2 - kS_1)}{k-1} \qquad (2.40)$$

persamaan (2.40) merupakan persamaan linier untuk *direct (closed) form* dari koordinat posisi kamera. Dapat diambil kesimpulan bahwa metode *rampal* :

1. Tidak memerlukan nilai pendekatan parameter.
2. Tidak memerlukan nilai pendekatan sudut rotasi.
3. Dapat digunakan tipe foto untuk di udara dan di darat .
4. Hanya digunakan satu foto dengan informasi koordinat foto dan koordinat objek.

2.4.2 Metode *Zhang*

Pada permasalahan sebelum-sebelumnya, telah dijelaskan bahwa untuk memperoleh enam parameter orientasi luar diperlukan nilai pendekatan dengan menggunakan prinsip *closed from solution.* Kemudian dilakukan analisa pada metode-metode tersebut, sehingga *Zeng and Wang (1992)* melakukan penelitian dengan menggunakan metode yang telah dijelaskan dan diuji oleh *Fischler and Bolles (1980)*. Metode itu dikenal dengan

metode permasalahan penentuan lokasi atau *Location determination Problem* (LDP) untuk analisa foto dan memperoleh posisi koordinat objek dengan menggunakan prinsip perkalian murni *(Fischler and Bolles, 1981; Zeng and Wang, 1992)*.

Metode yang dikembangkan oleh (*Zeng and Wang, 1992*) mencakup tiga tahapan inti yaitu :

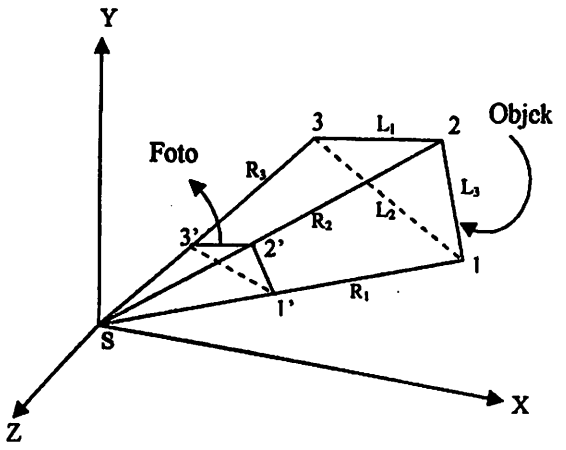
1. Penyelesaian untuk memperoleh posisi koordinat kamera ($X_L Y_L Z_L$).
2. Penyelesaian untuk memperoleh parameter rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ), berdasarkan atas algoritma *Pope-Hinsken* yang telah diuji oleh (*Hinsken, 1988*).
3. Mendiskusikan penyelesaian untuk memperoleh parameter reseksi dengan menggunakan prinsip kurva kritis.

   Tetapi dalam penjelasan didalam metode (*Zeng and Wang, 1992*) kali ini tidak dijelaskan tentang prinsip kurva kritis.

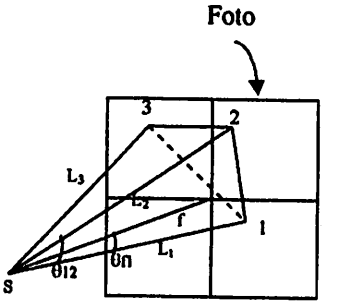
Ketiga tahapan ini juga dijelaskan dengan detail oleh *(Fischler and Bolles, 1981)*.

2.4.3 Penentuan Koordinat Kamera

Untuk memperoleh enam parameter orientasi luar maka (*Zeng and Wang, 1992*) terlebih dahulu menentukan posisi koordinat kamera dengan diketahui koordinat foto dan koordinat objek dari satu foto (*Gambar* 2.11).



Gambar a Ilustrasi dari piramid koordinat foto



Gambar b ilustrasi geometri dari koordinat foto dan koordinat kamera

Gambar 2.11 Hubungan antara pyramid koordinat doto dan koordinat objek

Dari *Gambar* 2.11 diatas dapat diketahui bahwa S merupakan posisi koordinat kamera, 1, 2, dan 3 merupakan koordinat titik kontrol objek, 1',2', dan 3' merupakan koordinat titik foto (dan tidak segaris), L_i dan R_i merupakan jarak antara jarak posisi koordinat objek dan koordinat titik foto pada piramid (*Zeng and Wang, 1992*). Dengan catatan urutan posisi titik 1',2', dan 3' searah dengan arah jarum jam. Dan sistem koordinat foto sebenarnya di S yang digunakan pada perhitungan. Misalnya x_i,y_i, adalah koordinat foto, dimana f merupakan jarak titik tengah foto dan arahnya berlawanan dengan arah sumbu z sehingga z = -f pada sistem koordinat foto, dan X_i, Y_i, dan Z_i merupakan koordinat objek pada sistem koordinat foto (*Zeng and Wang, 1992*). Dari gambar 2.11 dapat disusun suatu formula, sebagai berikut (*Zeng and Wang, 1992*):

$$
\begin{aligned}
x_1 &= -fx\frac{X_1}{Z_1}; \qquad y_1 = -fx\frac{Y_1}{Z_1} \\
x_2 &= -fx\frac{X_2}{Z_2}; \qquad y_1 = -fx\frac{Y_2}{Z_2} \\
x_3 &= -fx\frac{X_3}{Z_3}; \qquad y_1 = -fx\frac{Y_3}{Z_3}
\end{aligned}
\tag{2.41}
$$

$$
\begin{aligned}
(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2 &= L_3^2 \\
(X_2 - X_3)^2 + (Y_2 - Y_3)^2 + (Z_2 - Z_3)^2 &= L_1^2 \\
(X_3 - X_1)^2 + (Y_3 - Y_1)^2 + (Z_3 - Z_1)^2 &= L_2^2
\end{aligned}
\tag{2.42}
$$

Dimana, L_1, L_2, L_3 dihitung dengan menggunakan koordinat titik objek pada sistem koordinat foto. Dari persamaan (2.41) dan (2.42) diperoleh persamaan baru, sebagai berikut (*Zeng and Wang, 1992*):

$$
\begin{aligned}
AxZ_1^2 - DxZ_1xZ_2 + BxZ_2^2 &= L_3^2 \\
BxZ_2^2 - ExZ_2xZ_3 + CxZ_3^2 &= L_1^2 \\
CxZ_3^2 - FxZ_3xZ_1 + AxZ_1^2 &= L_2^2
\end{aligned}
\tag{2.43}
$$

Dimana :

$$
\begin{aligned}
A &= (x_1^2 + y_1^2 + f^2)/f^2 \\
B &= (x_2^2 + y_2^2 + f^2)/f^2 \\
C &= (x_3^2 + y_3^2 + f^2)/f^2 \\
D &= (2x_1x_2 + 2y_1y_2 + 2f^2)/f^2 \\
E &= (2x_2x_3 + 2y_2y_3 + 2f^2)/f^2 \\
F &= (2x_3x_1 + 2y_3y_1 + 2f^2)/f^2
\end{aligned}
\tag{2.44}
$$

A, B, C, D, dan E merupakan nilai konstanta yang dihitung dari koordinat foto dan fokus. Maka dapat disusun persamaan baru untuk mencari nilai konstanta dari D_1, E_1, dan F_1 dari *Persamaan* (2.44) menjadi (*Zeng and Wang, 1992*):

$$D_1 = D/\sqrt{AxB}$$
$$E_1 = E/\sqrt{BxC} \qquad (2.45)$$
$$F_1 = F/\sqrt{CxA}$$

Setelah ditemukan *Persamaan* (2.42) dengan asumsi itu maka diperoleh hubungan antara jarak posisi koordinat objek dan koordinat titik foto pada piramid, dapat ditulis suatu persamaan (*Zeng and Wang, 1992)* :

$$R_1^2 - D_1 x R_1 x R_2 + R_2^2 = L_3^2$$
$$R_2^2 - E_1 x R_2 x R_3 + R_3^2 = L_1^2 \qquad (2.46)$$
$$R_3^2 - F_1 x R_3 x R_1 + R_1^2 = L_2^2$$

Nilai konstanta dari perbandingan jarak pada koordinat titik objek yaitu K_{12} dan K_{32}, dapat ditulis sebagai berikut (*Zeng and Wang, 1992*):

$$K_{12} = \left(\frac{L_1}{L_2}\right)^2 \qquad K_{32} = \left(\frac{L_3}{L_2}\right)^2 \qquad (2.47)$$

Dari *Persamaan* (2.44) dan (2.45) diperoleh suatu bentuk persamaan baru yang mengandung operasi akar empat, menjadi (*Zeng and Wang, 1992*):

$$N_1 n^4 + N_2 n^3 + N_3 n^2 + N_4 n^1 + N_5 n^0 = 0 \qquad (2.48)$$

Dimana :

$$N_1 = (1-K_{12})^2 + K_{32}^2 - 2x(1-K_{12})xK_{32} - K_{32}xE_1^2 + 4xK_{32}x(1-K_{12})$$
$$N_2 = K_{32}xF_1x(-2-2xK_{32}+4xK_{12}+E_1^2) + K_{12}x(E_1xD_1+2xF_1 - 2xK_{12}xF_1) + E_1xD_1x(K_{32}-1)$$
$$N_3 = K_{32}x(K_{32}xF_1^2 - F_1xD_1xE_1 - 4xK_{12} + 2xK_{32} - E_1^2 - 2xK_{12}xF_1^2) + K_{12}x(K_{12}xF_1^2 + 2xK_{12} - F_1xE_1xD_1 - D_1^2) + D_1^2 - 2 + E_1^2 \qquad (2.49)$$
$$N_4 = K_{32}x(2xF_1 - 2xK_{32}xF_1 + E_1xD_1 + 4xK_{12}xF_1) + K_{12}x(E_1xD_1 - 2xK_{12}xF_1 + F_1xD_1^2 - 2xF_1) - D_1xE_1$$
$$N_5 = K_{32}x(K_{32}-2-2xK_{12}) + K_{12}x(K_{12}-D_1^2+2) + 1$$

Setelah diperoleh nilai N_i maka *Persamaan* (*2.49*) menjadi (*Zeng and Wang, 1992*):

$$M_1 n^4 + M_2 n^3 + M_3 n^2 + M_4 n^1 + M_5 n^0 = 0 \qquad (2.50)$$

M_i merupakan nilai koefisien dari bentuk *biqudratic polynomial*, dengan asumsi nilai M_i diperoleh dari :

$$M_1 = \frac{N_1}{N_1}; \quad M_2 = \frac{N_2}{N_1}; \quad M_3 = \frac{N_3}{N_1}; \quad M_4 = \frac{N_4}{N_1}; \quad M_5 = \frac{N_5}{N_1}; \qquad (2.52)$$

Persamaan (2.50) merupakan tipe persamaan aljabar akar empat dan mempunyai solusi yang umum. Meskipun metode solusi ini dapat diketahui, tetapi harus dilakukan dengan cara yang tepat sehingga memperoleh solusi yang pasti. Dengan demikian diperoleh hasil akar empat dari n_1,n_2,n_3, dan n_4 pada *Persamaan* (2.50), yaitu terdapat 8 solusi. Nilai yanga hanya nilai real positif yang diperlukan dalam metode *zhang's* dan setiap syarat *unknowns* dari kedua derajat. Dengan memecahkan persamaan aljabar akar empat tersebut satu dari empat nilai real positif yang dapat digunakan untuk menetukan koordinat posisi kamera dapat ditentukan (*Zeng and Wang, 1992*).

Dari tiap nilai positif n, dapat diketahui jarak antara koordinat posisi kamera dan koordinat objek (R_1, R_2, dan R_3) yaitu diperoleh dengan melakukan proses *intersection* (persilangan) titik dengan radius R_1, R_2, R_3 dan berpusat pada tiga titik kontrol *(Zeng and Wang, 1992)*. Persamaan untuk menghitung nilai R_i dapat dituliskan sebagai berikut :

$$R_1 = p = L_2 / \sqrt{n^2 - F_1 xn + 1} \qquad (2.51)$$

$$R_3 = nxp \tag{2.52}$$

Dengan menggunakan *Persamaan* (2.46) dapat dihitung nilai R_2 yang mempunyai dua solusi, yaitu (*Zeng and Wang, 1992*):

$$R_2 = (D_1 x R_1 \pm \sqrt{D_1^2 x R_1^2 - 4x(R_1^2 - L_3^2)})/2 \tag{2.53}$$

$$R_2 = (E_1 x R_3 \pm \sqrt{E_1^2 x R_3^2 - 4x(R_3^2 - L_1^2)})/2 \tag{2.54}$$

Sehingga dapat disusun persamaan untuk hubungan antara jarak R_1,R_2, dan R_3 dan koordinat titik objek dan koordinat kamera, yaitu (*Zeng and Wang, 1992*):

$$\begin{aligned}(X_1 - X_L)^2 + (Y_1 - Y_L)^2 + (Z_1 - Z_L)^2 &= R_1^2 \\ (X_2 - X_L)^2 + (Y_2 - Y_L)^2 + (Z_2 - Z_L)^2 &= R_2^2 \\ (X_3 - X_L)^2 + (Y_3 - Y_L)^2 + (Z_3 - Z_L)^2 &= R_3^2\end{aligned} \tag{2.55}$$

dimana X_i,Y_i dan Z_i adalah koordinat titik kontrol, X_L,Y_L dan Z_L adalah koordinat kamera. Maka *Persamaan* (2.55) dapat dijabarkan menjadi (*Awange and Grafarend September, 2004*) :

$$\begin{aligned}S_1^2 - S_2^2 &= X_1^2 - X_2^2 + Y_1^2 - Y_2^2 + Z_1^2 - Z_2^2 + a \\ S_2^2 - S_3^2 &= X_2^2 - X_3^2 + Y_2^2 - Y_3^2 + Z_2^2 - Z_3^2 + b\end{aligned} \tag{2.56}$$

dimana,

$$\begin{aligned}a &= 2X(X_2 - X_1) + 2Y(Y_2 - Y_1) + 2Z(Z_2 - Z_1) \\ b &= 2X(X_3 - X_2) + 2Y(Y_3 - Y_2) + 2Z(Z_3 - Z_2)\end{aligned} \tag{2.57}$$

Notasi a dan b dimisalkan sebagai hubungan antara parameter yang dicari pada formula disisi kanan dan sisi kiri pada *Persamaan* (2.56) setetah menghitung nilai T, yaitu merupakan nilai koreksi dari solusi koordinat

kamera yang telah ditentukan. Nilai T diperoleh dari hasil vektor 12, 13 dan 1S seperti :

$$T = \vec{1S} . (12 \times 13) \qquad (2.58)$$

Jika T > 0 maka nilai dari solusi koordinat posisi kamera adalah benar dan T < 0 maka nilai dari solusi koordinat posisi kamera adalah salah dengan menggunakan minimal tiga koordinat foto dan tiga koordinat objek.

2.4.4 Penentuan Parameter Rotasi

Didalam perhitungan fungsi trigonometri, terdapat permasalahan singular. Para ahli fotogrametri telah menyusun matriks rotasi dengan menggunakan parameter aljabar (*Thompson, 1959*) tetapi metode ini tidak membuktikan secara praktis (*Zeng and Wang, 1992*).

Pope (1970) mengusulkan menggunakan empat parameter aljabar dengan notasi yaitu *d,a,b* dan *c*, yang digunakan untuk menyusun matriks rotasi (R). *Hinsken (1988)* menurunkan satu set formula yang kompleks untuk membuat parameter (*Pope 1970*) dapat digunakan untuk perhitungan fotogrametri yang praktis. Secara detail algoritma tentang penyusunan matriks rotasi (R) dapat ditemukan pada *Hinsken (1988)* dan (Zeng and Wang, 1992). Secara praktis matriks rotasi dapat disusun menjadi *(Mikhail (2001); Bethel et al.; Thompson 1959; Pope 1970; Hinsken 1988; Zeng and Wang, 1992)*:

$$R=\begin{bmatrix} d^2+a^2-b^2-c^2 & 2(ab+cd) & 2(ac-bd) \\ 2(ab-cd) & d^2-a^2+b^2-c^2 & 2(bc+ad) \\ 2(ac+bd) & 2(bc-ad) & d^2-a^2-b^2+c^2 \end{bmatrix} \qquad (2.59)$$

dimana :

$$d^2+a^2-b^2+c^2=1 \qquad (2.60)$$

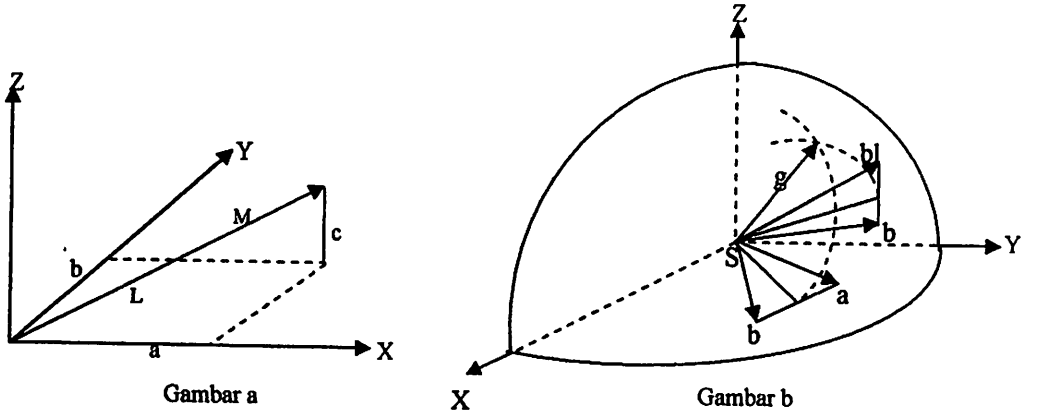
Hal itu telah dibuktikan secara praktis algoritma *Pope-Hinsken* tidak bermasalah untuk bentuk singular dan menuju ke sistem yang konvergen (*Zeng and Wang, 1992*). Untuk memberikan interpretasi fisik untuk parameter ini dapat diasumsikan menjadi (*Zeng and Wang, 1992*):

$$M=[a \quad b \quad c]^T \qquad (2.61)$$

Dengan adanya matriks parameter dan rotasi, maka penyusunan matriks menjadi *(Zeng and Wang, 1992)* :

$$RM=M \qquad (2.62)$$

Sekarang, setelah mentransformasikan M dengan R, maka M tetap tidak berubah sesuai dengan *Persamaan* (2.62), dan dapat diilustrasikan seperti *Gambar* dibawah ini (*Zeng and Wang, 1992*):



Gambar 2.12 Hubungan antara gambar a (Geometri a,b,c) dan gambar b (penetuan sumbu rotasi pada koordinat objek dan foto)

dimana a,b,c merupakan parameter dari matriks rotasi. Hasil dari proses transformasi R akan dihilangkan, dengan sistem koordinat yang dinotasikan denga sudut γ yang mengelilingi vektor M (sumbu rotasi x). Dengan melihat *Persamaan* (2.60) dan pada *Gambar* (2.12) maka dapat diperoleh persamaan :

$$|M| = L_m = \sqrt{a^2 + b^2 + c^2} = \sqrt{1 - d^2} \qquad (2.63)$$

$|M|$ = merupakan nilai magnitude dari M. Dengan memisalkan sudut antara vektor t dan R ditransformasikan terhadap t yang dinotasikan dengan simbol γ. Kemudian menjadi (*Zeng and Wang, 1992*):

$$\frac{t \circ Rt}{|t| x |t|} = \cos\gamma \qquad (2.64)$$

Dengan mengembangkan dan memanipulasi *Persamaan* (2.64) dapat diperoleh (*Zeng and Wang, 1992*):

$$L_m = \sqrt{(1 - \cos\gamma)/2} = \sqrt{1 - d^2} \qquad (2.65)$$

Jadi, sudut rotasi γ dari sistem koordinat telah dinyatakan pada syarat-syarat dari parameter rotasi d pada *Persamaan* (2.65). Sehingga akan diperoleh parameter d. Tahapan selanjutnya adalah menentukan sumbu rotasi, dengan asumsi bahwa *(Zeng and Wang, 1992)*:

al,bl : merupakkan dua unit vektor titik kontrol dari koordinat posisi kamera.

ap,bp : merupakan dua unit vektor titik foto yang berhubungan dengan koordinat posisi kamera.

g(XYZ) : sumbu rotasi sistem koordinat dari objek ke foto.

