SKRIPSI

PEMBUATAN PROGRAM BUNDLE ADJUSTMENT MULTI PHOTO KONVERGEN DENGAN BAHASA C#

(Studi kasus : Foto Terrestrial)

Di susun Oleh:

Nurhadi Bashit

10.25.904

JURUSAN TEKNIK GEODESI

FAKULTAS TEKNIK SIPIL DAN PERENCANAAN

INSTITUT TEKNOLOGI NASIONAL

MALANG

2012

PREFACE

PERKUMPULAN PENGELOLAAN PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNOLOGI SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
ERSERO) MALANG
NIAGA MALANG
Kampus I : Jl. Bendugan Sigura No. 2 Telp. (0341)551431 (Hunting), Fax. (0341)553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341)417634 Malang

LEMBAR PENGESAHAN

SKRIPSI

PEMBUATAN PROGRAM BUNDLE ADJUSTMENT MULTI PHOTO KONVERGEN DENGAN BAHASA C#

Telah Dipertahankan di Hadapan Panitia Penguji Skripsi Jenjang Strata-1 (S-1)

Pada hari : Rabu

Tanggal : 8 Agustus 2012

Dan diterima untuk memenuhi persyaratan guna memperoleh gelar Sarjana Teknik (ST)

Oleh :

Nurhadi Bashit

10.25.904

Panitia Ujian Skripsi

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

Anggota Penguji

Penguji I	Penguji II	Penguji III
(DK. Sunaryo, ST., MT)	(Dr. Edwin Tjahjadi, ST., M.Geom.Sc)	(Ir. M. Nurhadi, MT)

LEMBAR PERSETUJUAN
SKRIPSI

PEMBUATAN PROGRAM BUNDLE ADJUSTMENT MULTI PHOTO KONVERGEN DENGAN BAHASA C#

Diajukan Sebagai Salah Satu Syarat Memperoleh Gelar Sarjana Teknik Geodesi
S-1
Institut Teknologi Nasional Malang

Disusun Oleh :

Nurhadi Bashit
10.25.904

Menyetujui,

Dosen Pembimbing I	**Dosen Pembimbing II**
(Dr. Edwin Tjahjadi, ST., M.Geom.Sc)	**(Ir. M. Nurhadi, MT)**

Mengetahui

Ketua Jurusan Teknik Geodesi S-1

(Ir. Agus Darpono, MT)

PEMBUATAN PROGRAM BUNDLE ADJUSTMENT MULTI PHOTO KONVERGEN DENGAN BAHASA C#

Oleh: Nurhadi Bashit (1025904)
Dosen Pembimbing 1 : Dr. Edwin Tjahjadi, ST., M.Geom.Sc
Dosen Pembimbing 2 : Ir. M. Nurhadi, MT

ABSTRAK

Perkembangan teknologi dan dunia digital yang sangat cepat sehingga menuntut orang untuk memperoleh informasi dengan cepat. Perkembangan teknologi dan dunia digital merambah hingga cabang keilmuan bidang geodesi dengan adanya perkembangan sensor seperti kamera dan *scanner*. Salah satu dari dampak perkembangan tersebut adalah pada bidang fotogrametri. Perkembangan di bidang fotogrametri sangatlah mempengaruhi dalam sistem pengukuran. Sistem pengukuran tersebut dapat dilakukan dengan cepat, efisien dan ekonomis.

Dalam penelitian ini, pembuatan program *bundle adjustment* dalam *photogrammetry,* untuk jaringan pemotretan *multi photo* yang *konvergen* sehingga memperoleh parameter *exterior orientation* yang teratakan serta koordinat obyek yang fix. Program perhitungan *bundle adjustment* ini dibuat dalam bahasa C# dengan menggunakan *software Microsoft Visual Studio 2010*. Dari hasil program ini akan ditampilkan di *windows DOS* berupa parameter EO teratakan dan koordinat 3D yang fix.

Hasil dari penelitian ini adalah sebuah program perhitungan *bundle adjustment* untuk memperoleh parameter EO teratakan dan koordinat 3D yang fix. Hasil perhitungan program *bundle adjustment* dapat dijalankan di *windows DOS* dengan data awal yang disimpan dalam *text file*.

Kata Kunci: Program, *bundle adjustment*

SURAT PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini :

Nama	: Nurhadi Bashit
NIM	: 10.25.904
Program Studi	: Teknik Geodesi S-1
Fakultas	: Fakultas Teknik Sipil Dan Perencanaan

Menyatakan dengan sesungguhnya bahwa Skripsi saya dengan judul :

"PEMBUATAN PROGRAM BUNDLE ADJUSTMENT MULTI PHOTO KONVERGEN DENGAN BAHASA C#"

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

Malang,

Yang membuat pernyataan

Nurhadi Bashit

NIM : 10.25.904

KATA PENGANTAR

Puji syukur penulis panjatkan kepada ALLAH SWT, karena berkat rahmat-Nya, penulis dapat menyelesaikan skripsi yang berjudul:

"PEMBUATAN PROGRAM BUNDLE ADJUSTMENT MULTI PHOTO KONVERGEN DENGAN BAHASA C#"

dimana penulisan skripsi ini disusun sebagai salah satu syarat untuk meraih gelar Sarjana Teknik pada Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.

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

1. Bapak Ir. Soeparno Djiwo, MT selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. A. Agus Santosa, MT selaku Dekan Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.
3. Bapak Ir. Agus Darpono, MT selaku Ketua Jurusan Teknik Geodesi Institut Teknologi Nasional Malang.
4. Bapak Dr. Edwin Tjahjadi, ST., M.Geom.Sc selaku dosen Pembimbing I dan Dosen Penguji II.
5. Bapak DK. Sunaryo, ST., MT selaku Dosen Penguji I.
6. Bapak Ir. M. Nurhadi, MT selaku dosen Pembimbing II dan Dosen Penguji III.
7. Segenap dosen, staf pengajar dan *recording* Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.

8. Kedua orang tuaku Drs. R. Soebardo dan Ir. Endang Rachmayani dan mbakku Kusumawardani ST, yang selalu memberikan dukungan, semangat dan doa.
9. Teman seperjuangan mengerjakan skripsi Ernesto Dos Santos, dan temanku Taufiq Ihsanudin, Amd.
10. Teman-teman ITN yang selalu memberikan semangat dan doa.
11. Semua pihak yang telah membantu peneliti yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa penulisan laporan ini masih belum sempurna, baik dari segi materi, sistematika pembahasan, maupun susunan bahasa. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan. Hasil penelitian ini dan dengan segala keterbatasannya dipersembahkan kepada dunia pendidikan, semoga ada manfaatnya untuk pengembangan sumber daya manusia di negara tercinta ini.

Malang, 10 September 2012

Penulis

DAFTAR GAMBAR

Halaman

Gambar 2.1 contoh konfigurasi jaringan pemotretan *konvergen (Uffenkam, 1998)* 4

Gambar 2.2 Proses Bundle Multi Foto 6

Gambar 2.3 Rotasi sudut omega terhadap sumbu X_ω 10

Gambar 2.4 Rotasi sudut phi terhadap sumbu Y_φ 11

Gambar 2.5 Rotasi sudut kappa terhadapt sumbu $Z_{\omega\varphi}$ 12

Gambar 2.6 Hubungan keruangan antara titik A sistem koordinat sembarang dengan titik sistem koordinat kamera 14

Gambar 2.7 Struktur Persamaan Normal 26

Gambar 3.1 Tampilan awal Program C# 39

Gambar 3.2 Tampilan awal Program Australis 40

Gambar 3.3 Diagram Alir Penelitian (1) 41

Gambar 3.4 Diagram Alir Penelitian (2) 42

Gambar 3.5 Diagram Alir Program 46

Gambar 3.6 Input dari notepad 47

Gambar 3.7 Susunan Matrix N dan t (Fraser, 1997) 48

Gambar 3.8 Tampilan awal Visual Studio 2010 49

Gambar 3.9 Tampilan New Project 50

Gambar 3.10 Tampilan awal program dengan bahasa C# 51

Gambar 3.11 Tampilan awal class 51

Gambar 4.1 Tampilan Program Bundle Adjustment 58

Gambar 4.2 Susunan pajang fokus, jumlah foto, jumlah titik 58

Gambar 4.3 Susunan parameter EO 59

Gambar 4.4 Susunan Koordinat Foto 59

Gambar 4.5 Susunan Koordinat Objek Space 60

Gambar 4.6 Tampilan Data yang dibuka dalam Program Bundle Adjustment 61

Gambar 4.7 Hasil Perhitungan Menggunakan Program Bundle Adjustment 61

DAFTAR TABEL

Halaman

Tabel 3.1.Parameter IO 43

Tabel 3.2.Parameter EO 43

Tabel 3.3.Ketelitian Parameter EO 44

Tabel 3.4. Koordinat obyek (X_i,Y_i,Z_i) dan ketelitiannya 44

Tabel 3.5. Koordinat foto dan ketelitiannya pada Foto1 44

Tabel 3.6. Koordinat foto dan ketelitiannya pada Foto2 45

Tabel 3.7. Koordinat foto dan ketelitiannya pada Foto3 45

Tabel 3.8. Koordinat foto dan ketelitiannya pada Foto4 45

Tabel 4.1 Hasil Akhir Parameter EO 62

Tabel 4.2 Hasil Akhir Object Space Point final 62

Tabel 4.3 Hasil Koreksi Parameter EO 63

Tabel 4.4 Hasil Koreksi Koordinat Object Space 64

DAFTAR ISI

Halaman

HALAMAN JUDUL i

HALAMAN PENGESAHAN ii

HALAMAN PERSETUJUAN iii

ABSTRAK iv

SURAT PERNYATAAN KEASLIAN SKRIPSI v

HALAMAN PERSEMBAHAN vi

KATA PENGANTAR vii

DAFTAR GAMBAR ix

DAFTAR TABEL xi

DAFTAR ISI xii

BAB I PENDAHULUAN 1

I.1 Latar Belakang 2

I.2 Rumusan Masalah 2

I.3 Tujuan Penelitian 2

I.4 Batasan Masalah 2

I.5 Tinjauan Pustaka 3

BAB II DASAR TEORI 4

II.1 *Bundle Adjustment* dan Kesalahan Observasi 4

II.2 Metode Gabungan Kuadrat Terkecil (*Unified Least Square Method*) 7

II.2.1 *Adjustment of Indirect Observations* 7

II.2.2 *Adjustment of Observations Only* 8

II.3 *Conventional Bundle Adjustment* 9

II.3.1 Persamaan Kolinear 9

II.3.1.1 Matrik Rotasi 10

II.3.1.2 Transformasi Perspektif Pusat (*Central Perspective Transformation*) 13

II.3.1.3 Kondisi Kolinear 16

II.3.2 Persamaan Observasi 17

II.3.3 Matrik Bobot Observasi 17

II.3.4 Model Matematika 19

II.3.5 Linierisasi Persamaan Kolinier 20

II.3.6 Desain Matrik untuk *Bundle* Adjustment 21

II.3.7 Struktur dari Matrik Persamaan Normal 24

II.3.8 Penyelesaian Persamaan Normal *Bundle Adjustment* 28

II.4 Model Penilaian *Bundle Adjustment* 29

II.5 Pembuatan Algoritma untuk Program Perhitungan *Bundle Adjustment* 30

II.6 Pemrograman Menggunakan Bahasa C# 31

II.6.1 Karakteristik Bahasa Pemrograman C# 33
II.6.2 Tipe Data Dalam Bahasa Pemrograman C# 34
II.6.3 Definisi Dan Deklarasi Metode *Main( )* 35
II.6.4 Definisi Dan Deklarasi *Class* 35
II.6.5 Definisi Dan Deklarasi *Array* 36
II.6.6 *Build* Dan *Debug* 36

BAB III PELAKSANAAN PENELITIAN 38

III.1 Persiapan 38
III.1.1 Materi Penelitian 38
III.1.2 Alat Penelitian 39
III.2 Langkah Penelitian 40
III.3 Diagram Alir Program 46
III.4 Pembuatan Program *Bundle Adjustment* 49

BAB IV HASIL DAN ANALISA 57

IV.1 Hasil 57
IV.2 Analisa Output Program 62
IV.3 Pembahasan 63

MILIK
PERPUSTAKAAN
ITN MALANG

BAB V PENUTUP 66

V.1 Kesimpulan 66

V.2 Saran 66

DAFTAR PUSTAKA

LAMPIRAN

BAB I

PENDAHULUAN

I.1 Latar Belakang

Perkembangan teknologi dan dunia digital yang sangat cepat sehingga menuntut orang untuk memperoleh informasi dengan cepat. Perkembangan teknologi dan dunia digital merambah hingga cabang keilmuan bidang geodesi dengan adanya perkembangan sensor seperti kamera dan *scanner*. Salah satu dari dampak perkembangan tersebut adalah pada bidang fotogrametri. Perkembangan di bidang fotogrametri sangatlah mempengaruhi dalam sistem pengukuran. Sistem pengukuran tersebut dapat dilakukan dengan cepat, efisien dan ekonomis. Sehingga pemetaan menggunakan kamera non-metrik semakin cepat berkembang seiring perkembangan teknologi yang ada.

Pada kamera non-metrik, titik pusat fidusial foto elemen dari *principle point* (x_0, y_0) dan *principal distance* (f) harus ditentukan, hal ini dikarenakan semua sistem persamaan matematis yang digunakan dalam fotogrametri bergantung dari ketiga parameter ini (*Wolf dan Dewitt, 2000*). Data yang diperoleh tidak lepas dari bermacam-macam keselahan sehingga perlu dilakukan perhitungan *bundle adjustment* untuk meratakan data yang diperoleh.

Bundle adjustment merupakan suatu bagian penting dalam *photogrammetry* (*Mikhail et al., 2001*). Perhitungan *Bundle Adjustment* dalam *Photogrammetry* tidak terlepas dari proses *resection* dan *intersection* (*Mikhail et al., 2001*). Proses *Resection* merupakan proses penentuan posisi dan orientasi luar dalam tiap foto

sedangkan proses *Intersection* merupakan teknik untuk menentukan koordinat titik-titik obyek pada dua buah foto atau lebih yang saling bertampalan sehingga dapat diketahui posisi secara 3 dimensi (*Mikhail et al., 2001*).

Proses *bundle adjustment* sangat dibutuhkan untuk memperoleh hasil *Xi,Yi,Zi* yang akurat dan parameter orientasi kamera yang teliti (*Mikhail et al., 2001*). Melihat peranan penting dari *bundle adjustment* dalam mendukung proses *photogrammetry*, maka penelitian ini akan membahas tentang Pembuatan *Program Bundle Adjustment Multi Photo Konvergen* dengan *Bahasa C#*.

I.2 Rumusan Masalah

Rumusan masalah yang akan dibahas dalam skripsi ini adalah bagaimana proses pembuatan *program C# bundle adjustment multi photo* dalam *photogrammetry*, tanpa melibatkan parameter kalibrasi kamera?

I.3 Tujuan Penelitian

Maksud dan tujuan dari penelitian skripsi ini adalah pembuatan program *bundle adjustment* dalam *photogrammetry*, untuk jaringan pemotretan *multi photo* yang *konvergen* sehingga memperoleh parameter *exterior orientation* yang teratakan serta koordinat obyek yang fix.

I.4 Batasan Masalah

Batasan masalah penelitian ini yaitu membuat program *bundle adjustment multi photo* dengan parameter *exterior orientation* pendekatan dan *interior orientation* yang telah diketahui, tanpa melibatkan parameter kalibrasi.

I.5 Tinjauan Pustaka

Beberapa tinjauan pustaka telah dilakukan dalam menyusun penelitian, guna mengumpulkan informasi mengenai *bundle adjustment* dalam *close range photogrametry* yang didasarkan atas berbagai riset oleh para ilmuwan dalam bidang *photogrametry* antara lain:

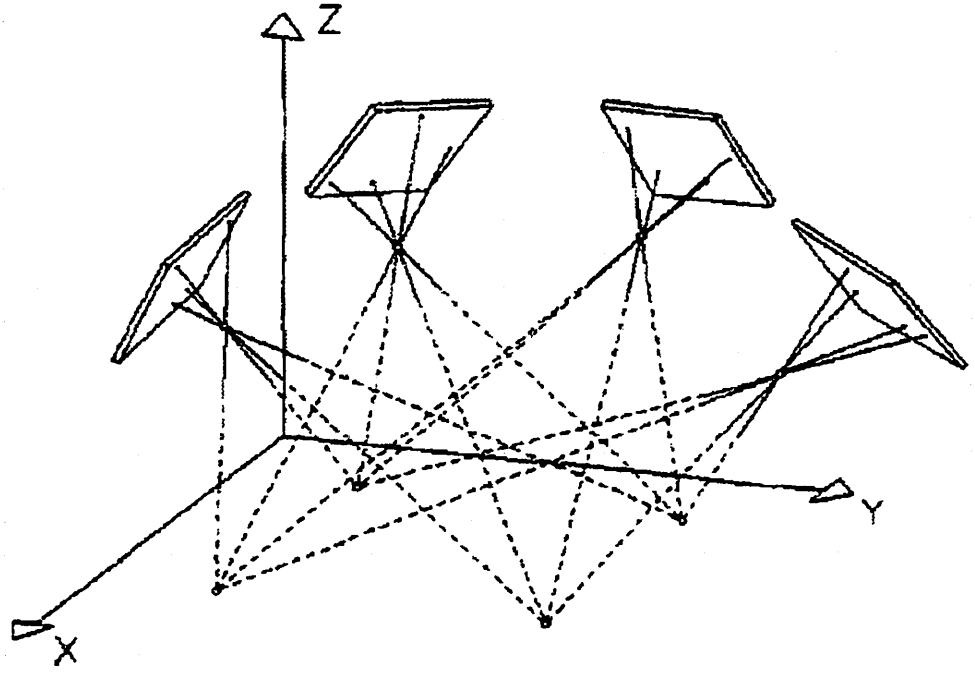
Bundle adjustment merupakan aplikasi yang paling sering digunakan dalam teori penyesuaian kuadrat terkecil pada *photogrammetry* karena metode yang digunakan lebih akurat dan fleksibel (*Mikhail et al., 2001*). *Bundle adjustment* yang paling sering diimplementasikan dengan menggunakan kuadrat terkecil terpadu yang memungkinkan perhitungan dari deviasi standard yang realistis untuk setiap objek titik koordinat ruang (*Mikhail et al., 2001*). Proses *Resection* merupakan proses penentuan posisi dan orientasi luar dalam tiap foto dan proses *Intersection* merupakan teknik untuk menentukan koordinat titik-titik obyek pada dua buah foto atau lebih yang saling bertampalan sehingga dapat diketahui posisi secara 3 dimensi (*Mikhail et al., 2001*).

Parameter-parameter yang digunakan dalam proses *bundle adjustmnent* adalah parameter IO (*interior orientation)*, EO (*exterior orientation*), serta koordinat obyek pendekatan dari obyek yang diamati (*Mikhail et al., 2001*).

BAB II

DASAR TEORI

Fotogrametri dengan jaringan pemotretan konvergen memiliki beberapa keunggulan dibandingkan geometri pemotretan normal. Salah satunya adalah penurunan jumlah gambar yang diperlukan untuk mencover seluruh permukaan objek. Keuntungan ini menyederhanakan desain jaringan dan meningkatkan secara teoritis keakuratan metode dan dapat diandalkan (*Mason, 1994*).



Gambar 2.1 contoh konfigurasi jaringan pemotretan konvergen(Uffenkam, 1998)

II.1. *Bundle Adjustment* dan Kesalahan Observasi

Dalam penelitian ini, akan dibahas secara khusus mengenai teknik perhitungan *bundle adjustment* dalam *photogrammetry*. *Photogrammetry* tidak terlepas dari proses *resection* dan *intersection*. Proses *resection* merupakan proses

penentuan posisi dan orientasi luar dalam tiap foto dan proses *intersection* merupakan teknik untuk menentukan koordinat titik-titik obyek pada dua buah foto atau lebih yang saling bertampalan sehingga dapat diketahui posisi secara 3 dimensi (*Mikhail et al, 2001*). Operasi ini akan dikombinasikan dalam proses triangulasi atau *block adjustment*, dimana parameter orientasi foto dan koordinat obyek akan dihitung secara serempak.

Penyesuaian dalam kuadrat terkecil membentuk dasar dari beberapa algoritma yang digunakan dalam proses perhitungan *photogrammetry*. Parameter-parameter yang digunakan dalam proses *bundle adjustmnent* adalah parameter IO (*interior orientation)*, EO (*exterior orientation*), serta koordinat obyek pendekatan dari obyek yang diamati.

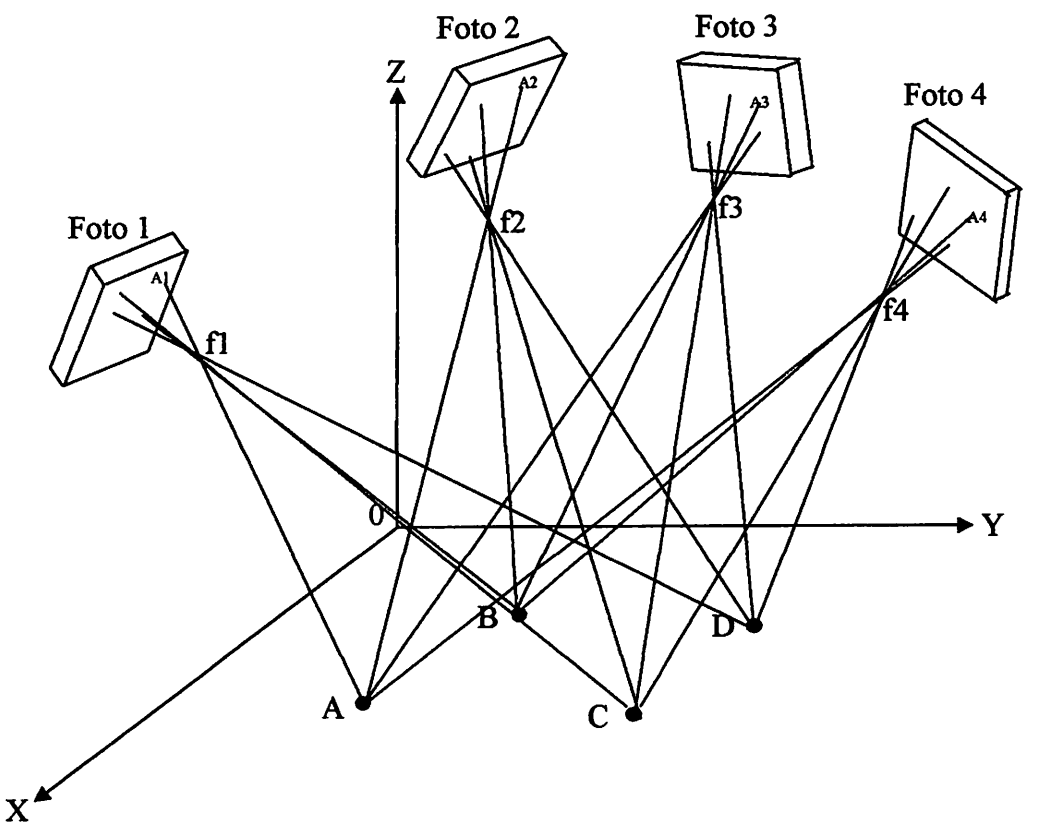
Mengenai alasan pemilihan *bundle adjustment* dalam perhitungan parameter-parameter *close range photogrammetry*, karena *bundle adjustment* (*Triggs et al, 2000*).

1. Fleksibilitas : *Bundle adjustment* dapat dengan baik memecahkan berbagai masalah antara lain berbagai bentuk 3 dimensi dan tipe kamera (titik, garis, kurva, surface, non-metrik, metrik, dsb), berbagai tipe skema (termasuk model dinamik dan artikulasi, skema constraints), sumber informasi (fitur 2 dimensi, tingkat intensitas, informasi 3 dimensi) dan *error models*.
2. Akurasi : *Bundle Adjusment* memberikan hasil yang presisi dan mudah dalam interpretasinya karena menggunakan model kesalahan statistik secara akurat.

3. Efisiensi : Algoritma *bundle* yang telah matang dan relatif efisien meskipun pada permasalahan yang sangat besar.

Dalam melakukan observasi maupun perhitungan, dapat terjadi tiga jenis kesalahan (*error*), yaitu *gross error*, *systematic error*, dan *random error*. Ukuran dari masing-masing kesalahan tersebut akan mempengaruhi keakuratan dan presisi dari parameter yang dihitung *(Wolf dan Ghilani, 1997)*.

