# SKRIPSI

**Pembuatan Paket Program Kalibrasi Foto Stereo Dengan Menggunakan Plat Datar Dan Bahasa Pemograman *Microsoft Visual Studio* C# 2008**


**Disusun Oleh :**

**YUSAK MNSEN**

**06.25.902**

**JURUSAN TEKNIK GEODESI**

**FAKULTAS TEKNIK SIPIL DAN PERENCANAAN**

**INSTITUT TEKNOLOGI NASIONAL**

**MALANG**


**2010**

# SKRIPSI

Pembuatan Pedal Program Realtime Pada Stereo Berbasis
Mikrokontroler Pilot Proyek Dan Efektor Pengolah Suara
Menggunakan Modul Studio Uc 32DSP

Disusun Oleh :

YUSAK MAULANA
06.52.005

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
INSTITUT TEKNOLOGI NASIONAL
MALANG
2010

# SKRIPSI

# Pembuatan Paket Program Kalibrasi Foto Stereo Dengan Menggunakan Plat Datar Dan Bahasa Pemograman *Microsoft Visual Studio* C# 2008

Diajukan untuk memenuhi persyaratan

dalam mencapai gelar sarjana S1 Teknik Geodesi


**Disusun Oleh :**

**YUSAK MNSEN**

**06.25.902**

**JURUSAN TEKNIK GEODESI**

**FAKULTAS TEKNIK SIPIL DAN PERENCANAAN**

**INSTITUT TEKNOLOGI NASIONAL**

**MALANG**


**2010**

# SKRIPSI

Pembuatan Paket Program Kalibrasi Foto Stereo Dengan
Menggunakan Plat Datar Dan Bahasa Pemograman
Microsoft Visual Studio C# 2008

Diajukan untuk memenuhi persyaratan
dalam memperoleh gelar sarjana ST Teknik Geodesi

Disusun Oleh :

YUSAK NUSI N

06.25.007

JURUSAN TEKNIK GEODESI

FAKULTAS TEKNIK SIPIL DAN PERENCANAAN

INSTITUT TEKNOLOGI NASIONAL

MALANG

2010

## LEMBAR PENGESAHAN

**Pembuatan Paket Program Kalibrasi Foto Stereo dengan Menggunakan Plat Datar dan Bahasa Pemograman *Microsoft Visual Studio C#2008***

**SKRIPSI**

Dipertahankan dihadapan Majelis Penguji Sidang Skripsi

Jenjang Starata Satu (S-1)

Pada hari : Jumat
Tanggal : 20 Agustus 2010
Dan diterima untuk memenuhi salah satu persyaratan guna memperoleh gelar Sarjana Teknik.

**Disusun oleh :**

**Yusak Mnsen 06.25.902**

**Panitia Ujian Tugas Akhir**

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

**Anggota Penguji**

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

**JURUSAN TEKNIK GEODESI**

**FAKULTAS TEKNIK SIPIL DAN PERENCANAAN**

**INSTITUT TEKNOLOGI NASIONAL**

**MALANG**

**2010**

**LEMBAR PERSETUJUAN**

**SKRIPSI**

**Pembuatan Paket Program Kalibrasi Foto Stereo dengan Menggunakan Plat Datar dan Bahasa Pemograman *Microsoft Visual Studio C#2008***

Diajukan Sebagai Salah Satu Syarat Memperoleh Gelar Sarjana Teknik Geodesi

S-1

Institut Teknologi Nasional Malang

Disusun Oleh :

**Yusak Mnsen**

**06.25.902**

**Meyetujui,**

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

**Mengetahui**

**Ketua Jurusan Teknik Geodesi S-1**

**Hery Purwanto, ST, M.Sc**

**Takut akan TUHAN adalah permulaan pengetahuan, tetapi orang bodoh menghina hikmat dan didikan.**
**(Amsal 1 : 7)**

Puji syukur kepada TUHAN YESUS yang maha pengasih dan penyang, yang sudah memberikan saya napas kehidupan, hikmad dan keindahan dalam segala kesenangan dan kesulitan saya.