Garis vektor dari ap dan al akan membentuk garis tegak lurus dan membagi dua bagian sudut antara ap dan al pada garis *sm*1*g*. Dengan menggunakan bp dan bl, dengan cara yang sama dapat dibentuk, sehingga terdapat perpotongan garis pada *sm*2*g* yang memberikan sumbu rotasi *g* pada *Gambar* (2.12) sehingga diperoleh bahwa (*Zeng and Wang, 1992*):

$$ap = \begin{pmatrix} ap_x & ap_y & ap_z \end{pmatrix}^T \qquad bp = \begin{pmatrix} bp_x & bp_y & bp_z \end{pmatrix}^T \tag{2.66}$$

Dengan arah parameter sumbu rotasi *g* adalah (*Zeng and Wang, 1992*):

$$\begin{aligned} p &= (ap_y - al_y)(bp_z - bl_z) - (ap_z - al_z)(bp_y - bl_y) \\ q &= (ap_z - al_z)(bp_x - bl_x) - (ap_x - al_x)(bp_z - bl_z) \\ r &= (ap_x - al_x)(bp_y - bl_y) - (ap_y - al_y)(bp_x - bl_x) \end{aligned} \tag{2.67}$$

Dari *Persamaan* (2.66) dan (2.67), maka dapat diketahui sudut rotasi dari sistem koordinat objek ke foto yaitu *γ*, menjadi (Zeng and Wang, 1992):

$$\cos\gamma = \frac{(gxap) \circ (gxal)}{|gxap| x |gxal|} \tag{2.68}$$

Dari *Persamaan* (2.65) dapat diketahui nilai dari parameter *d*, yaitu(*Zeng and Wang, 1992*) :

$$d = \sqrt{(1+\cos\gamma)/2} \tag{2.69}$$

Oleh karena nilai arah dan komponen koordinat sumbu rotasi *g* sepadan, maka persamaan (2.59) menjadi (*Zeng and Wang, 1992*):

$$\begin{aligned} a &= p\sqrt{(1-d^2)/(p^2+q^2+r^2} \\ b &= q\sqrt{(1-d^2)/(p^2+q^2+r^2} \\ c &= r\sqrt{(1-d^2)/(p^2+q^2+r^2} \end{aligned} \tag{2.70}$$

dimana, *a*,*b* dan *c* merupakan parameter matriks rotasi.

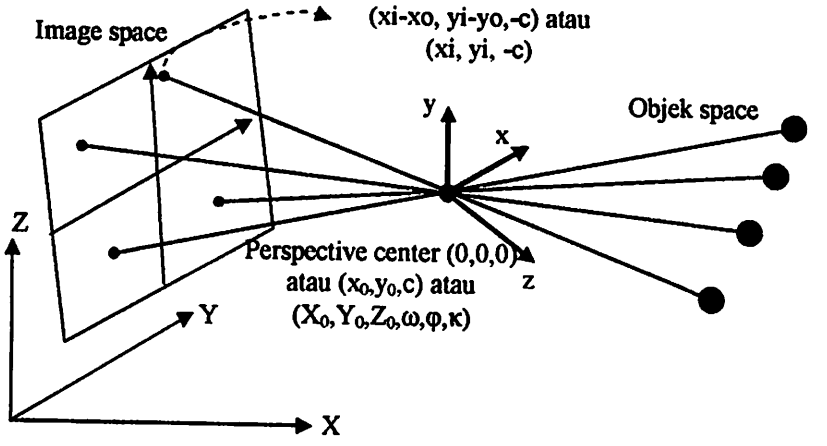
Untuk dapat diperoleh susunan matriks seperti pada *Persamaan* (2.68) digunakan 3 koordinat titik kontrol objek, cukup untuk menemukan nilai dari $d,a,b,$dan c, sehingga koordinat titik kontrol objek yang keempat digunakan untuk menguji kecocokan hasil dari 2 solusi akhir dari koordinat posisi kamera. Jika terdapat kecocokan antara vektor koordinat foto dan koordinat objek dengan petunjuk arah pada koordinat x pada foto dan koordinat X pada objek didefinisikan sebagai 1.0, maka solusi itulah yang paling tepat dan unik dari dua solusi akhir yang ada (*Zeng and Wang, 1992*).

2.4. *Resection*

Dasar persamaan untuk reseksi fotogrametri adalah persamaan proyeksi *(Thompson, 1966)*, yang menghubungkan foto dengan koordinat medan. Reseksi adalah prosedur yang digunakan untuk menentukan posisi dan orientasi dari kamera saat pengambilan foto. Teknik reseksi, atau sering disebut teknik perpotongan kebelakang adalah sebuah teknik untuk menentukan 6 elemen orientasi luar (ω, φ, κ, X_L, Y_L, Z_L) dari foto. Metode ini membutuhkan minimal tiga titik kontrol, yang dikenal dengan koordinat objek (X_i, Y_i, Z_i), yang diambil gambarnya lewat pemotretan sehingga dapat dihitung koordinat ruangnya *(Wolf & Dewitt, 2000)*.

Metode untuk evaluasi secara langsung pada enam parameter orientasi luar (*Eksterior Orientation)* diperoleh dari diukurnya koordinat foto pada bidang foto dengan tiga titik kontrol *non-linear* yang tidak memerlukan beberapa nilai pendekatan (*Zeng and Wang, 1992)*.

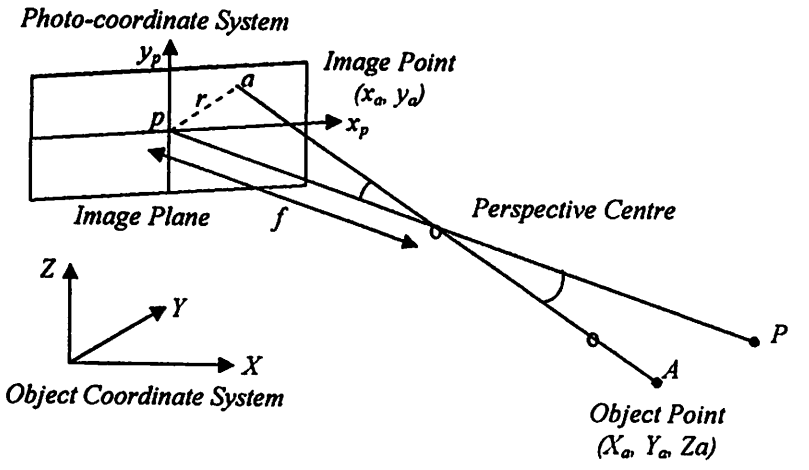
Prosedur ini memberikan koordinat secara langsung dari titik tengah pusat kamera. Bentuk secara aljabar akan digunakan pada matrikss rotasinya. Jika diperlukan, nilai untuk rotasi ω, φ dan k dapat di cari dari 9 elemen matriks rotasi dengan dimensi matriks 3x3 (*Cooper, 1987).*



Gambar 2.13 Teknik Resection

Hasil akhir reseksi adalah penentuan posisi dan orientasi dari kamera selama pemotretan. Koordinat titik tengah kamera dan orientasi yang terdiri atas parameter orientasi luar, dapat dipecahkan oleh hasil dari reseksi dalam bentuk 3D *(Awange and Grafarend, 2004).*

Reseksi ruang dengan kolinearitas meliputi formulasi yang disebut dengan persamaan kolinear (*collinearity equation*) untuk sejumlah koordinat titik kontrol objek (*Xi, Yi, Zi)* yang diketahui dan gambarnya tampak pada foto. Kemudian persamaan itu diselesaikan untuk enam parameter orientasi luar yang belum diketahui dan tampak pada foto. Kolinearitas dideskripsikan sebagai kondisi dimana kamera, beberapa titik objek, dan foto berada pada satu garis lurus pada ruang 3D.



Gambar 2.2 Kondisi Kolinearitas

Keterangan Gambar :

x_a, y_a : Koordinat foto

X_a, Y_a, Z_a : Koordinat titik objek

X_L, Y_L, Z_L : Koordinat kamera

f : Panjang fokus kamera

x_p, y_p : Koordinat dari titik tengah foto

Persamaan dasar dari kondisi kolinearitas bersifat nonlinier dan dapat dilinierkan dengan menggunakan teorema Taylor. Penggunaan teorema Taylor untuk menyelesaikan kolinearitas memerlukan pendekatan awal bagi semua unsur orientasi luar yang tidak diketahui. Dua persamaan menunjukkan kondisi kolinearitas untuk setiap titik pada foto, satu persamaan untuk koordinat foto x dan persamaan yang lain untuk koordinat foto y (*Wolf, 2000*).

$$x_a = x_0 - f\left[\frac{m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)}\right] \quad (2.71)$$

$$y_a = y_0 - f\left[\frac{m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)}\right] \quad (2.72)$$

dimana,

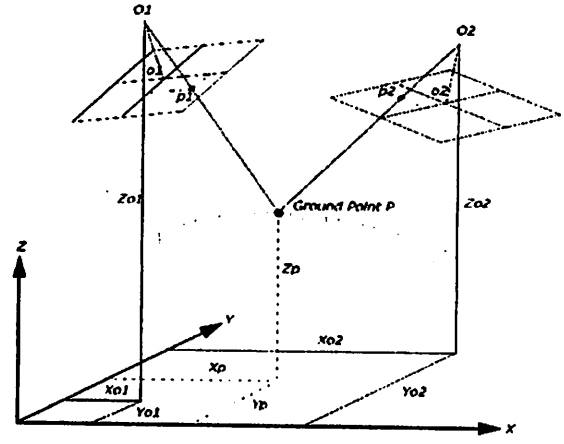
x_o, y_o	:	Koordinat foto titik a
x_a, y_a	:	Koordinat foto yang diukur
X_A, Y_A, Z_A	:	Koordinat objek untuk titik A
X_L, Y_L, Z_L	:	Koordinat kamera
f	:	Panjang fokus kamera,
m_{ij}	:	3 sudut matriks rotasi *ortogonal* (ω, φ, κ)

Persamaan (2.71) dan (2.72) merupakan persamaan *non linear* dan meliputi sembilan unsur yang belum diketahui, 3 sudut perputaran (ω, φ, κ) yang berhubungan dengan matriks m, 3 koordinat kamera (X_L, Y_L dan Z_L), 3 koordinat titik objek (X_A, Y_A dan Z_A) untuk titik A. Persamaan non linear dapat dilinearisasikan dengan menggunakan *teorema Taylor* (*Wolf, 2000*).

2.5. Intersection

Interseksi adalah salah satu teknik yang digunakan secara umum untuk menentukan koordinat titik-titik (X,Y,Z) yang terdapat pada daerah pertampalan dua buah foto yang telah diketahui parameter inerinsik dan parameter eksterinsiknya (*Wolf, 1993*). Dengan menggunakan persamaan

kondisi kolinearitas maka titik-titik koordinat obyek yang terdapat pada foto dapat diketahui posisinya secara tiga dimensi (*X, Y, Z*) (*Wolf, 1993*).



Gambar 2.14 Interseksi foto yang bertampalan pada Aerial Photogrammetry

Teknik interseksi ini juga dapat digunakan untuk menentukan titik *Ground Control Point's* (*GCP*'s) (*Geosystem, 2006b*). Dalam proses penentuan nilai koordinat obyek menggunakan persamaan kolinear terdapat dua cara yaitu linear dan non-linear (*Wolf, 1993*). Pertama, cara non-linear ini merupakan pengembangan dari persamaan segitiga sebangun antara sistem koordinat di foto dengan koordinat dalam ruang tiga dimensi. Hubungan ini dapat ditulis dalam sebuah persamaan matematika sebagai berikut :

$$\frac{x'_a}{X_A - X_L} = \frac{y'_a}{Y_A - Y_L} = \frac{z'_a}{Z_A - Z_L} \qquad (2.73)$$

Atau dalam bentuk sebagai berikut :

$$X_A - X_L = \frac{X_A - X_L}{z'_a} x'_a \qquad (2.74)$$

$$Y_A - Y_L = \frac{Y_A - Y_L}{z'_a} y'_a \qquad (2.75)$$

$$Z_A - Z_L = \frac{Z_A - Z_L}{z'_a} z'_a \qquad (2.76)$$

Dimana, X_A, Y_A, Z_A merupakan koordinat obyek, X_L, Y_L, Z_L merupakan posisi kamera dan x'_a, y'_a, z'_a merupakan koordinat foto sesuai dengan nilai orientasi foto dan dapat ditentukan menggunakan persamaan sebagai berikut :

$$\begin{aligned} x'_a &= r_{11}x_a + r_{12}y_a + r_{13}(-f) \\ y'_a &= r_{21}x_a + r_{22}y_a + r_{23}(-f) \\ z'_a &= r_{31}x_a + r_{32}y_a + r_{33}(-f) \end{aligned} \qquad (2.77)$$

Dengan melakukan subtitusi λ_a bagi $(Z_A - Z_L)/z'_a$ pada *Persamaan (2.73)-(2.75)* didapat sebuah persamaan untuk menentukan nilai koordinat titik *A* bagi foto *1* sebagai berikut :

$$\begin{aligned} X_A &= \lambda_{a1}x'_{a1} + X_{L1} \\ Y_A &= \lambda_{a1}y'_{a1} + Y_{L1} \\ Z_A &= \lambda_{a1}z'_{a1} + Z_{L1} \end{aligned} \qquad (2.78)$$

Dengan jalan yang sama dapat ditulis persamaan untuk foto *2* sebagai berikut :

$$\begin{aligned} X_A &= \lambda_{a2}x'_{a2} + X_{L2} \\ Y_A &= \lambda_{a2}y'_{a2} + Y_{L2} \\ Z_A &= \lambda_{a2}z'_{a2} + Z_{L2} \end{aligned} \qquad (2.79)$$

Susunan *Persamaan 2.77* sama dengan *Persamaan 2.78* yang membedakannya adalah proses penentuan nilai parameter λ_a. Untuk nilai

λ_a pada foto *1* dan *2* secara berurutan dapat ditentukan dengan cara sebagai berikut :

$$\lambda_{a1} = \frac{y'_{a1}(X_{L2} - X_{L1}) - x'_{a1}(Y_{L2} - Y_{L1})}{x'_{a1}\, y'_{a2} - x'_{a2}\, y'_{a1}} \qquad (2.80)$$

$$\lambda_{a2} = \frac{y'_{a2}(X_{L2} - X_{L1}) - x'_{a2}(Y_{L2} - Y_{L1})}{x'_{a2}\, y'_{a1} - x'_{a1}\, y'_{a2}} \qquad (2.81)$$

Kelemahan penggunaan cara kerja ini ialah bahwa pengulangan yang dilakukan tidak digunakan didalam penyelesaian dengan kuadrat terkecil. Akan tetapi cara kerja ini akan menyajikan nilai pendekatan awal yang baik.

Kedua, cara linear ini biasanya digunakan untuk melakukan perbaikan nilai pendekatan awal yang didapat dari cara non-linear seperti yang dijelaskan diatas. Persamaan umum yang digunakan adalah persamaan kolinear yang telah dilinearisasi seperti pada *persamaan:*

$$v_x + J = b_{11}d\omega + b_{12}d\varphi + b_{13}d\kappa - b_{14}dX_L - b_{15}dY_L - b_{16}dZ_L + b_{14}dX_A + b_{15}dY_A + b_{16}dZ_A \quad (2.82)$$

$$v_x + K = b_{11}d\omega + b_{12}d\varphi + b_{13}d\kappa - b_{14}dX_L - b_{15}dY_L - b_{16}dZ_L + b_{14}dX_A + b_{15}dY_A + b_{16}dZ_A \quad (2.83)$$

Akan tetapi, perlu diingat bahwa dalam proses interseksi, parameter yang akan ditentukan berupa parameter koordinat titik obyek dalam ruang tiga dimensi. Sehingga, *Persamaan 2.82* dan *2.83* direduksi menjadi persamaan baru yaitu (*Wolf & Dewitt, 2000*):

$$v_x + J = b_{14}dX_A + b_{15}dY_A + b_{16}dZ_A \qquad (2.84)$$

$$v_x + K = b_{14}dX_A + b_{15}dY_A + b_{16}dZ_A \qquad (2.85)$$

Dari persamaan diatas nilai parameter yang dicari berupa parameter koreksi untuk nilai pendekatan awal koordinat obyek (dX_a, dY_a, dZ_a) Keuntungan dari persamaan linear ini dapat diselesaikan dengan cara pengulangan hingga nilai koreksi dapat diabaikan.

2.6. Teknik Pembuatan Algoritma Dengan C#

Keseluruhan alur perancangan bahasa yang akan dimasukkan dalam program komputer yang diurutkan secara logis untuk menyelesaikan masalah dan yang disusun secara sistematis disebut dengan ***algoritma.***

Untuk memudahkan proses perhitungan, maka perlu dibuat Algoritma untuk mempermudah proses pembuatan program sederhana menggunakan bahasa pemrograman C# Visual Studio 2008 *(Suryo, 2007)*. Dengan bahasa pemrograman tersebut, akan dibuat fungsi-fungsi yang akan mewakili setiap tahapan perhitungan yang telah dijelaskan pada tiap langkah diatas. Setiap fungsi yang telah dibuat akan diuji kebenarannya dengan menggunakan data perhitungan secara manual.

Microsoft membuat C# seiring dengan pembuatan Framework .NET. Chief Architect dalam pembuatan C# adalah Anders Hejlsberg yang sebelumnya berperan dalam pembuatan Borland Delphi dan Turbo Pascal. C# menjanjikan produktifitas dan kemudahan yang ada di Visual Basic dengan kemampuan dan fleksibilitas yang ada di C/C++. C# menyederhanakan konsep-konsep sulit dari C dengan penambahan fitur-fitur.

C# berhasil distandarisasi oleh ECMA pada Desember 2001. Dengan standar tersebut siapa saja dapat membuat implementasi C#. Saat ini baru terdapat compiler C# buatan Microsoft dan compiler dari proyek Mono.

2.6.1 Perbandingan C# dengan Bahasa .NET Lain

C# adalah salah satu dari banyak bahasa yang bisa dipakai untuk pemrograman .NET. Kelebihan utama bahasa ini adalah sintaksnya yang mirip C, namun lebih mudah dan lebih bersih untuk membuat program .NET.

2.6.2 Ruang Lingkup Pemakaian C# dan Masa Depan Framework .NET

C# sebagai bahasa pemrograman untuk Framework .NET memiliki ruang lingkup penggunaan yang sangat luas. Pembuatan program dengan user interface *Windows* maupun *console* dapat dilakukan dengan C#. Karena *Framework* .NET memberikan fasilitas untuk berinteraksi dengan kode. C# juga dapat digunakan untuk pemrograman *web site* dan *web service.*

BAB III

PELAKSANAAN PENELITIAN

3.1 Materi Penelitian

Sumber data yang dipakai dalam penelitian ini adalah:

A. Data Foto.

Didalam data foto terdapat data intensitas cahaya, koordinat pixel, resolusi foto, dan panjang fokus yang akan dipakai dalam perhitungan pada masing-masing metode.