Bundle adjustment merupakan suatu bentuk penyesuaian yang ditujukan untuk melakukan estimasi secara simultan dalam penentuan nilai parameter orientasi dari koordinat obyek yang diinginkan. Proses tersebut meliputi penempatan foto-foto secara bebas dalam *object space* membentuk jaringan *photogrammetry*, seperti ditujukan pada gambar dibawah ini *(Uffenkam, 1998)* :



Gambar 2.2 Proses Bundle Multi Foto

Keterangan gambar:

f1, f2, f3, f4 = Fokus

A, B, C, D = Titik koordinat obyek

A1,A2,A3,A4 = Tampilan titik A di tiap-tiap foto

II.2 Metode Gabungan Kuadrat Terkecil (*Unified Least Square Method*)

Metode gabungan kuadrat terkecil (*Unified Least Square Method*) mengkombinasikan dua jenis persamaan fungsional yang berbeda yang umum digunakan dalam penyesuaian kuadrat terkecil, yaitu persamaan observasi (*observation equation*) dan persamaan kondisi (*condition equation*). Kedua tipe persamaan tersebut telah digunakan untuk merumuskan metode pemisahan kuadrat terkecil (*King, 1993*). Mikhail menyatakan bahwa terdapat dua metode dalam penyesuaian, yaitu *Adjustment of Indirect Observations* dan *Adjustment of Observations Only.*

MILIK
PERPUSTAKAAN
ITN MALANG

II.2.1 *Adjustment of Indirect Observations*

Persamaan dengan menggunakan metode ini mengandung observasi, konstanta dan parameter. Parameter merupakan sebuah variabel *stochastic* yang nilainya perlu diestimasikan selama proses kuadrat terkecil. Masing-masing persamaan mengandung hanya satu observasi. Model fungsional penyesuaian dari observasi tidak langsung (*Adjustment of Indirect Observations*) sebagai berikut :

$$l + v + B\Delta = d \quad \text{.....(2.1)}$$

atau

$$v + B\Delta = -l + d = f \quad \text{.....} \quad \text{.....} \quad (2.2)$$

dimana l dan v merupakan vektor observasi dan residu, d merupakan vektor konstanta, B merupakan matrik parameter koefisien, dan Δ merupakan vektor parameter (*King, 1993*).

II.2.2 *Adjustment of Observations Only*

Dalam metode ini, kombinasi observasi dan residunya sama dengan *physical constant*nya. Model fungsionalnya sebagai berikut:

$$A(l + v) = d \quad \text{.....} \quad (2.3)$$

atau

$$Av = d - Al = f \quad \text{.....} \quad \text{.....} \quad (2.4)$$

Dimana A merupakan matrik koefisien observasi, l dan v adalah vektor observasi dan residunya, d adalah vektor nilai konstan (*King, 1993*).

Dengan pendekatan gabungan kuadrat terkecil diasumsikan bahwa semua variabel yang terkadung dalam fungsi matematika ini merupakan observasi. Maka persamaanya sebagai berikut :

$$AV + B\Delta = f \quad \text{.....} \quad (2.5)$$

jika dikombinasikan dengan matrik bobot (W) akan menghasilkan persamaan sebagai berikut :

$$\begin{aligned} \Delta &= \left(B^T W B\right)^{-1} B^T W f \\ &= N^{-1} t \end{aligned} \quad \text{.....} \quad (2.6)$$

II.3 *Conventional Bundle Adjustment*

Conventional Bundle Adjustment, yang sering disebut dengan BA, merupakan bangun dasar dari *Least Square Bundle adjustment* yang dikembangkan oleh Brown pada tahun 1958 (*King, 1993*). Selain BA, terdapat beberapa model lain dalam *Bundle Adjustment*, seperti Model DLT, yang dikembangkan oleh Abdel-Aziz dan Karara serta Model Penyesuaian Observasi Tidak Langsung yang dikembangkan oleh Mikhail. Menurut hasil penelitian yang dilakukan oleh Ruther (1989), 80% dari program komputer yang menyediakan solusi untuk *photogrammetry* mengacu pada *bundle adjustment* (*King, 1993*). Hal ini menunjukkan bahwa *bundle adjustment* menjadi model perhitungan yang dominan diterapkan dalam ilmu *photogrammetry* dibandingkan dengan model lainnya.

Pada umumnya, *bundle adjustment* dipergunakan untuk menentukan parameter *interior orientation* (IO), *exterior orientation* (EO), dan koordinat objek yang dihitung secara bersamaan dengan menggunakan teknik hitung kuadrat terkecil (*Fraser, 1997*).

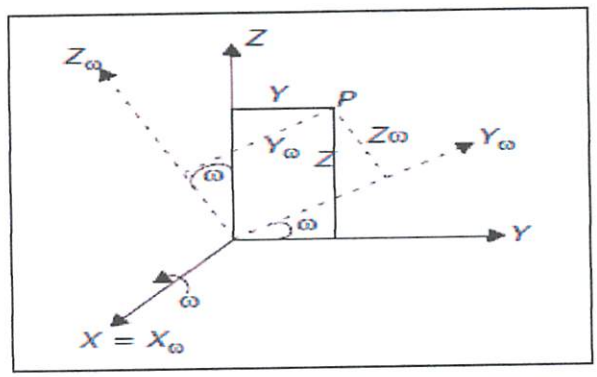
II.3.1 Persamaan Kolinear

Hubungan matematis antara titik konjugasi pada objek dengan titik konjugasi pada foto yang dijelaskan dengan transformasi projektif tiga dimensi disebut dengan persamaan kolinear. Disebut kolinear karena persamaan tersebut menunjukkan hubungan geometrik yang ideal antara sebuah titik objek, fotonya, dengan *imaging system perspective center.*

II.3.1.1 Matrik Rotasi

Rotasi matrik terdiri dari tiga parameter rotasi yaitu, omega (ω), phi (φ), dan kappa (κ). Dimana arah perputaran sumbu rotasi omega, phi dan kappa beserta sistem kaedah dan persamaannya dapat ditunjukkan pada gambar berikut :

1. Rotasi pertama yaitu omega (ω) dilakukan terhadap sumbu X_ω, dengan menggunakan sistem kaedah tangan kanan perputaran bernilai positif jika berlawanan arah jarum jam, seperti Gambar (2.3) (*Geosystem, 2006*):

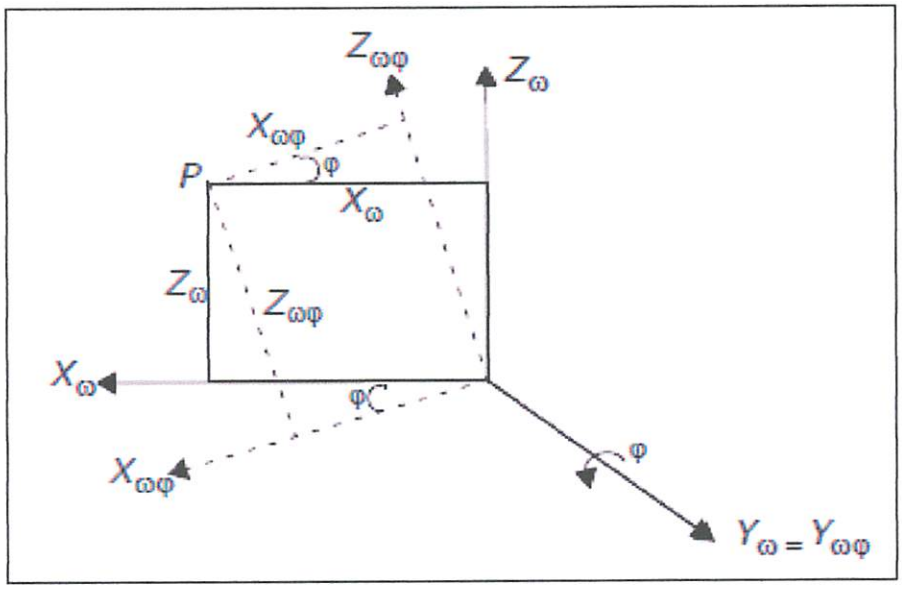


Gambar 2.3 Rotasi sudut omega terhadap sumbu X_ω

Dari Gambar (2.3) sistem rotasi $(X_\omega, Y_\omega, Z_\omega)$ dapat diperoleh persamaan vektornya $[X_\omega\ Y_\omega\ Z_\omega]^T = R_\omega [X\ Y\ Z]^T$, dimana (*Cooper dan Robson, 2001*):

$$R_\omega = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{bmatrix} \quad \text{.....(2.7)}$$

2. Rotasi kedua yaitu rotasi phi (φ) dilakukan terhadap sumbu Y_{φ}, dengan sistem kaedah tangan kanan perputaran bernilai positif jika berlawanan arah jarum jam seperti Gambar (2.4) (*Geosystem, 2006*).

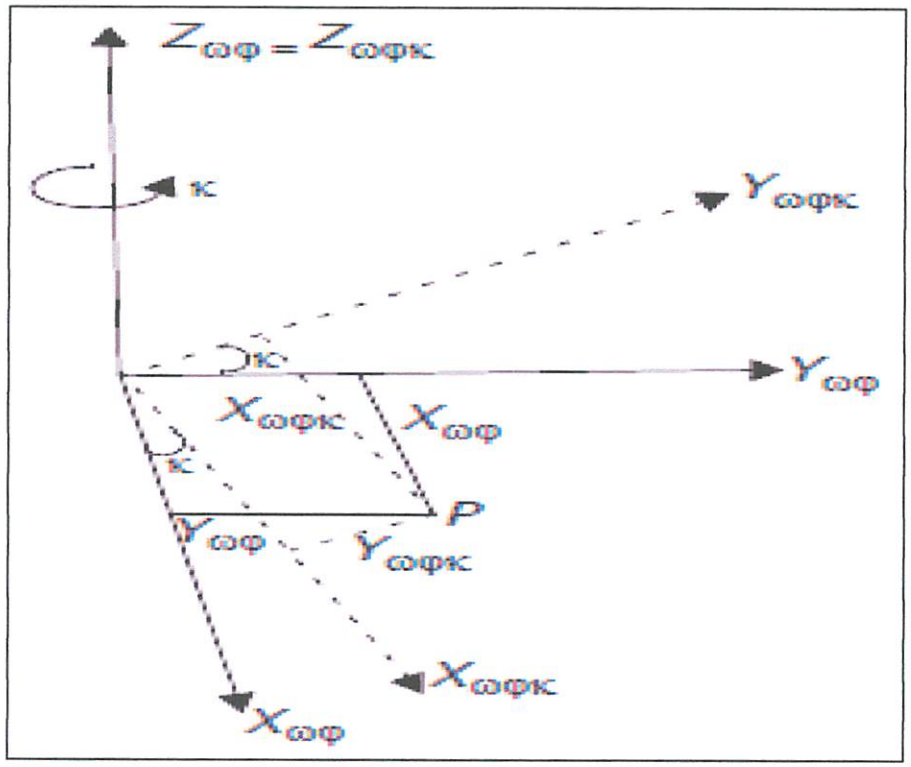


Gambar 2.4 Rotasi sudut phi terhadap sumbu Y_{φ}

Dari Gambar (2.4) sistem rotasi $(X_{\omega\varphi}, Y_{\omega\varphi}, Z_{\omega\varphi})$ dapat diperoleh vektornya $\left[X_{\omega\varphi}\, Y_{\omega\varphi}\, Z_{\omega\varphi}\right]^T = R_{\varphi}\left[X\, Y\, Z\right]^T$, dimana (*Cooper dan Robson, 2001*) :

$$R_{\varphi} = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix} \quad \text{.....(2.8)}$$

3. Rotasi ketiga yaitu rotasi kappa (κ) dilakukan terhadap sumbu $Z_{\omega\varphi}$, dengan sistem kaedah tangan kanan perputaran bernilai positif jika berlawanan arah jarum jam seperti gambar dibawah ini (*Geosystem, 2006*).



Gambar 2.5 Rotasi sudut kappa terhadapt sumbu $Z_{\omega\varphi}$

Dari Gambar (2.5) sistem rotasi ($X_{\omega\varphi\kappa}, Y_{\omega\varphi\kappa}, Z_{\omega\varphi\kappa}$) dapat diperoleh persamaan $\left[X_{\omega\varphi\kappa}\ Y_{\omega\varphi\kappa}\ Z_{\omega\varphi\kappa}\right]^T = R_\kappa \left[X\ Y\ Z\right]^T$ dimana (*Cooper dan Robson, 2001*) :

MILIK
PERPUSTAKAAN
ITN MALANG

$$R_\kappa = \begin{bmatrix} \cos\kappa & \sin\kappa & 0 \\ -\sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{.....(2.9)}$$

Dengan menggabungkan Persamaan (2.27, 2.28, 2.29) maka akan diperoleh persamaannya sebagai berikut (*Mikhail et al., 2001*):

$$R = R_\omega \, x \, R_\varphi \, x \, R_\kappa \quad \text{.....(2.10)}$$

dimana R merupakan matrik 3x3, dengan nilai elemen masing-masing matrik sebagai berikut :

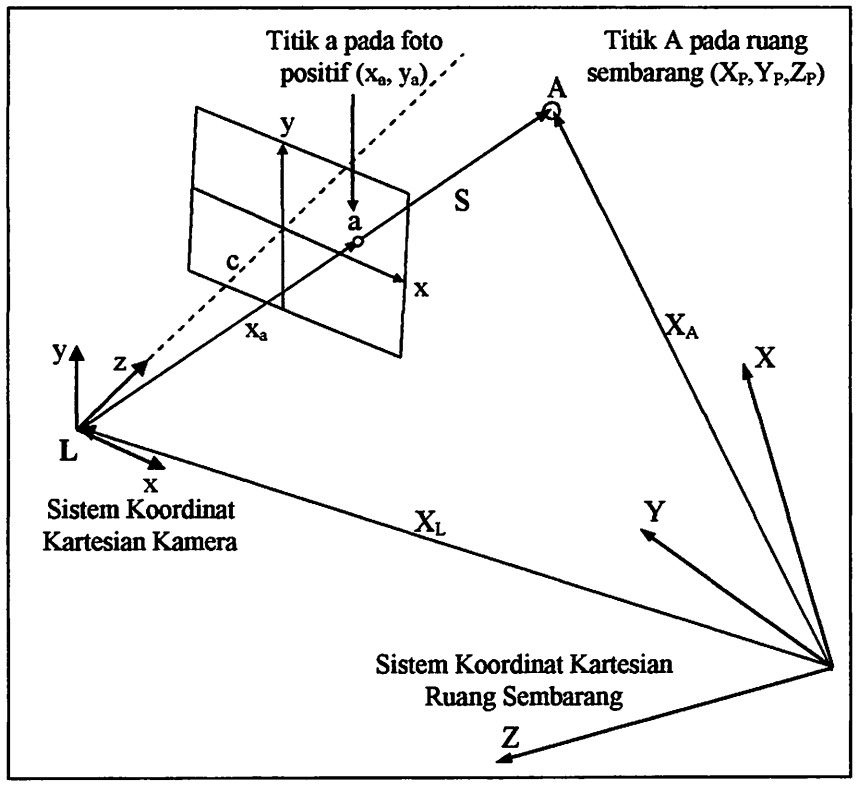
$$R=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \text{.....(2.11)}$$

dimana :

$$\begin{aligned} r_{11} &= \cos\varphi\cos\kappa \\ r_{12} &= \sin\omega\sin\varphi\cos\kappa+\cos\omega\sin\kappa \\ r_{13} &= -\cos\omega\sin\varphi\cos\kappa+\sin\omega\sin\kappa \\ r_{21} &= -\cos\varphi\sin\kappa \\ r_{22} &= -\sin\omega\sin\varphi\sin\kappa+\cos\omega\cos\kappa \\ r_{23} &= \cos\omega\sin\varphi\sin\kappa+\sin\omega\cos\kappa \\ r_{31} &= \sin\varphi \\ r_{32} &= -\sin\omega\cos\kappa \\ r_{33} &= \cos\omega\cos\varphi \end{aligned} \quad \text{.....(2.12)}$$

II.3.1.2 Transformasi Perspektif Pusat (*Central Perspective Transformation*)

Jika didalam kondisi yang ideal hubungan antara garis lurus Aa, sistem koordinat kamera dan sistem koordinat ruang dapat dijelaskan seperti Gambar (2.6), vektor S merupakan perpanjangan garis lurus Aa sebesar AL, dimana titik L berada pada bidang CCD/CMOS (*Cooper dan Robson, 2001*).



Gambar 2.6 Hubungan keruangan antara titik A sistem koordinat sembarang dengan titik sistem koordinat kamera

Berdasarkan hubungan antara vektor S, X_J dan X_L pada gambar 2.5, vektor posisi titik A pada sistem koordinat ruang dapat ditulis sebagai berikut :

$$X_A = X_L + S \quad \text{.....(2.13)}$$

Didalam proses pemotretan berkas sinar dari A ke L merupakan garis lurus (*Mikhail et al, 2001*), maka vektor S pada sistem koordinat kamera haruslah dikonversikan ke dalam sistem koordinat ruang dan dikalikan faktor skala perbesaran (*Cooper dan Robson, 2001*) maka persamaannya menjadi :

$$X_A = X_L + \lambda R^T x_a \quad \text{..... (2.14)}$$

atau jika dinyatakan dengan notasi matrik akan setara dengan :

$$\begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} = \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} + \lambda \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ -c \end{bmatrix} \quad \text{.....(2.15)}$$

dimana λ adalah faktor skala, x adalah vektor komponen titik a di dalam sistem koordinat kamera; dan X adalah vektor komponen titik A didalam sistem koordinat ruang (X_A, Y_A, Z_A) adalah koordinat titik A dan sistem ruang (X_L, Y_L, Z_L) merupaka koordinat titik L (pusat perspektif kamera) pada sistem ruang. Sedangkan R merupakan matrik rotasi dengan sudut perputaran omega, phi, dan kappa (ω, φ, dan κ) (*Mikhail et al., 2001*).

Dari Persamaan (2.14) dan (2.15) adalah sistem koordinat dari sistem koordinat kamera ke sistem koordinat ruang. Jika arah transformasi dibalik menjadi sistem koordinat ruang ke sistem koordinat kamera, maka persamaan (2.14) menjadi :

$$x_a = \lambda^{-1} R(X_A - XL) \quad \text{.....(2.16)}$$

atau jika dinyatakan dengan notasi matrik untuk titik sembarang akan setara dengan :

$$\begin{bmatrix} x - x_O \\ y - y_O \\ -c \end{bmatrix} = \lambda^{-1} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_A - X_L \\ Y_A - Y_L \\ Z_A - Z_L \end{bmatrix} \quad \text{.....(2.17)}$$

Dimana (x, y, $-c$) adalah nilai koordinat pada sistem kamera dan (X_A,Y_A,Z_A) adalah koordinat sistem ruang, serta (xo, yo) adalah koordinat titik tengah foto (*principal point*), dimana *principal point* merupakan nilai titik tengah *pixel* pada bidang CCD/CMOS yang tidak selalu berimpit dengan titik tengah foto (*Wolf dan Dewitt, 2000*).

II.3.1.3 Kondisi Kolinear

Dari Persamaan (2.17), karena faktor skala tidak dibutuhkan didalam pembahasan selanjutnya maka faktor skala ini dieliminir dengan membagi elemen baris pertama dan kedua dibagi dengan elemen baris ketiga pada persamaan (2.15), maka persamaannya akan menjadi (*Mikhail et al., 2001*):

$$x_a = x_o - c\left[\frac{r_{11}(X_A - X_L) + r_{12}(Y_A - Y_L) + r_{13}(Z_A - Z_L)}{r_{31}(X_A - X_L) + r_{32}(Y_A - Y_L) + r_{33}(Z_A - Z_L)}\right]$$
$$y_a = y_o - c\left[\frac{r_{21}(X_A - X_L) + r_{22}(Y_A - Y_L) + r_{23}(Z_A - Z_L)}{r_{31}(X_A - X_L) + r_{32}(Y_A - Y_L) + r_{33}(Z_A - Z_L)}\right] \quad \text{....(2.18)}$$

Persamaan ini merupakan persamaan garis lurus (persamaan kolinier) antara titik AaL seperti Gambar (2.6). Didalam Persamaan (2.18), jika titik-titik koordinat pada foto digital diukur (ditentukan nilainya) dan disimbolkan dengan $\underline{x}$, maka yang menjadi parameter (*unknowns*) adalah x_o dan y_o yang dikenal sebagai parameter orientasi dalam kamera; sedangkan parameter orientasi kamera ($\underline{O}$) atau orientasi dan posisi kamera dikenal ω, φ, κ, X_L, Y_L, Z_L; serta X, Y, Z yang merupakan titik-titik koordinat objek yang terekam pada saat pemotretan yang disimbolkan sebagai $\underline{X}$ (*Wolf dan Dewitt, 2000*).

Jika penulisan Persamaan (2.18) disederhanakan menjadi (*Fraser, 1997*) :

$$f(\underline{x}, \underline{O}, \underline{X})_{ij} = 0 \quad \text{.....(2.19)}$$

Keterangan :

i, j = foto yang ke-i dan titik objek yang ke-j yang terekam oleh kamera.

$\underline{x}$ =vektor yang berisikan nilai-nilai koordinat kamera untuk titik-titik obyek pada foto ke-i.

$\underline{O}$ = parameter orientasi luar kamera $(\omega_i, \varphi_i, \kappa_i, XL_i, YL_i, ZL_i)$ untuk foto ke-i

$\underline{X}$= Koordinat ruang untuk titik yang ke-j (X_j, Y_j, Z_j)

II.3.2 Persamaan Observasi

Metode gabungan kuadrat terkecil mensyaratkan agar persamaan dituliskan untuk masing-masing observasi yang secara fisik dibuat dan untuk masing-masing parameter yang digunakan dalam model fungsional. Persamaan ini disebut dengan persamaan observasi. Persamaan observasi dikembangkan dalam tiga kelompok variabel, yaitu (*King, 1993*) :

1. Titik koordinat foto
2. Parameter eksterior orientasi kamera
3. Titik koordinat objek

MILIK
PERPUSTAKAAN
ITN MALANG

II.3.3 Matrik Bobot Observasi

Salah satu keuntungan dari penerapan metode gabungan kuadrat terkecil adalah memungkinkan untuk menggabungkan parameter dan observasi yang digunakan dalam proses penyesuaian dengan mempertimbangkan bobot yang sesuai. Parameter atau observasi yang

nilainya reliabel dapat digabungkan dengan bobot yang tinggi dengan sebaliknya nilai yang tidak reliabel digabungkan dengan bobot yang rendah (*King, 1993*).

Bobot dari sebuah observasi bernilai proporsional dengan varian observasi (σ^2). Varian utama untuk tiap bobot dilambangkan dengan (σ_o^2). Bobot dari observasi dinyatakan sebagai berikut (*King, 1993*) :

$$W = \sigma_0^2 / \sigma^2 \quad \text{.....(2.20)}$$

Matrik bobot untuk *plate coordinates observation* dalam *i* pada foto *j* sebagai berikut (*Fraser, 1997*) :

$$W_{ij} = \begin{pmatrix} 1/\sigma_x^2 & 0 \\ 0 & 1/\sigma_y^2 \end{pmatrix}_{ij} \quad \text{.....(2.21)}$$

Pembahasan mengenai matrik bobot untuk kamera dan objek sebagai berikut (*King, 1993*).