Skripsi ini saya persembahkan buat semua yang menyayangi, mencintai dan mendidik saya sehingga saya bisa menempuh studi saya. -@Bapa Yeremias Mnsen(alm) yang selalu didalam sanubariku & Mama tercinta Paulian Kafiar yang selalu berdoa dan meneteskan air mata untuk keberhasilan anakmu ini, terima kasih banyak yusak sayang mama T_T; -@kk Amalek Mnsen sebagai pengganti tulang punggung dalam keluarga kami, yang memberi dorongan motivasi dan biaya tuk melanjutkan sekolah. @kk Nony, ponakan Jeremi, Gloria & Christian; -@kk Yansen Mnsen & kk Lina, ponakan Juan & Justin. -@ade Robertho Mnsen (Papen)lanjutkan ☺ ; -@tete S Kafiar; -@om Yoas Kafiar sekeluarga, -@om Pdt M. Kafiar sekeluarga; -@pd Bram Okoka sekeluarga, -@mm tua Aneta sekeluarga; -@om Frits sekeluarga; -@md Nace sekeluarga; -@md Yulia Sekeluarga; @pd Piter Auparai sekeluarga; -@pd Edu Dimo Sekeluarga; -@pd Maiko Koibur sekeluarga; -@kk Maikel Simbiak sekeluarga; @kk Leni Simbiak (almh); -@bpk Zius Moga sekeluarga; @Sanggar Gabriel; dan keluarga besar Mnsen-Kafiar yang tidak sebutkan satu-persatu; Terima kasih banyak tuk semua yang telah memebri dukungan doa, motivasi, semagat dan biaya kepada saya."GOD BLESS YOU ALL"

Ucapan terimakasih untuk para pendidik. -@Dosen pembimbing pak Edwin Tjahjadi (#_pak Boss), yang bersedia meluangkan waktu dan kesempatan dalam membimbing dan memberi arahan dalam penelitian selama ± 1,5 tahun; -@pak Hery Purwanto (#_the_Big Boss), sebagai bapaknya anak2nya Geodesi, terimakasih tuk motivasi dan bantuannya dalam penelitian. @Seluruh Dosen dan Staf di Jurusan Teknik Geodesi ITN-Malang.

Thank's for patner programmer (@sister #_Desi Saudale)tuk kerja samanya n akhirnya kita selesai jg projectnya hehehehe #_private, Void, Class ubi kayu _ect EErrroor…☺ //; @Team Rapid Mapping: //Trace.WriteLine (#_bli Agus, #_Ronald, #_pc Roger #_Via & #_Weny); @ Team Deformasi: //Trace_Writeline(#_Tanzil, #bli_Dody, #_Akbar, #_Alben, #_bli Gede & #_Eno); OverWrite_mathur suwun sing uaakeh nawak2 ☺.

Buat semua GEODET 03, GEODET 04, GEODET 05, GEODET 06 thank's yah, sudah menjadi bagian dari teman2 pache selama ini kontak2 eee klu ada proyek wkwkwkwkwk……Salam Satu Jiwa GEODESI ☺.

Ngombe kabeh sak gelas, mangan kabeh sak bungkus, mangan ora mangan sing penting ngumpul; inilah omah keluarga pache @FORMAT #_Fotografi Mahasiswa Teknik; #_Angkatan pdf 08, @Kading alias Kipli, @Christian tetep seperti dulu, @cak Gunarto, @Dodot PAP; @FMT 99.001 - FMT 12,…… ; #_tour malang-jogja-jember-bromo-sby-cangar-paralayang ect, hunting sampai sinting, mouting sampai mabok, ngombe sembarangalir, japrem, tura-turu ning embong, itulah kisah kita…….Big Family never End- #FORMAT mathur suwun sing uaaaakeh nawak-nawak ☺.

"SATU CITRA SERIBU BAHASA"

FMT 09.106

## Abstraksi

Kata Kunci : Distorsi Lensa, Stereo Kalibrasi, *EmguComputerVision CSharp MatrixLibrary (CSML).*

Distorsi lensa yang terdiri dari distorsi radial (K1, K2, K3), distorsi tangensial (P1, P2), dan distorsi akibat perbedaan penyekalaan dan ketidak ortogonal antara sumbu X dan Y (b1, b2) (Fraser, 1998).

Stereo kalibrasi adalah proses menghitung hubungan geometris antara dua kamera (Bradski & Kaehler, 2008). Geometri dari dua kamera ini berkaitan dengan nilai informasi parameter eksterinsik dan parameter instrinsik untuk tiap kamera. Parameter interinsik biasanya dipakai sebagai parameter untuk mendefinisikan nilai geometri dari kamera, sedangkan parameter ekstrinsik digunakan untuk menjelaskan hubungan secara geometri posisi kamera dan objek dalam suatu sistem koordinat referensi. Parameter ekstrinsik terdiri dari rotasi dan translasi.