B. Literatur dan bacaan yang berguna dalam proses penelitian.

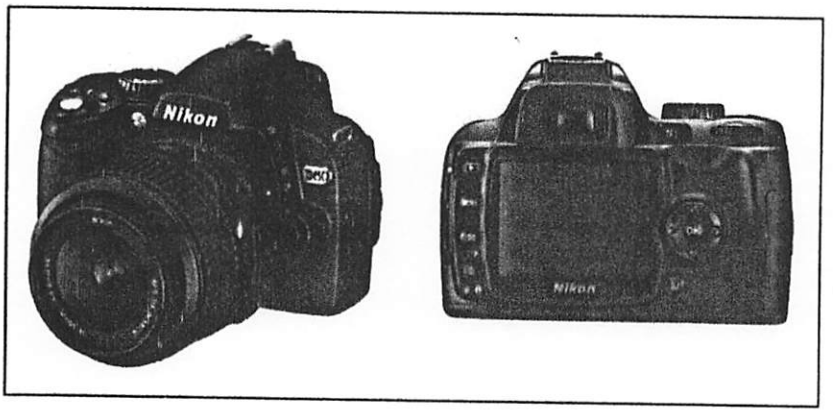
3.2 Peralatan Penelitian

Untuk mendukung pelaksanaan kegiatan penelitian ini, maka diperlukan beberapa perangkat keras (*hardware*) dan perangkat lunak (*software*) antara lain:

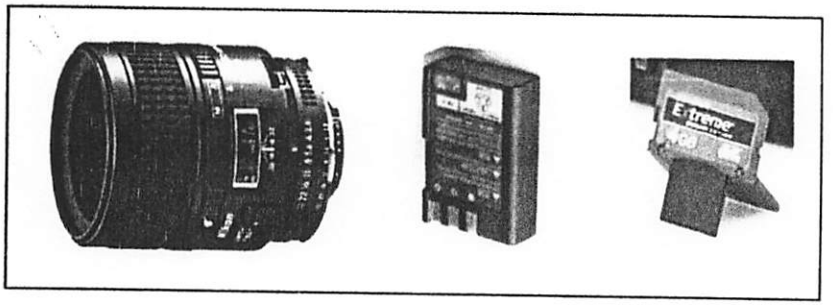
3.2.1 Perangkat keras (*hardware*), terdiri dari :

a. Kamera SLR Nikon D60 dengan spesifikasi kamera SLR Nikon D60 sebagai berikut :

- *Tipe Lensa* : *Single-lens reflex digital camera*
- *LCD screen* : *2.5-inch*
- *Image Sensor* : *23.6 x 15.8 mm CCD sensor*



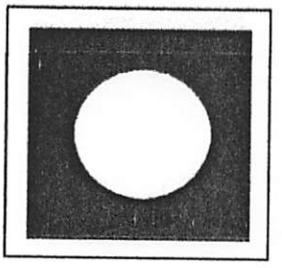
Gambar 3.1 Kamera SLR Nikon D 60 tampak depan



Gambar 3.2 Aksaesoris dari Kamera (Tipe Lensa, Baterai, Memori)

b. Striker (*Retro Reflective Target)*

Striker *retro reflective* berupa stiker berbentuk lingkaran berwarna hitam dengan background putih atau sebaliknya. Adapun contoh dari stiker *retro reflective target* adalah sebagai berikut :



Gambar 3.3 Striker Retro Target

Striker ini merupakan titik target yang dipasang pada objek yang digunakan untuk menentukan objek pemotretan. *Retro reflective target* sudah sering digunakan dalam aplikasi *survey* deformasi, dengan alasan karena *retro reflective target* mampu menghasilkan cahaya 2000 kali lebih terang dibandingkan dengan stiker biasa. sehingga mempermudah untuk menentukan koordinat nilai tengah dari target (Clarke dan Wang, 1999).

c. Satu unit komputer PC intel *duel core*

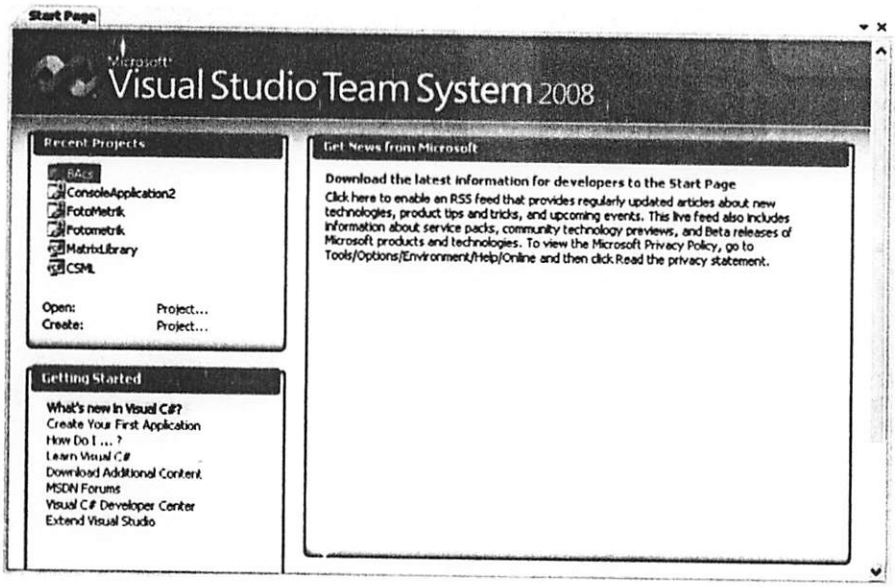
- *Procesor*
- *Hardisk*
- Memori
- *Mouse Optik + Keiborb*
- Monitor

3.2.2 Perangkat lunak (*software*), terdiri dari :

a. *Microsoft Word* 2003

b. *Microsoft Excel* 2003

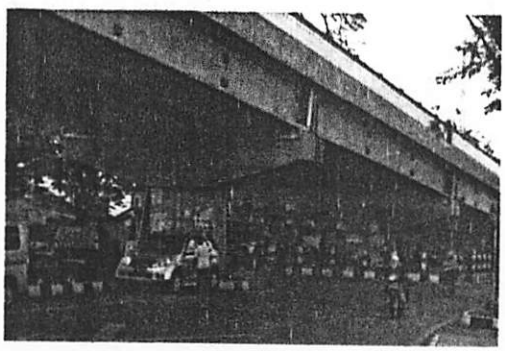
c. *Microsoft Visual Studio* 2008 (Bahasa Pemograman C#)



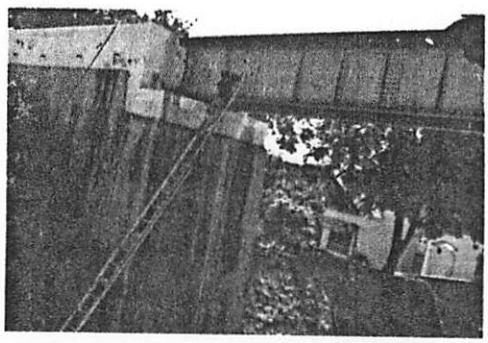
Gambar 3.4 Tampilan awal Program C#

3.3 Lokasi Penelitian

Penelitian dilakukan di dua tempat yaitu di Jembatan *flay-over* Arjosari Kota Malang dan Jembatan Rel Kereta Api Lawang, Kabupaten Malang. Objek yang diteliti yaitu tiang pancang jembatan *fly-over* yang ditempel stiker *retro-reflective target* sebanyak empat tiang dan sisi kanan dan kiri jembatan Rel Kereta Api di Lawang, Kabupaten Malang.

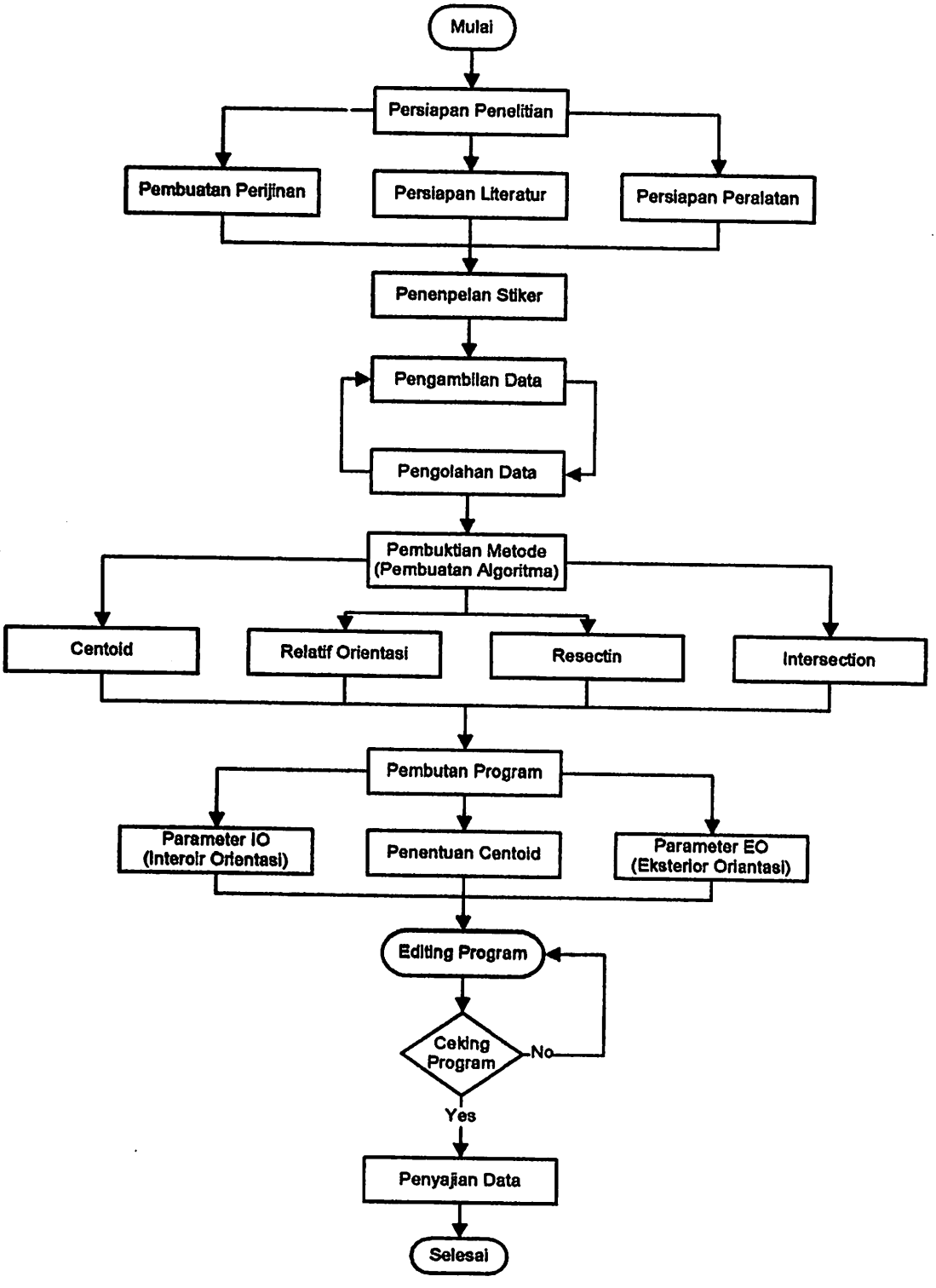


Gambar 3.5a Fly Over Arjosari Malang



Gambar 3.5b Jembatan Rel Kereta Api Lawang

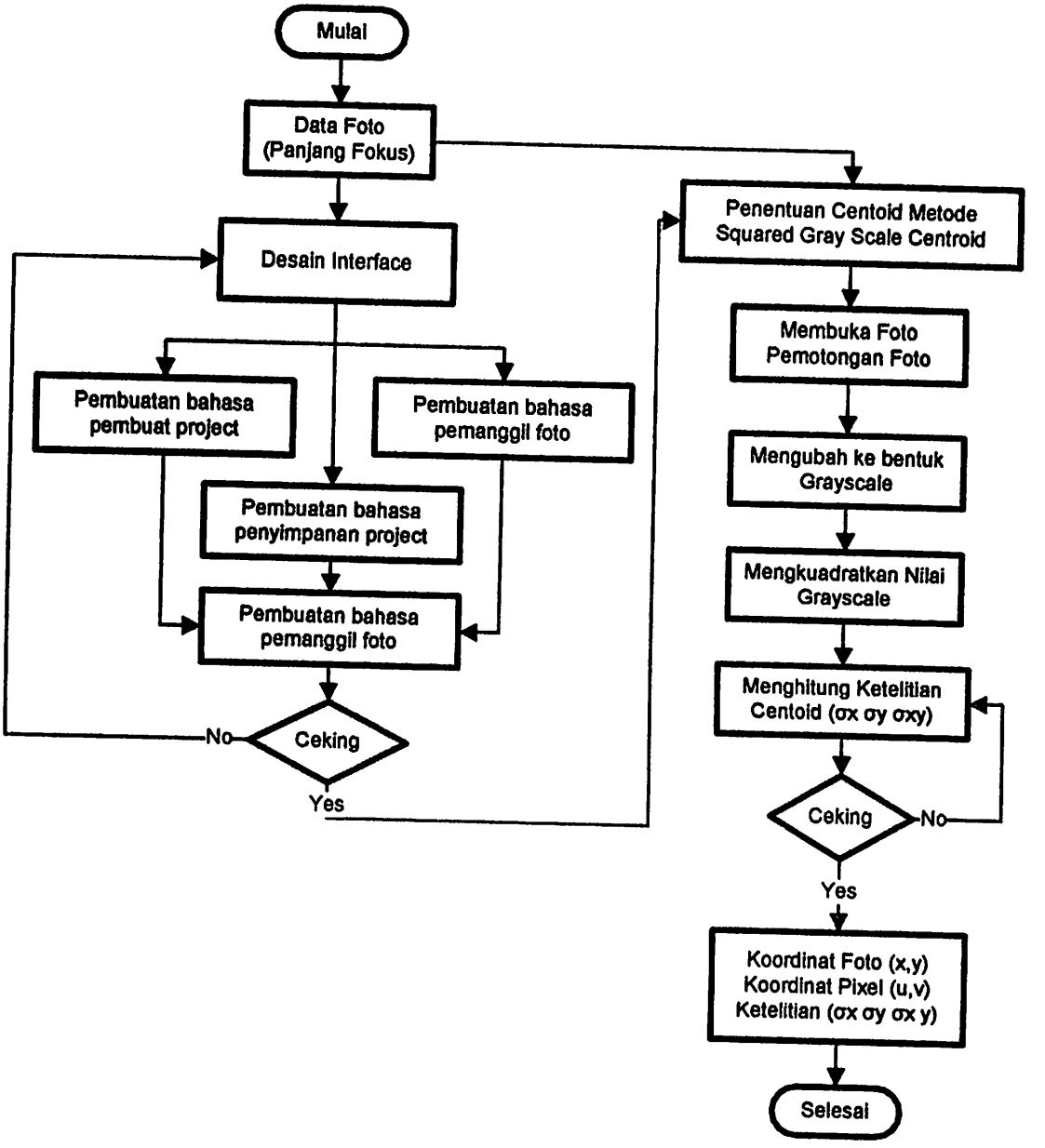
3.4 Dia gram Alir Penelitian



Penjelasan diagram alir penelitian

1. Pada tahap ini yang dilakukan adalah mempersiapkan segala sesuatu yang berkaitan dengan pelaksanaan penelitian yaitu pembuatan perizinan, berkaitan dengan perizinan lokasi penelitian, pengadaan kamera, pembuatan stiker, dan mencakup pemeriksaan kelayakan dan kesiapan alat dan bahan yang akan digunakan dalam proses pengambilan data.
2. Penempelan stiker dilakukan di dua lokasi penelitian yaitu jembatan rel kereta Lawang dan *Fly Over* Arjosari
3. Pengambilan data dengan melakukan pemotretan pada lokasi atau objek yang telah ditempel *retro reflektif target* dengan besar *overleping* minimal 60 % dan besar sudut pemotretan ideal adalah 120^0 (disesuaikan kondisi lapangan) dilakukan selama 6 bulan.
4. Pengolahan data dilakukan di laboratorium dengan menggunakan *software* Photomodeler untuk tiap sesi pemotretan.
5. Pebuktian metode dilakukan di laboratoriun sampai mendapat metode perhitungan yang paling stabil untuk digunakan dalam pembuatan algoritma.
6. Pembuatan algorima secara terperinci sehingga mempermudah pembuatan program untuk tiap topik pembahasan (*Centroid*, Relatif Orientasi, *Resection*, *Intersection*).
7. Pembuatan program dari *interface* dan mengkonfersi algoritma kedalam bentuk bahasa pemrograman C# Visual Studio 2008 untuk setiap topik penelitian.

3.5 Diagram Alir Pembuatan *Interface* dan *Centroid*



Penjelasan Diagram Alir Pembuatan *Interface* Dan Penentuan *Centroid*

1. Data awal yang disiapkan yaitu data foto dalam format Jpg, dan data panjang fokus yang didapat dari kamera.

2. Desain Interface

a. Mendesain tampilan utama program.

b. Membuat tols-tols sesuai kebutuhan

c. Mendesain Form Untuk Relatif orientasi, *resection, intersection.*

3. Penentuan Cenroid dilakukan dengan memanfaatkan fungsi-fungsi dalam software Visual Studio 2008.

a. Membuka data foto yang akan diproses.

b. Membuat Template berukuran 15x15 dengan cara memotong gambar sehingga mencakup satu retro-reflective target saja dan mengubah gambar template menjadi bentuk grayscale.

c. Menentukan nilai centroid dengan memasukkan persamaan pada metode grayscale centroid, pada persamaan ini terdapat beberapa parameter yang harus ditentukan yaitu:

1. Koordinat pixel x dan y
2. Seluruh intensitas cahaya pada *templated* dikuadratkan, sebelum dijumlahkan.

a. Menentukan sigma x (σx) dan sigma y (σy) sigma xy (σxy).

$$\sigma_x^2 = \frac{\sigma^2}{\Sigma li^2} \cdot \left(\sum xi - 2x \sum xi + mx^2 \right)$$

$$\sigma_y^2 = \frac{\sigma^2}{\Sigma li^2} \cdot \left(\sum yi - 2x \sum yi + my^2 \right)$$

dimana, $\sigma^2 = 1/12$ dan m adalah jumlah baris dan kolom dari matriks kovarian koordinat.

b. Mengkonversi koordinat pixel ke koordinat foto.

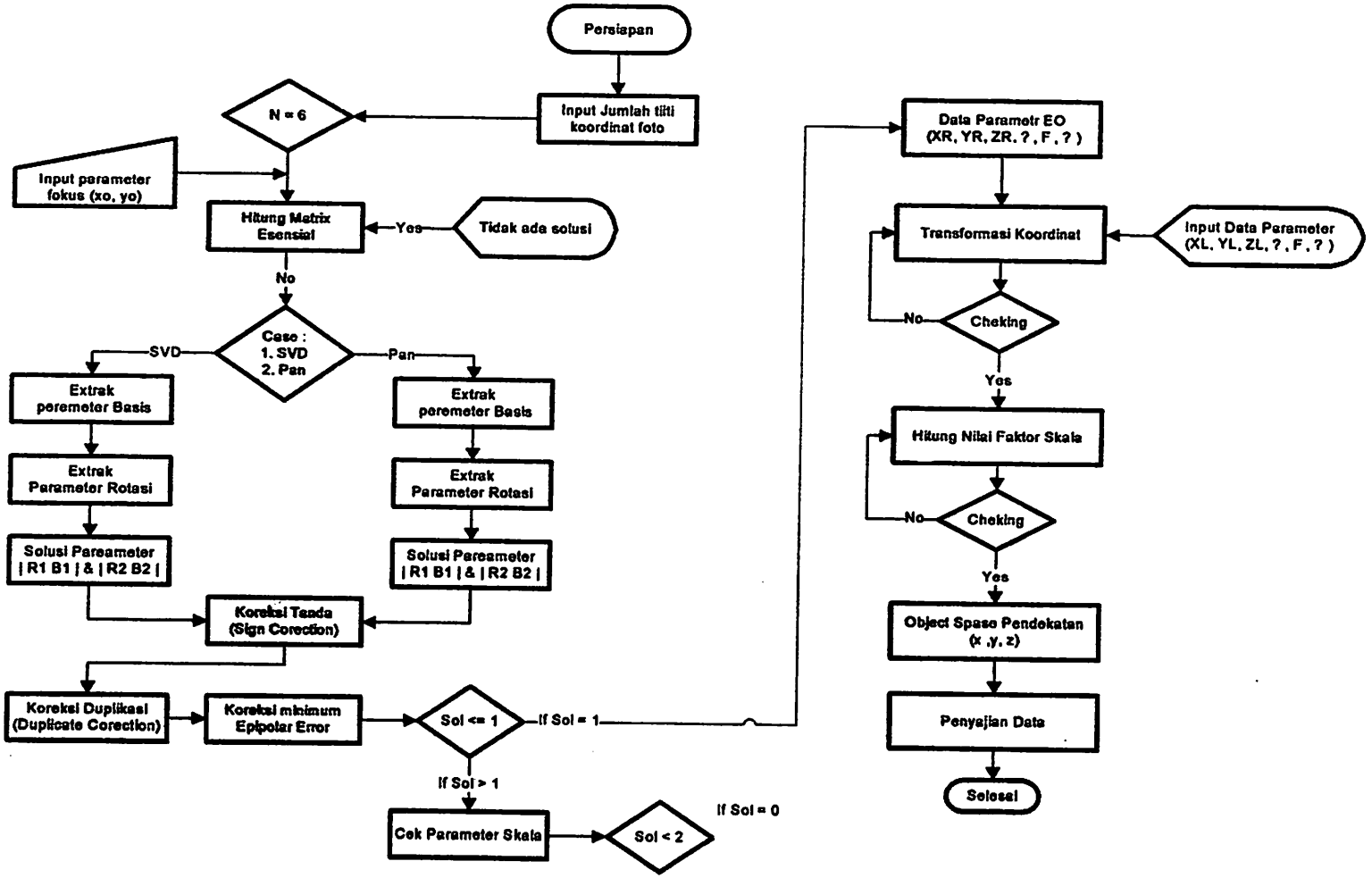
$$x=(x'-x'c)*xPixelSize$$
$$y=(y'c-y')*yPixelSize$$

dimana,

$$xc'\left[\frac{nx'}{2}\right]-0.5 \qquad yc'\left[\frac{ny'}{2}\right]-0.5$$

c. *Out put* dari hasil proses di simpan di *Notepad.*

3.6 Diagram Alir Relatof Orientasi dan Interseksi



Penjelasan Diagram Alir Perhitungan Relatif Orientssi

1. *Inputan* Data

Data-data yang digunakan dalam penelitian ini berupa pasangan foto *stereo* dalam berbagai konfigurasi pemotretan. Dari pasangan foto *stereo* tersebut dilakukan ekstraksi data koordinat obyek yang sama pada tiap foto. Sehingga, didapat data koordinat foto yang saling berkorespondensi (X_{n1}, Y_{n1}) (X_{n2}, Y_{n2}). Minimum data yang dibutuhkan untuk penelitian ini adalah enam titik korespondensi untuk tiap foto. Jika jumlah titik korespondesi kurang dari enam, proses perhitungan nilai orientasi awal dan koordinat obyek tidak dapat dilakukan. Selain data titik, data informasi mengenai geometri *interinsik* kamera yang berupa panjang fokus dan dua buah parameter *principle point* (X_0, Y_0) harus diketahui.