Matrik bobot diagonal untuk kamera *j* sebagai berikut :

$$W_j = \begin{bmatrix} 1/\sigma_\omega^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/\sigma_\varphi^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/\sigma_\kappa^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/\sigma_{XL}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/\sigma_{YL}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/\sigma_{ZL}^2 \end{bmatrix} \quad \text{.....(2.22)}$$

untuk keseluruhan *n* foto sebagai berikut :

$$W = \begin{bmatrix} W_1 & & & \\ & \ddots & & \\ & & W_j & \\ & & & \ddots \\ & & & W_n \end{bmatrix} \quad \text{.....(2.23)}$$

Matrik diagonal untuk titik objek *i* sebagai berikut :

$$W = \begin{bmatrix} 1/\sigma_X^2 & 0 & 0 \\ 0 & 1/\sigma_Y^2 & 0 \\ 0 & 0 & 1/\sigma_Z^2 \end{bmatrix} \quad \text{.....(2.24)}$$

dan untuk keseluruhan *m* titik objek sebagai berikut :

$$W = \begin{bmatrix} W_1 & & & \\ & \ddots & & \\ & & W_j & \\ & & & \ddots \\ & & & W_m \end{bmatrix} \quad \text{.....(2.25)}$$

II.3.4 Model Matematika

Untuk *m* titik didalam *n* foto dengan mengacu pada persamaan observasi digabungkan untuk membentuk persamaan pengamatan sebagai berikut (*King, 1993*) :

$$V + B\Delta = f \quad \text{.....(2.26)}$$

Dimana Δ merupakan matrik koreksi parameter yang dicari, *V* adalah matrik residu, *B* merupakan matrik koefisien dan *f* yaitu matrik observasi. Jika sekumpulan persamaan dikombinasikan dengan persamaan matrik bobotnya (*W*), solusi kuadrat terkecilnya sebagai berikut (*King, 1993*) :

$$\Delta = \left(B^T WB\right)^{-1} B^T Wf$$
$$= N^{-1} t \quad \text{.....(2.27)}$$

II.3.5 Linierisasi Persamaan Kolinier

Persamaan dasar dari *bundle adjustment* adalah persamaan kolinear, yang mendiskripsikan satuan dasar dari *photogrammetry*. Dengan menggunakan persamaan kolinear membuat bentuk persamaan dan proses penyelesaiannya akan lebih efisien (*Mikhail et al., 2001*). Bentuk linier dari persamaan kolinear untuk foto *i* dan titik *j*. Karena persamaan pengamatan garis lurus pada Persamaan (2.19) tidak linier, maka untuk menghitung parameter didalamnya harus dilinierkan terlebih dahulu dengan menggunakan deret Taylor sebagai berikut (*Stewart, 1999*):

$$\left(df/d\underline{x}.dx\right)_{ij} + \left(df/d\underline{O}.d\underline{O}\right)_{ij} + \left(df/d\underline{x}.d\underline{X}\right)_{ij} + f\left(\underline{x}, \underline{O}^0, \underline{X}^0\right)_{ij} = 0$$
.....(2.28)

Bentuk linier dari persamaan pada Persamaan (2.28) ini setara dengan persamaan hitung kuadrat terkecil (*least square observation*) menurut Mikhail et al. (2001) yaitu :

$$\underset{(2,1)}{v} + \underset{(2,6)(6,1)}{B_1}\ \underset{}{\delta_1} + \underset{(2,3)(3,1)}{B_2}\ \underset{}{\delta_2} = \underset{(2,1)}{f} \quad \text{.....(2.29)}$$

dengan :

$$O = O^0 + \delta_1, \quad X = X^0 + \delta_2 \quad \text{.....(2.30)}$$

Keterangan :

$v = \left[v_x \quad v_y\right]^T_{ij}$ =Residu dari koordinat foto

$f = f\left(\underline{x}, \underline{O^0}, \underline{X^0}\right)_{ij}$ =Observasi

$\delta_1 = \left[d\omega_i \; d\varphi_i \; d\kappa_i \; dXL_i \; dYL_i \; dZL_i\right]^T$ =Koreksi parameter EO 6m vektor)

$\delta_2 = \left[dX_j, dY_j, dZ_j\right]^T$ =Koreksi Koordinat koreksi XYZ (3n vektor)

II.3.6 Desain Matrik untuk *Bundle Adjustment*

Menurut Mikhail et al. (2001) dan Wolf dan Dewitt (2000), jika sebuah titik objek (objek ke-j) terekam pada sebuah foto (foto ke-i), maka dimensi-dimensi matrik didalam Persamaan (2.29) adalah sebagai berikut :

$$\underset{(2,2)}{B_{ij}} = \begin{pmatrix} \frac{\partial f_x}{dx} & \frac{\partial f_y}{dy} \\ \frac{\partial f_x}{dx} & \frac{\partial f_y}{dy} \end{pmatrix}_{ij} \quad \text{.....(2.31)}$$

$$\underset{(2,1)}{v_{ij}} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}_{ij} \quad \text{.....(2.32)}$$

Susunan matrik B_1 sebagai berikut (*Fraser, 1997*):

$$\underset{(2,6)}{B_{1ij}} = \begin{bmatrix} \left(\frac{\partial f_x}{\partial \omega}\right)_0 & \left(\frac{\partial f_x}{\partial \varphi}\right)_0 & \left(\frac{\partial f_x}{\partial \kappa}\right)_0 & \left(\frac{\partial f_x}{\partial X_L}\right)_0 & \left(\frac{\partial f_x}{\partial Y_L}\right)_0 & \left(\frac{\partial f_x}{\partial Z_L}\right)_0 \\ \left(\frac{\partial f_y}{\partial \omega}\right)_0 & \left(\frac{\partial f_y}{\partial \varphi}\right)_0 & \left(\frac{\partial f_y}{\partial \kappa}\right)_0 & \left(\frac{\partial f_y}{\partial X_L}\right)_0 & \left(\frac{\partial f_y}{\partial Y_L}\right)_0 & \left(\frac{\partial f_y}{\partial Z_L}\right)_0 \end{bmatrix}_{ij}$$

.....(2.33)

atau secara sederhana sebagai berikut (*Wolf dan Dewitt, 2000*).

$$\underset{(2,6)}{B_{1ij}} = \begin{bmatrix} (b_{11})_0 & (b_{12})_0 & (b_{13})_0 & (-b_{14})_0 & (-b_{15})_0 & (-b_{16})_0 \\ (b_{21})_0 & (a_{22})_0 & (b_{23})_0 & (-b_{24})_0 & (-b_{25})_0 & (-b_{26})_0 \end{bmatrix}_{ij} \quad \text{....(2.34)}$$

Susunan matrik B_2 sebagai berikut (*Fraser, 1997*) :

$$B_{\substack{2ij \\ (2,3)}} = \begin{bmatrix} \left(\frac{\partial f_x}{\partial X_i}\right)_0 & \left(\frac{\partial f_x}{\partial X_i}\right)_0 & \left(\frac{\partial f_x}{\partial X_i}\right)_0 \\ \left(\frac{\partial f_x}{\partial X_i}\right)_0 & \left(\frac{\partial f_x}{\partial X_i}\right)_0 & \left(\frac{\partial f_x}{\partial X_i}\right)_0 \end{bmatrix}_{ij} \quad \text{.....(2.35)}$$

atau (*Wolf dan Dewitt, 2000*)

$$B_{\substack{2ij \\ (2,3)}} = \begin{bmatrix} (b_{14})_0 & (b_{15})_0 & (b_{16})_0 \\ (b_{24})_0 & (b_{25})_0 & (b_{26})_0 \end{bmatrix}_{ij} \quad \text{.....(2.36)}$$

B_1 mempunyai dimensi 2x6 sedangkan B_2 mempunyai dimensi 2x3 dimana untuk tiap titik objek ke-j yang terekam pada foto ke-i.

Susunan matrik f sebagai berikut (*Fraser, 1997*) :

$$f_{ij} = \begin{bmatrix} J \\ K \end{bmatrix}_{ij} \quad \text{.....(2.37)}$$

Penjelasan elemen matrik B dan matrik f (*Wolf and Dewitt, 2000*):

$$b_{11} = \frac{f}{q^2}[r(-m_{33}\Delta Y + m_{32}\Delta Z) - q(-m_{13}\Delta Y + m_{12}\Delta Z)]$$

$$b_{12} = \frac{f}{q^2}[r(\cos\phi\,\Delta X + \sin\omega\sin\phi\,\Delta Y - \cos\omega\sin\phi\,\Delta Z) -$$
$$q(-\sin\phi\cos\kappa\,\Delta X + \sin\omega\cos\phi\cos\kappa\,\Delta Y - \cos\omega\cos\phi\cos\kappa\,\Delta Z)]$$

$$b_{13} = \frac{-f}{q}(m_{21}\Delta X + m_{22}\Delta Y + m_{23}\Delta Z)$$

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

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

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

$$b_{21} = \frac{f}{q^2}[s(-m_{33}\Delta Y + m_{32}\Delta Z) - q(-m_{23}\Delta Y + m_{22}\Delta Z)]$$

$$b_{22} = \frac{f}{q^2}[s(\cos\phi\,\Delta X + \sin\omega\sin\phi\,\Delta Y - \cos\omega\sin\phi\,\Delta Z) -$$
$$q(-\sin\phi\cos\kappa\,\Delta X - \sin\omega\cos\phi\sin\kappa\,\Delta Y + \cos\omega\cos\phi\sin\kappa\,\Delta Z)]$$

$$b_{23} = \frac{f}{q}(m_{11}\Delta X + m_{12}\Delta Y + m_{13}\Delta Z)$$

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

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

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

$$J = x_a - x_o + f\frac{r}{q}$$
$$K = y_a - y_o + f\frac{s}{q} \quad \text{.....(2.38)}$$

Untuk matrik δ_{1i} dan δ_{2j} struktur matriknya sebagai berikut (*Mikhail et al. 2001*) :

$$\underset{(6m,1)}{\delta_{1i}} = \begin{bmatrix} \delta\omega \\ \delta\varphi \\ \delta\kappa \\ \delta X_L \\ \delta Y_L \\ \delta Z_L \end{bmatrix}_i \quad \text{.....(2.39)}$$

$$\underset{(3n,1)}{\delta_{2j}} = \begin{bmatrix} \delta X \\ \delta Y \\ \delta Z \end{bmatrix}_j \quad \text{.....(2.40)}$$

Dimana δ_1 besar dimensinya 6mx1 dan δ_2 besar dimensinya 3nx1

II.3.7 Struktur dari Matrik Persamaan Normal

Jika kita sekarang akan membentuk persamaan normal yang berhubungan dengan sebuah foto untuk satu titik, maka akan diperoleh persamaan sebagai berikut (*Mikhail et al., 2001*) :

$$\begin{bmatrix} B_{1ij}^T W_{ij} B_{1ij} & B_{1ij}^T W_{ij} B_{2ij} \\ B_{2ij}^T W_{ij} B_{1ij} & B_{2ij}^T W_{ij} B_{2ij} \end{bmatrix} \begin{bmatrix} \dot{\delta}_{1i} \\ \ddot{\delta}_{2j} \end{bmatrix} = \begin{bmatrix} B_{1ij}^T W_{ij} f_{ij} \\ B_{2ij}^T W_{ij} f_{ij} \end{bmatrix} \quad \text{.....(2.41)}$$

dimana W_{ij} merupakan matrik bobot 2x2 (*inverse* dari matrik *covarian* dari ketelitian pengukuran foto digital) yang berkaitan dengan koordinat foto dari titik j pada foto i. Persamaan tersebut juga dapat ditulis sebagai berikut :

$$\begin{bmatrix} \dot{N}_i & \overline{N}_{ij} \\ \overline{N}_{ij}^T & \ddot{N}_j \end{bmatrix} \begin{bmatrix} \dot{\delta}_i \\ \ddot{\delta}_j \end{bmatrix} = \begin{bmatrix} \dot{t}_i \\ \ddot{t}_j \end{bmatrix} \quad \text{atau} \quad N\delta = t \quad \text{.....(2.42)}$$

Perlu diperhatikan bahwa $\dot{N}$ mengandung koefisien dari parameter foto, $\ddot{N}$ mengandung koefisien parameter titik (koordinat), $\overline{N}$ mengacu pada keduanya.

Kita sekarang membuat asumsi penting bahwa kesalahan dari tiap-tiap titik pada koordinat foto tidak berhubungan dengan kesalahan pada koordinat foto lainnya, baik titik-titik pada foto yang sama maupun foto yang berbeda untuk titik yang sama, meskipun koordinat x dan y pada suatu foto mungkin saja memiliki hubungan.

Bundle adjustment pada umumnya diterapkan dengan menggunakan gabungan kuadrat terkecil, yang tidak mengubah bentuk dasar algoritma secara signifikan. Matrik bobot dari observasi parameter untuk setiap foto

dan titik. W_i dan W_j ditambahkan pada masing-masing N_i dan N_j (*Mikhail et al., 2001*).

Dari pengembangan persamaan kolinier (2.28), maka akan dapat dibentuk persamaan yang elemen di dalamnya didefinisikan sebagai berikut: $\underset{(6x6)}{\dot{N}_{ij}} = \sum_{j=1}^{n} B_{1ij}^{T} W_{ij} B_{1ij}$; $\underset{(6x1)}{\dot{t}_i} = \sum_{j=1}^{n} B_{1ij}^{T} W_{ij} f_{ij}$

$$\underset{(3x3)}{\ddot{N}_{ij}} = \sum_{i=1}^{m} B_{2ij}^{T} W_{ij} B_{2ij} \; ; \; \underset{(3x1)}{\ddot{t}_j} = \sum_{i=1}^{m} B_{2ij}^{T} W_{ij} f_{ij} \quad \text{.....(2.43)}$$

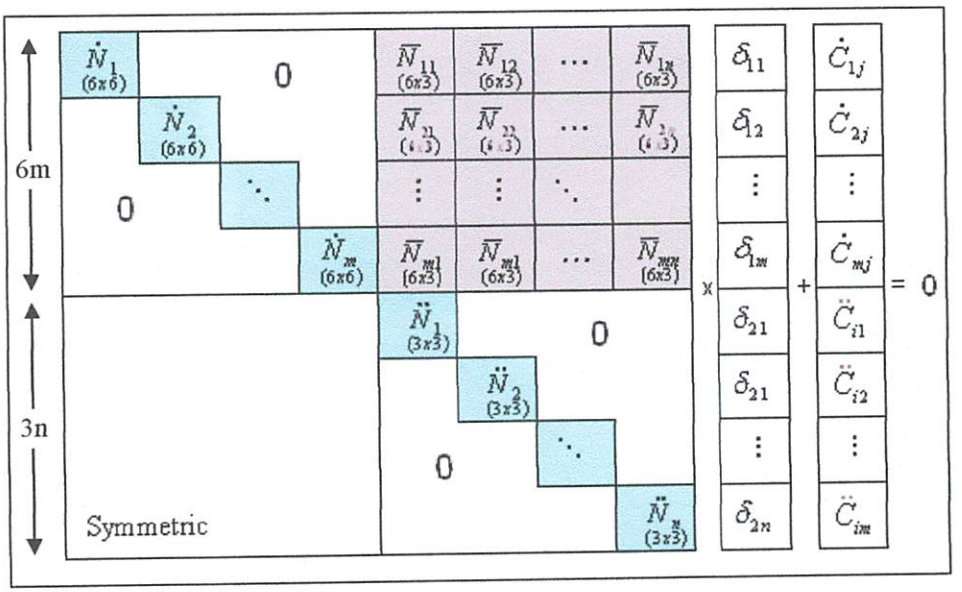
$$\underset{(6x3)}{\overline{N}_{ij}} = B_{1\ ij}^{T} W_{ij} B_{2ij}$$

Dimana $\dot{N}$ dan $\ddot{N}$ adalah sub matrik dari matrik blok-diagonal, dimana blok $\dot{N}$ mengacu pada parameter EO dan $\ddot{N}$ mengacu pada koordinat titik-titik obyek serta $\overline{N}$ mengacu pada keduanya.

Penyusunan matrik N dapat disajikan sebagai berikut dengan *m* banyak foto dan *n* banyak titik obyek :

$$\underset{(6m+3n)\,(6m+3n)}{N} = \left[\begin{array}{cccc|cccc} \dot{N}_{1j} & 0 & \cdots & 0 & \overline{N}_{11} & \overline{N}_{12} & \cdots & \overline{N}_{1n} \\ 0 & \dot{N}_{2j} & \cdots & 0 & \overline{N}_{21} & \overline{N}_{22} & \cdots & \overline{N}_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{N}_{mj} & \overline{N}_{m1} & \overline{N}_{m2} & \cdots & \overline{N}_{mn} \\ \hline \overline{N}_{11}^{T} & \overline{N}_{21}^{T} & \cdots & \overline{N}_{m1}^{T} & \ddot{N}_{i1} & 0 & \cdots & 0 \\ \overline{N}_{12}^{T} & \overline{N}_{22}^{T} & \cdots & \overline{N}_{m2}^{T} & 0 & \ddot{N}_{i1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{N}_{1n}^{T} & \overline{N}_{2n}^{T} & \cdots & \overline{N}_{mn}^{T} & 0 & 0 & \cdots & \ddot{N}_{i1} \end{array}\right] \quad \text{....(2.44)}$$

Struktur persamaan normal *bundle adjustment* juga dapat disusun seperti Gambar (2.7) sebagai berikut :



Gambar 2.7 Struktur Persamaan Normal

dimana :

$$\underset{(6m+3n),1}{\delta} = \begin{bmatrix} \delta_{11} \\ \delta_{12} \\ \vdots \\ \delta_{1m} \\ \delta_{21} \\ \delta_{21} \\ \vdots \\ \delta_{2n} \end{bmatrix}; \qquad \underset{(6m+3n),1}{C} = \begin{bmatrix} \sum_{j=1}^{n} \dot{C}_{1j} \\ \sum_{j=1}^{n} \dot{C}_{2j} \\ \vdots \\ \sum_{j=1}^{n} \dot{C}_{mj} \\ \sum_{i=1}^{m} \ddot{C}_{i1} \\ \sum_{i=1}^{m} \ddot{C}_{i2} \\ \vdots \\ \sum_{i=1}^{m} \ddot{C}_{im} \end{bmatrix} \quad \text{..... (2.45 dan 2.46)}$$

Persamaan diatas adalah teknik *Bundle Adjustment Normal Equation* untuk mendapatkan nilai parameter dan koordinat titik objek didalam sistem kartesian 3D.

Solusi hitung kuadrat terkecil dapat ditentukan dengan menggunakan matriks *inverse Cayley N-1*. Tetapi matrik *N* memiliki *rank defect* sebesar parameter penentuan datum (*Cooper and Cross, 1988; 1991*), yaitu 7. *Singularitas* matrik *N* ini dapat diselesaikan dengan menambahkan parameter datum secara implisit untuk membentuk persamaan normal yang non-singular seperti:

$$\begin{bmatrix} \dot{N}_{ij} & \overline{N}_{ij} & 0 \\ \overline{N}_{ij}^{T} & \dot{N}_{ij} & G \\ 0 & G^{T} & 0 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ k \end{bmatrix} + \begin{bmatrix} \dot{C}_i \\ \ddot{C}_j \\ 0 \end{bmatrix} = 0 \quad \text{.....(2.47)}$$

Dimana *G* adalah matrik transformasi *Helmert* dan *k* adalah faktor pengali *Lagrangian*. Matrik *G* didefinisikan sebagai (*Fraser, 1997*) :

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & Z_j & -Y_j & X_j \\ 0 & 1 & 0 & -Z_j & 0 & X_j & Y_j \\ 0 & 0 & 1 & Y_j & -X_j & 0 & Z_j \end{pmatrix} \quad \text{.....(2.48)}$$

Teknik hitungan yang dipakai untuk menyelesaikan Persamaan (2.47) ada beberapa macam, seperti misalnya teknik *minimum constraint, S-transformation, Pseudo Inverse, dan Free-Net Adjustment (Tjahjadi, 2008b)*. Tetapi, teknik yang paling sesuai didalam kasus ini adalah teknik yang dapat mengoptimalkan tingkat keakurasian koordinat titik-titik obyek. Dengan kata lain, harus dipilih suatu teknik yang dapat meminimumkan matrik kovarian dari titik-titik objek.

II.3.8 Penyelesaian Persamaan Normal *Bundle Adjustment*

Untuk penyelesaian persamaan normal diatas dapat diselesaikan sebagai berikut (*Fraser, 1997*) :

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = -\begin{bmatrix} A_1^T PA_1 & A_1^T PA_2 \\ A_2^T PA_1 & A_2^T PA_2 \end{bmatrix}^{-1} \begin{bmatrix} A_1^T Pw \\ A_2^T Pw \end{bmatrix} \quad \text{.....(2.47)}$$

Penyelesaian dari teknik *Bundle Adjustment* bertujuan untuk mendapatkan nilai parameter EO dan koordinat titik objek didalam sistem kartesian 3D.

$O = O^0 + \delta_1$: (Parameter orientasi luar (EO) yang terkoreksi)

$X = X^0 + \delta_2$: (Koordinat titik objek yang terkoreksi)

Jika titik-titik objek ini hendak dihitung dengan tingkat keakurasian yang lebih tinggi lagi, maka kesalahan sistematis kamera harus dimodelkan.

Jumlah persamaan yang digunakan untuk (3n + 6m) parameter adalah (2mn + 7). Kondisi berikut harus dipenuhi untuk memungkinkan penyesuaian bundel untuk bekerja (*wang dan Clarke, 1998*):

$$2mn + 7 \geq 3n + 6m \quad \text{.....(2.48)}$$

Minimal empat titik obyek yang diperlukan pada umumnya (ketika $m \geq 3$) dan minimal lima titik obyek diperlukan jika hanya dua foto yang digunakan. Minimal dua kamera yang diperlukan pada umumnya (jika $n \geq 5$) dan minimal tiga kamera yang diperlukan jika titik pada obyek yang satu terlibat. Namun, sebenarnya tiga titik obyek yang cukup untuk menentukan parameter EO yang disediakan bahwa semua muncul pada

bidang gambar kamera dan dua kamera yang memadai untuk memecahkan koordinat 3D dari titik-titik obyek(*wang dan Clarke, 1998*).

n = Jumlah titik

m = Jumlah foto

MILIK
PERPUSTAKAAN
ITN MALANG

II.4 Model Penilaian *Bundle Adjustment*

Kinerja dari *bundle adjustment* dinilai dengan beberapa kriteria, antara lain tingkat akurasi dan presisinya. Penilaian keakuratan *bundle adjustment* pada umumnya dilakukan dengan cara membandingkan koordinat titik objek yang dihitung dari *bundle adjustment* dengan nilai yang diperoleh dari hasil pengamatan (*King, 1993*).

Kita dapat menilai ketepatan atau presisi suatu perhitungan dengan melihat nilai kovarian dari parameternya. Varian dari satu parameter dapat dikatakan sebagai sebaran dari nilai parameter itu. Semakin tinggi nilai varian dari parameter semakin tidak baik parameter tersebut, parameter yang sempurna akan mempunyai nilai varian nol. Sebuah persamaan kuadrat kecil akan menghasilkan varian dari parameternya. Varian ini sering disebut dengan standar deviasi, dimana standar deviasi merupakan varian yang berbentuk akar kuadrat dan memiliki satuan yang sama dengan parameternya (*Mikhail et al, 2001*).

Matriks kovarian merupakan hasil simpangan yang umumnya diperoleh dari *bundle adjustment* simultan secara langsung dan dalam waktu bersamaan saat perhitungan parameter yang tidak diketahui (*wang dan Clarke, 1998*).

II.5 Pembuatan Algoritma untuk Program Perhitungan *Bundle Adjustment*

Definisi algoritma secara umum adalah urutan langkah logis tertentu untuk memecahkan suatu masalah.Yang ditekankan adalah urutan langkah logis, yang berarti algoritma harus mengikuti suatu urutan tertentu, tidak boleh melompat-lompat. *(Microsoft Press Computer and Internet Dictionaary 1997, 1998)*

Dalam bidang komputer, algoritma sangat diperlukan dalam menyelesaikan berbagai masalah pemrograman, terutama dalam komputasi numeris. Tanpa algoritma yang dirancang baik maka proses pemrograman akan menjadi salah, rusak, atau lambat dan tidak efisien.

Berikut adalah Algoritma yang digunakan untuk menyusun program perhitungan *bundle adjustment* :

1. Input :
   a. Parameter Orientasi Luar $(\omega_i, \varphi_i, \kappa_i, XL_i, YL_i, ZL_i)$ pendekatan
   b. Koordinat *Object Space Point (X, Y, Z)* Pendekatan
   c. Koordinat foto (x,y) dan ketelitian (σx, σy) serta Parameter *Interior Orientation (IO)*

2. Proses
   a. Menghitung Matrik Rotasi (R)
   b. Menghitung Parameter *q, r, s*
   c. Menghitung Matrik *B1, B2, f, w*
   d. Menghitung Matrik $\dot{N}_{ij}, \ddot{N}_{ij}, \overline{N}_{ij}, \dot{C}i_j, \ddot{C}_{ij}, G$
   e. Menyusun Matrik N dan t

f. Menghitung matrik Koreksi (δ1 dan δ2)

g. Menghitung Nilai Residu (v)

h. Menghitung Nilai Akhir

i. Menghitung Standard Deviasi

3. Output

a. Parameter Orientasi Luar $(\omega_i, \varphi_i, \kappa_i, XL_i, YL_i, ZL_i)$ final

b. Koordinat *Object Space Point (X, Y, Z)* final

c. Ketelitian *Varian-Kovarian*

II.6 Pemrograman Menggunakan Bahasa C#

Bahasa pemrograman C# dikembangkan oleh Microsoft sebagai bahasa yang simple, modern, *general-purpose*, dan berorientasi obyek (*SmithDev, 2009*). Kehadiran C# memeberi suntikan optimisme bagi para programmer untuk dapat mengembangkan aplikasi yang bedaya guna dengan lebih cepat dan mudah. Bahasa *C#* merupakan *general-purpose language*, yaitu bahasa pemrograman yang dapat digunakan untuk tujuan apa saja Dengan bahasa *C#*, kita dapat membangun beragam aplikasi, mulai dari pemrograman sistem, aplikasi cerdas (*artificial intelligence*), sistem pakar, *utility*, *driver*, database, *browser*, *network programming*, *sistem operasi*, *game*, *virus*, dan lainnya (*SmithDev, 2009*).

C# memiliki kesamaan bahasa dengan *C*, *C++*, dan *Java*, sehingga memudahkan *developer* yang sudah terbiasa dengan bahasa *C* untuk menggunakannya, *C#* mengambil fitur-fitur terbaik dari ketiga bahasa tersebut dan juga menambahkan fitur-fitur baru. *C#* merupakan bahasa pemrograman

Object Oriented dan memiliki *class library* yang sangat lengkap yang berisi *pre-built* component sehingga memudahkan programer untuk men-*develop* program lebih cepat. *C#* juga distandarkan oleh *Ecma International* pada bulan desember 2002 (*Ridi Ferdiana, 2006*).

Bahasa pemrograman C# merupakan gabungan dari kecanggihan bahasa keluarga C (C, C++, Objective-C, Java, dan sebagainya) dengan kemudahan bahasa pemrograman Visual Basic. Dalam hal ini spesifikasi bahasa pemrograman C# pada awalnya ditulis oleh Anders Hejlsberg dan Scott Wiltamuth dari Microsoft Corp. (*Ridi Ferdiana, 2006*).