*Emgu computer vision* merupakan pengembangan dari *OpenCV (Open Computer Vision) dimana OpenCV merupakan* suatu *library* yang dikembangkan oleh *developer-developer Intel Corporation. Library* ini terdiri dari fungsi-fungsi Komputer *Vision* dan *API (Aplication Programming Interface)* untuk *image processing high level* maupun *low level* dan sebagai optimasi aplikasi *realtime* (Wikipedia, 2009) .

*CSharp Matrix Library (CSML)* adalah suatu *class library* yang berbasis *.Net* yang menyediakan berbagai fungsi matriks untuk perhitungan aljabar liner. Komponen-komponen di dalam *CSharp Matrix Library* terdiri dari beberapa metode diantaranya adalah pengoprasian matriks (perkalian matriks, penjumlahan matriks, Eksponen dan sistem persamaan linear), manipulasi matriks (*Concatenation, insertion, transpose, inverse, flipping, simetrizing dan extraction*), dekomposisi matriks dan arimatika kompleks matriks (hanzzoid, 2007).

## PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini :

**Nama** : **Yusak Mnsen**

**NIM** : **06.25.902**

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

**Fakultas** : **Fakultas Teknik Sipil Dan Perencanaan**

Menyatakan dengan sesungguhnya bahwa Skripsi saya dengan judul :

**"Evaluasi Metode Penentuan Pendekatan Parameter *Exterior Orientation* (EO) dengan Menggunakan Metode *Closed Form Solution* untuk Pemotretan pada Multi Foto Konvergen (*Studi Kasus : Jembatan Rel Kereta Api Lawang dan Fly Over Arjosari Malang*)"** adalah hasil karya saya sendiri, bukan merupakan duplikat serta tidak mengutip atau menyadur dari hasil karya orang lain kecuali disebutkan sumbernya.

Malang, 29 September 2010

Yang membuat pernyataan


Yusak Mnsen

NIM : 06.25.902

# KATA PENGANTAR

Puji syukur penulis panjatkan kepada ALLAH SWT, karena berkat rahmat-Nya, penulis dapat menyelesaikan skripsi yang berjudul **"Pembuatan Paket Program Kalibrasi Foto Stereo dengan Menggunakan Plat Datar dan Bahasa Pemograman *Microsoft Visual Studio C#2008*"**, 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 Prof. Dr. Eng. Ir. Abraham Lomi, MSEE selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. A. Agus Santosa, MT selaku Dekan Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.
3. Bapak Heri Purwanto, ST., M.Sc. selaku Ketua Jurusan Teknik Geodesi Institut Teknologi Nasional Malang dan selaku Dosen Pembimbing I.
4. Bapak Dr. Edwin Tjahjadi, ST., MGeom.Sc. selaku Dosen Pembimbing II dan Dosen Penguji III.
5. Bapak Hery Purwanto, ST, M.Sc. selaku Dosen Penguji I.
6. Bapak Ir. Agus Darpono, MT. selaku Dosen Penguji II.
7. Segenap dosen, staff pengajar dan *recording* Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.

8. Bapak (alm), Ibu, Kakak dan Adikku, yang selalu memberikan dukungan, semangat dan doa.
9. Team *Rapid Mapping* dan team Deformasi yang selalu memberikan kerja sama dan dukungannya.
10. Keluarga besar Mnsen-Kafiar yang selalu memberikan dukungan, doa dan Semangat.
11. Keluarga besar Unit Kegiatan Mahasiswa FOTOGRAFI MAHASISWA TEKNIK (FORMAT) ITN Malang.
12. 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, 29 September 2010

Penulis

## DAFTAR ISI

## DAFTAR GAMBAR

## DAFTAR LAMPIRAN

# BAB I

# PENDAHULUAN



## I.1. Latar Belakang Penelitian

Perkembangan teknologi dan dunia digital membawa banyak dampak dalam bidang Fotogrametri. Beberapa tahun belakangan ini keberhasilan yang menonjol didalam fotogrametri yang disebabkan oleh perkembangan ketelitian kamera, karena kamera merupakan salah satu instrument yang terpenting dan alat yang utama dalam fotogrametri.