2. Penentuan nilai Parameter Essential Matrik

Dengan menggunakan nilai parameter koordinat foto dan parameter interinsik kamera dapat dilakukan proses perhitungan nilai matrik essensial. Adapun proses perhitungan matrik essensial akan dijabarkan dalam langkah-langkah sebagai berikut:

a. Menysusun parameter matri koefisien *A* seperti yang telah dijelaskan pada *persamaan (2.64)* yang telah dibahas pada *bab-dua*. Untuk

memudahkan maka akan ditulis kembali sebuah susunan dari matrik A sebagai berikut:

$$A=\begin{pmatrix} x_1x_2 & y_1x_2 & f_1x_2 & x_1y_2 & y_1y_2 & f_1y_2 & x_1f_2 & y_1f_2 & f_1f_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1n}x_{2n} & y_{1n}x_{2n} & f_{1n}x_{2n} & x_{1n}y_{2n} & y_{1n}y_{2n} & f_{1n}y_{2n} & x_{1n}f_{2n} & y_{1n}f_{2n} & f_{1n}f_{2n} \end{pmatrix} \qquad (3.1)$$

b. Sesuai dengan persamaan (2.63), hubungan antara koefisien A dan matrik essensial dituliskan dalam bentuk $AE = 0$. Sehingga, untuk mendapatkan nilai matrik essensial, dilakukan proses dekomposi matrik koefisien A menggunakan metode *singular value decompotion* (SVD) sebagai berikut:

$$[U,S,V]=svd(A) \qquad (3.2)$$

Jika jumlah $n\_point \geq 8$, maka solusi e didapat menggunakan persamaan:

$$E=\sqrt{2}\times V(:,end) \qquad (3.3)$$

Dimana $V(:,end)$ merupakan nilai dari kolom terakhir matrik V yang dihasilkan dari proses dekomposi matrik A. Akan tetapi, jika $n\_point \geq 6$ dan $n\_point < 8$ solusi dari nilai e didapat sebagai berikut:

$$E=\sqrt{2}\times V(:,7:9) \qquad (3.4)$$

Dengan nilai $V(:,7:9)$ merupakan nilai dari matrik V pada kolom ke-7 sampai dengan ke-9 hasil dekomposi matrik A. Karena jumlah solusi matrik essensial menggunakan enam titik lebih dari satu, maka perlu dilakukan sebuah konstrain terhadap nilai tiga-dimensi minimum sebagai berikut:

$$E = xE_x + yE_y + zE_z \tag{3.5}$$

3. Menetukan Nilai Enam Parameter *Ekstrinsik* Kamera

Dalam proses penentuan nilai parameter *ekstrinsik* kamera menggunakan matrik essensial terdapat dua cara, yaitu menggunakan metode pan dan metode dekomposisi matrik *singular* seperti yang telah dijelaskan pada bab-dua.

a. Metode dekomposisi matrik singular (SVD)

Untuk mendapatkan nilai parameter *ekstrinsik* kamera menggunakan metode dekomposisi matrik singular, terdapat dua tahapan, pertama penentuan nilai basis kamera dan keda penentuan nilai parameter rotasi. Adapun proses perhitungan kedua nilai tersebut akan dijelaskan dalam *poin-poin* dibawah ini.

- **Proses penentuan nilai basis kamera**

  Nilai basis kamera ini didapat dari nilai dekomposi matrik essensial kedalam tiga buah matrik *singular*.

$$[U,S,V] = svd(E)$$

Dengan nilai parameter basis.

$$B_1 = \begin{vmatrix} v_{13} \\ v_{23} \\ v_{33} \end{vmatrix} \quad dan \quad B_2 = -\begin{vmatrix} v_{13} \\ v_{23} \\ v_{33} \end{vmatrix} \tag{3.6}$$

Dimana vektor v merupakan nilai dari kolom terakhir dari matrik V hasil dekomposisi matrik essensial menggunakan SVD.

➢ **Proses penentuan nilai rotasi kamera**

Dengan menggunakan nilai tiga matrik singular hasil dekomposisi matrik essensial diatas, didapat nilai rotasi sebagai berikut.

$$R_1 = UWV^T \text{ dan } R_2 = UW^TV^T \tag{3.7}$$

Dimana nilai U dan V merupakan nilai matrik singular hasil dekomposisi matrik essensial dan W merupakan matrik yang bernilai:

$$W = \begin{vmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix} \tag{3.8}$$

Hasil dekomposisi diatas belum dapat digunakan. Untuk dapat digunakan, matrik tersebut harus terlebih dahulu dilakukan konstrain terhadap nilai determinannya sebagai berikut:

Jika nilai $\det(R_1) < 1$ maka nilai $R_1 = -R_1$.

Jika nilai $\det(R_2) < 1$ maka nilai $R_2 = -R_2$.

➢ **Solusi untuk nilai basis dan rotasi kamera**

Dari proses penentuan nilai parameter *eksterinsik* kamera menggunakan metode dekomposisi matrik *singular* ini didapat dua pasang solusi parameter *ekstrinsik* sebagai berikut:

Solusi pertama : $P = |R_1 \| B_1|$ (3.9)

Solusi kedua : $P = |R_2 \| B_2|$ (3.10)

b. **Metode *Pan***

Tujuan dari metode ini sama dengan metode dekomposisi matrik *singular* diatas. Yang membedakan proses ini ialah dari segi proses perhitungannya. Proses perhitungan nilai basis dan rotasi kamera menggunakan metode ini akan diuraikan pada langkah-langkah sebagai berikut:

➢ **Proses Penentuan Nilai Basis Kamera**

Untuk menghitung nilai basis kamera menggunakan metode ini, pertama harus ditentukan nilai dari *vector* b_{XY}, b_{XZ}, b_{YZ}, menggunakan persamaan:

$$b_{XY} = -(e_{11}e_{21} + e_{12}e_{22} + e_{13}e_{23}) \quad (3.11)$$

$$b_{XZ} = -(e_{11}e_{31} + e_{12}e_{32} + e_{13}e_{33}) \quad (3.12)$$

$$b_{YZ} = -(e_{21}e_{31} + e_{22}e_{32} + e_{23}e_{33}) \quad (3.13)$$

dapat dituliskan kembali sebagai berikut :

$$b_{XY} = -(e_{11}e_{21} + e_{12}e_{22} + e_{13}e_{23}) \quad (3.14)$$

$$b_{XZ} = -(e_{11}e_{31} + e_{12}e_{32} + e_{13}e_{33}) \quad (3.15)$$

$$b_{YZ} = -(e_{21}e_{31} + e_{22}e_{32} + e_{23}e_{33}) \quad (3.16)$$

Sehingga, menggunakan ketiga nilai *vector* diatas dapat dihitung nilai elemen dari tiap *vector* basis sebagai berikut.

$$b_X = \sqrt{\frac{b_{XY}b_{XZ}}{b_{YZ}}}; \quad b_Y = \sqrt{\frac{b_{XY}b_{YZ}}{b_{XZ}}}; \quad b_Z = \sqrt{\frac{b_{XZ}b_{YZ}}{b_{XY}}} \quad (3.17)$$

Selanjutnya, menentukan tanda dari tiap nilai dari elemen *vector* basis b_X, b_Y, b_Z sebagai berikut:

Jika nilai $b_{XY} > 0 \,\&\, b_{XZ} < 0 \,\&\, b_{YZ} < 0$, maka nilai b_X, b_Y, b_Z ialah:

$$B = \begin{vmatrix} b_X \\ b_Y \\ -b_Z \end{vmatrix} \qquad (3.18)$$

Jika nilai $b_{XY} < 0 \,\&\, b_{XZ} > 0 \,\&\, b_{YZ} < 0$, maka nilai b_X, b_Y, b_Z ialah:

$$B = \begin{vmatrix} b_X \\ -b_Y \\ b_Z \end{vmatrix} \qquad (3.19)$$

Jika nilai $b_{XY} < 0 \,\&\, b_{XZ} < 0 \,\&\, b_{YZ} > 0$, maka nilai b_X, b_Y, b_Z ialah:

$$B = \begin{vmatrix} b_X \\ -b_Y \\ -b_Z \end{vmatrix} \qquad (3.20)$$

Dan yang terakhir, apabila kesceluruhan nilai dari *vector* b_{XY}, b_{XZ}, b_{YZ} positif maka *vector* b_X, b_Y, b_Z juga akan bernilai positif. Dari proses penentuan basis inipun terdapat dua solusi untuk basis yaitu $B_1 = B$ dan $B_2 = -B$.

- **Proses Penentuan Nilai Rotasi Kamera**

Proses penentuan nilai rotasi kamera dihitung untuk tiap nilai parameter basis yang telah didapat sebelumnya. Adapun proses perhitungannya secara ringkas akan akan diberikan sebagai berikut:

Jika nilai $b_X > b_Y \,\&\, b_Z$, maka nilai $r_{11}, r_{12}, r_{21}, r_{22}, r_{31}$ dan r_{32} sebagai berikut:

$$\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ r_{31} & r_{32} \end{pmatrix} = \begin{pmatrix} b_Z & 0 & b_Z \\ -b_Y & b_X & 0 \\ b_X^2 & b_X b_Y & b_X b_Z \end{pmatrix}^{-1} \begin{pmatrix} e_{21} & e_{22} \\ e_{31} & e_{32} \\ \det\begin{pmatrix} e_{22} & e_{23} \\ e_{32} & e_{33} \end{pmatrix} & -\det\begin{pmatrix} e_{21} & e_{23} \\ e_{31} & e_{33} \end{pmatrix} \end{pmatrix} \quad (3.21)$$

Jika nilai $b_Y > b_X \,\&\, b_Z$, maka nilai $r_{11}, r_{12}, r_{21}, r_{22}, r_{31}$ dan r_{32} sebagai berikut :

$$\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ r_{31} & r_{32} \end{pmatrix} = \begin{pmatrix} 0 & -b_Z & b_Y \\ -b_Y & b_X & 0 \\ b_X b_Y & b_Y^2 & b_Y b_Z \end{pmatrix}^{-1} \begin{pmatrix} e_{11} & e_{12} \\ e_{31} & e_{32} \\ -\det\begin{pmatrix} e_{12} & e_{13} \\ e_{32} & e_{33} \end{pmatrix} & \det\begin{pmatrix} e_{11} & e_{13} \\ e_{31} & e_{33} \end{pmatrix} \end{pmatrix} \quad (3.22)$$

Jika nilai $b_Z > b_X \,\&\, b_Y$, maka nilai $r_{11}, r_{12}, r_{21}, r_{22}, r_{31}$ dan r_{32} sebagai berikut :

$$\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ r_{31} & r_{32} \end{pmatrix} = \begin{pmatrix} 0 & -b_Z & b_Y \\ b_Z & & -b_X \\ b_X b_Z & b_Y b_Z & b_Z^2 \end{pmatrix}^{-1} \begin{pmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \\ \det\begin{pmatrix} e_{12} & e_{13} \\ e_{22} & e_{23} \end{pmatrix} & -\det\begin{pmatrix} e_{11} & e_{13} \\ e_{21} & e_{23} \end{pmatrix} \end{pmatrix} \quad (3.23)$$

Dan untuk nilai r_{13}, r_{23}, r_{33} dapat ditentukan dengan menggunakan :

$$r_{13} = \det\begin{pmatrix} r_{21} & r_{22} \\ r_{31} & r_{32} \end{pmatrix}; \quad r_{13} = \det\begin{pmatrix} r_{21} & r_{22} \\ r_{31} & r_{32} \end{pmatrix}; \quad r_{13} = \det\begin{pmatrix} r_{21} & r_{22} \\ r_{31} & r_{32} \end{pmatrix} \quad (3.24)$$

Dari proses diatas, apabila nilai inputan basis ialah B_I maka akan menghasilkan rotasi R_I dan sebaliknya.

➢ **Solusi untuk nilai basis dan rotasi kamera**

Jumlah solusi yang akan didapat untuk parameter basis dan rotasi kamera adalah sebagai berikut :

Solusi pertama : $P = |R_1 \| B_1|$ (*3.25*)

Solusi kedua : $P = |R_2 \| B_2|$ (3.26)

4. Koreksi Tanda (+/-) Untuk Tiap Pasang Solusi

Pada proses ini akan dilakukan koreksi tanda dari dua kemungkinan solusi posisi kamera. Proses ini dilakukan dengan mempertimbangkan bahwa tiap titik konjugasi $(x_1, y_1) \rightarrow (x_2, y_2)$ memiliki nilai kedalaman $z_1, z_2 > 0$. Jika diasumsikan pada foto stereo, kamera *1* memiliki nilai parameter *ekstrinsik* $P = |I \| 0|$ dan kamera *2* memiliki nilai $P = |R \| - B|$ maka didapat hubungan $z_1 x_2 = R(z_1 x_1 - B)$. Sebagai contoh, untuk menentukan nilai dari persamaan $x_1 = aB + b(R^T x_2)$ seharusnya didapat nilai koefisien $a, b > 0$. Sehingga, dapat disimpulkan bahwa dalam proses ini akan dilakukan perubahan nilai tanda dari parameter B dan menghapus seluruh kemungkinan solusi posisi kamera yang salah.

5. **Koreksi Nilai Duplikasi**

Data awal dari proses koreksi nilai duplikasi ini merupakan nilai parameter basis dan rotasi kamera hasil koreksi tanda yang telah dilakukan diatas. Untuk koreksi nilai parameter basis, apabila selisih dari dua solusi memiliki nilai kurang dari batasan $(B_1 - B_2 < threshold)$, maka kedua parameter akan dikatakan sama dan salah satunya akan dihilangkan. Sedangkan untuk parameter rotasi kamera, apabila memenuhi syarat $R_1^T R_2 - I < threshold$, maka salah satu parameter rotasi akan dihilangkan. Nilai *threshold* untuk solusi basis dan rotasi pada proses koreksi duplikasi ini akan diberikan sebesar $1e^{-4}$.

6. **Koreksi Nilai Kesalahan Proyeksi Epipolar Terkecil**

Tahapan ini dilakukan untuk mendapatkan nilai parameter basis dan rotasi kamera yang memiliki nilai kesalahan terkecil. Kesalahan ini didefinisikan dalam sebuah persamaan sebagai berikut (*Triggs, 2000*):

$$Err = \sqrt{\sum \frac{(f_x f_y)^2}{((f_x f_x^T)(x_{2i} x_{2i}^T)) + ((x_{1i} x_{1i}^T)(f_y^T f_y))}} \qquad (3.27)$$

Dimana i = 1,2,....,n-titik, x_{1i} merupakan koordinat foto-*1* $(x_{1i}\ y_{1i}\ -f)$, x_{2i} merupakan koordinat foto-2 $(x_{2i}\ y_{2i}\ -f)$ dan f_x, f_y secara urut merupakan nilai dari $x_{1i}E$, Ex_{2i}^T.

7. Reduksi Nilai Kesalahan Skala

Dalam hal ini, proses reduksi nilai kesalahan skala dilakukan jika nilai solusi parameter basis dan rotasi kamera masih berjumlah lebih dari satu pada proses koreksi kesalahan proyeksi epipolar terkecil. Adapun nilai parameter basis dan rotasi kamera yang dianggap benar, apabila memenuhi syarat $\lambda > 0$. Nilai λ dapat dihitung menggunakan persamaan (3.11) atau (3.12).

8. Perhitungan Interseksi

Tahap-tahap yang dilakukan dalam proses penentuan koordinat *object space* dengan menggunakan metode intersection adalah:

a. Menggunakan data dari proses relatif orientas untuk foto kiri yang dianggap posisi kamera dan omega, phi, kappa-nya nol $(x = 0, y = 0, z = 0, \omega = 0, \sigma = 0, \kappa = 0)$, untuk foto kanan posisi kamera dan Omega, Phi, Kappa-nya adalah hasil dari perhitungan Relatif Orientasi.
b. Melakukan perhitungan perhitungan transformasi koordinat untuk tiap titik, dalam hal ini koordinat foto kiri dan kanan (x, y, z) dengan matriks rotasi masing-masing foto.

c. Menghitung nilai faktor skala untuk tiap titik interseksi.

d. Melakukan perhitungan interseksi dengan mengiterasi sampai mendapatkan ketelitian yang paling baik (iterasi terakir) merupakan object *spase poin* pendekatan dari 2 foto

e. Penyajian data *object spase poin* pendekatan dalam bentuk NotePad.

f. Selesai.