Pada akhir tahun 2005 *Microsot* merilis *.NET Framework* 2.0 bersamaan dengan paket *Visual* Studio. Otomatis versi dari *C#* juga diperbaharui menjadi *C# 2.0* yang berjalan diatas *.NET Framework 2.0*. Pada versi baru ini banyak sekali fitur-fitur yang ditambahkan terutama pada pengembangan aplikasi berbasis *web* dengan *ASP.NET* seperti (*master page, site map control, user login,* dll), juga penambahan *generic collection* yang sangat membantu programer bekerja dengan *object-object collection* dan *list* (*Budi Hartanto, 2008*).

Bahasa pemrograman C# dirancang oleh Microsoft Corp. sebagai bahasa pemrograman yang berdaya guna, aman serta mudah digunakan. Sebagai bagian dari platform .NET, bahasa pemrograman C# dirancang untuk bekerja sangat baik diatas framework .NET, yang mampu digunakan untuk menulis perangkat lunak handal demi layanan yang cepat. Bahasa pemrograman C# juga dapat digunakan untuk mengembangkan aplikasi-aplikasi sarana bergerak (*mobile application*),

aplikasi berbasis Web (*Web-based applications*), serta aplikasi berskala besar (*Enterprise*) (*Andrew Troelsen, 2007*).

II.6.1 Karakteristik Bahasa Pemrograman C#

Secara umum bahasa pemrograman C# memiliki karakteristik umum seperti tertulis dibawah ini (*Faraz Rasheed, dkk., 2006*):

1. Tidak ada alokasi memori secara manual menggunakan pointer (dalam hal ini mirip dengan bahasa pemrograman Java).
2. Manajemen memori otomatis menggunanakan salah satu fiturnya yang dinamakan *garbage collection* (hal ini juga mirip dengna bahasa Java).
3. Mendukung konstruksi kelas, antar muka, struktur, dan *enumerasi* seperti bahasa pemrograman berorientasi objek lainnya (misalnya C++ atau Java).
4. Mendukung LINQ (Language Integrated Query) yang memungkinkan aplikasi yang ditulis menggunakan bahasa pemrograman C# mampu berinteraksi dan bekerja sama dengan berbagai jenis format data dimana hal ini sangat penting saat kita membuat aplikasi bahasa C# yang mengakses sistem basis data relasional (RDBMS-*Relational Database Management System*).
5. Mendukung tipe data dan kelas generic (mirip dengan C++ dan Java).
6. Mendukung operator delegasi (= >).

II.6.2 Tipe Data Dalam Bahasa Pemrograman C#

Bahasa pemrograman C#, tentu saja berbasis pada framework .NET. Salah satu bagian dari framework .NET ini adalah berbagi CTS (*common type system*) yang ada di dalamnya (*Mark Horner, 2006*). CTS mendefinisikan bagaimana tipe-tipe data dideklarasikan dan digunakan dalam semua bahasa pemrograman yang tercakup dalam framework .NET.

Dalam bahasa pemrograman C#, semua tipe data merupakan bagian dari *namespase System*. Pada kenyataannya bahasa pemrograman C# mendukung dua jenis tipe data, yaitu tipe data nilai (*value data type*) dan tipe data rujukan (*refference data type*) (*Faraz Rasheed, 2006*) yang keduanya berbasis pada kelas dasar objek (*System.Object*).

Tipe data rujukan (*refference data type*) menyimpan objek-objek yang memiliki rujukan pada data actual dan mereka disimpan dalam *heap* yang lebih permanen dan memiliki ruang memori yang lebih besar (*Michael McMillan, 2006*). Tipe data dasar yang bersifat *deafault* adalah bertipe nilai (*value data type*) dan mereka disimpan dalam suatu stack dimemori komputer. Meski demikian, tentu saja kita dapat menyimpan tipe data dalam *heap* dan merujuknya sebagai suatu objek jika memang dikehendaki (*Faraz Rasheed, 2006*).

Bahasa pemrograman C# bekerja dengan peubah seperti bahasa pemrograman Java (*Andrew Troelsen, 2007*). Semua peubah harus didekalarasikan sebelum digunakan. Selain itu, juga mirip dengan bahasa pemrograman Java, peubah itu dapat diinisialisasi dengan nilai default saat

mereka dideklarasikan dan beberapa peubah yang memiliki tipe yang sama dapat dideklarasikan pada saat yang bersamaan (*Nugroho Adi, 2009*).

II.6.3 Definisi Dan Deklarasi Metode *Main( )*

Metode *Main( )* (dalam program kita tertulis sebagai *static void Main (string [] args)*) yang merupakan fungsi utama sebuah program C#. eksekusi program bahasa C# selalu dimulai dari metode *Main ( )* ini. Metode *Main ( )* tertulis sebagai *static* supaya bias dieksekusi oleh CLR tanpa perlu membuat objek program terlebih dahulu. Kata kunci *void* menunjukkan bahwa metode *Main ( )* tidak menghasilkan kembalian apapun. Selanjutnya, pernyaan *(string [] args)* akan digunakan saat kita melewatkan parameter tertentu pada metode *Main ( )* (*Nugroho Adi, 2009*).

II.6.4 Definisi Dan Deklarasi *Class*

Dalam platform .NET (karena .NET merupakan platform pemrograman yang berorientasi objek, bentukan yang paling mendasar dalam aplikasi adalah kelas, class (Faraz Rasheed, 2006). Secara formal, kelas merupaka tipe data bentukan pengguna (UDT atau ADT) yang di dalamnya memuat data (sering disebut sebagai atribut atau *member variable*) dan metode yang beroperasi pada data yang bersangkutan (misalnya *constructor*, *event*, *function*, dan sebagainya). Secara bersamaan, himpunan data merepresentasikan *state* dari objek bentukan dari kelas yang

bersangkutan. Di sini dapat disimpulkan bahwa kelas merupakan sejumlah objek yang masing-masing memiliki *state* (*Nugroho Adi, 2009*). Kecanggihan bahasa-bahasa pemrograman berorientasi objek seperti C# adalah kemampuannya dalam melakukan pengelompokan data dan metode/fungsi terkait dalam suatu definisi kelas, di mana hal ini sesuai dengan fakta yang memang ada di dunia.

II.6.5 Definisi Dan Deklarasi *Array*

Larik (*array*) adalah kelompok peubah tunggal atau multidimensi. Larik adalah sejumlah nilai bertipe sama yang bisa dirujuk menggunakan nama peubah yang sama diikuti indeksnya (posisi nilai tertentu dalam larik) (*Nugroho Adi, 2009*). Larik sangat bermanfaat juka digunakan secara baik. Saat mendekalarasikan larik menggunakan bahasa C# bilangan yang digunakan dalam deklarasi larik merepresentasikan jumlah total elemen dalam lari yang dibuat (*Nugroho Adi, 2009*). Pembuatan suatu larik yang isinya beragam yang semuanya disebut objek. Aplikasi penting yang sering digunakan dalam kaitan dengan larik adalah larik 2 dimensi (matriks) (*Nugroho Adi, 2009*).

II.6.6 *Build* Dan *Debug*

Build adalah tingkat-tinggi membangun sistem untuk mengelola C# dan proyek *C*. Hal ini didasarkan pada file-file konfigurasi yang adalah file

teks sederhana. Dalam bertentangan dengan lain membangun sistem file konfigurasi compiler dan *platform*-independen dan hanya berisi sebanyak mungkin informasi mutlak diperlukan untuk membangun sistem untuk mengetahui bagaimana untuk menghasilkan binari. Sebagai *build* mendukung banyak kompiler dan platform keluar dari kotak pilihan dan diterjemahkan dalam file konfigurasi untuk opsi baris perintah yang diharapkan oleh sebuah kompiler yang dipilih file-file konfigurasi ditulis sekali dan digunakan di mana-mana. Program dilaksanakan di standar C# atau C standar dapat dengan mudah dibangun dengan build pada *platform* yang berbeda tidak peduli apa *compiler* yang digunakan.

Debug adalah proses menemukan dan memperbaiki atau melewati *bugs (error)* dalam program kode komputer atau rekayasa perangkat keras. Untuk *debug* atau perangkat keras program ini adalah untuk mulai dengan masalah, mengisolasi sumber masalah, dan kemudian memperbaikinya.

BAB III

PELAKSANAAN PENELITIAN

III.1 Persiapan

Sebelum melakukan sebuah penelitian, diperlukan suatu persiapan yang matang guna memperlancar proses penelitian sampai penyajian hasil dari batasan penelitian yang dibahas. Untuk memperoleh hasil yang maksimal maka ada beberapa hal yang harus dipersiapkan terlebih dahulu, sebagai berikut :

III.1.1 Materi Penelitian

Adapun materi yang digunakan sebagai bahan dalam penelitian ini meliputi nilai interior orientasi, dan nilai pendekatan eksterior orientasi yang disesuaikan dengan batasan penelitian ini :

1. Data foto merupakan data dasar yang dibutuhkan dalam proses *Bundle Adjustment Close Range Photogrammetry* yang berupa data digital.
2. Data parameter *Interior Orientation* (IO) pendekatan yaitu (*Xo,Yo, f*) dan koordinat foto (*x,y*) serta ketelitian (σ_x, σ_y) dari proses *centroid.* Parameter interior mendefinisikan geometri internal sebuah kamera antara lain *perspektif center*, *principle point*, serta panjang fokus dari kamera.
3. Data parameter *Exterior Orientation* (EO) pendekatan yaitu ($X_L, Y_L, Z_L, \omega, \varphi, \kappa$) tiap foto dari proses *Relative Orientation* dan Proses *Resection.*

4. Data parameter *Object space* pendekatan yaitu (X_i , Y_i ,, Z_i) dari proses *Intersection.*

MILIK
PERPUSTAKAAN
ITN MALANG

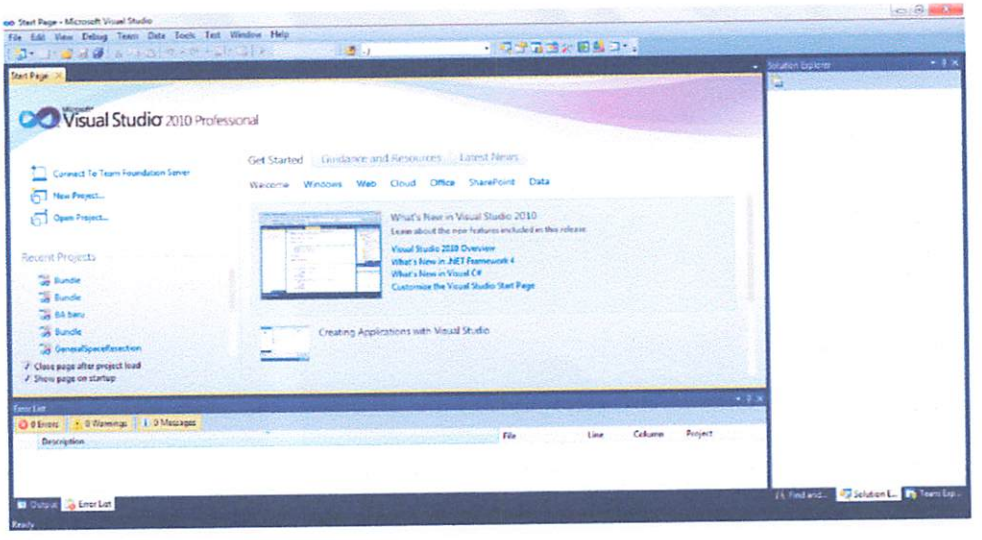
III.1.2 Alat Penelitian

Adapun alat dan bahan yang dibutuhkan dalam proses penelitian ini baik itu perangkat lunak (*software*) maupun perangkat keras (*hardware*) antara lain :

1. Perangkat Lunak

   1) Microsoft Visual Studio 2010

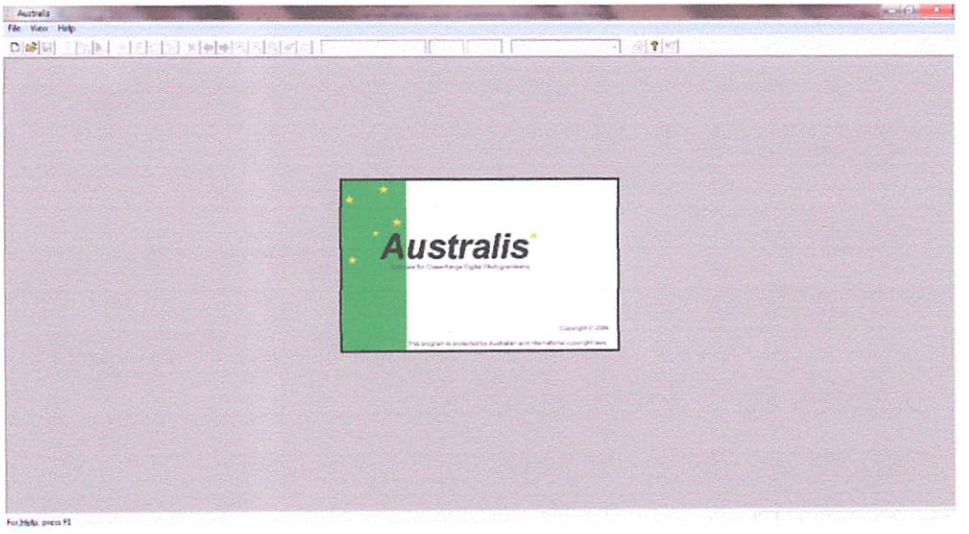
      Microsoft Visual Studio 2010 merupakan salah satu perangkat lunak yang berorientasi Object (*Object-oriented programming* disingkat OOP). Konsep pemograman orientasi objek menekankan pada : Kelas, objek, method, event serta action.



Gambar 3.1 Tampilan awal Program C#

2) Australis

Australis merupakan salah satu perangkat lunak untuk memperoleh parameter EO pendekatan dan koordinat obyek 3 dimensi pendekatan.



Gambar 3.2 Tampilan awal Program Australis

2. Perangkat Keras

1) Perangkat komputer Intel Core i-3

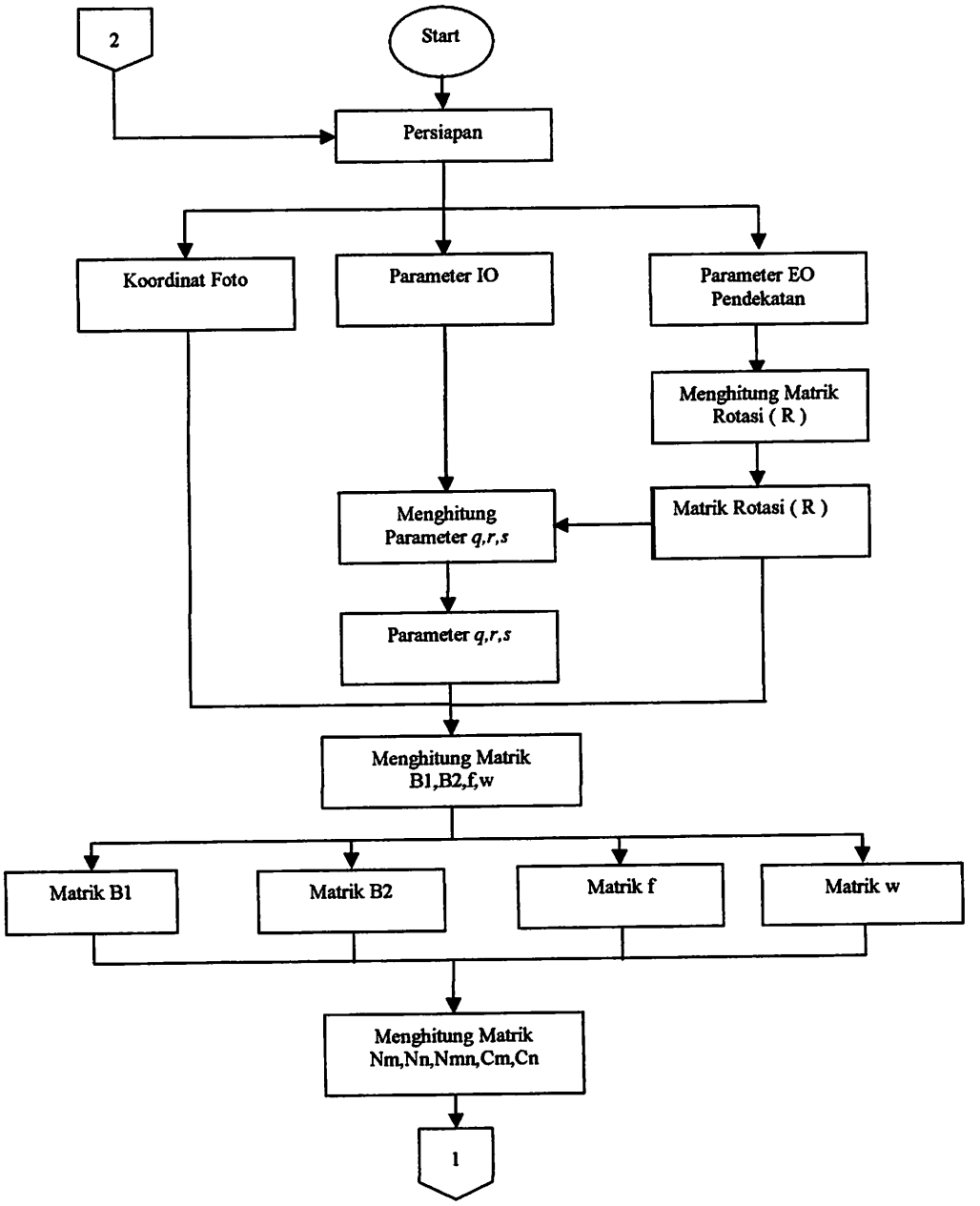
2) RAM 1 GB

MILIK
PERPUSTAKAAN
ITN MALANG

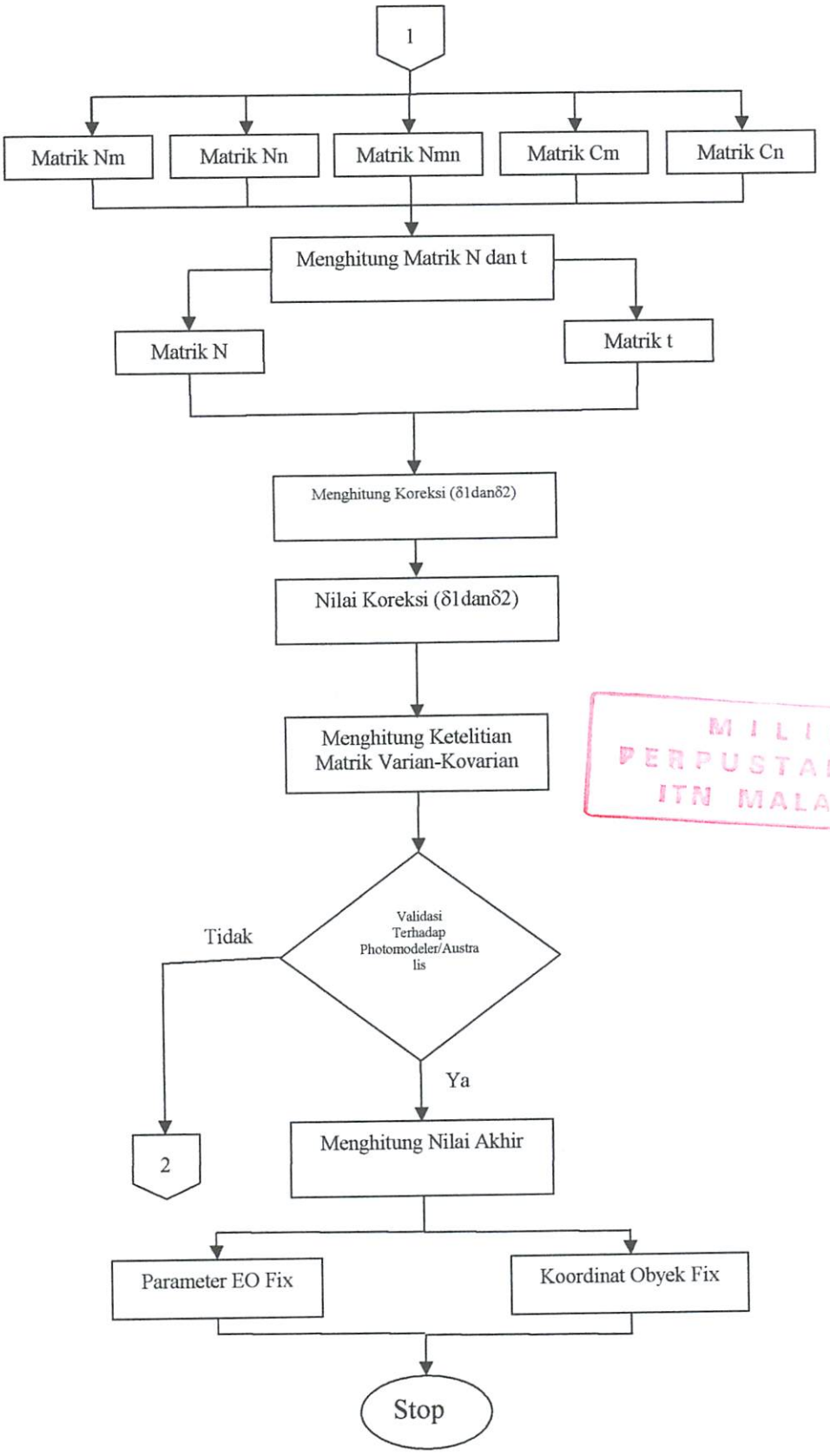
III.2 Langkah Penelitian

Dalam proses penelitian haruslah dibuat suatu kerangka pekerjaan yang sistematis agar mudah dipahami dan mempermudah dalam penelitian. Adapun langkah atau alur penelitian yang akan dilakukan sebagai berikut :

Diagram alir penelitian :



Gambar 3.3 Diagram Alir Penelitian (1)



MILIK PERPUSTAKAAN ITN MALANG

Gambar 3.4 Diagram Alir Penelitian (2)

Keterangan Diagram Alir Penelitian:

1. Tahap Persiapan

   Tahapan pertama yang harus dilakukan adalah menyiapkan segala unsur-unsur yang dapat digunakan untuk mendukung kelancaran proses penelitian, baik berupa lokasi penelitian, materi, alat dan bahan penelitian, metode penelitian dan metode perhitungan yang akan digunakan.

2. Tahap Perolehan Data

   Didalam tahap ini dilakukan proses perhitungan dengan menentukan data awal koordinat foto (x,y) dan ketelitiannya, parameter *Interior Orientation*(*Xo,Yo,f*), dan koordinat obyek (X_i,Y_i,Z_i), dan estimasi parameter EO pendekatan.