Didalam kamera terdapat bidang sensor *CCD (charge coupled device)* dan susunan lensa, dimana susunan lensa ini mentransformasikan gambaran atau suatu objek di permukaan bumi ke bidang foto dijital. Karena *CCD* dan susunan lensa yang tidak sempurna, sehingga proses perekaman yang dilakukan akan memiliki kesalahan. Oleh karena itu untuk mendapatkan nilai ukuran yang memiliki tingkat akurasi, presisi dan reliability yang tinggi maka kamera harus dilakukan kalibrasi. Karena proses kalibrasi tersebut dapat menentukan besarnya penyimpangan-penyimpangan yang terjadi serta dapat mengeliminasi kesalahan sistematik yang terdapat pada unsur intrinsik dari kamera tersebut. Kalibrasi kamera dilakukan untuk menentukan parameter distorsi (Nuraini, 2007).

Distorsi dengan kata lain adalah pembiasan, distorsi terjadi karena sinar terbelok atau berubah arahnya sehingga setelah sinar-sinar tersebut menembus lensa akan keluar dengan arah yang tidak sejajar lagi dengan arahnya sewaktu datang *(Wolf, 1993)*.

Distorsi lensa dapat menyebabkan bergesernya titik pada foto dari posisi yang sebenarnya, sehingga memberikan ketelitian pengukuran yang tidak baik, namun tidak mempengaruhi kualitas ketajaman citra yang dihasilkan. Distorsi lensa dapat dibagi menjadi distorsi radial dan distorsi tangensial serta distorsi affinity *(Wolf, 1993)*.

Dengan demikian untuk memudahkan dan menghemat waktu serta biaya yang tinggi, perlu adanya suatu kerangka berpikir yang sistematis dan dapat diadopsi oleh komputer sehingga lebih memudahkan dalam melakukan proses kalibrasi kamera non metrik, maka penulis bermaksud untuk menulis sebuah skripsi yang berjudul " ***Pembuatan Paket Program Kalibrasi Foto Stereo dengan menggunakan Plat Datar dan Bahasa Pemograman Microsoft Visual Studio C# 2008***"

## I.2. Rumusan Masalah

Berdasarkan latar belakang yang telah disampaikan sebelumnya, maka rumusan masalah yang akan dibahas dalam penelitian ini adalah bagaimana membuat suatu program aplikasi untuk mempermudah proses perhitungan kalibrasi foto *strereo*.

## I.3. Batasan Masalah

1. Pembuatan paket program kalibrasi foto stereo dengan menggunakan plat datar dan bahasa pemograman *Microsoft Visual Studio C# 2008*.
2. Program yang dibuat terbatas untuk perhitungan kalibrasi foto *strereo* yang menghasilkan parameter *Ekstinsic*, *Intrinsic* dan *Baseline*.

## I.4. Tujuan Penelitian

Tujuan dari penelitian adalah untuk memudahkan proses perhitungan kalibrasi foto *strereo* agar dalam proses pengolahan data mendapatkan hasil yang lebih cepat, teliti dan efisien.

## I.5. Manfaat Penelitian

Dari hasil penelitian ini diharapkan dapat memberikan manfaat yang cukup besar bagi pengguna yang banyak berkaitan dengan pekerjaan *close range photogrammetry* terutama dalam proses pengolahan data untuk menghitung parameter-parameter kalibrasi foto *strereo*.

## I.6. Tinjauan Pustaka

Beberapa tinjauan pustaka telah dilakukan dalam menyusun penelitian, guna mengumpulkan informasi mengenai proses *seft calibration*t dalam *close range photogrammetry* yang didasarkan atas berbagai riset oleh para ilmuan dalam bidang *close range photogrammetry* antara lain :

Pada umumnya kamera non-metrik tidak mempunyai susunan lensa yang sempurna, sehingga akan mengakibatkan terjadinya kesalahan pada proses perekaman (Nuraini, 2007).

Kalibrasi kamera dilakukan untuk menentukan parameter internal kamera (*IO*) yang meliputi *priciple distace* (c), titik pusat fidusial foto ($x_0$, $y_0$) (Dörstel, et. al, 2001).

Distorsi lensa yang terdiri dari distorsi radial ($K_1$, $K_2$, $K_3$), distorsi tangensial ($P_1$, $P_2$), dan distorsi akibat perbedaan penyekalaan dan ketidak ortogonal antara sumbu X dan Y ($b_1$, $b_2$) (Fraser, 1998).