BAB IV

HASIL DAN PEMBAHASAN

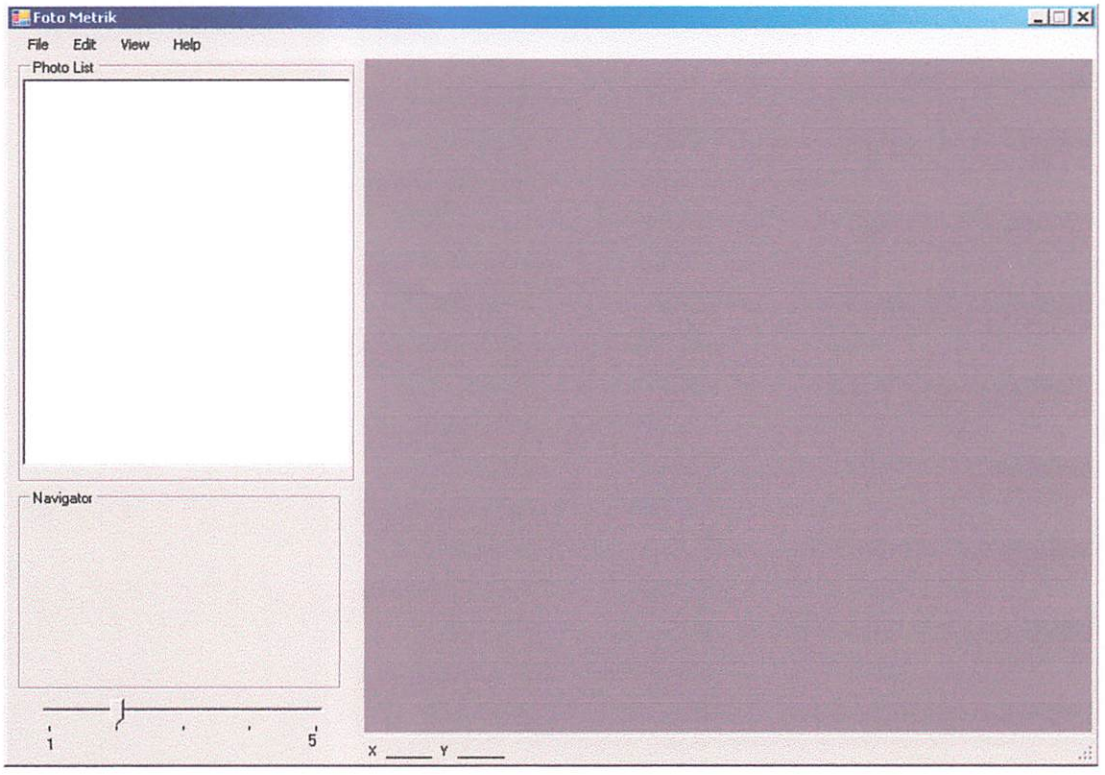
4.1. Hasil

Untuk mengetahui tingkat keberhasilan dari penelitian, perlu dilakukan proses analisa atau pembahasan dari sebuah hasil yang telah dicapai selama pelaksanaan penelitian. Parameter keberhasilan dapat diukur dengan membandingkan tujuan dari penelitian dan hasil yang dicapai. Adapun beberapa parameter yang telah dihasilkan selama pelaksanaan penelitian akan disajikan dibawah ini.

Hasi dari penelitian ini adalah sebuah program perhitungan *Object Spase Poin* pendekatan dengan metode *Clos Range Photogrametry.*

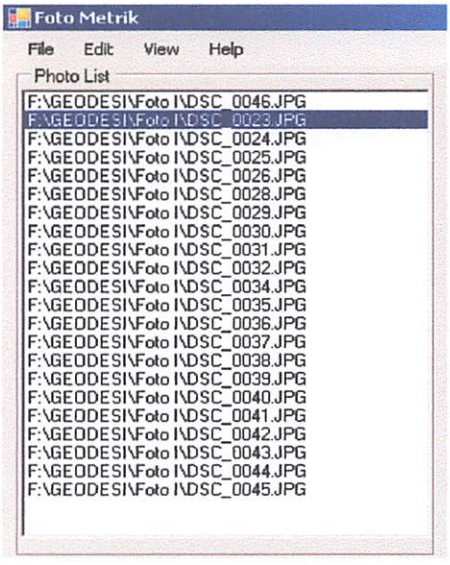
4.1.1. Desain Interface

A. Tampilan Utama dari program terdiri dari beberapa bagian penting yaitu *Photo List, Navigator, Image Box Menu File, Menu Edit, Menu View,* dan *Menu Help* seperti tampak pada Gambar 4.1 dibawah ini:



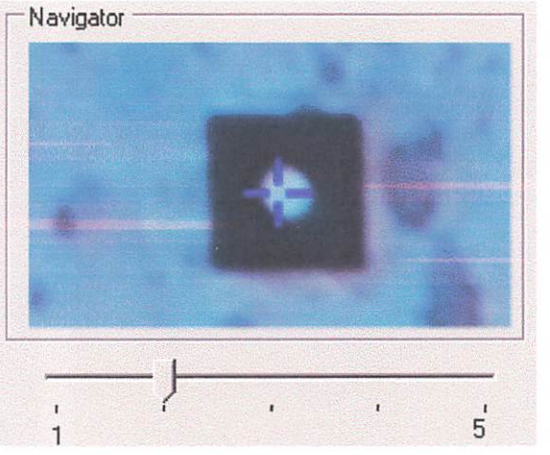
Gambar 4.1 Tampilan Utama Program

B. Box Photo list adalah tempat lis nama foto yang di masukan dalam *project.*



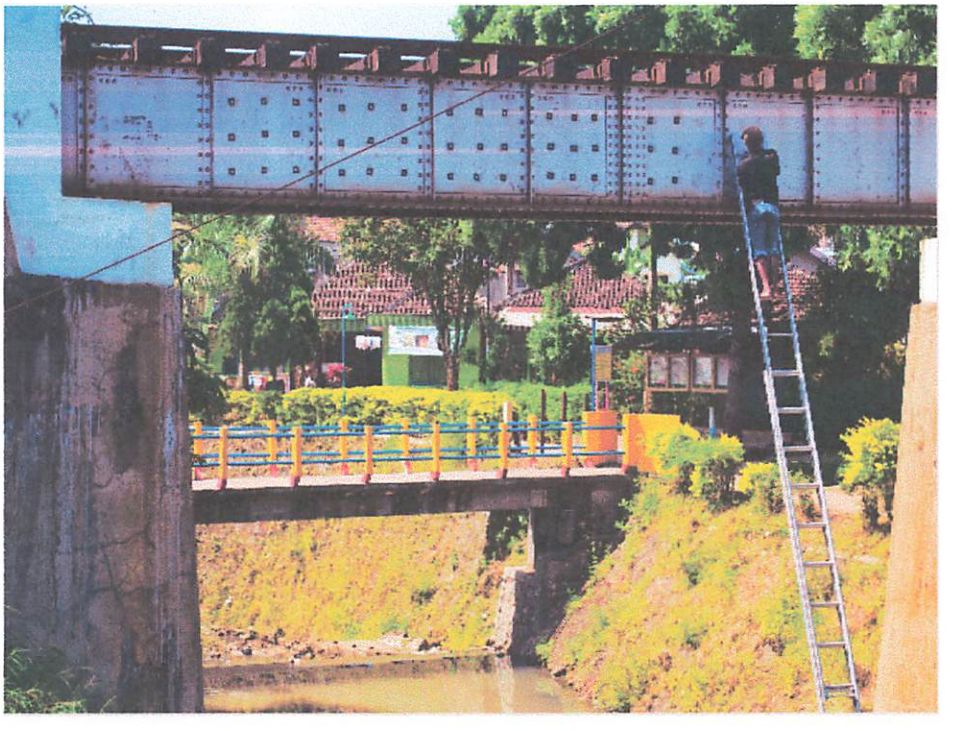
Gambar 4.2 Box Photo List

C. *Navigator* merupakan form perbesaran dari *object* yang di tunjuk pada *Image box*.



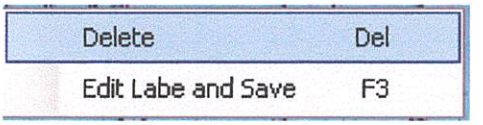
Gambar 4.3 Box Navigator

D. *Image Box* berfungsi untuk menampilkan foto



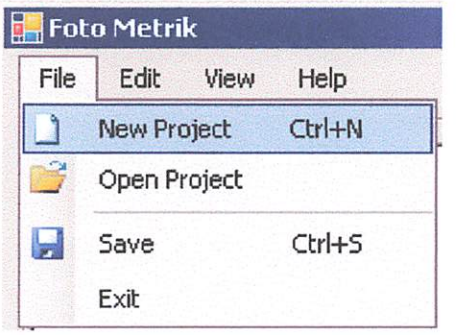
Gambar 4.4 Foto Pada Image Box

E. *Menu* Pilihan untuk menghapus dan mengganti nama titik untuk di simpan di list data hasil proses



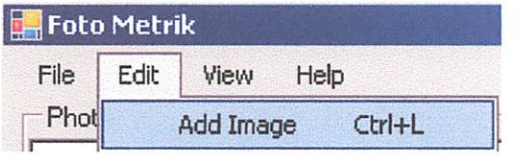
Gambar 4.5 Foto pada Image Box

F. *Menu File* terdiri dari 4 sub menu.



Gambar 4.6 Menu File

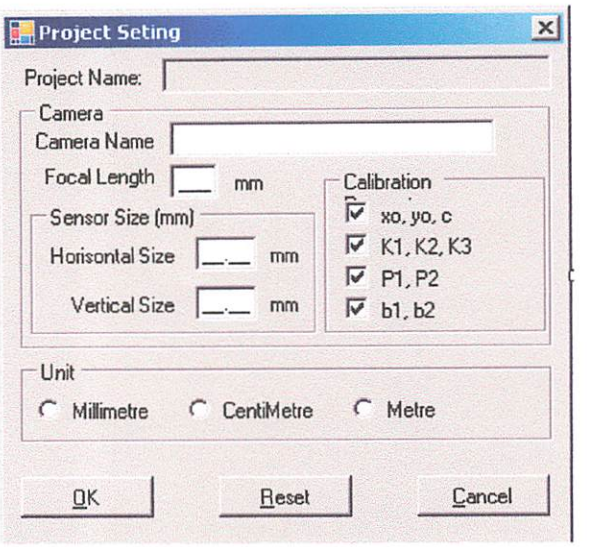
G. *Menu Edit* terdiri dari 1 sub nenu yaitu *Add Image* yang berfungsi untuk menaambil foto.



Gambar 4.7 Menu Edit

H. Menu View Terdiri dar 4 sub menu yaitu:

- *Sub Menu Project Seting* berfungsi untuk mengarut parameter dari kamera.

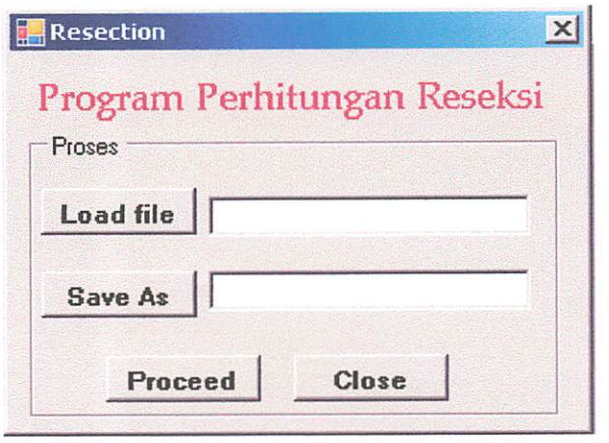


Gambar 4.9 Menu View

- *Sub Menu Relatif Orintasi* merupakan sub menu untuk menjalankan proses hitung *Relatif Orientasi* dan *Intersection.*

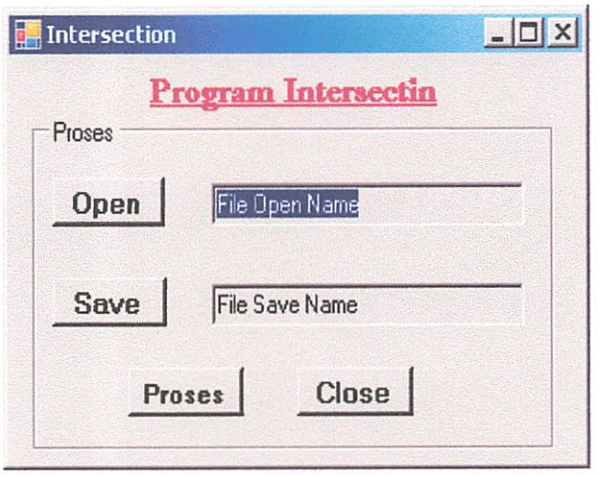
Gambar 4.10 Form perhitungan Relatif Orientasi

- *Sub Menu Resection* untuk mejalamkan proses hitung *Resection.*



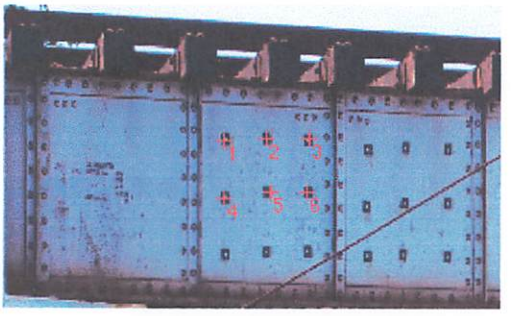
Gambar 4.11 Form perhitungan Resection

- *Sub Menu Intersection* untuk mejalamkan proses hitung *Intersection.*



Gambar 4.12 Form perhitungan Resection

4.1.2. Proses Penentuan *Centroid*



Gambar 4.13 Form Penentuan Centroid

Hasil dari proses *cenroid* dengan program

1	-5.0219	4.786422	931.1204	338.2156
2	-4.2733	4.713898	1080.84	352.7205
3	-3.64908	4.70295	1205.684	354.91
4	-2.68348	4.612813	1398.805	372.9374
5	-2.06891	4.634037	1521.717	368.6927
6	-1.38401	4.61545	1658.699	372.41
7	-0.45651	4.536612	1844.199	388.1776
8	0.183534	4.528469	1972.207	389.8063
9	0.706145	4.521936	2076.729	391.1129
10	1.624718	4.444766	2260.444	406.5469
11	2.142539	4.430651	2364.008	409.3698
12	2.554262	4.42283	2446.352	410.934
13	3.754594	4.382369	2686.419	419.0261
14	4.267812	4.377684	2789.062	419.9631

Dari hasil penentuan *Centroid* dengan Program didapat nomor titik (kolom 1), koordinat foto x, y (kolom 2 dan 3) dan koordinat pixel u, v (kolom 4 dan 5). Hasil dari proses *centoid* di simpan dalam Notepad dengan extensi .dfm.

4.1.3. Perhitungan Relatif Orientasi

Dari proses pelaksanaan penelitian yang telah dilakukan baik dilaboratorium maupun dilapangan dan dengan menggunakan berbagai jenis data yang telah dikumpulkan. Didapat beberapa parameter berupa parameter matrik essensial, parameter *ekstrinsik* kamera pendekatan yang terdiri dari enam parameter $(X_L, Y_L, Z_L, \omega, \varphi, \kappa)$ dan tiga parameter posisi obyek pendekatan dalam ruang tiga dimensi (X, Y, Z). Selain tiga parameter diatas dihasilkan pula suatu fungsi matematika yang ditulis dalam sebuah bahasa pengrograman C # Visual Studio 2008 untuk dapat dipergunakan secara fleksibel dengan berbagai kondisi data.

Final Projective Matrix

0.9655482615409970	0.0772646698482488	-0.2438318975765590	-0.2351093493525030
0.0728692900524894	-0.9976775488998430	-0.0238680306535022	0.2634920317658960
-0.2442740905873130	0.0049217845387694	-0.9695124343381950	-0.9355723077576220

Tiga kolom pertama adalah matrix rotasi foto kanan (Foto dua) sedangkan kolom ke-empat merupakan koordinat posisi kamera dua.

4.1.4. Perhitungan Interseksi

Dalam proses perhitungan dengan menggunakan metode *intersection* ada sembilan parameter yang tidak diketahui antara lain: ω, σ, κ (omega, phi, kappa) ini merupakan tiga parameter rotasi dimana parameter ini berhubungan dengan *m*, tiga parameter posisi kamera (X_L, Y_L, Z_L) dimana keenam parameter ini terdapat dalam *Eksterior Orientation* (orientasi luar). Sedangkan tiga

parameter lainnya merupakan koordinat titik objek (X_A, Y_A, Z_A). Pada proses intersection parameter ω, σ, κ (omega, phi, kappa), parameter posisi kamera (X_L, Y_L, Z_L), dan koordinat titik objek (X_A, Y_A, Z_A) pendekatan dicari dengan menggunakan metode *resection*. Berikut ini adalah nilai-nilai dari parameter-parameter ω, σ, κ (omega, phi, kappa), parameter posisi kamera (X_L, Y_L, Z_L), dan koordinat titik objek (X_A, Y_A, Z_A) pendekatan yang di dapatkan dari pengolahan menggunakan Program yang dibuat antara lain:

No Titik	X	Y	Z
1	-0.11975	0.081051	-0.38848
2	-0.10003	0.096761	-0.41335
3	-0.0822	0.114192	-0.4448
4	-0.05932	0.130115	-0.46728
5	-0.03559	0.147869	-0.49386
6	-0.00924	0.165954	-0.51894
7	0.013694	0.14161	-0.42739
8	0.029398	0.108346	-0.3264
9	0.060033	0.108165	-0.20636
10	0.188784	0.411021	-4.43115

4.2. Pembahsan

Penggunaan aplikasi diatas sangatlah sederhana, hanya dengan menggunakan parameter interior kamera berupa panjang fokus (f), dua parameter principle point (x_o, y_o), dan nilai koordinat obyek pada tiap foto $(x_1, y_1, f_1, x_2, y_2, f_2)$ dapat dilakukan perhitungan untuk nilai parameter ekstrinsik dari kedua kamera dan koordinat 3-dimensi obyek.

4.2.1. Essential Matrik

Seperti yang telah dijelaskan pada Bab 2, matrik essensial merupakan matrik spesial dari persamaan koplanar. Dengan menggunakan data foto dan dilakukan proses perhitungan nilai matrik essensial dengan menggunakan fungsi essential maka didapat nilai sebagai berikut:

Matriks esensial

$$E = \begin{pmatrix} -0.7003 & 0.0494 & 0.7121 \\ -0.4397 & -0.8157 & -0.3759 \\ 0.5623 & -0.5764 & 0.5930 \end{pmatrix}$$

Matriks U

$$U = \begin{pmatrix} -0.7003 & 0.0494 & 0.7121 \\ -0.4397 & -0.8157 & -0.3759 \\ 0.5623 & -0.5764 & 0.5930 \end{pmatrix}$$

Matriks S

$$S = \begin{pmatrix} 1.0050 & 0 & 0 \\ 0 & 0.9949 & 0 \\ 0 & 0 & 0.0017 \end{pmatrix}$$

Matriks V

$$V = \begin{pmatrix} 0.0317 & -0.7609 & 0.6481 \\ -0.8125 & 0.3580 & 0.4601 \\ -0.5821 & -0.5411 & -0.6069 \end{pmatrix}$$

Dari keempat matrik diatas, E merupakan matrik essensial, U, S dan V merupakan dekomposisi matrik essensial menggunakan teknik

singular value decompotion (SVD). Parameter matrik essensial diatas masih dipengaruhi oleh kesalahan (noise). Hal ini dapat dilihat dari komposisi matrik S. Secara teori, nilai matrik S seharusnya memiliki nilai diagonal $|1 \quad 1 \quad 0|$. Sehingga perlu dilakukan suatu konstrain terhadap matrik essensial dengan mengubah nilai diagonalnya dengan nilai $|1 \quad 1 \quad 0|$ seperti dibawah ini.

Matriks esensial

$$E = \begin{pmatrix} -0.7003 & 0.0494 & 0.7121 \\ -0.4397 & -0.8157 & -0.3759 \\ 0.5623 & -0.5764 & 0.5930 \end{pmatrix}$$

Matriks U

$$U = \begin{pmatrix} -0.5252 & 0.4658 & 0.7121 \\ -0.8457 & -0.3789 & -0.3759 \\ 0.0948 & -0.7996 & 0.5929 \end{pmatrix}$$

Matriks S

$$S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Matrix V

$$V = \begin{pmatrix} -0.4384 & -0.6227 & -0.6481 \\ -0.4261 & 0.7789 & -0.4602 \\ -0.7914 & -0.0744 & 0.6068 \end{pmatrix}$$

Pada tabel diatas, didapat nilai matrik essensial dan dekomposisinya yang baru, setelah dilakukan proses konstrain.