   1 .Parameter *Interior Orientation*(*Xo,Yo,f*)

Parameter IO	
Xo	0
Yo	0
f (millimeter)	18.556

Tabel 3.1.Parameter IO

2 .Parameter *Exterior Orientation*(ω, φ, κ, XL, YL, ZL)

Foto	Omega (radians)	Phi (radians)	Kappa (radians)	XL (milimeter)	YL (milimeter)	ZL (milimeter)
1	-0.000006	-0.000009	0.000008	-0.000007	0.000002	0.00001
2	-0.600814	0.094842	0.126491	0.157251	0.560701	-0.053418
3	-0.889227	-1.386234	-1.013069	-0.680837	0.125193	-0.721659
4	-1.214648	-0.869058	-1.187392	-0.640585	0.611842	-0.592235

Tabel 3.2.Parameter EO

3 . Ketelitian Parameter *Exterior Orientation*(sω, sφ, sκ, sXL, sYL, sZL)

Foto	sOmega (radians)	sPhi (radians)	sKappa (radians)	sXL (milimeter)	sYL (milimeter)	sZL (milimeter)
1	0.00001	1.20000	0.00001	0.00001	1.20000	0.00001
2	0.00001	1.20000	0.00001	0.00001	1.20000	0.00001
3	0.00001	1.20000	0.00001	0.00001	1.20000	0.00001
4	0.00001	1.20000	0.00001	0.00001	1.20000	0.00001

Tabel 3.3.Ketelitian Parameter EO

4 . Koordinat obyek 3D (X_i, Y_i, Z_i) dan ketelitiannya

Titik	X (milimeter)	Y (milimeter)	Z (milimeter)	SX (milimeter)	SY (milimeter)	SZ (milimeter)
1	-0.248212	0.127585	-0.924662	0.000010	1.200000	0.000010
2	-0.043758	0.108611	-0.750067	0.000010	0.000010	0.000010
3	0.109730	0.081328	-0.610558	-0.500000	-3.100000	-0.500000
4	-0.264773	0.055156	-0.894553	0.000010	0.400000	0.000010
5	-0.060100	0.054513	-0.725221	0.000010	0.200000	0.000010
6	0.100197	0.042822	-0.593114	-0.200000	-1.200000	-0.200000
7	-0.151294	-0.004943	-0.773228	0.100000	0.200000	0.100000
8	0.087079	-0.011657	-0.579469	0.000010	1.000000	0.000010
9	-0.180456	-0.083628	-0.770431	0.000010	-0.300000	0.000010
10	0.068379	-0.107470	-0.564724	0.000010	0.100000	0.000010

Tabel 3.4. Koordinat obyek (X_i, Y_i, Z_i) dan ketelitiannya

5 . Koordinat foto (x,y) dan ketelitiannya (σ_x, σ_y)

FOTO 1	X (milimeter)	Y (milimeter)	Sx (milimeter)	Sy (milimeter)
1	-4.9864400	2.5620800	0.0000100	1.2000000
2	-1.0750600	2.6860700	0.0000100	0.0000100
3	3.3248800	2.4666600	-0.5000000	-3.1000000
4	-5.4971000	1.1426200	0.0000100	0.4000000
5	-1.5281900	1.3957200	0.0000100	0.2000000
6	3.1304300	1.3404500	-0.2000000	-1.2000000
7	-3.6248000	-0.1162800	0.1000000	0.2000000
8	2.7893500	-0.3706700	0.0000100	1.0000000
9	-4.3459800	-2.0139500	0.0000100	-0.3000000
10	2.2499100	-3.5332800	0.0000100	0.1000000

Tabel 3.5. Koordinat foto dan ketelitiannya pada Foto1

FOTO 2	X (milimeter)	Y (milimeter)	Sx (milimeter)	Sy (milimeter)
1	-5.4421300	3.2292900	0.0000100	1.2000000
2	-2.5942100	0.7956000	0.0000100	0.0000100
3	0.2990300	-2.0959700	-0.5000000	-3.1000000
4	-5.7976400	1.8089200	0.0000100	0.4000000
5	-3.0476600	-0.4369400	0.0000100	0.2000000
6	-0.0591500	-3.0825100	-0.2000000	-1.2000000
7	-4.4772300	-0.6205500	0.1000000	0.2000000
8	-0.4930500	-4.2372600	0.0000100	1.0000000
9	-4.9006000	-1.7755200	0.0000100	-0.3000000
10	-1.0534200	-5.9539700	0.0000100	0.1000000

Tabel 3.6. Koordinat foto dan ketelitiannya pada Foto2

FOTO 3	X (milimeter)	Y (milimeter)	Sx (milimeter)	Sy (milimeter)
1	-6.5572900	1.9216100	0.0000100	1.2000000
2	1.0592700	2.3382500	0.0000100	0.0000100
3	4.6266000	2.2332400	-0.5000000	-3.1000000
4	-5.1275600	-0.9990700	0.0000100	0.4000000
5	1.9736300	0.8064900	0.0000100	0.2000000
6	5.1780000	1.3448300	-0.2000000	-1.2000000
7	0.5978200	-1.7389300	0.1000000	0.2000000
8	5.6970200	0.0386300	0.0000100	1.0000000
9	0.9776300	-4.6663800	0.0000100	-0.3000000
10	6.3742100	-2.3098000	0.0000100	0.1000000

Tabel 3.7. Koordinat foto dan ketelitiannya pada Foto3

FOTO 4	X (milimeter)	Y (milimeter)	Sx (milimeter)	Sy (milimeter)
1	-5.5784900	-3.2028300	0.0000100	1.2000000
2	0.4461300	-0.6074900	0.0000100	0.0000100
3	3.9904700	0.6258400	-0.5000000	-3.1000000
4	-4.5761300	-4.9837700	0.0000100	0.4000000
5	0.9663500	-1.8689400	0.0000100	0.2000000
6	4.2748800	-0.1643600	-0.2000000	-1.2000000
7	-0.5418200	-4.2049800	0.1000000	0.2000000
8	4.4416400	-1.2168400	0.0000100	1.0000000
9	-0.5241200	-5.9009200	0.0000100	-0.3000000
10	4.5469300	-2.8701800	0.0000100	0.1000000

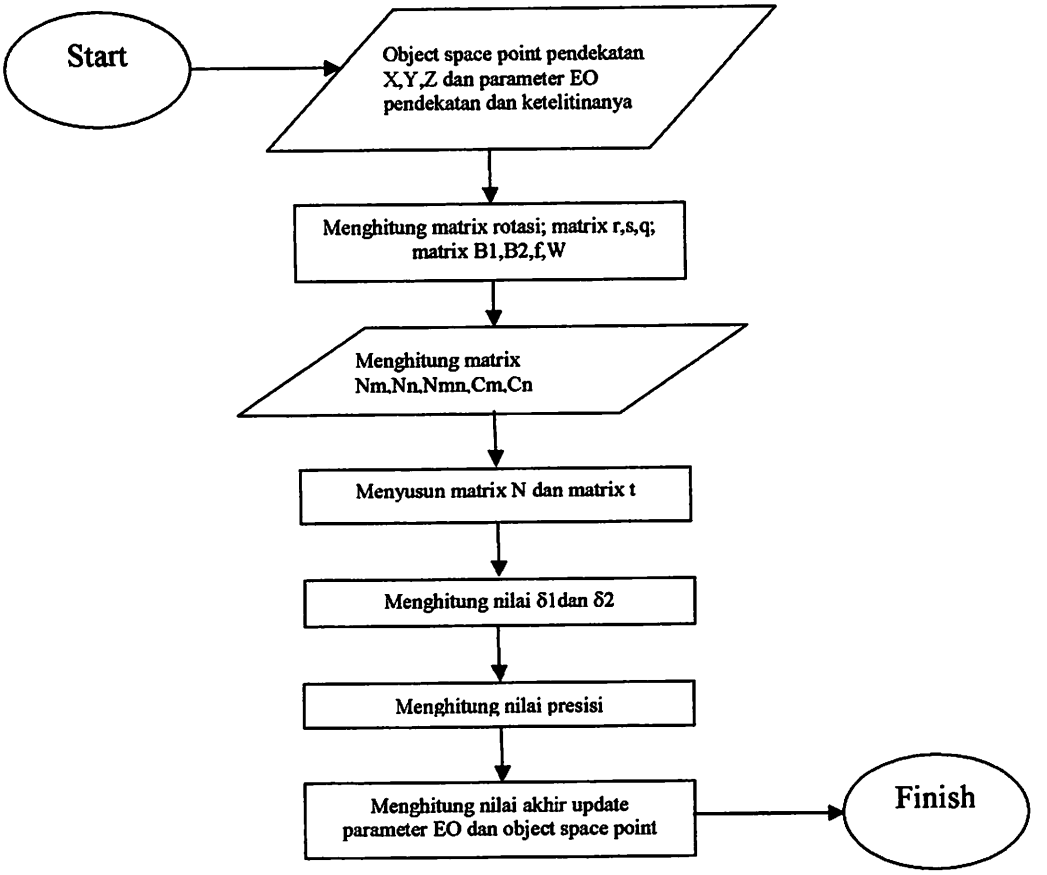
Tabel 3.8. Koordinat foto dan ketelitiannya pada Foto4

3. Program *Bundle Adjusment*

Pada bagian ini dilakukan pembuatan program tentang *bundle adjustment multi photo*. Jika hasilnya benar maka proses penelitian dianggap selesai dan jika tidak maka kembali mengecek pembuatan program yang kemungkinan terjadi kesalahan.

III.3 Diagram Alir Program

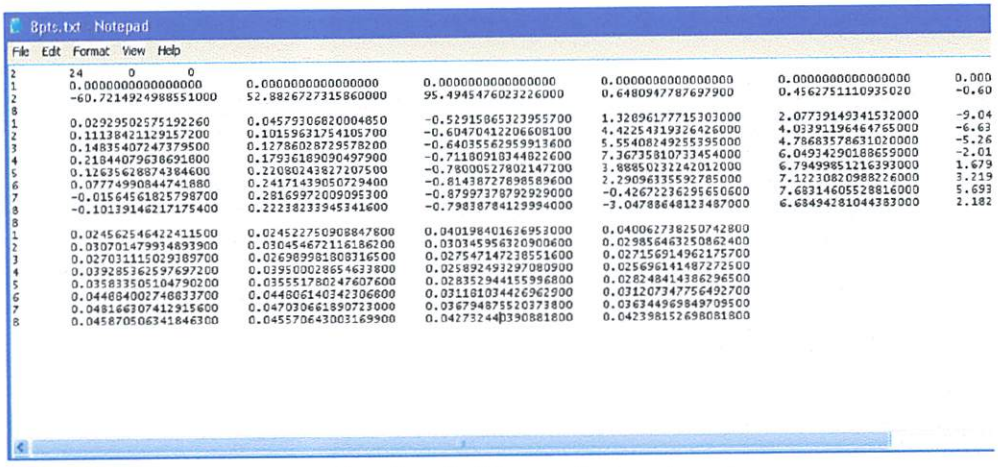
Dalam proses pembuatan program haruslah dibuat suatu kerangka pekerjaan yang sistematis agar mudah dipahami dan mempermudah dalam pembuatan program. Adapun langkah atau alur pembuatan program yang akan dilakukan sebagai berikut:



Gambar 3.5 Diagram Alir Program

Keterangan diagram alir program:

1. Dari data koordinat foto (x,y) dan ketelitian (σ_x, σ_y) serta Parameter *Interior Orientation* (IO), Parameter *Exterior Orientation* (EO) pendekatan yaitu ω, φ, κ, XL, YL, ZL, diperoleh melalui proses *Relative Orientation* (RO), kemudian akan didapatkan juga koordinat titik-titik objek (Xi, Yi, Zi) pendekatan yang diperoleh menggunakan proses interseksi. Data awal tersebut akan disimpan didalam bentuk file *".txt"* yang akan digunakan sebagai *input* awal untuk proses lebih lanjut didalam program ini. Berikut adalah bentuk format penulisan data awal didalam file *".txt"* yang dapat dibaca dengan fasilitas *"notepad* "didalam *"windows"*.

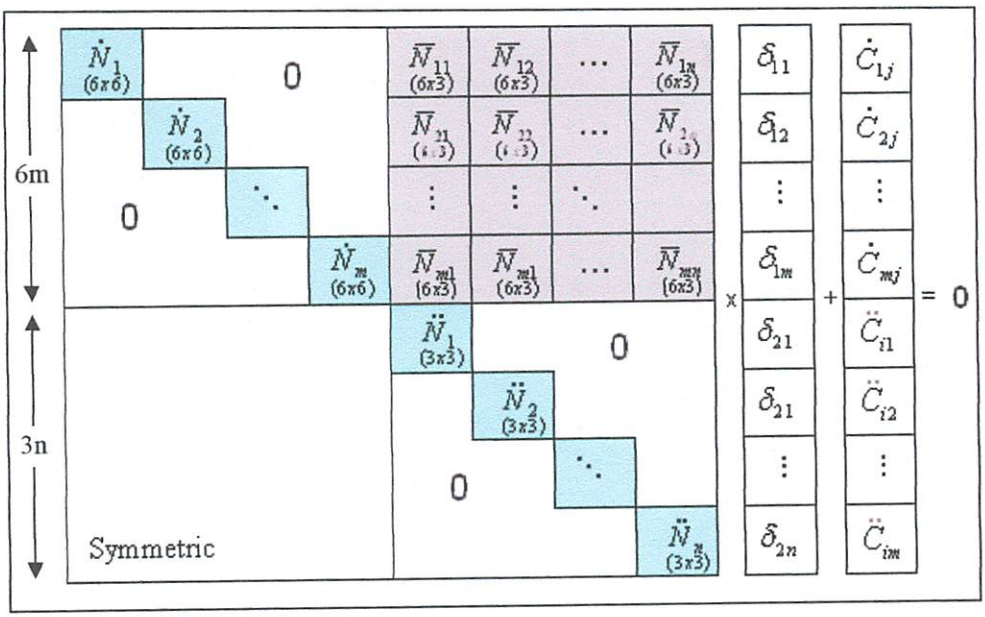


Gambar 3.6 Input dari notepad

2. Setelah setiap data divariabelkan maka akan dimulai proses perhitungan untuk mendapatkan matrix rotasi dari parameter *EO* pendekatan,matrix r ,s ,q dari koordinat *object space* pendekatan yang kemudian akan diteruskan secara berurutan hingga mendapatkan matrix. Dari perhitungan tesebut akan

didapatkan matrix B_1, B_2, L, kemudian akan dilanjutkan dengan perhitungan untuk mendapatkan elemen-elemen matrix $\dot{N}_{ij}, \ddot{N}_{ij}, \overline{N}_{ij}, \dot{C}i_j, \ddot{C}_{ij}, G$

3. Setelah matrik $\dot{N}_{ij}, \ddot{N}_{ij}, \overline{N}_{ij}, \dot{C}i_j, \ddot{C}_{ij}, G$ didapatkan maka langkah selanjutnya adalah menyusun matrik N dan matrix t. Berikut adalah susunan matriknya:



Gambar 3.7 Susunan Matrix N dan t (Fraser, 1997)

4. Setelah matrik N dan t tersusun maka proses selanjutnya adalah menghitung matrix koreksi δ_1 dan δ_2

5. Menghitung nilai ketelitian dapat dilakukan jika matrik koreksi telah didapat . Didalam proses ini user akan melakukan verifikasi terhadap nilai ketelitian yang didapatkan. Jika nilai residu tidak sesuai maka perlu dilakukan proses iterasi hingga mendapatkan nilai ketelitian yang sesuai.

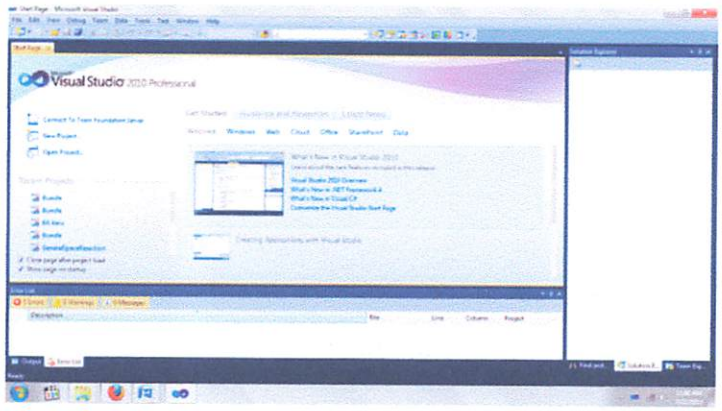
Dalam program ini telah disediakan fasilitas iterasi hingga nilai ketelitian yang sesuai didapatkan. Perancang telah membuat hasil *output* sesuai dengan format input data dengan melakukan update pada setiap parameter *EO* dan *object space point* pendekatan(X,Y,Z) dengan menambahkan nilai koreksi yang didapatkan pada proses sebelumnya sehingga proses iterasi dapat dilakukan dengan menjalankan program dari awal lagi dengan mengganti input dengan *output* hasil proses sebelumnya.

6. Jika ketelitian telah sesuai maka akan dilanjutkan dengan menghitung nilai akhir dengan cara mengupdate data *input* dengan nilai koreksi yang telah didapatkan dalam proses ini,selanjutnya adalah menghitung nilai standar deviasi.

III.4 Pembuatan Program *Bundle Adjustment*

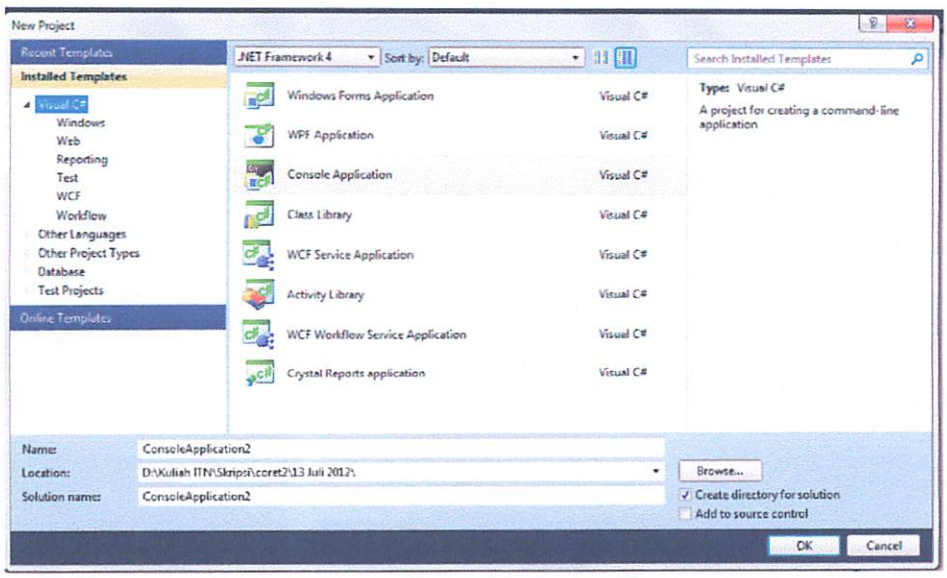
Dalam proses pembuatan program harus mengikuti diagram alir penelitian serta diagram alir pembuatan program untuk mempermudah dalam pembuatan program. Adapun langkah dalam pembuatan program yang akan dilakukan sebagai berikut :

1. Buka aplikasi *Visual Studio 2010* dengan melakukan *double*-klik pada *icon Visual Studio 2010* yang berada pada *desktop*. Sehingga, akan muncul sebuah tampilan awal aplikasi sebagai berikut.



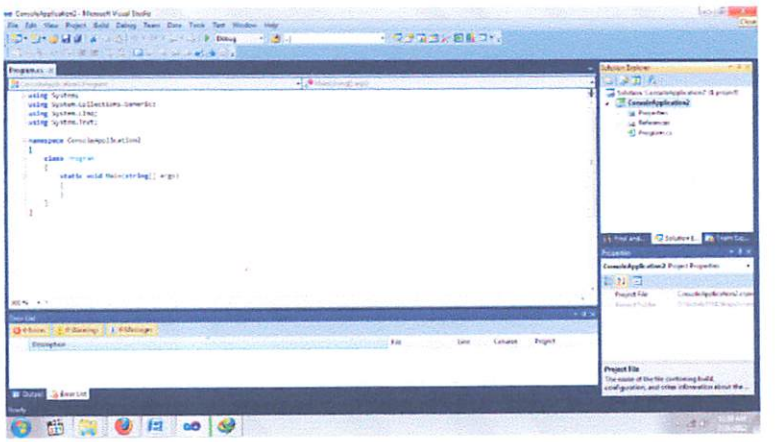
Gambar 3.8 Tampilan awal Visual Studio 2010

2. Untuk memulai *Project* baru, klik *New Project*. Kemudian akan muncul sebuah jendela *New Project* dan selanjutnya atur agar menggunakan bahasa C# pada *Installed Templates* kemudian pilih *Visual C#* kemudian pilih *Console Application* serta tempat penyimpanan program yang dibuat dengan cara isikan *Name* (nama folder program) serta *Location* (tempat penyimpanan program) dan *Solution name* (nama program yang dibuat), pengaturan awal *Project* dalam tampilan *New Project* seperti dibawah ini:



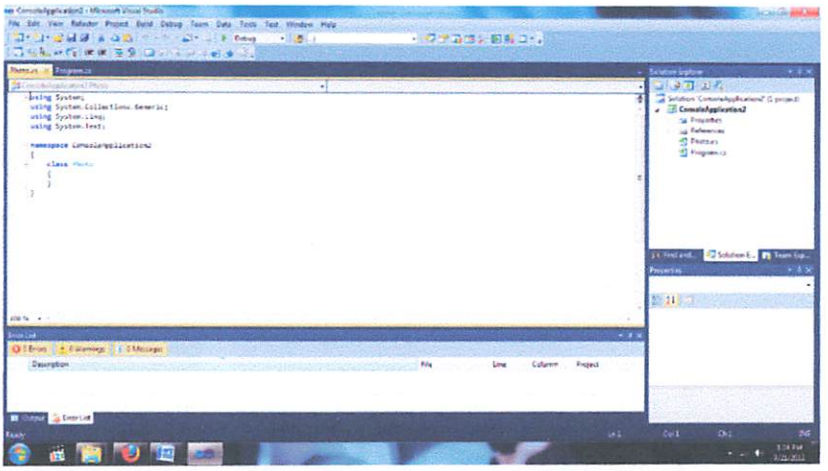
Gambar 3.9 Tampilan New Project

Setelah mengatur bahasa C# yang digunakan dalam bahasa pemrograman dan menggunakan *Console Application* sebagai tampilan program serta tempat penyimpanan program yang akan dibuat, selajutnya klik *OK*. Maka akan muncul tampilan awal program seperti dibawah ini:



Gambar 3.10 Tampilan awal program dengan bahasa C#

3. Data-data koordinat foto (x,y) dan ketelitian (σ_x, σ_y) serta Parameter *Interior Orientation* (IO), Parameter *Exterior Orientation* (EO) pendekatan yaitu ω, φ, κ, XL, YL, ZL dan koordinat titik-titik objek (Xi, Yi, Zi) harus terlebih dahulu dilakukan pendefisinian variabel yang ada dalam kelas tertentu. Dalam pembuatan kelas untuk mendefinisikan variabel perlu melakukan langkah-langkah pilih *Project* kemudian pilih *Add Class* selanjutnya masukkan *Name* (nama kelas yang dibuat) kemudian klik *Add* maka akan muncul kelas yang baru dibuat seperti dibawah ini:



Gambar 3.11 Tampilan awal class

4. Selanjutnya memasukkan kode untuk pendefinisian variabel data tiap kelas dengan contoh kode seperti dibawah ini:

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

namespace Bundle
{
   class Photo
   {
      private string photoname;
      private ParameterEO parametereo;
      private List<point2D> ptList;

      public void InputData(string photName, ParameterEO paramEO,
List<point2D> p2Dlist)
      {
         this.photoname = photName;
         this.parametereo = paramEO;
         this.ptList = p2Dlist;
      }
      public string Photoname
      {
         get { return photoname; }
         set { photoname = value; }
      }
      public ParameterEO Parametereo
      {
```

```
            get { return parametereo; }
            set { parametereo = value; }
        }
        public List<point2D> ListPoint2D
        {
            get { return ptList; }
            set { ptList = value; }
        }
    }
}
```

5. Setelah selesai membuat seluruh kelas sesuai dengan variabel data yang ada, kemudian memasukkan kode untuk pemanggilan data input dari text file pada program dengan kode sebagai berikut:

PERPUSTAKAAN
ITN MALANG

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

namespace Bundle
{
    class Program
    {
        static void Main(string[] args)
        {
            //Masukan File
            string word = null;
            string jumlahfoto;
            Console.WriteLine("Masukan path file = ");
            word = Console.ReadLine();//untuk membaca path file
            TextReader tr = new StreamReader(word);
            Console.WriteLine(tr.ReadToEnd());
            jumlahfoto = Console.ReadLine();//untuk membaca jumlah foto.
            //Eksekusi Perhitungan
            bacadata baca = new bacadata(word);
            baca.data2D();
            Console.ReadLine();
        }
    }
}
```

6. Setelah melakukan input data dalam program dari data text file, selanjutnya memisahkan antar data seperti data koordinat foto (x,y) dan ketelitian (σ_x, σ_y) serta Parameter *Interior Orientation* (IO), Parameter *Exterior Orientation* (EO) pendekatan yaitu ω, φ, κ, XL, YL, ZL dan koordinat titik-titik objek (Xi, Yi, Zi) kedalam kelas-kelas yang telah dibuat, seperti contoh dibawah ini:

PERPUSTAKAAN
ITN MALANG

```
#region Baca data
      public bacadata(string fileName)
      {
        this.fileName = fileName;
      }
      public bool BacaData()
      {
        if (!File.Exists(this.fileName))
        {
          Console.WriteLine(String.Format("{} File Tidak Ada",
this.fileName));
        }
        string[] bacaText = File.ReadAllLines(this.fileName);
        int rows = bacaText.Count();
        string[] pC;
        for (int i = 0; i < rows; i++)
        {
          String[] data = bacaText[i].Split(new char[] { ' ', '\t' });
          String[] ptnbr;
          String[] eox;
          String[] optx;
          if (i == 0)
          {
            pC = bacaText[i].Split(new char[] { ' ', '\t' });
            this.focal = double.Parse(pC[0]);
            this.jf = int.Parse(pC[1]);
            this.jo = int.Parse(pC[2]);
            Console.WriteLine("Panjang Fokus ={0} , jumlah Foto ={1}",
focal, jf);
          }
          else if (data.Length == 1)
          {
            ptnbr = bacaText[i].Split(new char[] { ' ', '\t' });
```

```
            }
            else if (data.Length == 13)
            {
                eox = bacaText[i].Split(new char[] { ' ', '\t' });
                inputEO(eox);
            }
            else if (data.Length == 5)
            {
                optx = bacaText[i].Split(new char[] { ' ', '\t' });
                baca_p2(optx);
            }
            else if (data.Length == 7)
            {
                String[] opt = bacaText[i].Split(new char[] { ' ', '\t' });
                baca_p3(opt);
            }
        }
        return true;
    }
    public void baca_p3(string[] photocoord3D)
    {
        Point3D p3d = new Point3D();

        List<string> lst = new List<string>();
        foreach (string a in photocoord3D)
        {
            lst.Add(a);
        }
        p3d.Label = lst[0];
        p3d.X = double.Parse(lst[1]);
        p3d.Y = double.Parse(lst[2]);
        p3d.Z = double.Parse(lst[3]);
        p3d.SX = double.Parse(lst[4]);
        p3d.SY = double.Parse(lst[5]);
        p3d.SZ = double.Parse(lst[6]);

        ListPoint3D.Add(p3d);
    }
```

7. Seletah melakukan pemisahan antara data input yang ada, selajutkan memasukkan kode perhitungan untuk memperoleh matrix B_1, B_2, L, contoh hasil perhitungan matrix B_1, B_2, L terdapat pada lampiran ke 1. Kemudian akan dilanjutkan dengan perhitungan untuk mendapatkan elemen-elemen

matrix $\dot{N}_{ij},\ddot{N}_{ij},\overline{N}_{ij},\dot{C}i_j,\ddot{C}_{ij},G$, contoh hasil perhitungan matrix $\dot{N}_{ij},\ddot{N}_{ij},\overline{N}_{ij},\dot{C}i_j,\ddot{C}_{ij},G$ terdapat pada lampiran ke 2.