# BAB II
# DASAR TEORI

## II.1 Pendahuluan

Pada umumnya kamera non-metrik tidak mempunyai susunan lensa yang sempurna, sehingga akan mengakibatkan terjadinya kesalahan pada proses perekaman (Nuraini 2007). Oleh karena itu perlu dilakukan pengkalibrasian kamera untuk dapat menentukan besarnya penyimpangan-penyimpangan yang terjadi. Kalibrasi kamera dilakukan untuk menentukan parameter internal kamera (*IO*) yang meliputi *priciple distace* (c), titik pusat fidusial foto ($x_0$, $y_0$) (Dörstel, et. al, 2001). Distorsi lensa yang terdiri dari distorsi radial ($K_1$, $K_2$, $K_3$), distorsi tangensial ($P_1$, $P_2$), dan distorsi akibat perbedaan penyekalaan dan ketidak ortogonal antara sumbu X dan Y ($b_1$, $b_2$) (Fraser, 1998).

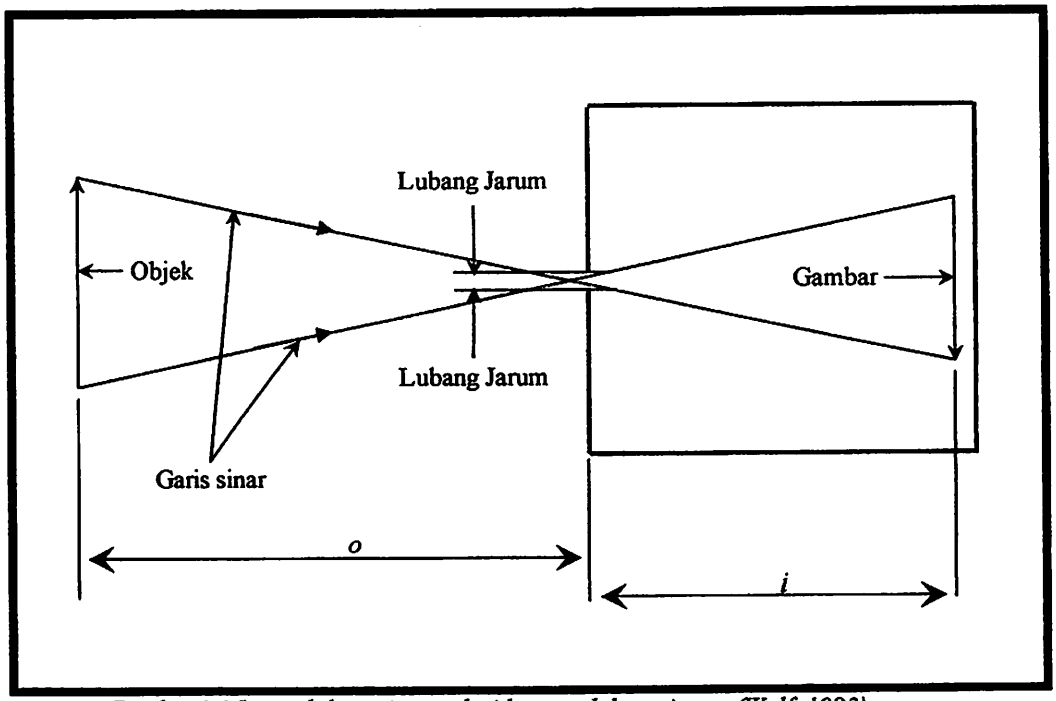
Maksud dari penulisan ini yang ingin dicapai yaitu untuk mengetahui model kamera serta defenisi dari parameter yang dikalibrasi, teori-teori yang digunakan untuk pengkalibrasian kamera sehingga dapat menentukan nilai yang tepat bagi sejumlah konstanta untuk perhitungan distorsi 2 buah kamera *CCTV*. Adapun tujuan yang ingin dicapai dari penulisan ini adalah membuat paket program kalibrasi foto *stereo* dengan menggunakan plat datar.

## II.2 Distorsi Lensa

Fungsi utama dari suatu lensa ialah mengumpulkan berkas sinar yang berasal dari seluruh titik yang membentuk objek dan mengumpulkannya kearah titik api yang terletak pada jarak tertentu di sisi lain di balik lensa (Wolf,1983). Perkakas yang paling mudah dan sederhana guna

memperagakan fungsi suatu lensa ialah sebuah lubang jarum yang secara teoretis memungkinkan masuknya sebuah sinar tunggal yang berasal dari setiap titik dari objek (Wolf, 1983).