4.2.2. Parameter Ekstrinsik Kamera

Penentuan parameter ekstrinsik ini berdasarkan atas dua metode yaitu Metode Matrik Ssingular (SVD) dan Metode Pan. Kedua metode ini sama-sama menggunakan parameter matrik essensial sebagai data inputan awal untuk menghasilkan enam parameter ekstrinsik masing-masing kamera pada foto stereo. Akan tetapi, terdapat sedikit perbedaan. Metode Pan menggunakan matrik essensial yang telah di transposekan, sedangkan Metode dekomposisi matrik singular sebaliknya. Masih menggunakan data foto yang sama dengan yang digunakan pada penentuan essensial matrik, didapat nilai parameter untuk kedua kamera stereo dengan masing-masing metode sebagai berikut:

Tabel 4.1 Data Parameter ekstrinsik kamera stereo pada balok kalibrasi

Metode dekomposisi matrik singular (SVD)						
ID Photo	***Omega***	***Phi***	***Kappa***	X_L	Y_L	Z_L
Photo 1	0	0	0	0	0	0
Photo 2	-1.2367	52.5864	-1.2367	0.6467	0.4624	-0.6066
Metode Pan						
ID Photo	***Omega***	***Phi***	***Kappa***	X_L	Y_L	Z_L
Photo 1	0	0	0	0	0	0
Photo 2	-1.2367	55.0334	-1.2367	0.6514	0.4658	-0.599

Tabel 4.2 Selisih hasil perhitungan parameter ekastrinsik dari kedua metode

Selisih Nilai Parameter Ekstrinsik						
ID Photo	*Δ Ομεγα (δ)*	*Δ Πηι (δ)*	*Δ Καππα (δ)*	$\Delta \Xi_A$	$\Delta \Psi_A$	ΔZ_A
Photo 2	0	2.447	0	0.0047	0.0034	0.0076

Dari hasil yang ditampilkan diatas, nilai selisih terbesar untuk parameter sudut rotasi yaitu φ (phi) dengan besaran 2.4470°, dari kedua sudut rotasi tersebut bernilai positif sehingga arah rotasi berlawanan dengan arah jarum jam. Sedangkan nilai selisih terbesar untuk parameter translasi yaitu ZL dengan nilai 0.0076. Posisi kamera dalam hal ini memiliki nilai skala dan satuan sembarang, hal ini disebabkan oleh nilai parameter matrik essensial merupakan matrik up to scale (skala sembarang).

4.2.3. Parameter Koordinat Obyek

Parameter koordinat obyek, merupakan parameter yang dihasilkan melalui proses interseksi. Proses ini membutuhkan nilai enam parameter ekstrinsik untuk masing-masing kamera stereo. Dengan menggunakan nilai parameter eksterinsik yang telah dihasilkan pada proses dekomposisi matrik essensial, didapat nilai koordinat obyek 3-dimensi sebagai berikut:

No Titik	X	Y	Z
1	-0.11975	0.081051	-0.38848
2	-0.10003	0.096761	-0.41335
3	-0.0822	0.114192	-0.4448
4	-0.05932	0.130115	-0.46728
5	-0.03559	0.147869	-0.49386
6	-0.00924	0.165954	-0.51894
7	0.013694	0.14161	-0.42739
8	0.029398	0.108346	-0.3264
9	0.060033	0.108165	-0.20636
10	0.188784	0.411021	-4.43115

Nilai pendekatan awal orientasi kamera dan koordinat obyek yang didapat dari matrik essensial diatas cukup baik. Ini dapat dibuktikan dengan hasil visualisasi terhadap nilai koordinat obyek dan koordinat posisi kamera dalam bentuk 3-dimensi. Dari visualisasi tersebut, didapat bentuk yang hampir sama dan menyerupai bentuk aslinya. Adapun contoh hasil visualisasi tersebut dapat dilihat pada gambar dibawah ini.

Selain dengan proses diatas, terdapat cara lain untuk membuktikan bahwa nilai awal orientasi kamera dan koordinat 3-dimensi. Objyek diatas memiliki nilai yang cukup baik. Cara tersebut dapat dilakukan dengan proses iterasi terhadap keseluruhan parameter yang diataranya ialah parameter ekstrinsik kamera untuk dua buah foto stereo dan tiga parameter koordinat obyek 3-dimensi. Proses iterasi dilakukan dengan menggunakan teknik relative orientasi menggunakan persamaan kolinear yang telah terliniearisasi. Apabila dalam proses iterasi nilai koreksi untuk tiap parameter ekstrinsik kamera dan koordinat 3-dimensi selalu mengecil pada tiap iterasi, maka nilai parameter tersebut dapat dikatakan benar.

BAB V

PENUTUP

5.1. Kesimpulan

Kesimpulan yang dapat diambil selama proses pelaksanaan penelitian mengenai "Pembuatan program penentuan restitusi *bundle adjustmen* untuk studi deformasi " ialah sebagai berikut:

1. Program fotometrik hasil penelitian dapat berfungsi dengan baik, ditunjang dengan kemudahan dalam proses memasukan data dan kemampuan dalam memproses data dengan jumlah banyak dalam waktu yang relatf singkat.
2. Program Fotometrik dapat mengambil koordinat foto secara otomatis dan cepat.
3. Perhitungan Relatif Orientasi dan *Intersection* dengan menggunakan Program telah disusun sesuai dengan algoritma sehingga hasil yang dikeluarkan adalah hasil yang teliti.

5.2. Saran

1. Program Fotometrik ini dapat dikembangkan dengan menambahkan media untuk menggambarkan hasil tiga dimensi dari titik yang terhitung.
2. Untuk pembuatan Program disarankan untuk menggunakan Visual C# kerena C# mempunyai banyak kelebihan dan mudah dioperasikan oleh siapa saja.

Lampiran 1. Source Code Penyimpanan Project

```
private void SaveasProject()
{
    if (!_lastSaved && String.IsNullOrEmpty(_projectName))
    {
        using (SaveFileDialog dlg = new SaveFileDialog())
        {
            // Display a dialog for saving the album
            dlg.Title = "Save Project";
            dlg.DefaultExt = "dfm";
            dlg.Filter = "Album files (*.dfm)|*.dfm|" + "All files|*.*";
            dlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            dlg.RestoreDirectory = true;

            if (dlg.ShowDialog() == DialogResult.OK)
            {
                _projectName = dlg.FileName;
                if (SaveProject())
                    _lastSaved = true;
                else
                    _lastSaved = false;
                Version ver = new Version(Application.ProductVersion);
                Text = String.Format("{3} - Deformasi {0:0}.{1:0}.{2:0}",
                      ver.Major, ver.Minor, ver.Build, _projectName);
            }
        }
    }
    else if (!_lastSaved && !String.IsNullOrEmpty(_projectName))
    {
        if (SaveProject())
            _lastSaved = true;
        else
            _lastSaved = false;
```

```
        }
    }

    protected override void OnLoad(EventArgs e)
    {
        if (Manager == null)
        {
            this.Show();
        }
        base.OnLoad(e);
    }
    protected override void OnFormClosing(FormClosingEventArgs e)
    {
        if (Manager != null)
        {
            if (SaveAndCloseAlbum() == false)
                e.Cancel = true;
            else
                e.Cancel = false;
        }
        base.OnFormClosing(e);
    }
    #endregion
```

Lampiran 2. Suorce Code Penmanggil Foto

```
private void undoToolStripMenuItem_Click(object sender, EventArgs e)
{
    NewAlbum();
    using (OpenFileDialog dlg = new OpenFileDialog())
    {
        dlg.Title = "Add Image(s) to Project";
        dlg.Multiselect = true;
        dlg.Filter = "Image Files (JPEG, TIF, GIF, BMP, etc.)|"
           + "*.jpg;*.jpeg;*.gif;*.bmp;"
           + "*.tif;*.tiff;*.png|"
           + "JPEG files (*.jpg;*.jpeg)|*.jpg;*.jpeg|"
           + "GIF files (*.gif)|*.gif|"
           + "BMP files (*.bmp)|*.bmp|"
           + "TIFF files (*.tif;*.tiff)|*.tif;*.tiff|"
           + "PNG files (*.png)|*.png|"
           + "All files (*.*)|*.*";
        dlg.InitialDirectory = Environment.CurrentDirectory;
        dlg.RestoreDirectory = true;
        Crosses tempcrosses = null;
        if (dlg.ShowDialog() == DialogResult.OK)
        {
            string[] files = dlg.FileNames;
            int index = 0;
            try
            {
                foreach (string s in files)
                {
                    Debug.WriteLine(s);
                    Photograph photo = new Photograph(s);

                    index = Manager.AlbumContent.IndexOf(photo);
                    if (index < 0)
```

```
        {
            Manager.AlbumContent.Add(photo);
            int idx = Manager.AlbumContent.IndexOf(photo);
            photo.SetPhotoIndex(idx);
        }
    }
    if (PhotoLis.Items.Count == 0)
    {
        foreach (string s in files )
        {
            if (tempcrosses == null)
            {
                tempcrosses = new Crosses();
                if (tempcrosses != null)
                {
                    PhotoLis.Items.Add(s);
                    tempcrosses.ImageName = s;
                    Trace.WriteLine("s " + tempcrosses.ImageName.ToString());
                    _crossesOnEachImage.CrossesOnImage.Add(tempcrosses);
                    tempcrosses = null;
                }
            }
        }
    }
    else
    {
        foreach (string s in files)
        {
            Debug.WriteLine(s);
            Photograph photo = new Photograph(s);
            index = Manager.AlbumContent.IndexOf(photo);
            if (index < 0)
            {
                Manager.AlbumContent.Add(photo);
                int idx = Manager.AlbumContent.IndexOf(photo);
```

```
                        photo.SetPhotoIndex(idx);   //set photo & its thumbnail index
                    }
                }
                foreach (string s in files)
                {
                    if (!PhotoLis.Items.Contains(s))
                    {
                        if (tempcrosses == null)
                        {
                            tempcrosses = new Crosses();
                            if (tempcrosses != null)
                            {
                                PhotoLis.Items.Add(s);
                                tempcrosses.ImageName = s;
                                Trace.WriteLine("s " + tempcrosses.ImageName.ToString());
                                _crossesOnEachImage.CrossesOnImage.Add(tempcrosses);
                                tempcrosses = null;
                            }
                        }
                    }
                }
            }
        }
        catch (ArgumentException ex)
        {
            MessageBox.Show("Unable to open file:" + ex.Message);
        }
        finally
        {
            Trace.WriteLine(_crossesOnEachImage.ToString());
            _lastSaved = false;
        }
    }
}
```

Lampiran 3. Source code Pembuka Project

```
private void openToolStripMenuItem1_Click(object sender, EventArgs e)
{
    using (OpenFileDialog dlg = new OpenFileDialog())
    {
        dlg.Title = "Open Project";
        dlg.DefaultExt = "dfm";
        dlg.Filter = "Album files (*.dfm)|*.dfm|"
            + "All files|*.*";
        dlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        dlg.RestoreDirectory = true;
        if (dlg.ShowDialog() == DialogResult.OK)
        {
            _projectName = dlg.FileName;
            ReadProject();
            _lastSaved = false;
            Version ver = new Version(Application.ProductVersion);
            Text = String.Format("{3} - Deformasi {0:0}.{1:0}.{2:0}",
             ver.Major, ver.Minor, ver.Build, _projectName);
            //_isPlatePointsAvailable = false;
        }
    }
}
```

Lampiran 4. Source code penentuan centoid

```
int[,] T = new int[15, 15];
int[,] Tress = new int[15, 15];
private int[] K;
private int[] H;
private double XPixel;
private double YPixel;
private double XFoto;
private double YFoto;
private double sigmaX;
private double sigmaY;
private double SigmaXY;


private void imageBox1_MouseClick(object sender, MouseEventArgs e)
{
    Point pt = ConvertPtFromPBoxToImageCoordinates(e.Location);
    MouseEventArgs ne = new MouseEventArgs(e.Button, e.Clicks, pt.X, pt.Y, e.Delta);
    e = ne;
    if (e.Button == MouseButtons.Left && img != null)
    {
        Bitmap original = new Bitmap(imageBox1.Image.Bitmap);

        T = new int[15, 15]; //_____________________ matrik Gray Value

        for (int i = (e.X) - 7; i <= (e.X) + 7; i++)
        {
            for (int j = (e.Y) - 7; j <= (e.Y) + 7; j++)
            {
                T[i - ((e.X) - 7), j - ((e.Y) - 7)] = ((original.GetPixel(i, j).B) +
                        (original.GetPixel(i, j).G) + (original.GetPixel(i, j).R)) / 3;
            }
```

```
        }
    }
}


        var SumT = (from int val in T select val).Sum(); //___Garay___

        //Console.WriteLine();
        K = new int[15]; //_____________________________ Matrik X

        for (int i = (e.X) - 7; i <= (e.X) + 7; i++)
        {
            K[i - ((e.X) - 7)] = i;
         }

        H = new int[15];  //_____________________________ Matrik Y
        for (int j = (e.Y) - 7; j <= (e.Y) + 7; j++)
        {
            H[j - ((e.Y) - 7)] = j;
        }

        ////_____________________________Metode Tres Hold_____________________________;

        // untuk menampilkan nilai maksimal
        //var SumT = (from int val in T select val).Sum(); //___Garay___
        var SumX = (from int val in K select val).Sum(); //___________X
        var SumY = (from int val in H select val).Sum(); //___________Y
        var max = (from int val in T select val).Max();
        var SumT1 = (from int val in T select val).Average();

        // untuk menampilkan standar perhitungan
        var tot = max - (SumT1 / 2);
        double SumTotal = 0;

        for (int i = 0; i < 15; i++)
```

```
{
    for (int j = 0; j < 15; j++)
    {
        if (T[i, j] > tot)
        {
            Tress[i, j] = 1;
            SumTotal += 1;

        }
        else { Tress[i, j] = 0; }
    }
}
atasX = 0;//_____ X
for (int i = 0; i < 15; i++)

    for (int j = 0; j < 15; j++)
    {
        atasX += T[j, i] * K[i];
    }
atasY = 0;  //_____ Y
for (int i = 0; i < 15; i++)

    for (int j = 0; j < 15; j++)
    {
        atasY += T[j, i] * H[j];
    }

// Jumlah Kuadrat X
SumX2 = 0;//_____ X
for (int i = 0; i < 15; i++)
{
    SumX2 += K[i] * K[i];
}
// Jumlah Kuadrat X
SumY2 = 0;  //_____ Y
```

```
            for (int j = 0; j < 15; j++)
            {
                SumY2 += H[j] * H[j];
            }

            XPixel = atasX / SumT;
            YPixel = atasY / SumT;
            X2 = XPixel * XPixel;
            Y2 = YPixel * YPixel;

            SumT2 = SumT * SumT;

            sigmaX = Math.Sqrt((SumX2 - 2 * (XPixel * SumX) + (15 * X2)) /
                    (12 * (SumTotal * SumTotal)));
            sigmaY = Math.Sqrt((SumY2 - 2 * (YPixel * SumY) + (15 * Y2)) /
                    (12 * (SumTotal * SumTotal)));
            SigmaXY = (((SumX + SumY) - (XPixel * SumY) - (YPixel * SumX))
                    + (15 * XPixel * YPixel) / (12 * (SumTotal * SumTotal)));

            //_____________ KOnversi Koordinat_________


            XFoto = (XPixel - ((Wx / 2) - 0.5)) * 0.005;
            YFoto = (((Hy / 2) - 0.5) - YPixel) * 0.005;

        }
    }
```