8. Kemudian memasukkan kode untuk pembuatan matrik normal, setelah itu menghitung matrix koreksi δ_1 dan δ_2 „ contoh hasil perhitungan matrix koreksi δ_1 dan δ_2 terdapat pada lampiran ke 3. Setelah memperoleh matrix koreksi δ_1 dan δ_2 , kemudian membuat kode untuk perhitungan parameter EO dan koordinat obyek 3D sehingga memperoleh parameter EO dan koordinat obyek 3D akhir, contoh hasil perhitungan parameter EO dan koordinat obyek 3D akhir terdapat pada lampiran ke 3. Selanjutnya membuat kode untuk perhitungan standard deviasi dan berakhir program yang dibuat, kode keseluruhan program terdapat pada lampiran ke 4.

BAB IV

HASIL DAN ANALISA

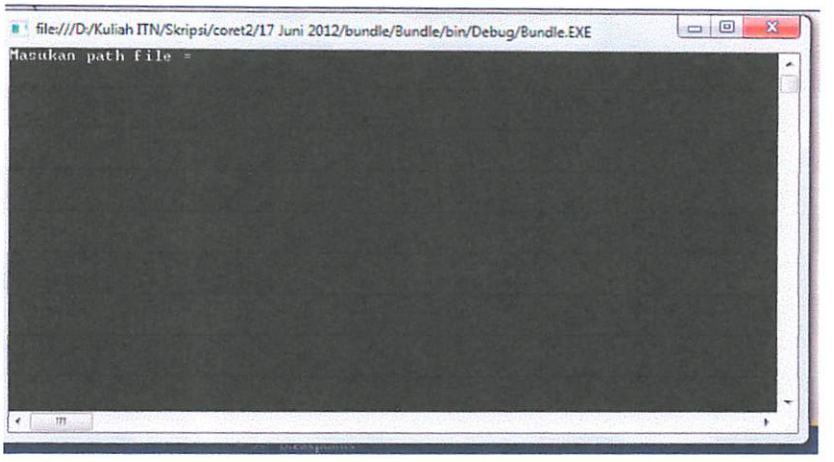
Untuk mengetahui tingkat keberhasilan dari penelitian, perlu dilakukan proses analisa atau pembahasan dari sebuah hasil yang telah dicapai selama proses pelaksanaan penelitian. Parameter keberhasilan dapat diukur dengan membandingkan tujuan dari penelitian dan hasil yang dicapai. Adapun beberapa parameter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

IV.1 Hasil

Didalam pembuatan program perhitungan bundle adjustment banyak terdapat point-point penting yang harus diperhatikan baik didalam algoritma maupun didalam listing program itu sendiri. Algoritma yang tepat serta mudah dipahami akan sangat membantu terhadap ketepatan, ketelitian serta kestabilan suatu program. Demikian juga dengan dengan Listing program,dengan perintah-perintah yang tepat, efisien serta flesibel, kinerja dan kestabilan suatu program dapat terjaga didalam menghadapi kemungkinan-kemungkinan yang terjadi ketika program tersebut dijalankan. Oleh karena itu perlu dibahas secara jelas langkah-langkah didalam menjalankan program, sebagai berikut:

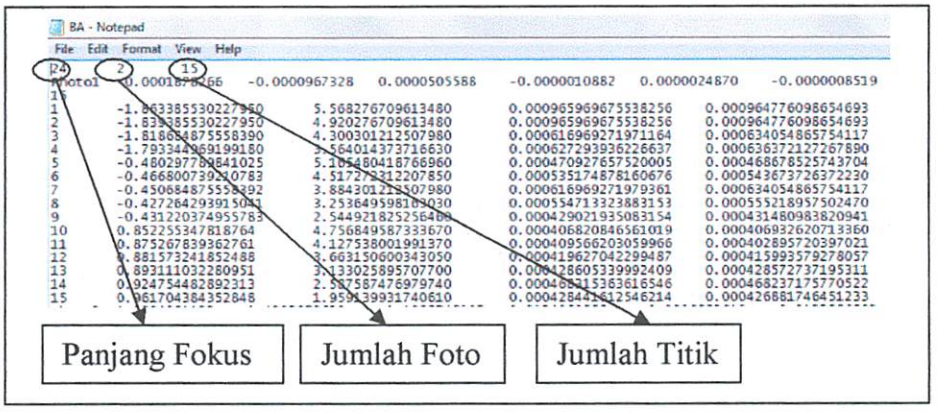
1. Buka program aplikasi *bundle adjustment* yang telah dibuat dalam *shortcut bundle adjustment*, dengan cara klik 2 (dua) kali pada *shortcut* program *bundle adjustment* yang telah dibuat sehingga program terbuka

dalam bentuk *Windows DOS*. Maka akan muncul tampilan program seperti dibawah ini:

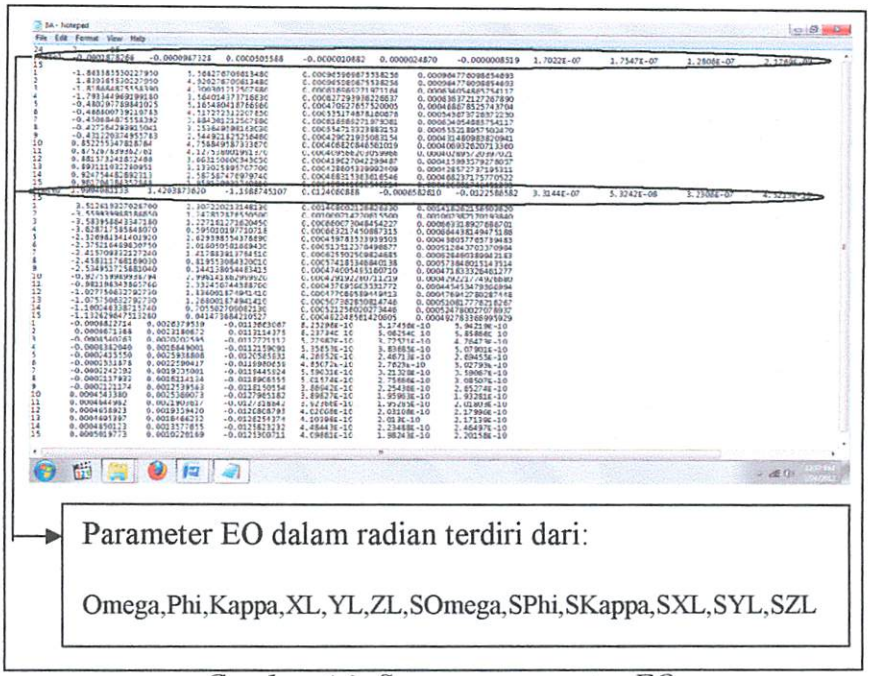


Gambar 4.1 Tampilan Program Bundle Adjustment

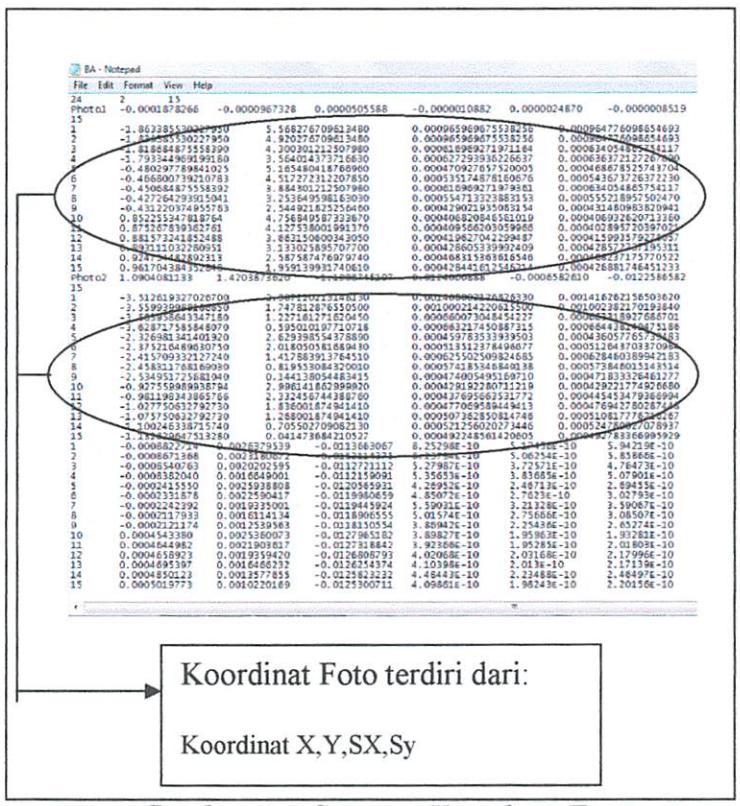
2. Mempersiapkan data diinputkan adalah parameter EO pendekatan, *object space* pendekatan, dan titik-titik koordinat foto. Ketiga data ini akan menjadi data inputan dalam program bundle adjustment yang akan dilakukan dalam bentuk notepad (*file* dalam bentuk format *.txt*), seperti dibawah ini:



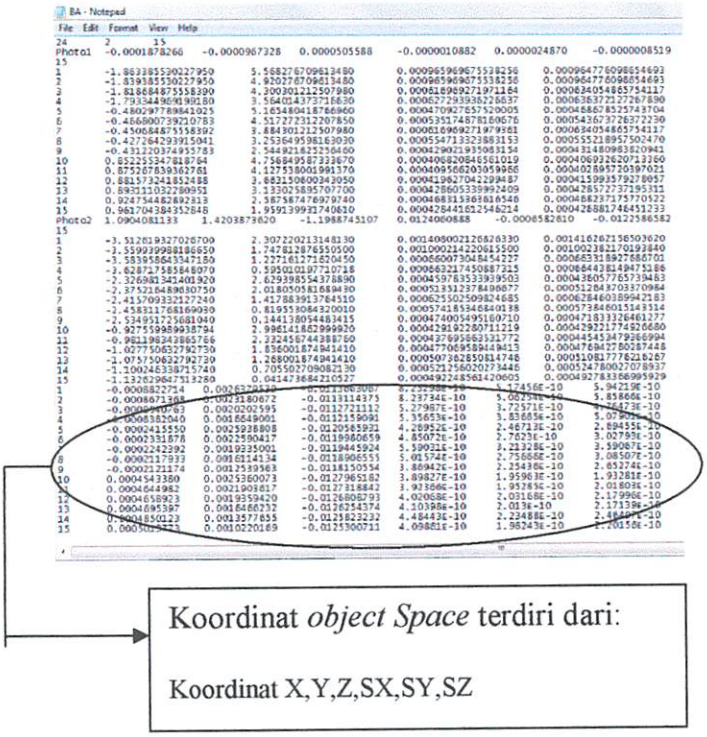
Gambar 4.2 Susunan pajang fokus, jumlah foto, jumlah titik



Gambar 4.3 Susunan parameter EO

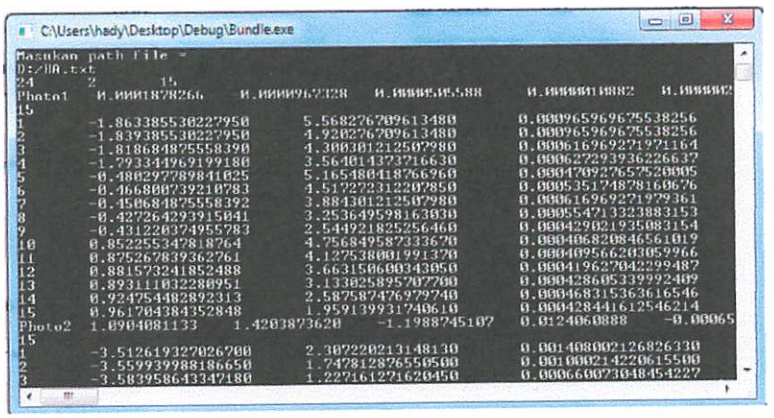


Gambar 4.4 Susunan Koordinat Foto



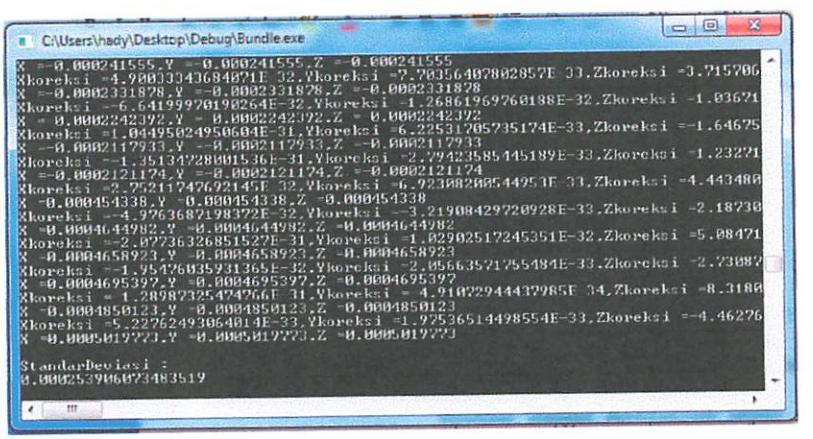
Gambar 4.5 Susunan Koordinat Objek Space

3. Setelah tampilan program *bundle adjustment* terbuka langkah dan data dalam *text file* telah siap selanjutnya memasukkan data dalam bentuk *text file* dengan cara ketik data yang ada dalam bentuk *text file* yang telah disiapkan seperti contoh: " D:/BA.txt", data *text file* yang telah tersimpat dalam folder D dengan nama BA.txt akan terbuka dalam program *bundle adjustment* sehingga akan program akan membaca data yang ada. Maka program *bundle adjustment* akan menbaca data BA.txt seperti dibawah ini:



Gambar 4.6 Tampilan Data yang dibuka dalam Program Bundle Adjustment

4. Setelah data dalam bentuk *text file* terbaca dalam program *bundle adjustment*, selanjutnya tekan *Enter* 1 (satu) kali untuk menjalankan perhitungan program *bundle adjustment* maka data yang terbuka diproses hinggal memperoleh *object space coordinat* yang final serta parameter EO yang final serta memperoleh standar devisiasi, maka selesai penggunaan program *bundle adjustment* yang digunakan, seperti contoh dibawah ini:



Gambar 4.7 Hasil Perhitungan Menggunakan Program Bundle Adjustment

BAB V

PENUTUP

V.1 Kesimpulan

Setelah penulis melakukan seluruh rangkaian penelitian dan menyelesaikan penulisan laporan dalam rangkaian "Pembuatan Program *Bundle Adjustment Multi Photo Konvergen* Dengan Bahasa C#" dapat ditarik beberapa kesimpulan sebagai berikut:

1. Program *Bundle Adjustment* membutuhkan inputan awal parameter IO kamera, parameter EO masing-masing kamera, koordinat *object space point*, dan koordinat foto masing-masing kamera.
2. Program *Bundle Adjustment* menghasilkan parameter EO terkoreksi, koreksi parameter EO, koordinat *object space point* terkoreksi, dan koreksi koordinat *object space point.*
3. Program *Bundle Adjustment* yang dibuat dapat dibuka menggunakan *Windows DOS*, dengan data input berupa *text file*.

V.2 Saran

Dalam pengerjaan penelitian Tugas Akhir ini tentu nya penulis mempunyai banyak pendapat tentang segala hal baik masalah teknis maupun non-teknis. Dan adapun saran-saran jika sekiranya program perhitungan bundle

adjustment ini akan coba disempurnakan oleh perancang program yang lain, antara lain:

1. Perhitungan *bundle adjustment* harus menambahkan parameter datum untuk solusi hitung kuadrat terkecil jika data parameter EO pendekatan dari 2 foto belum terdefinisikan datum awal disalah satu foto.
2. Dalam metode perhitungan *bundle adjustment* harus menambahkan parameter datum untuk menghindari adanya *Matrix Singular*, adalah matrix yang memiliki Determinant=0.
3. Program *bundle adjustment* dapat disimpan dalam format *text file*.
4. Pembuatan program *bundle adjustment multi photo* dengan menggunakan parameter kalibrasi kamera.

DAFTAR PUSTAKA

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

Fraser, C. S. 1997. *Photogrametric Orientation : Transformation from Image to Object Space.* Photogrammetry 451-447. Melbourne, The University of Melbourne.

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

King, B. A. 1993. *Methods For The Photogrammetric adjustment of Bundles of Constrained Streospairs.* Departement of Civil Engineering And Surveying. New South Wales, The University of Newcastle.

Mikhail, J. S. Bethel, et al. 2001. *Introduction To Modern Photogrammetry.* New York, John Wiley & Sons, Inc.

Ruther, H. 1989. An Overview of Software in Non-topographic Photogrammetry. In : Non-topographic Photogrammetry.2nd Edition. *American Society for Photogrammetry and Remote Sensing. Falls Church, Virginia, USA., pp129-145.*

Tjahjadi, E. 2008b. *Ketelitian Pengukuran Dari Kamera Non Metrik.* Jurusan Teknik Geodesi. Malang, Institut Teknologi Nasional.

Triggs, B., P. McLauchlan, F., et al. 2000. *Bundle Adjustment□A Modern Synthesis.* Lecture Notes in Computer Science 1883: 298-372.

Uffenkamp, V. 1998. *State Of The Art Of High Precision Industrial Photogrammetry.* Institut für Photogrammetrie und Bildverarbeitung. Berlin, Technische Universität Braunschweig

Wang, X. and T. A. Clarke. 1998. Separate Adjustment of Close Range Photogrammetric Measurements. *ISPRS XXXII.*

Wolf, P. R. and B. A. Dewitt. 2000. *Element Of Photogrammetry With Applications in GIS.* Madison, Thomas Casson

Wolf, P. R. and Ghilani D. C. 1997. *Adjustment computations.*

Ferdiana, Ridi. *Membangun Aplikasi Smart Client Dengan Visual C#* dan *Visual Web Developer Express.* Penerbit ANDI OFFSET, Yogyakarta, 2006.

SmithDev. *Cara Mudah Menguasai Microsoft C# 2008.* Penerbit ANDI OFFSET, Yogyakarta, 2009.

Hartanto, Budi. *Memahami Visual C# .NET Secara Mudah.* Penerbit ANDI OFFSET, Yogyakarta, 2008.

Troelsen, Andrew. *Pro C# 2008 and the .NET 3.5 Platform (4th Edition).* Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York. NY 10013, 2007.

Rasheed, Faraz, Tore Nestinius, Jonathan Warthington, Lee Addy Wright. C# School. 2006. www.programmersheaven.com.

McMillan, Michael. *Data Structure and Algorithm in C#.* Pulatsky Technical College, University of Cambridge, 2006. www.cambridge.com.

Nugroho, Adi. *Algoritma dan Struktur Data dengan C#*. Penerbit ANDI OFFSET, Yogyakarta. 2009.

Mason, S.O., 1994. Expert system-based design of photogrammetric networks. Dissertation ETH Nr. 10475. Institute for Geodesy and Photogrammetry, Swiss Federal Institute of Technology (EYH), p.50.

LAMPIRAN 1

Contoh Hasil Perhitungan *Matrix* B_1, B_2, L

A. Contoh hasil perhitungan matrix B_1

B1	FOTO KE 1						FOTO KE 2												FOTO KE m					
TITIK KE 1	0.687	19.89	2.56	-20.1	-0	5.387	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-18.9	-0.69	4.981	2E-04	-20.1	-2.77	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	-8.17	21.05	1.681	-20.7	3.456	9.89	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	-17.7	-2.98	6.926	2.542	-19	10.35	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12.63	8.207	-3.32	-12.3	5.965	-25.6
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-7.78	18.08	4.809	-20.7	-19.3	4.044
TITIK KE 2	0.103	18.6	2.18	-20.1	-0	0.949	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-18.8	-0.1	0.878	2E-04	-20.1	-2.36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	-4.57	18.78	0.792	-21.4	0.604	5.778	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	-18.1	-2.45	2.587	2.667	-18.6	11.32	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10.83	7.123	-0.25	-7.9	3.222	-21
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-6.53	17.23	2.706	-13.6	-17.6	2.389
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
TITIK KE n	0.159	18.66	-2.16	-20.1	-0	-1.48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-18.8	-0.16	-1.37	1E-04	-20.1	2.332	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	0	0	0	0	0	0	-2.96	18.41	-3.03	-19.1	-0.78	3.189	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	-18.8	-2.35	0.352	2.737	-14.2	13.29	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12.41	7.133	-2.05	-5.35	1.828	-16.9
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-5.55	17.48	1.451	-12.1	-12.9	2.283

Keterangan :

n = Jumlah Titik

m = Jumlah Foto

B. Contoh hasil perhitungan matrix B_2

B2	TITIK KE 1			TITIK KE 2						TITIK KE n		
FOTO KE 1	20.0678	0.0001	-5.3867	0	0	0	0	0	0	0	0	0
	-0.0002	20.0677	2.7689	0	0	0	0	0	0	0	0	0
FOTO KE 2	17.8524	-2.1251	-7.2518	0	0	0	0	0	0	0	0	0
	-2.0295	17.1681	-7.5902	0	0	0	0	0	0	0	0	0
FOTO KE 3	18.4066	-7.4864	39.1386	0	0	0	0	0	0	0	0	0
	2.2272	41.2133	5.2320	0	0	0	0	0	0	0	0	0
FOTO KE 4	13.0874	-6.5950	25.0545	0	0	0	0	0	0	0	0	0
	20.3096	19.1759	-3.9621	0	0	0	0	0	0	0	0	0
FOTO KE 1	0	0	0	24.7388	0.0002	-1.4430	0	0	0	0	0	0
	0	0	0	-0.0002	24.7388	3.5821	0	0	0	0	0	0
FOTO KE 2	0	0	0	21.3832	-0.6036	-5.7781	0	0	0	0	0	0
	0	0	0	-2.6675	18.6332	-11.3223	0	0	0	0	0	0
FOTO KE 3	0	0	0	1.2071	-3.5938	29.1674	0	0	0	0	0	0
	0	0	0	0.9326	29.3393	3.7888	0	0	0	0	0	0
FOTO KE 4	0	0	0	5.2014	-0.9560	22.7165	0	0	0	0	0	0
	0	0	0	14.5419	18.0584	-2.5886	0	0	0	0	0	0
	0	0	0	0	0	0	...	...	...	0	0	0

	0	0	0	0	0	0	...	...	...	0	0	0
	0	0	0	0	0	0	...	...	...	0	0	0
	0	0	0	0	0	0	...	...	...	0	0	0
	0	0	0	0	0	0	...	...	...	0	0	0
	0	0	0	0	0	0	...	...	...	0	0	0
	0	0	0	0	0	0	...	...	...	0	0	0
	0	0	0	0	0	0	...	...	...	0	0	0
FOTO KE 1	0	0	0	0	0	0	0	0	0	32.8578	0.0003	3.9788
	0	0	0	0	0	0	0	0	0	-0.0002	32.8578	-6.2531
FOTO KE 2	0	0	0	0	0	0	0	0	0	22.6592	0.4374	-4.5101
	0	0	0	0	0	0	0	0	0	-3.5993	14.8629	-18.7971
FOTO KE 3	0	0	0	0	0	0	0	0	0	-5.9340	-2.0144	25.3428
	0	0	0	0	0	0	0	0	0	6.8697	23.7878	2.4701
FOTO KE 4	0	0	0	0	0	0	0	0	0	1.0385	1.7724	19.5791
	0	0	0	0	0	0	0	0	0	13.7001	13.4018	-2.6459

Keterangan :

n = Jumlah Titik

m = Jumlah Foto

C. Contoh hasil perhitungan matrix *L*

TITIK KE 1	FOTO KE 1	-0.0053847 0.0016287
	FOTO KE 2	0.0049202 0.0003637
	FOTO KE 3	0.0055237 -0.0016675
	FOTO KE 4	-0.0076943 0.0014501
TITIK KE 2	FOTO KE 1	0.0074340 -0.0009094
	FOTO KE 2	-0.0073001 0.0040906
	FOTO KE 3	0.0017594 -0.0014579
	FOTO KE 4	-0.0013144 -0.0006107
		... ...
		... ...

		... ...
		... ...
TITIK KE n	FOTO KE 1	0.0030867 -0.0020719
	FOTO KE 2	0.0014870 0.0076038
	FOTO KE 3	0.0042786 0.0028088
	FOTO KE 4	-0.0007817 -0.0072771

Keterangan :

n = Jumlah Titik

m = Jumlah Foto

MILIK
PERPUSTAKAAN
ITN MALANG

LAMPIRAN 2

Contoh Hasil Perhitungan *Matrix* $\dot{N}_{ij}, \ddot{N}_{ij}, \overline{N}_{ij}, \dot{C}i_j, \ddot{C}_{ij}, G$

A. Contoh hasil perhitungan matrix $\dot{N}_{ij}$

FOTO KE 1	3493.7	28.443	-217.2	-14.64	3750.4	111.56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	28.443	3631.9	97.694	-3823	14.641	256.17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-217.2	97.694	116.09	-105.7	-234.9	0.0005	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-14.64	-3823	-105.7	4027.1	-0.003	-254	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3750.4	14.641	-234.9	-0.003	4027.1	114.24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	111.56	256.17	0.0005	-254	114.24	135.77	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FOTO KE 1	0	0	0	0	0	0	3606.6	-542.8	-488.6	500.44	2978.1	-2518	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	-542.8	3777.5	-171.7	-3963	554.41	837.89	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	-488.6	-171.7	181	181.17	-475.8	305.19	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	500.44	-3963	181.17	4185.9	-585.8	-822.5	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	2978.1	554.41	-475.8	-585.8	2897.3	-1916	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	-2518	837.89	305.19	-822.5	-1916	1826.5	0	0	0	0	0	0	0	0	0	0	0	0
FOTO KE 1	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	0	0	0	0	0	0
FOTO KE m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1905.9	-236.3	-459	24.9	1434.3	-2680
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-236.3	3734.8	329.19	-3308	-2555	-1028
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-459	329.19	175.74	-262.5	-530.5	535.25
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24.9	-3308	-262.5	3097.2	2119.2	1207.2

	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1434.3	-2555	-530.5	2119.2	2649.8	-1163
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-2680	-1028	535.25	1207.2	-1163	4329.1

Keterangan :

n = Jumlah Titik

m = Jumlah Foto

B. Contoh hasil perhitungan matrix $\ddot{N}_{ij}$

TITIK KE 1	1653.06	184.349	757.334	0	0	0	0	0	0	0	0	0	0	0	0
	184.349	2867.76	-377.93	0	0	0	0	0	0	0	0	0	0	0	0
	757.334	-377.93	2349.51	0	0	0	0	0	0	0	0	0	0	0	0
TITIK KE 2	0	0	0	1317.21	218.043	-9.7961	0	0	0	0	0	0	0	0	0
	0	0	0	218.043	2160.3	-180.99	0	0	0	0	0	0	0	0	0
	0	0	0	-9.7961	-180.99	1564.33	0	0	0	0	0	0	0	0	0
TITIK KE 3	0	0	0	0	0	0	1720.96	202.863	61.7121	0	0	0	0	0	0
	0	0	0	0	0	0	202.863	2167.41	-180.1	0	0	0	0	0	0
	0	0	0	0	0	0	61.7121	-180.1	1377.63	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	...	...	...	0	0	0
	0	0	0	0	0	0	0	0	0	...	...	...	0	0	0
	0	0	0	0	0	0	0	0	0	...	...	...	0	0	0
TITIK KE n	0	0	0	0	0	0	0	0	0	0	0	0	1877.21	317.234	-53.135
	0	0	0	0	0	0	0	0	0	0	0	0	317.234	2053.4	-479.86
	0	0	0	0	0	0	0	0	0	0	0	0	-53.135	-479.86	1467.3

Keterangan :

n = Jumlah Titik

m = Jumlah Foto

C. Contoh hasil perhitungan matrix $\overline{N}_{ij}$

	TITIK KE 1			TITIK KE 2			...			TITIK KE n		
FOTO KE 1	13.7915	-379.47	-56.06	2.55167	-465.39	-67.536	...	...	...	5.24015	-617.95	118.233
	399.21	-13.793	-109.06	460.082	-2.552	-27.205	...	...	...	613.044	-5.2396	75.2332
	51.3816	99.9588	-0.0001	53.9218	21.7244	0.00044	...	...	...	-70.861	-45.087	-0.0006
	-402.72	0.00115	108.099	-496.45	0.0007	28.9575	...	...	...	-659.38	-0.0012	-79.847
	0.00115	-402.71	-55.565	0.00079	-496.45	-71.885	...	...	...	-0.0014	-659.38	125.483
	108.099	-55.565	-36.683	23.4898	-58.312	-9.8135	...	...	...	-48.766	76.6376	-20.49
FOTO KE 2	-109.79	-287.22	193.878	-49.346	-334.5	231.309	...	...	...	0.53712	-280.54	366.516
	381.866	-95.942	-130.02	408.112	-57.073	-80.718	...	...	...	425.536	-26.822	-38.913
	15.9593	115.328	-64.759	10.0244	47.7247	-33.863	...	...	...	-69.97	3.90544	7.05815
	-374.43	87.5939	130.709	-464.36	62.6105	93.3529	...	...	...	-443.17	32.3077	34.8095
	100.318	-334.05	119.379	62.6105	-347.56	207.484	...	...	...	33.4819	-211.72	270.848
	155.557	156.704	-150.29	93.3529	207.484	-161.58	...	...	...	24.4223	198.939	-264.22
FOTO KE 3	1.02873	211.704	4.68687	3.41207	99.8418	18.2407	...	...	...	-11.006	64.6791	148.433
	264.506	612.454	570.913	29.068	445.103	392.07	...	...	...	48.0744	352.147	292.564
	53.1396	-308.62	109.611	-1.4016	-163.9	64.2364	...	...	...	-21.446	-119.2	-74.343
	222.795	-13.024	474.65	6.46617	-22.178	157.956	...	...	...	-73.941	-174.6	94.1524
	-36.464	-1900.1	-100.1	-24.91	-896.8	-35.17	...	...	...	-175.28	-565.86	-2.0965
	-866.77	91.6223	-1846.1	-40.796	-6.6761	-911.01	...	...	...	131.554	-7.5998	-639.31
FOTO KE 4	7.21292	-232.49	347.192	-38.603	-128.21	262.818	...	...	...	-63.123	-52.362	257.616
	474.673	292.645	133.964	287.536	304.253	117.21	...	...	...	246.875	246.896	93.4029

	54.1497	114.138	-102.35	38.0615	49.1011	-12.628	...	...	...	17.7495	15.8105	-44.02
	-581.05	-314.73	-227.34	-239.46	-238.75	-144.26	...	...	...	-171.79	-172.1	-72.644
	-313.14	-408.71	225.763	-239.35	-321.12	118.776	...	...	...	-175.09	-169.9	69.9686
	-252.53	246.185	-656.69	-74.316	63.1915	-482.5	...	...	...	13.7349	0.65465	-336.84

Keterangan :

n = Jumlah Titik

m = Jumlah Foto

D. Contoh hasil perhitungan matrix $\dot{C}i_j$

FOTO KE 1	0.0263971
	0.0329618
	-0.0055211
	-0.0514526
	0.0288142
	0.0132795
FOTO KE 2	0.0512961
	0.0214933
	-0.0196667
	-0.0104456
	0.0704859
	-0.0266641
FOTO KE 3	0.0167093
	0.0481653
	-0.0217721
	-0.0749476
	-0.1218708
	0.0525071
FOTO KE 4	0.0350156
	0.0070397
	-0.0288198
	0.0196601

	0.0031402
	-0.0610936

Keterangan :

n = Jumlah Titik

m = Jumlah Foto

E. Contoh hasil perhitungan matrix $\ddot{C}_{ij}$

TITIK KE 1	0.0057510
	-0.0030539
	0.0040152
TITIK KE 2	0.0019437
	-0.0007378
	-0.0006032
TITIK KE 3	-0.0006947
	-0.0008571
	-0.0025334
TITIK KE 4	0.0056485
	-0.0012145
	0.0037237
TITIK KE 5	0.0014928
	-0.0005827
	-0.0016786
TITIK KE 6	-0.0012373
	-0.0004796
	-0.0033090
TITIK KE 7	0.0036256
	0.0020502
	-0.0011998
TITIK	0.0009411

KE 8	0.0024854
	-0.0040567
TITIK KE 9	0.0041606
	0.0039995
	-0.0014267
TITIK KE 10	0.0011462
	0.0048707
	-0.0050818

Keterangan :

n = Jumlah Titik

m = Jumlah Foto

F. Contoh hasil perhitungan matrix *G*

TITIK KE 1	1	0	0	0	-0.9247	-0.1276	-0.2482
	0	1	0	0.92466	0	-0.2482	0.12759
	0	0	1	0.12759	0.24821	0	-0.9247
TITIK KE 2	1	0	0	0	-0.7501	-0.1086	-0.0438
	0	1	0	0.75007	0	-0.0438	0.10861
	0	0	1	0.10861	0.04376	0	-0.7501
TITIK KE 3	1	0	0	0	-0.6106	-0.0813	0.10973
	0	1	0	0.61056	0	0.10973	0.08133
	0	0	1	0.08133	-0.1097	0	-0.6106
TITIK KE 4	1	0	0	0	-0.8946	-0.0552	-0.2648
	0	1	0	0.89455	0	-0.2648	0.05516
	0	0	1	0.05516	0.26477	0	-0.8946
TITIK KE 5	1	0	0	0	-0.7252	-0.0545	-0.0601
	0	1	0	0.72522	0	-0.0601	0.05451
	0	0	1	0.05451	0.0601	0	-0.7252
TITIK KE 6	1	0	0	0	-0.5931	-0.0428	0.1002
	0	1	0	0.59311	0	0.1002	0.04282
	0	0	1	0.04282	-0.1002	0	-0.5931
TITIK KE 7	1	0	0	0	-0.7732	0.00494	-0.1513
	0	1	0	0.77323	0	-0.1513	-0.0049
	0	0	1	-0.0049	0.15129	0	-0.7732

TITIK KE 8	1	0	0	0	-0.5795	0.01166	0.08708
	0	1	0	0.57947	0	0.08708	-0.0117
	0	0	1	-0.0117	-0.0871	0	-0.5795
TITIK KE 9	1	0	0	0	-0.7704	0.08363	-0.1805
	0	1	0	0.77043	0	-0.1805	-0.0836
	0	0	1	-0.0836	0.18046	0	-0.7704
TITIK KE 10	1	0	0	0	-0.5647	0.10747	0.06838
	0	1	0	0.56472	0	0.06838	-0.1075
	0	0	1	-0.1075	-0.0684	0	-0.5647

Keterangan :

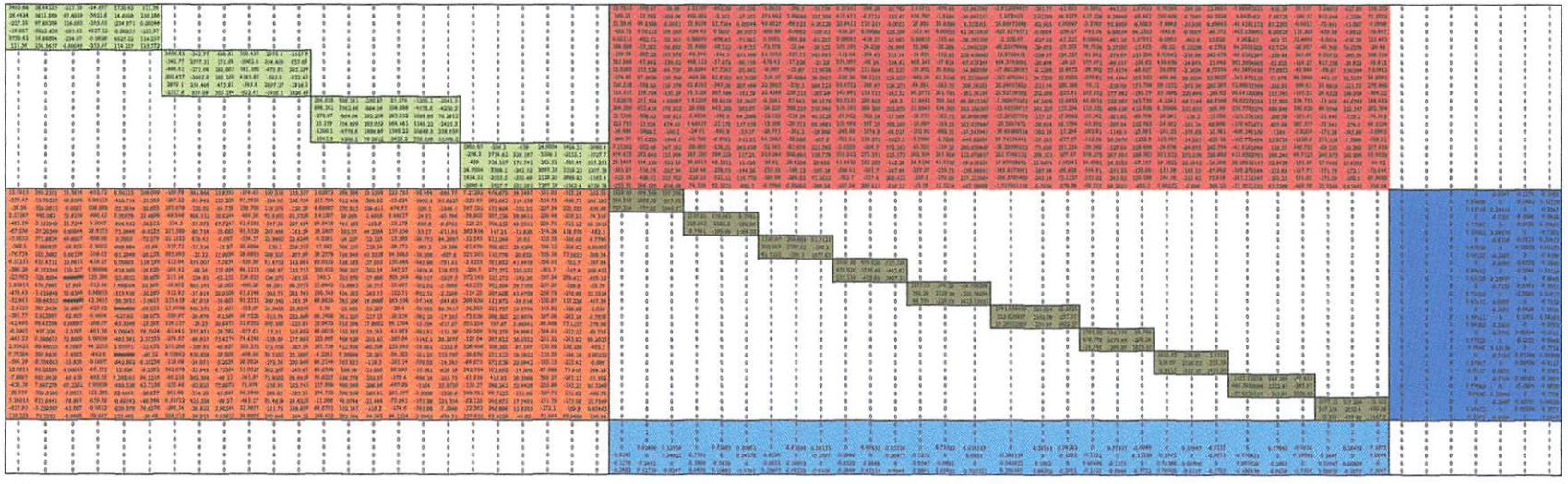
n = Jumlah Titik

m = Jumlah Foto

G. Contoh bentuk matrix *T Normal*

Matrix (C_i)	FOTO KE 1	0.0263971
		0.0329618
		-0.0055211
		-0.0514526
		0.0288142
		0.0132795
	FOTO KE 2	0.0512961
		0.0214933
		-0.0196667
		-0.0104456
		0.0704859
		-0.0266641
	FOTO KE 3	0.0167093
		0.0481653
		-0.0217721
		-0.0749476
		-0.1218708
		0.0525071
	FOTO KE 4	0.0350156
		0.0070397
		-0.0288198
		0.0196601
		0.0031402
		-0.0610936
Matrix (C_j)	TITIK KE 1	0.0057510
		-0.0030539
		0.0040152
	TITIK KE 2	0.0019437
		-0.0007378
		-0.0006032
	TITIK KE 3	-0.0006947
		-0.0008571
		-0.0025334
	TITIK KE 4	0.0056485
		-0.0012145
		0.0037237
	TITIK KE 5	0.0014928
		-0.0005827
		-0.0016786
	TITIK KE 6	-0.0012373
		-0.0004796
		-0.0033090
	TITIK KE 7	0.0036256
		0.0020502
		-0.0011998
	TITIK KE 8	0.0009411
		0.0024854
		-0.0040567
	TITIK KE 9	0.0041606
		0.0039995
		-0.0014267
	TITIK KE 10	0.0011462
		0.0048707
		-0.0050818
		0
		0
		0
		0
		0
		0
		0

H. Contoh bentuk matrix *N Normal*



Keterangan:

= Matrix $\dot{N}_{v}$

= Matrix $\overline{N}_{v}$

= Matrix $\dot{N}^{*}_{v}$

= Matrix G^{T}

= Matrix $\overline{N}_{v}^{T}$

= Matrix G

LAMPIRAN 3

Contoh Hasil Perhitungan Koreksi δ_1 dan δ_2 Parameter EO Dan Koordinat Obyek 3D Akhir

A. Contoh hasil perhitungan matrix δ_1

FOTO KE 1	0.0064947
	-0.0301152
	-0.0006849
	-0.0283493
	-0.0060862
	0.0039998
FOTO KE 2	-0.0082901
	-0.0046533
	-0.0005684
	-0.0043947
	0.0034663
	-0.0075727
FOTO KE 3	0.0033228
	-0.0000121
	0.0032229
	-0.0006091
	-0.0001480
	0.0001956
FOTO KE 4	0.0002917
	0.0005726
	0.0000103

	0.0004906
	0.0001122
	0.0002200

B. Contoh hasil perhitungan matrix δ_2

TITIK KE 1	0.00025706
	-0.00006274
	0.00001672
TITIK KE 2	-0.00027626
	0.00015356
	0.00004212
TITIK KE 3	0.00015603
	0.00018512
	-0.00000614
TITIK KE 4	0.00029981
	-0.00013188
	-0.00001228
TITIK KE 5	-0.00028810
	0.00009322
	0.00000993
TITIK KE 6	0.00011567
	0.00007540
	-0.00004687
TITIK KE 7	-0.00018848
	0.00002665
	0.00005561
TITIK KE 8	0.00006622

	-0.00007060
	-0.00007193
TITIK KE 9	-0.00014933
	0.00001642
	0.00005332
TITIK KE 10	0.00000737
	-0.00028516
	-0.00004048

C. Contoh hasil parameter EO akhir

Foto	Omega (radians)	Phi (radians)	Kappa (radians)	XL (milimeter)	YL (milimeter)	ZL (milimeter)
1	0.01946	-0.0904	-0.0021	-0.0851	-0.0182	0.01201
2	-0.6257	0.08088	0.12478	0.14406	0.5711	-0.0762
3	-0.8793	-1.3863	-1.0034	-0.6827	0.12475	-0.7211
4	-1.2138	-0.8673	-1.1874	-0.6391	0.61218	-0.5916

D. Contoh hasil koordinat obyek 3 dimensi akhir

Titik	Koordinat X (milimeter)	Koordinat Y (milimeter)	Koordinat Z (milimeter)
1	-0.2484	-0.2484	-0.2484
2	-0.043709	-0.043709	-0.043709
3	0.108901	0.108901	0.108901
4	-0.264313	-0.264313	-0.264313
5	-0.059975	-0.059975	-0.059975
6	0.100665	0.100665	0.100665
7	-0.150739	-0.150739	-0.150739
8	0.08706	0.08706	0.08706
9	-0.179556	-0.179556	-0.179556
10	0.067984	0.067984	0.067984

LAMPIRAN 4

Kode Keseluruhan Program

A. Source code untuk class Program

Class program bertujuan untuk memjalankan program yang telah dibuat.

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

namespace Bundle_Adjustment_Multi
{
    class Program
    {
        static void Main(string[] args)
        {

            //Masukan File
            string word = null;
            string jumlahfoto;
            Console.WriteLine("Masukan path file = ");
            word = Console.ReadLine();//untuk membaca path file

            TextReader tr = new StreamReader(word);
            Console.WriteLine(tr.ReadToEnd());
            jumlahfoto = Console.ReadLine();//untuk membaca jumlah foto.
            //Eksekusi Perhitungan
            Adjustment baca = new Adjustment(word);
            baca.data2D();
            Console.ReadLine();
        }
    }
}
```

B. Source code untuk class point2D

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

namespace Bundle_Adjustment_Multi
{
   class point2D
   {
      private string label;
      private double x;
      private double y;
      private double sx;
      private double sy;

      public string Label
      {
         get { return label; }
         set { label = value; }
      }
      public double X
      {
         get { return x; }
         set { x = value; }
      }
      public double Y
      {
         get { return y; }
         set { y = value; }
      }
      public double SX
      {
         get { return sx; }
         set { sx = value; }
```

```
        }
        public double SY
        {
            get { return sy; }
            set { sy = value; }
        }
    }
}
```

C. Source code untuk class point3D

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

namespace Bundle_Adjustment_Multi
{
    class Point3D
    {
        private string label;
        private double x;
        private double y;
        private double z;
        private double sx;
        private double sy;
        private double sz;

        public string Label
        {
            get { return label; }
            set { label = value; }
        }
        public double X
        {
            get { return x; }
            set { x = value; }
        }
        public double Y
        {
            get { return y; }
            set { y = value; }
        }
        public double Z
```

```
        {
            get { return z; }
            set { z = value; }
        }
        public double SX
        {
            get { return sx; }
            set { sx = value; }
        }
        public double SY
        {
            get { return sy; }
            set { sy = value; }
        }
        public double SZ
        {
            get { return sz; }
            set { sz = value; }
        }
    }
}
```

D. Source code untuk class parameterEO

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

namespace Bundle_Adjustment_Multi
{
    class ParameterEO
    {
        private double xL;
        private double yL;
        private double zL;
        private double omega;
        private double phi;
        private double kappa;


        public double XL
        {
            get { return xL; }
            set { xL = value; }
```

```
        }
        public double YL
        {
            get { return yL; }
            set { yL = value; }
        }
        public double ZL
        {
            get { return zL; }
            set { zL = value; }
        }
        public double Omega
        {
            get { return omega; }
            set { omega = value; }
        }
        public double Phi
        {
            get { return phi; }
            set { phi = value; }
        }
        public double Kappa
        {
            get { return kappa; }
            set { kappa = value; }
        }
    }
}
```

E. Source code untuk class Photo

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

namespace Bundle_Adjustment_Multi
{
    class Photo
    {
        private string photoname;
        private ParameterEO parametereo;
        private List<point2D> ptList;
```