Lampiran 7. Relatif Orintasi

```
//menghitung RSQ
 double[] rsq = new double[3];
 rsq = ProjectRSQ(ref _omega, ref_phi, ref _kappa, ref _xL, ref_yL, ref_zL, ref xc, ref yc, ref zc);
```

```
            PrintVectorDebug(ref rsq, "rsq");

    //Matrix A
    double a = Math.PI / 180;

    A[k, 0] = (_focus / (rsq[2] * rsq[2])) * ((rsq[0] * ((-rotasi[2, 2] * (yc - YL)) + (rotasi[2, 1] *
        (zc - ZL)))) - (rsq[2] * ((-rotasi[0, 2] * (yc - YL)) + (rotasi[0, 1] * (zc - ZL)))));
    A[k, 1] = (_focus / (rsq[2] * rsq[2])) * ((rsq[0] * ((Math.Cos(Phi * a) * (xc - XL)) +
        (Math.Sin(Omega * a) * Math.Sin(Phi * a) * (yc - YL)) - (Math.Cos(Omega * a) * Math.Sin(Phi *
        a) * (zc - ZL)))) - rsq[2] * ((-Math.Sin(Phi * a) * Math.Cos(Kappa * a) * (xc - XL)) +
        (Math.Sin(Omega * a) * Math.Cos(Phi * a) * Math.Cos(Kappa * a) * (yc - YL)) - (Math.Cos(Omega *
        a) * Math.Cos(Phi * a) * Math.Cos(Kappa * a) * (zc - ZL))));
    A[k, 2] = (-_focus / rsq[2]) * ((rotasi[1, 0] * (xc - XL)) + (rotasi[1, 1] * (yc - YL)) +
         (rotasi[1, 2] * (zc - ZL)));
    A[k, 3] = -(_focus / (rsq[2] * rsq[2])) * ((rsq[0] * rotasi[2, 0]) - (rsq[2] * rotasi[0, 0]));
    A[k, 4] = -(_focus / (rsq[2] * rsq[2])) * ((rsq[0] * rotasi[2, 1]) - (rsq[2] * rotasi[0, 1]));
    A[k, 5] = -(_focus / (rsq[2] * rsq[2])) * ((rsq[0] * rotasi[2, 2]) - (rsq[2] *
        rotasi[0, 2]));
    A[k + 1, 0] = (_focus / (rsq[2] * rsq[2])) * (rsq[1] * ((-rotasi[2, 2] * (yc - YL)) + (rotasi[2, 1]
        * (zc - ZL))) - rsq[2] * ((-rotasi[1, 2] * (yc - YL)) + (rotasi[1, 1] * (zc - ZL))));
    A[k + 1, 1] = (_focus / (rsq[2] * rsq[2])) * ((rsq[1] * ((Math.Cos(Phi * a) * (xc - XL)) + (Math.Sin
        (Omega * a) * Math.Sin(Phi * a) * (yc - YL)) - (Math.Cos(Omega * a) * Math.Sin(Phi * a) * (zc -
        ZL)))) -rsq[2] * ((Math.Sin(Phi * a) * Math.Sin(Kappa * a) * (xc - XL)) - (Math.Sin(Omega * a)
        * Math.Cos(Phi * a) * Math.Sin(Kappa * a) * (yc - YL)) + (Math.Cos(Omega * a) * Math.Cos(Phi *
        a) * Math.Sin(Kappa * a) * (zc - ZL))));
    A[k + 1, 2] = (_focus / rsq[2]) * ((rotasi[0, 0] * (xc - XL)) + (rotasi[0, 1] * (yc - YL)) +
        (rotasi[0, 2] * (zc - ZL)));
    A[k + 1, 3] = -(_focus / (rsq[2] * rsq[2])) * ((rsq[1] * rotasi[2, 0]) - (rsq[2] * rotasi[1, 0]));
    A[k + 1, 4] = -(_focus / (rsq[2] * rsq[2])) * ((rsq[1] * rotasi[2, 1]) - (rsq[2] * rotasi[1, 1]));
    A[k + 1, 5] = -(_focus / (rsq[2] * rsq[2])) * ((rsq[1] * rotasi[2, 2]) - (rsq[2] * rotasi[1, 2]));


    //Matrix L

    L[k, 0] = (xp - _xo) + (_focus * (rsq[0] / rsq[2]));
```

```
L[k + 1, 0] = (yp - _yo) + (_focus * (rsq[1] / rsq[2]));
objectspacepoint(spapoint);


private double[] ProjectRSQ(ref double Omega, ref double Phi, ref double Kappa, ref double XL,
      ref double YL, ref double ZL, ref double Xc, ref double Yc, ref double Zc)
   {
      double[] rsq = new double[3];

      double[,] M_rot = new double[3, 3];
      M_rot = ProjectRotation(ref Omega, ref Phi, ref Kappa);

      rsq[0] = (M_rot[0, 0] * (Xc - XL)) + (M_rot[0, 1] * (Yc - YL)) + (M_rot[0, 2] * (Zc - ZL));
      rsq[1] = (M_rot[1, 0] * (Xc - XL)) + (M_rot[1, 1] * (Yc - YL)) + (M_rot[1, 2] * (Zc - ZL));
      rsq[2] = (M_rot[2, 0] * (Xc - XL)) + (M_rot[2, 1] * (Yc - YL)) + (M_rot[2, 2] * (Zc - ZL));

      return rsq;
   }

    //Matrix Rotasi

   private double[,] ProjectRotation(ref double Omega, ref double Phi, ref 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.Cos(Omega * a) * Math.Sin(Kappa * a) + Math.Sin(Omega * a) * Math.Sin(Phi * a) *
          Math.Cos(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;
    }

    private CSML.Matrix Reshape3x3to9x1(CSML.Matrix m)
    {
        CSML.Matrix temp = CSML.Matrix.HorizontalConcat(m.Column(1),
        CSML.Matrix.HorizontalConcat(m.Column(2), m.Column(3)));
        return temp;
    }

    private CSML.Matrix ComputeFinalSolution(ref CSML.Matrix pmTotal)
    {
        CSML.Matrix result = RotationFixSigns(ref pmTotal); //ok
        CSML.Matrix duplicateRemove = DuplicateRemoval(ref result);//ok
        CSML.Matrix Rt = FurtherDuplicateRemoval(ref duplicateRemove);//ok
        CSML.Matrix finalRt = DeletePoorFits(Rt);
        System.Diagnostics.Trace.WriteLine("Final Rt: \n" + finalRt.ToString());

        return finalRt;
    }

    private CSML.Matrix DeletePoorFits(CSML.Matrix m)
    {
        int n = m.ColumnCount / 4;
        double[] errors = new double[n];
        for (int i = 0; i < n; i++)
        {
            CSML.Matrix projectiveMatrix = m.Extract(1, 3, 1 + i * 4, 4 + i * 4);
            //extract R & t
            CSML.Matrix R = projectiveMatrix.Extract(1, 3, 1, 3);
```

```
            CSML.Matrix t = projectiveMatrix.ExtractColumn(4);
            CSML.Matrix tt = SkewMatrix(new double[3] {t[1,1].Re,t[2,1].Re,t[3,1].Re});

            errors[i] = Math.Sqrt(EpipolarErrorProjection(R * tt) / _photopairlist.Count);
            System.Diagnostics.Trace.WriteLine("error: " + errors[i]);
        }

        int index = 0;
        for (int j = 0; j < n-1; j++)
        {
            if ((errors[j]*max_rel_fit_err) < (errors[j + 1]*max_rel_fit_err))
            {
                index = j;
            }
            else
                index = j + 1;
        }

        CSML.Matrix mat = m.Extract(1, 3, 1 + index * 4, 4 + index * 4);
        return mat;
    }

    //r(3x3)
    private double EpipolarErrorProjection(CSML.Matrix r)
    {
        int n = _photopairlist.Count;
        double er = 0.0;
        for (int i = 0; i < n; i++)
        {
            //Left photo
            CSML.Matrix xp = new CSML.Matrix(3, 1);
            xp[1, 1].Re = _photopairlist[i].XLeft;
            xp[2, 1].Re = _photopairlist[i].YLeft;
            xp[3, 1].Re = _focalLengthLeft;
```

```
            //right photo
            CSML.Matrix yp = new CSML.Matrix(3, 1);
            yp[1, 1].Re = _photopairlist[i].XRight;
            yp[2, 1].Re = _photopairlist[i].YRight;
            yp[3, 1].Re = _focalLengthRight;

            CSML.Matrix xf = xp.Transpose() * r;
            CSML.Matrix fy = r * yp;
            double b = (xf * yp).ColumnSum(1).Re;
            double c = ((xf * xf.Transpose()) * (yp.Transpose() * yp) +
                        (xp.Transpose() * xp) * (fy.Transpose() * fy)).ColumnSum(1).Re;
            er = er + (b * b) / c;

        }
        return er;
    }

    private CSML.Matrix SkewMatrix(double[] v)
    {
        System.Diagnostics.Debug.Assert(v.Length == 3);
        double[,] m = new double[3, 3] { { 0, -v[2], v[1] },
                                         { v[2], 0, -v[0] },
                                         { -v[1], v[0], 0 } };
        return new CSML.Matrix(m);
    }
    #endregion

    #region Essential Matrix From Homography

    private double[,] ProjectiveMatrixFromHomography(double[,] h)
    {
        /// svd: h(3x3) = U(3x3) S(3) VT(3x3) = UWV
        double[,] U = new double[3, 3];
        double[] S = new double[3];
        double[,] VT = new double[3, 3];
```

```
if (SVD(h, 3, 3, 2, 2, 2, ref S, ref U, ref VT))
{
}
double s1 = S[0] / S[1];
double s3 = S[2] / S[1];
double a1 = Math.Sqrt(1 - (s3 * s3));
double b1 = Math.Sqrt((s1 * s1) - 1);
//a1.ToString() + " , " + b1.ToString());
double[,] ab = Unitize(a1, b1);
double[,] cd = Unitize(1 + s1 * s3, a1 * b1);
double[,] ef = Unitize(-b1 / s1, -a1 / s3);
double[,] v1 = new double[3, 1];
double[,] v3 = new double[3, 1];
for (int i = 0; i < 3; i++)
{
    v1[i, 0] = VT[0, i];
    v3[i, 0] = VT[2, i];
}

CSML.Matrix v1_csml = new CSML.Matrix(v1);
CSML.Matrix v3_csml = new CSML.Matrix(v3);
CSML.Matrix n1_csml = (ab[0, 1] * v1_csml) - (ab[0, 0] * v3_csml);
CSML.Matrix n2_csml = (ab[0, 1] * v1_csml) + (ab[0, 0] * v3_csml);
double invd = s1 - s3;

CSML.Matrix VT_csml = new CSML.Matrix(VT);
CSML.Matrix U_csml = new CSML.Matrix(U);

double[,] constant1 = new double[3, 3] { { cd[0, 0], 0, cd[0, 1] }, { 0, 1, 0 },
                                         { -cd[0, 1], 0, cd[0, 0] } };
CSML.Matrix constant1_csml = new CSML.Matrix(constant1);
CSML.Matrix R1_csml = U_csml * constant1_csml * VT_csml;
//System.Diagnostics.Trace.WriteLine("R1_csml: " + R1_csml.ToString());

double[,] constant2 = new double[3, 3] { { cd[0, 0], 0, -cd[0, 1] }, { 0, 1, 0 },
```

```
                                                { cd[0, 1], 0, cd[0, 0] } };
        CSML.Matrix constant2_csml = new CSML.Matrix(constant2);
        CSML.Matrix R2_csml = U_csml * constant2_csml * VT_csml;

        CSML.Matrix t1_csml = (ef[0, 0] * v1_csml) + (ef[0, 1] * v3_csml);
        //System.Diagnostics.Trace.WriteLine("t1_csml: " + t1_csml.ToString());

        CSML.Matrix t2_csml = (ef[0, 0] * v1_csml) - (ef[0, 1] * v3_csml);

        double[,] side = new double[3, 1] { {0}, {0}, {1} };
        CSML.Matrix side_csml = new CSML.Matrix(side);

        if ((side_csml.Transpose() * n1_csml).RowSum(1).Re < 0)
        {
            t1_csml = -1 * t1_csml;
            n1_csml = -1 * t1_csml;
        }

        if ((side_csml.Transpose() * n2_csml).RowSum(1).Re < 0)
        {
            t2_csml = -1 * t2_csml;
            n2_csml = -1 * n2_csml;
        }
        CSML.Matrix hz1_csml = CSML.Matrix.VerticalConcat(R1_csml, t1_csml);
        CSML.Matrix hz3_csml = CSML.Matrix.VerticalConcat(R2_csml, t2_csml)
        CSML.Matrix M_csml = CSML.Matrix.VerticalConcat(hz1_csml, hz3_csml);

        return (CSML2Array(ref M_csml));
    }


    private double[,] Unitize(double a, double b)
    {
        double[,] t = new double[1, 2];
        double sum = Math.Sqrt(a * a + b * b);
```

```
        t[0, 0] = a / sum;
        t[0, 1] = b / sum;
        return t;
    }
    private double[,] ComputeHomography()
    {
        double[,] A = new double[3 * PointNumber, 9];
        CSML.Matrix a = new CSML.Matrix(3 * PointNumber, 9);
        for (int i = 0; i < PhotoPairList.Count; i++)
        {
            //Photo 1
            CSML.Matrix photo1 = new CSML.Matrix(1, 3);
            photo1[1,1].Re = PhotoPairList[i].XLeft;
            photo1[1,2].Re = PhotoPairList[i].YLeft;
            photo1[1,3].Re = FocalLengthLeft;

            //Photo2
            CSML.Matrix photo2 = new CSML.Matrix(1, 3);
            photo2[1,1].Re = PhotoPairList[i].XRight;
            photo2[1,2].Re = PhotoPairList[i].YRight;
            photo2[1,3].Re = FocalLengthRight;

            CSML.Matrix nom = new CSML.Matrix(1, 1);
            nom = photo2 * photo2.Transpose();
            double d = nom.RowSum(1).Re;
            CSML.Matrix temp = CSML.Matrix.Eye(3) - ((photo2.Transpose() * photo2) / d);

            //Kronecker
            CSML.Matrix a1 = photo1[1, 1] * temp;
            CSML.Matrix a2 = photo1[1, 2] * temp;
            CSML.Matrix a3 = photo1[1, 3] * temp;
            //Rearrange a
            a.Insert(i * 3 + 1, 1, a1);
            a.Insert(i * 3 + 1, 4, a2);
            a.Insert(i * 3 + 1, 7, a3);
```

```
}
/// 2. calculate SVD
/// svd: A(3nx9) = U(3nx3n) S(9) V(9x9) = UWV --> alglib: V=VT
A = CSML2Array(ref a);
//PrintMatrixDebug(ref A, "A");
int row = A.GetUpperBound(0) + 1;    //=3n
int col = A.GetUpperBound(1) + 1;    //=9
double[,] U = new double[row, row];
double[] S = new double[col];
double[,] V = new double[col, col];
SVD(A, row, col, 2, 2, 2, ref S, ref U, ref V);
///condition
double[,] cond = new double[1, 2];
cond[0, 0] = S[7] / S[0];
cond[0, 1] = S[8] / S[7];
///Homography
double[,] P = new double[3, 3];
//convert vector to matrix h
double[] tempH = new double[col];
for (int i = 0; i < col; i++)
{
    tempH[i] = V[col-1, i];
}
//double[,] h = new double[3, 3];
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
        P[i, j] = Math.Sqrt(3.0) * tempH[i + j * 3];
}
double ok = 0.0;
///Optional Sign hack
CSML.Matrix Homography = new CSML.Matrix(P);
for (int i = 0; i < PhotoPairList.Count; i++)
{
    //Photo 1
```

```
            CSML.Matrix photo1 = new CSML.Matrix(1, 3);
            photo1[1, 1].Re = PhotoPairList[i].XLeft;
            photo1[1, 2].Re = PhotoPairList[i].YLeft;
            photo1[1, 3].Re = FocalLengthLeft;

            //Photo2
            CSML.Matrix photo2 = new CSML.Matrix(1, 3);
            photo2[1, 1].Re = PhotoPairList[i].XRight;
            photo2[1, 2].Re = PhotoPairList[i].YRight;
            photo2[1, 3].Re = FocalLengthRight;

            ok = ok + (photo2 * Homography * photo1.Transpose()).RowSum(1).Re;
        }

        if (ok < (PhotoPairList.Count / 2))
            Homography = -1.0 * Homography;
        return (CSML2Array(ref Homography));
    }
    private void ExtractEssentialMatrix()
    {
        double[,] A = new double[9,9];
        /// 6 points method
        if (_pointNumber >= 6 && _pointNumber < 8)
        {
            ///1. Compose the Matrix A, ...., Extract final E from  6 or 7 points
            Essential=ExtractEssentialMatrix6Or7Points(ref A);
        }
        else if (_pointNumber >= 8)//max: 8 points only
        {
            ///1. Compose the Matrix A, ...., Extract final E from 8 points
            Essential = ExtractEssentialMatrixAt8Points(ref A);
        }
    }

    private double[,] ExtractEssentialMatrix6Or7Points(ref double[,] a)
```

```
{
    ExtractA(ref a);
    /// 2. calculate SVD
    /// svd: A(nx9) = U(nxn) S(n) V(9x9) = UWV' --> alglib: V=VT
    int row = a.GetUpperBound(0) + 1;
    int col = a.GetUpperBound(1) + 1;
    //System.Diagnostics.Trace.WriteLine("row, col: " + row.ToString() + " " + col.ToString());
    double[,] U = new double[row, row];
    double[] S = new double[row];
    double[,] V = new double[col, col];
    SVD(a, row, col, 2, 2, 2, ref S, ref U, ref V);              /// save the condition
    double num = S[0];
    for (int i = S.GetLowerBound(0); i <= S.GetUpperBound(0); i++)
        S[i] = S[i] / num;
    Condition = new double[2] { S[5], S[6] };
    /// 3. Calculate an Approx. of Essential Matrix,
    double[,] Ex = new double[3, 3];
    double[,] Ey = new double[3, 3];
    double[,] Ez = new double[3, 3];
    ApproximateEssential(ref V, ref Ex, ref Ey, ref Ez);

    /// 4. Calculate the E
    double[,] E = new double[3, 3];
    double[] condition = new double[0];
    Calculate_E(ref Ex, ref Ey, ref Ez, ref E, ref condition);

    //save E and condition to class member
    Condition = condition;
    return E;
}

private double[,] ExtractEssentialMatrixAt8Points(ref double[,] a)
{
    ExtractA(ref a);
    /// 2. calculate SVD
```

```
    /// svd: A(nx9) = U(nxn) S(n) V(9x9) = UWV --> alglib: V=V'
    /// svd: A(nx9) = U(nxn) S(n) V(9x9) = UWV --> alglib: V=V'
    int row = a.GetUpperBound(0) + 1;
    int col = a.GetUpperBound(1) + 1;
    //System.Diagnostics.Trace.WriteLine("row, col: " + row.ToString() + " " + col.ToString());
    double[,] U = new double[row, row];
    double[] S = new double[row];
    double[,] V = new double[col, col];
    SVD(a, row, col, 2, 2, 2, ref S, ref U, ref V);

    /// 3. Save the Condition
    double num = S[0];
    for (int i = S.GetLowerBound(0); i <= S.GetUpperBound(0); i++)
        S[i] = S[i] / num;  //all condition values
    //PrintVectorDebug(ref S, "normalised S" );
    double[] tempcondition = new double[2];
    tempcondition[0] = S[S.GetUpperBound(0) - 1];
    tempcondition[1] = S[S.GetUpperBound(0)];
    Condition = tempcondition;
    //PrintVectorDebug(ref Condition, "Condition");

    ///4.Extract real E
    Essential = Compute_E8(ref V);
    //PrintMatrixDebug(ref Essential, "Essential");
    return Essential;
}

private double[,] Compute_E8(ref double[,] v)
{
    int row = v.GetUpperBound(0);
    int col = v.GetUpperBound(1);
    double[] temp = new double[col + 1];
    for (int i = v.GetLowerBound(1); i <= col; i++)
    {
        temp[i] = v[row, i];
```

```
        }

        //convert vector to Matrix E
        double[,] e = new double[3, 3];
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
                e[i, j] = Math.Sqrt(2.0) * temp[i + j * 3];
        }
        return e;
    }
    private void ExtractA(ref double[,] a)
    {
        //int row = a.GetUpperBound(0);
        for (int i = 0; i < 6; i++)
        {
            System.Diagnostics.Trace.WriteLine("Point Numbers: " + _photopairlist.Count.ToString());
            //if (i > 5
            //    return;
            double x1 = _photopairlist.ElementAt(i).XLeft;
            double x2 = _photopairlist.ElementAt(i).XRight;
            double y1 = _photopairlist.ElementAt(i).YLeft;
            double y2 = _photopairlist.ElementAt(i).YRight;

            a[i, 0] = x1 * x2;
            a[i, 1] = y2 * x1;
            a[i, 2] = _focalLengthRight * x1;
            a[i, 3] = x2 * y1;
            a[i, 4] = y1 * y2;
            a[i, 5] = _focalLengthRight * y1;
            a[i, 6] = x2 * _focalLengthLeft;
            a[i, 7] = y2 * _focalLengthLeft;
            a[i, 8] = _focalLengthLeft * _focalLengthRight;

        }
```

```
}

    /// svd: N(9x10) = U(9x9) S(9x10) VT(10x10) = UWV
    double[,] U = new double[9, 9];
    double[] S = new double[9];
    double[,] VT = new double[10, 10];

    if (SVD(N, 9, 10, 2, 2, 2, ref S, ref U, ref VT))
    {
    }
    double num = S[0];
    for (int i = S.GetLowerBound(0); i <= S.GetUpperBound(0); i++)
        S[i] = S[i] / num;

    double x3 = VT[10 - 1, 1 - 1];
    double y3 = VT[10 - 1, 7 - 1];
    double z3 = VT[10 - 1, 10 - 1];

    double a = 0, x = 0, y = 0, z = 0;
    if (Math.Abs(z3) > Math.Max(Math.Abs(x3), Math.Abs(y3)))
    {
        a = z3; x = VT[10 - 1, 6 - 1] / a; y = VT[10 - 1, 9 - 1] / a; z = 1;    //xzz, yzz, zzz
    }
    else if (Math.Abs(y3) > Math.Max(Math.Abs(x3), Math.Abs(z3)))
    {
        a = y3; x = VT[10 - 1, 4 - 1] / a; y = 1; z = VT[10 - 1, 8 - 1] / a;    //xyy, yyy, yyz
    }
    else
    {
        a = x3; x = 1; y = VT[10 - 1, 2 - 1] / a; z = VT[10 - 1, 3 - 1] / a;    //xxx, xxy, xxz
    }

    CSML.Matrix temp = Ex * x + Ey * y + Ez * z;
    e = CSML2Array(ref temp);       //Essential Matrix
    //PrintMatrixDebug(ref e, "E");
```

```
        double[] v1 = new double[10]{x*x*x,x*x*y,x*x*z,x*y*y, x*y*z, x*z*z, y*y*y, y*y*z, y*z*z, *z*z};
        CSML.Matrix V = new CSML.Matrix(10, 1);
        for (int i = 0; i < 10; i++)
        {
            V[i + 1, 1].Re = VT[10 - 1, i];
            V[i + 1, 1].Re /= a;
            V[i + 1, 1].Re -= v1[i];
        }

        double norm = V.Norm();
        //save the condition
        cond = AssemblyTheCondition(ref Condition, ref norm, ref S);
        //PrintVectorDebug(ref cond, "cond");
    }

    double[] AssemblyTheCondition(ref double[] a, ref double b, ref double[] s)
    {
        double[] result = new double[12];
        result[0] = a[0];
        result[1] = a[1];
        result[2] = b;
        for (int i = 3; i < 12; i++)
            result[i] = s[i - 3];

        return result;
    }

    private void InsertColumn(ref CSML.Matrix N, int colIndex, ref CSML.Matrix m)
    {
        CSML.Matrix col1 = m.Column(1);
        CSML.Matrix col2 = m.Column(2);
        CSML.Matrix col3 = m.Column(3);
        for (int i = 1; i <= 3; i++)//row
        {
            for (int j = 1; j <= 3; j++)//vector
```

```
        {
            if (i == 1)
                N[j, colIndex] = col1[j];
            if (i == 2)
                N[i + 2 + j - 1, colIndex] = col2[j];
            if (i == 3)
                N[i + 3 + j, colIndex] = col3[j];
        }
    }
}

private void ReadFile(ref string file)
{
    string[] readData = File.ReadAllLines(file);

    for (int i = 0; i < readData.Count(); i++)
    {
        if (i == 0)
        {
            String[] firstLineData = readData[0].Split(new char[] { ' ', '\t' });
            if (firstLineData.Length == 3)
            {
                //Extract data eksterior orientasi untuk foto kiri
                X_camLeft = Double.Parse(firstLineData[0]);
                Y_camLeft = Double.Parse(firstLineData[1]);
                Z_camLeft = Double.Parse(firstLineData[2]);
            }
        }

        else if (i == 1)
        {
            String[] secondLineData = readData[1].Split(new char[] { ' ', '\t' });
            if (secondLineData.Length == 3)
            {
                //Extract data eksterior orientasi untuk foto kanan
```

```
                X_camRight = Double.Parse(secondLineData[0]);
                Y_camRight = Double.Parse(secondLineData[1]);
                Z_camRight = Double.Parse(secondLineData[2]);
            }
        }

        else if (i == 2)
        {
            String[] thirdLineData = readData[2].Split(new char[] { ' ', '\t' });
            if (thirdLineData.Length == 3)
            {
                PointNumber = int.Parse(thirdLineData[0]);
                FocalLenghtLeft = Double.Parse(thirdLineData[1]) * -1.0;
                FocalLenghtRight = Double.Parse(thirdLineData[2]) * -1.0;
            }
        }
        else
        {
            String[] data = readData[i].Split(new char[] { ' ', '\t' });
            if (data.Length == 5)
            {
                //Extract data titik foto pada tiap foto
                PhotoPair pair = new PhotoPair();
                pair.Label = data[0];
                pair.XLeft = Double.Parse(data[1]);
                pair.YLeft = Double.Parse(data[2]);
                pair.XRight = Double.Parse(data[3]);
                pair.YRight = Double.Parse(data[4]);

                PhotoPairList.Add(pair);
            }
        }
    }
}
```

```
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++)
    {
        System.Diagnostics.Trace.Write(v[i].ToString() + "\t");
    }

    System.Diagnostics.Trace.WriteLine("");
}
```