```
        public void InputData(string photName, ParameterEO paramEO,
List<point2D> p2Dlist)
        {
            this.photoname = photName;
            this.parametereo = paramEO;
            this.ptList = p2Dlist;
        }
        public string Photoname
        {
            get { return photoname; }
            set { photoname = value; }
        }
        public ParameterEO Parametereo
        {
            get { return parametereo; }
            set { parametereo = value; }
        }
        public List<point2D> ListPoint2D
        {
            get { return ptList; }
            set { ptList = value; }
        }
    }
}
```

F. Source code untuk class Adjustment

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

using Accord.Math;
using System.Collections;

namespace Bundle_Adjustment_Multi
{
    class Adjustment
    {
        #region Field
        List<double[,]> B1 = new List<double[,]>();
        List<double[,]> B2 = new List<double[,]>();
        List<double[,]> L = new List<double[,]>();
        public Photo foto = new Photo();
```

```
        List<Point3D> pt3D = new List<Point3D>();
        List<point2D> listpoint2D = new List<point2D>();
        List<ParameterEO> parametereo = new List<ParameterEO>();
        List<Photo> ph = new List<Photo>();
        double focal;
        int Jumlah_Foto;
        int Jumlah_Obyek;
        string fileName;
        #endregion

        #region properties
        public List<double[,]> MatrixL
        {
            get { return L; }
        }
        public List<double[,]> MatrixB1
        {
            get { return B1; }
        }
        public List<double[,]> MatrixB2
        {
            get { return B2; }
        }
        public List<Point3D> ListPoint3D
        {
            get { return pt3D; }
        }
        public List<Photo> ListPhoto
        {
            get { return ph; }

        }
        public List<point2D> ListPoint2D
        {
            get { return listpoint2D; }
        }
        public List<ParameterEO> ListPhotoEO
        {
            get { return parametereo; }
        }
        #endregion


    #region properties
        public List<double[,]> MatrixL
        {
            get { return L; }
```

```
}
public List<double[,]> MatrixB1
{
   get { return B1; }
}
public List<double[,]> MatrixB2
{
   get { return B2; }
}
public List<Point3D> ListPoint3D
{
   get { return pt3D; }
}
public List<Photo> ListPhoto
{
   get { return ph; }

}
public List<point2D> ListPoint2D
{
   get { return listpoint2D; }
}
public List<ParameterEO> ListPhotoEO
{
   get { return parametereo; }
}
#endregion

#region Baca data
public Adjustment(string fileName)
{
   this.fileName = fileName;
}
public bool BacaData()
{
   if (!File.Exists(this.fileName))
   {
      Console.WriteLine(String.Format("{} File Tidak Ada", this.fileName));
   }
   string[] bacaText = File.ReadAllLines(this.fileName);
   int rows = bacaText.Count();
   string[] pC;

   for (int i = 0; i < rows; i++)
   {
      String[] data = bacaText[i].Split(new char[] { ' ', '\t' });
      String[] PeO;
```

```
            String[] point2d;
            String[] point3d;
            if (i == 0)
            {
                pC = bacaText[i].Split(new char[] { ' ', '\t' });
                this.focal = double.Parse(pC[0]);
                this.Jumlah_Foto = int.Parse(pC[1]);
                this.Jumlah_Obyek = int.Parse(pC[2]);
                Console.WriteLine("Panjang Fokus ={0} , jumlah Foto ={1}", focal,
Jumlah_Foto);
            }
            else if (data.Length == 13)
            {
                PeO = bacaText[i].Split(new char[] { ' ', '\t' });
                inputEO(PeO);
            }
            else if (data.Length == 5)
            {
                point2d = bacaText[i].Split(new char[] { ' ', '\t' });
                baca_p2(point2d);
            }
            else if (data.Length == 7)
            {
                point3d = bacaText[i].Split(new char[] { ' ', '\t' });
                baca_p3(point3d);
            }
        }
        return true;
    }
    public void baca_p3(string[] Koordinat3D)
    {
        Point3D p3d = new Point3D();
        List<string> lst = new List<string>();
        foreach (string P3d in Koordinat3D)
        {
            lst.Add(P3d);
        }
        p3d.Label = lst[0];
        p3d.X = double.Parse(lst[1]);
        p3d.Y = double.Parse(lst[2]);
        p3d.Z = double.Parse(lst[3]);
        p3d.SX = double.Parse(lst[4]);
        p3d.SY = double.Parse(lst[5]);
        p3d.SZ = double.Parse(lst[6]);

        ListPoint3D.Add(p3d);
    }
```

```
public void baca_p2(string[] Koordinat2D)
{

   point2D pt2D = new point2D();
   List<string> lst = new List<string>();
   foreach (string a in Koordinat2D)
   {
      lst.Add(a);
   }
   pt2D.Label = lst[0];
   pt2D.X = double.Parse(lst[1]);
   pt2D.Y = double.Parse(lst[2]);
   pt2D.SX = double.Parse(lst[3]);
   pt2D.SY = double.Parse(lst[4]);
   ListPoint2D.Add(pt2D);
   foto.ListPoint2D = listpoint2D;
}
public void inputEO(string[] ParametrsEO)
{
   ParameterEO ParamsEO = new ParameterEO();
   List<string> lst = new List<string>();
   foreach (string s in ParametrsEO)
   {
      lst.Add(s);
   }
   ParamsEO.Omega = double.Parse(lst[1]);
   ParamsEO.Phi = double.Parse(lst[2]);
   ParamsEO.Kappa = double.Parse(lst[3]);
   ParamsEO.XL = double.Parse(lst[4]);
   ParamsEO.YL = double.Parse(lst[5]);
   ParamsEO.ZL = double.Parse(lst[6]);

   ListPhotoEO.Add(ParamsEO);
   foto.Parametereo = ParamsEO;
   //Console.WriteLine("omega ={0} , phi ={1}, kappa ={2}, xl ={3}, yl
={4}, zl ={5}", ParamsEO.Omega, ParamsEO.Phi, ParamsEO.Kappa,
ParamsEO.XL, ParamsEO.YL, ParamsEO.ZL);
}
#endregion

#region Metod
public void data2D()
{
BacaData();
data2D(ListPoint2D, ListPoint3D, ListPhotoEO);
}
```

```
    public void data2D(List<point2D> p2d, List<Point3D> p3d,
List<ParameterEO> eo)
    {
      #region
      for (int iter = 0; iter < 3; iter++)
      {
        double[,] NNormal = new double[((6 * ListPhotoEO.Count) + (3 *
ListPoint3D.Count) + 7), ((6 * ListPhotoEO.Count) + (3 * ListPoint3D.Count) +
7)];
        double[,] TNormal = new double[((6 * ListPhotoEO.Count) + (3 *
ListPoint3D.Count) + 7), 1];
        int d = iter + 1;
        int h = eo.Count;
        List<double[,]> bd1 = new List<double[,]>();
        List<double[,]> bd2 = new List<double[,]>();
        List<double[,]> ml = new List<double[,]>();
        double[,] B1total = new double[h * ListPoint3D.Count * 2, 6 * h];
        double[,] B2total = new double[h * ListPoint3D.Count * 2, 3 *
ListPoint3D.Count];
        double[,] Ltotal = new double[h * ListPoint3D.Count * 2, 1];
        double[,] Helmert = new double[3 * ListPoint3D.Count, 7];
        double[,] Standar_Deviasi = new double[1,1];
        Console.WriteLine("Iterasi : " + d);

        for (int j = 0; j < p3d.Count; j++)
        {
          #region Bentuk Bdots1,Bdots2,Dot3,Ldot
          double[,] Bdots1 = new double[2 * ListPhotoEO.Count, 6 *
ListPhotoEO.Count];
          double[,] Bdots2 = new double[2 * ListPhotoEO.Count, 3];
          double[,] Ldots = new double[2 * ListPhotoEO.Count, 1];
          double[,] Helmerts = new double[3, 7];
          #endregion

          #region Input B1,B2,L
          List<string> label = new List<string>();
          List<double> x = new List<double>();
          List<double> y = new List<double>();
          List<double> sx = new List<double>();
          List<double> sy = new List<double>();
          for (int i = 0; i < p2d.Count; i++)
          {
            if (p3d[j].Label == p2d[i].Label)
            {
              sx.Add(p2d[i].SX);
              sy.Add(p2d[i].SY);
              x.Add(p2d[i].X);
```

```
                y.Add(p2d[i].Y);
            }
        }
        List<double[,]> B1 = Bdot(ListPoint3D);
        List<double[,]> B2 = BBdots2(ListPoint3D);
        ml = Epsilon(p3d[j].X, p3d[j].Y, p3d[j].Z, x, y);
        foreach (double[,] a in B1)
        {
            MatrixB1.Add(a);
        }
        foreach (double[,] b in B2)
        {
            MatrixB2.Add(b);
        }
        foreach (double[,] s in ml)
        {
            MatrixL.Add(s);
        }
        #endregion

        #region print Matrix B1, Matrix B2, Matrix L
        //printmatrixDebug(this.MatrixB1[j], "Matirx B1");
        //printmatrixDebug(this.MatrixB2[j], "Matirx B2");
        //printmatrixDebug(this.MatrixL[j], "Matirx L");
        #endregion

        #region bdot,bdotdot,Ldot
        for (int g = 0; g < ListPhotoEO.Count; g++)
        {
            int n = j * ListPhotoEO.Count;
            int kolom = g * 6;
            int baris = g * 2;

            #region review
            double[,] mBdot = MatrixB1[n + g];
            Bdots1[baris, kolom] = mBdot[0, 0]; Bdots1[baris, kolom + 1] =
mBdot[0, 1]; Bdots1[baris, kolom + 2] = mBdot[0, 2]; Bdots1[baris, kolom + 3] =
mBdot[0, 3]; Bdots1[baris, kolom + 4] = mBdot[0, 4]; Bdots1[baris, kolom + 5] =
mBdot[0, 5];
            Bdots1[baris + 1, kolom] = mBdot[1, 0]; Bdots1[baris + 1, kolom
+ 1] = mBdot[1, 1]; Bdots1[baris + 1, kolom + 2] = mBdot[1, 2]; Bdots1[baris +
1, kolom + 3] = mBdot[1, 3]; Bdots1[baris + 1, kolom + 4] = mBdot[1, 4];
Bdots1[baris + 1, kolom + 5] = mBdot[1, 5];

            double[,] mBBdots2 = MatrixB2[n + g];
            Bdots2[baris, 0] = mBBdots2[0, 0]; Bdots2[baris, 1] =
mBBdots2[0, 1]; Bdots2[baris, 2] = mBBdots2[0, 2];
```

```
                Bdots2[baris + 1, 0] = mBBdots2[1, 0]; Bdots2[baris + 1, 1] =
mBBdots2[1, 1]; Bdots2[baris + 1, 2] = mBBdots2[1, 2];

                double[,] mldot = MatrixL[n + g];
                Ldots[baris, 0] = mldot[0, 0];
                Ldots[baris + 1, 0] = mldot[1, 0];
                #endregion
            }
            bd1.Add(Bdots1);
            bd2.Add(Bdots2);
            #endregion

            #region Matrix Bdots1,Bdots2,Ldots
            //printmatrixDebug(Bdots1, "Bdots1");
            //printmatrixDebug(Bdots2, "Bdots2");
            //printmatrixDebug(Ldots, "matrix lBdots");

            double[,] bdoet1 = bd1[j];
            double[,] bdoet2 = bd2[j];
            for (int Pe0 = 0; Pe0 < ListPhotoEO.Count; Pe0++)
            {
                int baris1 = j * (h * 2);
                int m = j * 3;
                for (int baris2 = 0; baris2 < ListPhotoEO.Count * 2; baris2++)
                {
                    for (int kolom = 0; kolom < ListPhotoEO.Count * 6; kolom++)
                    {
                        B1total[baris1 + baris2, kolom] = bdoet1[baris2, kolom];
                    }

                    B2total[baris1 + baris2, m] = bdoet2[baris2, 0]; B2total[baris1 +
baris2, m + 1] = bdoet2[baris2, 1]; B2total[baris1 + baris2, m + 2] =
bdoet2[baris2, 2];
                    Ltotal[baris1 + baris2, 0] = Ldots[baris2, 0];
                }
            }
            #endregion

            #region Matrix Helmert
            Helmerts[0, 0] = 1; Helmerts[0, 4] = ListPoint3D[j].Z; Helmerts[0, 5]
= -ListPoint3D[j].Y; Helmerts[0, 6] = ListPoint3D[j].X;
            Helmerts[1, 1] = 1; Helmerts[1, 3] = -ListPoint3D[j].Z; Helmerts[1,
5] = ListPoint3D[j].X; Helmerts[1, 6] = ListPoint3D[j].Y;
            Helmerts[2, 2] = 1; Helmerts[2, 3] = ListPoint3D[j].Y; Helmerts[2, 4]
= -ListPoint3D[j].X; Helmerts[2, 6] = ListPoint3D[j].Z;
            //printmatrixDebug(Helmerts, "matrix helmert");
```

```
            int barish = 3 * j;
            Helmert[barish, 0] = Helmerts[0, 0]; Helmert[barish, 1] = Helmerts[0,
1]; Helmert[barish, 2] = Helmerts[0, 2]; Helmert[barish, 3] = Helmerts[0, 3];
Helmert[barish, 4] = Helmerts[0, 4]; Helmert[barish, 5] = Helmerts[0, 5];
Helmert[barish, 6] = Helmerts[0, 6];
            Helmert[barish + 1, 0] = Helmerts[1, 0]; Helmert[barish + 1, 1] =
Helmerts[1, 1]; Helmert[barish + 1, 2] = Helmerts[1, 2]; Helmert[barish + 1, 3] =
Helmerts[1, 3]; Helmert[barish + 1, 4] = Helmerts[1, 4]; Helmert[barish + 1, 5] =
Helmerts[1, 5]; Helmert[barish + 1, 6] = Helmerts[1, 6];
            Helmert[barish + 2, 0] = Helmerts[2, 0]; Helmert[barish + 2, 1] =
Helmerts[2, 1]; Helmert[barish + 2, 2] = Helmerts[2, 2]; Helmert[barish + 2, 3] =
Helmerts[2, 3]; Helmert[barish + 2, 4] = Helmerts[2, 4]; Helmert[barish + 2, 5] =
Helmerts[2, 5]; Helmert[barish + 2, 6] = Helmerts[2, 6];
            #endregion
        }
        #region Print Matrix Herlmert, B1,B2,L
        double[,] HelmeTrans = Matrix.Transpose<double>(Helmert);
        //printmatrixDebug(Helmert, "total Helmert");
        //printmatrixDebug(B1total, "B1total");
        //printmatrixDebug(B2total, "B2 gabung");
        //printmatrixDebug(Ltotal, "L Gabung");
        #endregion

        #region Proses Hitung matrix N_DOT N_DOTDOT  N_STRIP T_DOT
T_DOTDOT
        double[,] N_DOT = TransposeM(B1total, B1total);
        double[,] N_DOTDOT = TransposeM(B2total, B2total);
        double[,] N_STRIP = TransposeM(B1total, B2total);
        double[,] N_STrans = Matrix.Transpose<double>(N_STRIP);
        double[,] T_DOT = TransposeM(B1total, Ltotal);
        double[,] T_DOTDOT = TransposeM(B2total, Ltotal);
        #endregion

        #region Print Out matrix N_DOT N_DOTDOT  N_STRIP T_DOT
T_DOTDOT
        //printmatrixDebug(N_DOT, "matrix NDOT");
        //printmatrixDebug(N_DOTDOT, "matrix N_DOTDOT");
        //printmatrixDebug(T_DOT, "matrix T DOT");
        //printmatrixDebug(T_DOTDOT, "Matrix T_DOTDOT");
        //printmatrixDebug(N_STRIP, "matrix N Strip");
        #endregion

        #region Bentuk Normal Equation
        #region Bentuk Normal Matrix
        for (int barisN = 0; barisN < ListPhotoEO.Count * 6; barisN++)
        {
```

```
            for (int kolomNdot = 0; kolomNdot < ListPhotoEO.Count * 6;
kolomNdot++)
            {
              NNormal[barisN, kolomNdot] = N_DOT[barisN,
kolomNdot];//ndot
            }
            for (int barisNT = 0; barisNT < ListPoint3D.Count * 3; barisNT++)
            {
              for (int kolomNST = 0; kolomNST < ListPhotoEO.Count * 6;
kolomNST++)
              {
                int barisNST = barisNT + ((ListPhotoEO.Count * 6));
                NNormal[barisNST, kolomNST] = N_STrans[barisNT,
kolomNST];//NStrip Transpose
              }
            }
            for (int kolomns = 0; kolomns < 3 * ListPoint3D.Count; kolomns++)
            {
              int kolomNS = kolomns + ((ListPhotoEO.Count * 6));
              NNormal[barisN, kolomNS] = N_STRIP[barisN, kolomns];
            }
            for (int barisn2 = 0; barisn2 < 3 * ListPoint3D.Count; barisn2++)
            {
              for (int kolomn2 = 0; kolomn2 < 3 * ListPoint3D.Count;
kolomn2++)
              {
                int kolomN2 = kolomn2 + ((ListPhotoEO.Count * 6));
                int barisN2 = barisn2 + ((ListPhotoEO.Count * 6));
                NNormal[barisN2, kolomN2] = N_DOTDOT[barisn2,
kolomn2];
              }
            }
            for (int barisgT = 0; barisgT < 7; barisgT++)
            {
              for (int kolomgT = 0; kolomgT < 3 * ListPoint3D.Count;
kolomgT++)
              {
                int kolomGT = kolomgT + ((ListPhotoEO.Count * 6));
                int barisGT = barisgT + ((ListPhotoEO.Count * 6) +
(ListPoint3D.Count * 3));
                NNormal[barisGT, kolomGT] = HelmeTrans[barisgT,
kolomgT];
              }
            }
            for (int barisg = 0; barisg < 3 * ListPoint3D.Count; barisg++)
            {
              for (int kolomg = 0; kolomg < 7; kolomg++)
```

```
                {
                    int kolomG = kolomg + ((ListPhotoEO.Count * 6) +
(ListPoint3D.Count * 3));
                    int barisG = barisg + ((ListPhotoEO.Count * 6));
                    NNormal[barisG, kolomG] = Helmert[barisg, kolomg];
                }
            }

        }
        #endregion

        #region Bentuk T Normal Matrix
        for (int baris = 0; baris < ListPhotoEO.Count * 6; baris++)
        {
            TNormal[baris, 0] = T_DOT[baris, 0];
        }
        for (int S = 0; S < 3 * ListPoint3D.Count; S++)
        {
            int baris = S + ((ListPhotoEO.Count * 6));
            TNormal[baris, 0] = T_DOTDOT[S, 0];
        }
        #endregion
        //printmatrixDebug(TNormal, "matrix T Normal");
        //printmatrixDebug(NNormal, "matrix N Normal");
        #endregion

        #region Matrix X dan Standar Deviasi
        double[,] Matrix_X = InverseM(NNormal, TNormal);
        printmatrixDebug(Matrix_X, "Matrix_X");

        double[,] VTV = TransposeM(Ltotal, Ltotal);
        Standar_Deviasi =
Accord.Math.Matrix.Sqrt(Accord.Math.Matrix.Divide(VTV, ((2 * Jumlah_Foto *
Jumlah_Obyek) - (6 * Jumlah_Foto + 3 * Jumlah_Obyek))));
        printmatrixDebug(Standar_Deviasi, " Standar Deviasi");
        #endregion

        #region Koreksi Parameter
        for (int i = 0; i < Jumlah_Foto; i++)
        {

            ListPhotoEO[i].Omega = ListPhotoEO[i].Omega + Matrix_X[i*6, 0];
            ListPhotoEO[i].Phi = ListPhotoEO[i].Phi + Matrix_X[i * 6 + 1, 0];
            ListPhotoEO[i].Kappa = ListPhotoEO[i].Kappa + Matrix_X[i * 6 + 2,
0];
            ListPhotoEO[i].XL = ListPhotoEO[i].XL + Matrix_X[i * 6 + 3, 0];
            ListPhotoEO[i].YL = ListPhotoEO[i].YL + Matrix_X[i * 6 + 4, 0];
```

```
            ListPhotoEO[i].ZL = ListPhotoEO[i].ZL + Matrix_X[i * 6 + 5, 0];
            Console.WriteLine("Omega ={0},Phi ={1},Kappa ={2},XL ={3},YL
={4},ZL ={5}", ListPhotoEO[i].Omega, ListPhotoEO[i].Phi,
ListPhotoEO[i].Kappa, ListPhotoEO[i].XL, ListPhotoEO[i].YL,
ListPhotoEO[i].ZL);
            Console.WriteLine("Koreksi Omega ={0},Koreksi Phi ={1},Koreksi
Kappa ={2},Koreksi XL ={3},Koreksi YL ={4},Koreksi ZL ={5}", Matrix_X[i *
6, 0], Matrix_X[i * 6 + 1, 0], Matrix_X[i * 6 + 2, 0], Matrix_X[i * 6 + 3, 0],
Matrix_X[i * 6 + 4, 0], Matrix_X[i * 6 + 5, 0]);
          }
          for (int l = 0; l < ListPoint3D.Count; l++)
          {
            ListPoint3D[l].X = ListPoint3D[l].X + Matrix_X[l + 1 +
(Jumlah_Foto * 6), 0];
            ListPoint3D[l].Y = ListPoint3D[l].Y + Matrix_X[l + 2 +
(Jumlah_Foto * 6), 0];
            ListPoint3D[l].Z = ListPoint3D[l].Z + Matrix_X[l + 3 +
(Jumlah_Foto * 6), 0];
            Console.WriteLine("Xkoreksi ={0},Ykoreksi ={1},Zkoreksi ={2}",
Matrix_X[l + 1 + (Jumlah_Foto * 6), 0], Matrix_X[l + 2 + (Jumlah_Foto * 6), 0],
Matrix_X[l + 3 + (Jumlah_Foto * 6), 0]);
            Console.WriteLine("X ={0},Y ={1},Z ={2}", ListPoint3D[l].X,
ListPoint3D[l].X, ListPoint3D[l].X);
          }


        #endregion

      }
      #endregion
    }
    #endregion

    #region Inverse,Transpose
    double[,] InverseM(double[,] a, double[,] b)
    {
      double[,] hasil = Matrix.Multiply(Matrix.PseudoInverse(a), b);
      return hasil;
    }

    double[,] TransposeM(double[,] a, double[,] b)
    {
      double[,] result = Matrix.Multiply(Matrix.Transpose<double>(a), b);
      return result;
    }
    #endregion
```

```
#region Matriks B
public List<double[,]> Bdot(List<Point3D> p3d)
{
    List<double[,]> b1 = new List<double[,]>();
    for (int k = 0; k < p3d.Count; k++)
    {
        for (int i = 0; i < ListPhotoEO.Count; i++)
        {
            double[,] bdot = new double[2, 6];
            double[,] rot = new double[3, 3];
            double so = Math.Sin(ListPhotoEO[i].Omega);
            double co = Math.Cos(ListPhotoEO[i].Omega);
            double sp = Math.Sin(ListPhotoEO[i].Phi);
            double cp = Math.Cos(ListPhotoEO[i].Phi);
            double sk = Math.Sin(ListPhotoEO[i].Kappa);
            double ck = Math.Cos(ListPhotoEO[i].Kappa);

            //menghitung elements matriks rotasi dengan mengunakan data
parameter EO
            double m11 = cp * ck; double m12 = so * sp * ck + co * sk; double
m13 = -co * sp * ck + so * sk;
            double m21 = -cp * sk; double m22 = -so * sp * sk + co * ck; double
m23 = co * sp * sk + so * ck;
            double m31 = sp; double m32 = -so * cp; double m33 = co * cp;

            rot[0, 0] = m11; rot[0, 1] = m12; rot[0, 2] = m13;
            rot[1, 0] = m21; rot[1, 1] = m22; rot[1, 2] = m23;
            rot[2, 0] = m31; rot[2, 1] = m32; rot[2, 2] = m33;

            double DX = p3d[k].X - ListPhotoEO[i].XL, DY = p3d[k].Y -
ListPhotoEO[i].YL, DZ = p3d[i].Z - ListPhotoEO[i].ZL;
            //r,s,q
            double r = rot[0, 0] * DX + rot[0, 1] * DY + rot[0, 2] * DZ;
            double s = rot[1, 0] * DX + rot[1, 1] * DY + rot[1, 2] * DZ;
            double q = rot[2, 0] * DX + rot[2, 1] * DY + rot[2, 2] * DZ;
            //form b-dot(terhadap parameter unknown)
            bdot[0, 0] = (focal / (q * q)) * (r * (-m33 * DY + m23 * DZ) - q * (-
m13 * DY + m12 * DZ));
            bdot[0, 1] = (focal / (q * q)) * (r * (cp * DX + so * sp * DY - co * sp
* DZ) - q * ((-sp * ck * DX) + (so * cp * ck * DY) - (co * cp * ck * DZ)));
            bdot[0, 2] = -((focal / q) * (m21 * DX + m22 * DY + m23 * DZ));
            bdot[0, 3] = -((focal / (q * q)) * (r * m31 - q * m11));
            bdot[0, 4] = -(focal * (r * m32 - q * m12) / (q * q));
            bdot[0, 5] = -(focal * (r * m33 - q * m13) / (q * q));

            bdot[1, 0] = (focal / (q * q)) * (s * (-m33 * DY + m32 * DZ) - q * (-
m23 * DY + m22 * DZ));
```

```
                bdot[1, 1] = (focal / (q * q)) * (s * (cp * DX + so * sp * DY - co * sp
* DZ) - q * (sp * sk * DX - so * cp * sk * DY + co * cp * sk * DZ));
                bdot[1, 2] = (focal / q) * (m11 * DX + m12 * DY + m13 * DZ);
                bdot[1, 3] = -(focal * (s * m31 - q * m21) / (q * q));
                bdot[1, 4] = -(focal * (s * m32 - q * m22) / (q * q));
                bdot[1, 5] = -(focal * (s * m33 - q * m23) / (q * q));
                b1.Add(bdot);
            }
        }
        return b1;
    }
    public List<double[,]> BBdots2(List<Point3D> p3d)
    {
        List<double[,]> b2 = new List<double[,]>();
        for (int k = 0; k < p3d.Count; k++)
        {
            for (int i = 0; i < ListPhotoEO.Count; i++)
            {
                double[,] B2 = new double[2, 3];
                double[,] rot = new double[3, 3];
                double so = Math.Sin(ListPhotoEO[i].Omega);
                double co = Math.Cos(ListPhotoEO[i].Omega);
                double sp = Math.Sin(ListPhotoEO[i].Phi);
                double cp = Math.Cos(ListPhotoEO[i].Phi);
                double sk = Math.Sin(ListPhotoEO[i].Kappa);
                double ck = Math.Cos(ListPhotoEO[i].Kappa);

                //menghitung elements matriks rotasi dengan mengunakan data
parameter EO
                double m11 = cp * ck; double m12 = so * sp * ck + co * sk; double
m13 = -co * sp * ck + so * sk;
                double m21 = -cp * sk; double m22 = -so * sp * sk + co * ck; double
m23 = co * sp * sk + so * ck;
                double m31 = sp; double m32 = -so * cp; double m33 = co * cp;

                rot[0, 0] = m11; rot[0, 1] = m12; rot[0, 2] = m13;
                rot[1, 0] = m21; rot[1, 1] = m22; rot[1, 2] = m23;
                rot[2, 0] = m31; rot[2, 1] = m32; rot[2, 2] = m33;

                double DX = p3d[k].X - ListPhotoEO[i].XL, DY = p3d[k].Y -
ListPhotoEO[i].YL, DZ = p3d[k].Z - ListPhotoEO[i].ZL;
                //r,s,q
                double r = rot[0, 0] * DX + rot[0, 1] * DY + rot[0, 2] * DZ;
                double s = rot[1, 0] * DX + rot[1, 1] * DY + rot[1, 2] * DZ;
                double q = rot[2, 0] * DX + rot[2, 1] * DY + rot[2, 2] * DZ;

                B2[0, 0] = ((focal / (q * q)) * (r * m31 - q * m11));
```

```
                B2[0, 1] = (focal * (r * m32 - q * m12) / (q * q));
                B2[0, 2] = (focal * (r * m33 - q * m13) / (q * q));

                B2[1, 0] = (focal * (s * m31 - q * m21) / (q * q));
                B2[1, 1] = (focal * (s * m32 - q * m22) / (q * q));
                B2[1, 2] = (focal * (s * m33 - q * m23) / (q * q));
                b2.Add(B2);
            }
        }
        return b2;
    }
    #endregion

    #region Matriks L
    public List<double[,]> Epsilon(double xa, double ya, double za,
List<double> x, List<double> y)
    {
        List<double[,]> l = new List<double[,]>();
        for (int i = 0; i < ListPhotoEO.Count; i++)
        {
            double[,] ldot = new double[2, 1];
            double[,] M = new double[3, 3];
            double xo =0, yo = 0;
            double DX = xa - ListPhotoEO[i].XL, DY = ya - ListPhotoEO[i].YL,
DZ = za - ListPhotoEO[i].ZL;

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

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

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

```
            M[2, 2] = Math.Cos(ListPhotoEO[i].Omega) *
Math.Cos(ListPhotoEO[i].Phi);

            //r,s,q
            double r = M[0, 0] * DX + M[0, 1] * DY + M[0, 2] * DZ;
            double s = M[1, 0] * DX + M[1, 1] * DY + M[1, 2] * DZ;
            double q = M[2, 0] * DX + M[2, 1] * DY + M[2, 2] * DZ;

            ldot[0, 0] = (x[i] - xo) + (focal * r / q);
            ldot[1, 0] = (y[i] - yo) + (focal * s / q);
            l.Add(ldot);
        }
        return l;
    }
    #endregion

    #region PrintOut Matrik
    private void printmatrixDebug(double[,] a, string name)
    {
        int lower1 = a.GetLowerBound(0);
        int upper1 = a.GetUpperBound(0);
        int lower2 = a.GetLowerBound(1);
        int upper2 = a.GetUpperBound(1);
        Console.WriteLine("\n" + name + " : ");
        for (int i = lower1; i <= upper1; i++)
        {
            for (int j = lower2; j <= upper2; j++)
            {
                Console.Write(a[i, j].ToString() + "\t");
            }
            Console.WriteLine("");
        }
    }

    private void PrintVectorDebug(ref double[] v, string name)
    {
        int lower = v.GetLowerBound(0);
        int upper = v.GetUpperBound(0);
        int length = v.Length;
        Console.WriteLine("\n" + name + ":");
        for (int i = lower; i <= upper; i++)
        {
            Console.Write(v[i].ToString() + "\t");
        }

        Console.WriteLine("");
    }
```

```
        #endregion
    }
}
```