Lampiran 8. Intersection

```
////transformasi koordinat untuk tiap titik

        private double _omega;
        private double _phi;
        private double _kappa;
        private double _xL;
        private double _yL;
        private double _zL;
        private int _jmlpoint;
        private double _xo;
        private double _yo;
        private double _focus;
        Collection<CSML.Matrix> objspace = new Collection<CSML.Matrix>();
        List<PhotoPair> _photopairlist;
```

```
for (int i = 0; i < PointNumber; i++)
{
XYZtransLeft[i, 0] = PhotoPairList[i].XLeft * M_left[0, 0] + PhotoPairList[i].YLeft * M_left[1, 0] +
                     _focalLengthLeft * M_left[2, 0];
XYZtransLeft[i, 1] = PhotoPairList[i].XLeft * M_left[0, 1] + PhotoPairList[i].YLeft * M_left[1, 1] +
                     _focalLengthLeft * M_left[2, 1];
XYZtransLeft[i, 2] = PhotoPairList[i].XLeft * M_left[0, 2] + PhotoPairList[i].YLeft * M_left[1, 2] +
                     _focalLengthLeft * M_left[2, 2];
XYZtransRight[i, 0] = PhotoPairList[i].XRight * M_right[0, 0] + PhotoPairList[i].YRight * M_right[1, 0] +
                     _focalLengthRight * M_right[2, 0];
XYZtransRight[i, 1] = PhotoPairList[i].XRight * M_right[0, 1] + PhotoPairList[i].YRight * M_right[1, 1] +
                     _focalLengthRight * M_right[2, 1];
XYZtransRight[i, 2] = PhotoPairList[i].XRight * M_right[0, 2] + PhotoPairList[i].YRight * M_right[1, 2] +
                     _focalLengthRight * M_right[2, 2];
}
////Menghitung nilai faktor skala untuk tiap titik
double[] scale = new double[PointNumber];
for (int j = 0; j < PointNumber; j++)
      {
      scale[j] = (XYZtransLeft[j, 1] * (KT[0, 0] - X_camLeft) - XYZtransLeft[j, 0] * (KT[1, 0] -
                  Y_camLeft)) / (XYZtransLeft[j, 0] * XYZtransRight[j, 1] - XYZtransRight[j, 0] *
                  XYZtransLeft[j, 1]);

//Menghitung nilai koordinat objek pendekatan untuk tiap titik
double[,] threeDobject = new double[PointNumber, 3];
for (int k = 0; k < PointNumber; k++)
{
      threeDobject[k, 0] = scale[k] * XYZtransRight[k, 0] + KT[0, 0];
      threeDobject[k, 1] = scale[k] * XYZtransRight[k, 1] + KT[1, 0];
      threeDobject[k, 2] = scale[k] * XYZtransRight[k, 2] + KT[2, 0];
}
```

```
// proses least square untuk tiap koordinat pendekatan
for (int i = 0; i < _pointNumber; i++)
{

      rsq_left[0, 0] = M_left[0, 0] * (XYZ_obj[0] - X_camLeft) + M_left[0, 1] * (XYZ_obj[1] - Y_camLeft) +
                M_left[0, 2] * (XYZ_obj[2] - Z_camLeft);
      rsq_left[0, 1] = M_left[1, 0] * (XYZ_obj[0] - X_camLeft) + M_left[1, 1] * (XYZ_obj[1] - Y_camLeft) +
                M_left[1, 2] * (XYZ_obj[2] - Z_camLeft);
      rsq_left[0, 2] = M_left[2, 0] * (XYZ_obj[0] - X_camLeft) + M_left[2, 1] * (XYZ_obj[1] - Y_camLeft) +
                M_left[2, 2] * (XYZ_obj[2] - Z_camLeft);
      rsq_right[0, 0] = M_right[0, 0] * (XYZ_obj[0] - KT[0, 0]) + M_right[0, 1] * (XYZ_obj[1] - KT[1, 0]) +
                M_right[0, 2] * (XYZ_obj[2] - KT[2, 0]);
      rsq_right[0, 1] = M_right[1, 0] * (XYZ_obj[0] - KT[0, 0]) + M_right[1, 1] * (XYZ_obj[1] - KT[1, 0]) +
            M_right[1, 2] * (XYZ_obj[2] - KT[2, 0]);
      rsq_right[0, 2] = M_right[2, 0] * (XYZ_obj[0] - KT[0, 0]) + M_right[2, 1] * (XYZ_obj[1] - KT[1, 0]) +
                M_right[2, 2] * (XYZ_obj[2] - KT[2, 0]);

      //Menyusun Matrik koefisien A untuk foto kiri dan kanan
      double[,] A = new double[4, 3];
      A[0, 0] = -1.0 * _focalLengthLeft / 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 * _focalLengthLeft / 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 * _focalLengthLeft / 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 * _focalLengthLeft / 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 * _focalLengthLeft / 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 * _focalLengthLeft / 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 * _focalLengthRight / 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 * _focalLengthRight / 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 * _focalLengthRight / 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 * _focalLengthRight / 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 * _focalLengthRight / 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 * _focalLengthRight / Math.Pow(rsq_right[0, 2], 2) * (rsq_right[0, 1] * M_right[2, 2]
      - rsq_right[0, 2] * M_right[1, 2]);


//Menyusun matrik observasi b
double[,] b = new double[4, 1];
b[0, 0] = xy_obj[0] + (-1.0 * _focalLengthLeft) * rsq_left[0, 0] / rsq_left[0, 2];
b[1, 0] = xy_obj[1] + (-1.0 * _focalLengthLeft) * rsq_left[0, 1] / rsq_left[0, 2];
b[2, 0] = xy_obj[2] + (-1.0 * _focalLengthRight) * rsq_right[0, 0] / rsq_right[0, 2];
b[3, 0] = xy_obj[3] + (-1.0 * _focalLengthRight) * rsq_right[0, 1] / rsq_right[0, 2];

////Koreksi nilai koordinat untuk tiap titik
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];
double[,] Hasil = CSML2Array(ref XYZobjfnl);

// RMS error untuk tiap iterasi
CSML.Matrix So2 = ((A_csml * x - b_csml).Transpose() * (A_csml * x - b_csml));


  private CSML.Matrix Quad_to_Rot(ref CSML.Matrix q)
  {
      q = q / q.Norm();
      CSML.Matrix R = new CSML.Matrix(3, 3);
      R[1, 1].Re = Math.Pow(q[4, 1].Re, 2) + Math.Pow(q[1, 1].Re, 2) - Math.Pow(q[2, 1].Re, 2) -
            Math.Pow(q[3, 1].Re, 2);
      R[1, 2].Re = 2 * ((q[1, 1].Re * q[2, 1].Re) - (q[3, 1].Re * q[4, 1].Re));
      R[1, 3].Re = 2 * ((q[4, 1].Re * q[2, 1].Re) + (q[1, 1].Re * q[3, 1].Re));
```

```
    R[2, 1].Re = 2 * (q[4, 1].Re * q[3, 1].Re) + (q[1, 1].Re * q[2, 1].Re);
    R[2, 2].Re = Math.Pow(q[4, 1].Re, 2) - Math.Pow(q[1, 1].Re, 2) + Math.Pow(q[2, 1].Re, 2) -
         Math.Pow(q[3, 1].Re, 2);
    R[2, 3].Re = 2 * ((q[2, 1].Re * q[3, 1].Re) - (q[4, 1].Re * q[1, 1].Re));
    R[3, 1].Re = 2 * ((q[1, 1].Re * q[3, 1].Re) - (q[4, 1].Re * q[2, 1].Re));
    R[3, 2].Re = 2 * ((q[4, 1].Re * q[1, 1].Re) + (q[2, 1].Re * q[3, 1].Re));
    R[3, 3].Re = Math.Pow(q[4, 1].Re, 2) - Math.Pow(q[1, 1].Re, 2) - Math.Pow(q[2, 1].Re, 2) +
         Math.Pow(q[3, 1].Re, 2);
    return R;
}
```

BUKU CATATAN BIMBINGAN
TUGAS AKHIR

Nama	:	**Alben S. Liu**
NIM	:	**05.25.012**
Judul Skripsi	:	**Pembuatan Progran Untuk Penentuan Restitusi Bundle Adjustment dengan C# Visual Studio 2008** (Studi Kasus : Jembatan Rel Kereta Api Lawang dan Fly Over Arjosari)

Pembimbing : 1. Heri Purwanto, ST, MSc.
2. DR. Edwin TjahJadi, ST, M.Geom.SC

PROGRAM SARJANA STRATA SATU (S-1)
JURUSAN TEKNIK GEODESI
INSTITUT TEKNOLOGI NASIONAL
MALANG
2009

PETUNJUK

1. Buku Catatan Tugas Akhir ini wajib dimiliki oleh setiap mahasiswa Program Sarjana Jurusan Teknik Geodesi ITN Malang yang mengambil Tugas Akhir.
2. Buku ini dimaksudkan untuk membantu kelancaran proses bimbingan.
3. Buku ini harus dibawa dan diisi pada saat bimbingan.
4. Materi diskusi, arahan bimbingan, serta informasi lain yang berkaitan dengan Tugas Akhir harus diisikan pada kolom yang tersedia serta ditandatangani/diparaf oleh Pembimbing.
5. Jika diperlukan sewaktu–waktu Ketua Jurusan Teknik Geodesi ITN Malang dapat meminjam buku ini untuk mengetahui perkembangan penulisan Tugas Akhir mahasiswa yang bersangkutan.
6. Buku ini harus diserahkan ke jurusan Teknik Geodesi sebelum Sidang/Evaluasi Tugas Akhir dilaksanakan.

Malang, ... 2009

Ketua Program Studi

Program Sarjana

Jurusan Teknik Geodesi

Institut Teknologi Nasional Malang

Hery Purwanto, ST, Msc

CATATAN PROSES BIMBINGAN TUGAS AKHIR

Hari / Tanggal	Materi Diskusi / Catatan Bimbingan	Paraf
1/8 '10.	Perbaiki Daftar isi Lengkapi Daftar pustaka	
2/8 '10.	Perbaiki Bab I. Setiap kalimat memiliki SP	
3/8 '10.	Perbaiki Bab I	
9/8 '10.	Acc Bab I Perbaiki Bab 2 Perhatikan S P O/K ! Lanjutkan ke Bab 3	

CATATAN PROSES BIMBINGAN TUGAS AKHIR

Hari / Tanggal	Materi Diskusi / Catatan Bimbingan	PARAF
16/8 '10.	Acc Bab 5 Acc Seminar	

Hari / Tanggal	Materi Diskusi / Catatan Bimbingan	Par…
	Lakukan pembetulan penulisan sesuai arahan. - penulisan kalimat lebih kurang mengena dari segi content. - gt tidak perlu bingkai biar seragam.	
	perbaiki BAB II penulisan yg masih belum konsisten Revisi pelaksanaan BAB III sesuai dg saran & lanjutkan BAB IV, V	
	BAB I, II, III Acc. Lakukan pembetulan pada BAB IV pelaksanaan dg validasi data.	
	Buat uraian singkat pelaksanaan dengan hasil analisa untuk menjawab tujuan penelitian	

CATATAN PROSES BIMBINGAN TUGAS AKHIR

Hari / Tanggal	Materi Diskusi / Catatan Bimbingan	Paraf
12/8 '10.	Perbaiki Bab 3	
14/8 '10.	Perbaiki lagi Bab 2 ACC Bab 3 Lanjut ke Bab 4 & 5	
15/8 '10.	ACC Bab 2 & 4 Perbaiki Bab 5	

CATATAN PROSES BIMBINGAN TUGAS AKHIR

Hari / Tanggal	Materi Diskusi / Catatan Bimbingan	PARAF
	Revisi BAB IV, V sesuai tambahan dosen.	
	ACC BAB IV, V ACC ACC di semarkan	

CATATAN PROSES BIMBINGAN TUGAS AKHIR

Hari / Tanggal	Materi Diskusi / Catatan Bimbingan	PARAF

SURAT KETERANGAN

Yang bertanda tangan dibawah ini :

Nama : Alben S. Liu

NIM : 05. 25. 012

Jurusan : Teknik Geodesi

Telah menyelesaikan skripsi dan layak untuk diseminarkan dihadapan sidang Jurusan. Adapun Judul Skripsi tersebut adalah :

"Pembuatan Program Untuk Penentuan Restitusi Bundle Adjustment

Dengan C# Visual Studio 2008"

Demikian surat keterangan ini, agar dapat digunakan sebagai rekomendasi mahasiswa untuk Seminar Hasil.

Malang, ...2010

Dosen Pembimbing I	Dosen Pembimbing II
(Hery Purwanto, ST Msc)	(Dr. M. Edwin Tjahjadi, ST.,M.Geom.Sc)

Mengetahui

Jurusan Teknik Geodesi S-1

Ketua

18/08/2010

(Hery Purwanto, ST.,MSc)

Nomor :

Lampiran :

Perihal :

Kepada : Yth. Ketua Jurusan Teknik Geodesi
Fakultas Teknik Sipil dan Perencanaan
Institut Teknologi Nasional Malang
di Malang

Dengan Hormat,

Yang bertanda tangan dibawah ini :

Nama : **Dr. M. Edwin Tjahjadi, ST.,M.Geom.Sc**

Dosen Pembimbing ke : II (Dua)

Memberitahukan bahwa, nilai yang saya berikan kepada mahasiswa bimbingan skripsi sebagai berikut :

Nama : **ALBEN S. LIU**

NIM : **05.25.012**

Judul Skripsi : Pembuatan Program Untuk Penentuan Restitusi Bundle Adjustment Dengan C# Visual Studio 2008

Nilai Skripsi : (.....)

Demikian surat ini dibuat, atas bantuannya diucapkan banyak terimakasih

Malang , 16 Agustus -2010

Hormat Kami,

(Dr. M. Edwin Tjahjadi, ST.,M.Geom.Sc)

Nomor :

Lampiran :

Perihal :

Kepada : Yth. Ketua Jurusan Teknik Geodesi
Fakultas Teknik Sipil dan Perencanaan
Institut Teknologi Nasional Malang
di Malang

Dengan Hormat,

Yang bertanda tangan dibawah ini :

Nama : **Hery Purwanto, ST MSc**

Dosen Pembimbing ke : I (Satu)

Memberitahukan bahwa, nilai yang saya berikan kepada mahasiswa bimbingan skripsi sebagai berikut :

Nama : **ALBEN S. LIU**

NIM : **05.25.012**

Judul Skripsi : Pembuatan Program Untuk Penentuan Restitusi Bundle Adjustment Dengan C# Visual Studio 2008

Nilai Skripsi : (.....)

Demikian surat ini dibuat, atas bantuannya diucapkan banyak terimakasih

Malang, ...-.....-2010

Hormat Kami,

08/08/2010

(Hery Purwanto, ST MSc)

DAFTAR PUSTAKA

Ahmad, A.B., 2005. Analisis Ke Atas Prestasi Kamera Digital Kompak Untuk Aplikasi Fotogrametri Jarak Dekat, Universiti Teknologi Malaysia, Skudai Malaysia.

Brown, D.C., 1974. Evolution, Application and Potential of The Bundle Method of Photogrammetric Triangulation, Geodetic Services, Inc., Melbourne, Florida.

Chen, Y.-Q., 1983. ANALYSIS OF DEFORMATION SURVEYS - A GENERALIZED METHOD, University Of New Brunswick, Fredericton, N.B.

Chrzanowski, A., 1989. IMPLEMENTATION OF TRIGONOMETRIC HEIGHT TRAVERSING IN GEODETIC LEVELLING OF HIGH PRECISION, University Of New Brunswick, Fredericton, N.B.

Cooper, M.A.R. and Robson, S., 2001. Theory Of Close Range Photogrammetry. Wittles Publishing, London.

Dorstel, C., Jacobsen, K. and Stallmann, D., 2004. DMC – PHOTOGRAMMETRIC ACCURACY – CALIBRATION ASPECTS AND GENERATION OF SYNTHETIC DMC IMAGES, University of Hannover, Germany.

Geosystem, L., 2006. Leica Photogrammetry Suite Project Manager, Leica Geosystems Geospatial Imaging, United States of America.

Hanifa, N.R., 2007. Analisis Deformasi Menggunakan Teknik Close Range Photogrammetry, Institut Teknologi Bandung, Bandung.

Heindl, F.J., 1981. Direct Editing of Normal Equations of the Banded-Bordered Form. Photogrammetric Engineering And Remote Sensing, 47(4): 489-493.

Kuang, S.-L., 1991. Optimizing and Design of Deformation Monitoring Schemes, University of New Brunswick, Frederiction, N.B.

Mikhail, E.M., Bethel, J.S. and McGlone, J.C., 2001. Introduction To Modern Photogrammetry. John Wiley & Sons. INC., New York.

Schut, G., H., 1980. Block Adjustment of Bundles. The Canadian Surveyor, 34(2): 139-152.

Shirkhani, A., Varshosaz, D.M. and Saadatseresht, D.M., 2006. 3D Coordinate Measurement of Dam by Close Range Photogrammetry, K.N. Toosi University of Technology, Tehran.

Tjahjadi, E., 2008a. Ketelitian pengukuran dari kamera non metrik, Institut Teknologi Nasional, Malang.

Tjahjadi, E., 2008b. PRECISION FEATURE EXTRACTION FROM UNMANNED AERIAL PLATFORMS, Institut Teknologi Nasional, Malang.

Triggs, B., McLauchlan, P., F., Hartley, R., I. and Fitzgibbon, A., W. , 2000a. Bundle Adjustment - A Modern Synthesis. Lecture Notes in Computer Science 1883: 298-372.

Triggs, B., McLauchlan, P., Hartley, R. and Fitzgibbon, A., 2000b. Bundle Adjustment —A Modern Synthesis, INRIA Rh^one-Alpes, Montbonnot, France.

V-Stars, 2005. Basics of Photogrammetry.

Wang, X. and Clarke, T.A., 1999. SEPARATE ADJUSTMENT OF CLOSE RANGE PHOTOGRAMMETRIC MEASUREMENTS, City University, London.

Wolf, P.R. and Dewitt, B.A., 2004. *Element Of Photogrammetry with Application in GIS. Mc Graw Hill, New York.*

Andrew Troelsen, 2008. *Pro C# 2008 and the.NET 3.5 Platform*

Jhon Sharp, 2008 Visual C# Step by Step

Wiley Publishing, Inc, 2008. *Beginning* Microsoft® Visual C#® 2008

Pranoto, Suryo M, 2007. C_Sharp_ Part 1 - Pengenalan Logika Basic

Pranoto, Suryo M, 2007. C# Part 2 – Class Dan Array

Pranoto, Suryo M, 2007. C_Sharp_ Part 3 - Static And Function

Rachmatullah, Agro, 2002. Mempelajari C# Bahasa Pemrograman *Modern*

Brown Erik, 2007. *Windows Forms in Action 2nd Edition - Manning*