Dengan alasan bahwa lubang jarum hanya memungkinkan untuk dilalui sinar yang sangat sedikit sehingga sinarnya sangat lemah dan tidak sesuai untuk pekerjaan fotogrametri, maka lubang jarum tersebut diganti dengan lensa kaca (Wolf, 1983). Keunggulan suatu lensa jika dibandingkan dengan dengan lubang jarum ialah adanya peningkatan jumlah sinar yang melaluinya. Sebuah lensa mengumpulkan seluruh berkas sinar yang berasal dari setiap titik pada objek dan bukan hanya sebuah sinar tunggal saja (Wolf, 1983).




*Gambar 2.1 Lensa lubang jarum dari kamera lubang jarum (Wolf, 1983).*
*Dimana o adalah jarak objek dan i adalah jarak gambar*

Ketidakstabilan susunan lensa yang akan mengakibatkan tidak berimpitnya sumbu fokus antar lensa (Luhmann et al, 2006). Sehingga, setelah sinar-sinar tersebut menembus lensa akan keluar dengan arah yang tidak sejajar lagi dengan arahnya sewaktu datang atau disebut distorsi lensa (Wolf, 1983). Distorsi lensa menyebabkan bergesernya titik dari posisi sebenarnya atau merusak nilai geometrinya sehingga memberikan ketelitian letak atau posisi pengukuran yang tidak baik (Wolf , 1983). Tetapi, distorsi lensa tidak dapat mengurangi kualitas atau ketajaman gambar (Wolf, 1983).

Apabila betul-betul terjadi distorsi yang disebabkan oleh lensa kamera, maka para ahli fotogrametri akan memperoleh kesalahan dalam pengukuran bagi letak-letak gambar dalam foto yang dihasilkannya. Distorsi lensa dibedakan menjadi tiga, yakni; distorsi radial ($K_1$, $K_2$, $K_3$), distorsi tangensialn ($P_1$, $P_2$), dan distorsi akibat perbedaan penyekalaan dan ketidak ortogonal antara sumbu X dan Y ($b_1$, $b_2$) (Fraser, 1998). Namun hanya distorsi radial yang memiliki pengaruh yang signifikan terhadap nilai geometri suatu gambar atau foto (Jedlička, Potůčková, 1999).

### II.2.1 Distorsi radial

Sesuai dengan namanya distorsi radial, menyebabkan semua bagian gambar diubah letaknya menurut arah jari-jari, bermula dari sumbu optik. Keadaan ini disebabkan oleh kesalahan dalam pengasahan bagian-bagian lensa. Nilai distorsi radial merupakan perpindahan secara radial suatu titik dari posisi sebenarnya terhadap posisi dari *principle point* ($x_p$, $y_p$), dengan indikator bila nilainya positif maka pergeserannya mengarah keluar dan jika nilainya negatif maka pergeserannya mengarah kedalam (Wolf, 1983).

Secara teoritik koreksi radial dilakukan setelah dilakukan reduksi gambar ketitik utama dan koreksi oleh kesalahan pengkerutan atau pemekaran. Jumlah pengkerutan atau pemekaran pada foto ditentukan di dalam kalibrasi kamera. Sehingga koordinat foto dapat dikoreksi. Bila $x_m$ dan $y_c$ merupakan ukuran fidusial pada positif, sedangkan $x_c$ dan $y_c$ merupakan jarak fidusial terkalibrasi, maka koordianat foto pada titik "a" dapat dihitung sebagai berikut:

$$x_a' = \left(\frac{x_c}{x_m}\right) x_a$$
$$y_a' = \left(\frac{y_c}{y_m}\right) y_a \qquad (2.1)$$

Dimana $x_a'$ dan $y_a'$ adalah koordinat foto terkoreksi dan $x_a$ dan $y_a$ merupakan koordinat terukur *(Wolf, 1993)*. Persamaan polinomial berdasarkan pada teori desain kamera adalah sebagai berikut (Wolf and Dewitt 2004) :

$$\Delta x = k_1 r^3 + K_2 r^5 + K_3 r^7$$
$$\Delta y = K_1 r^3 + K_2 r^5 + K_3 r^7 \qquad (2.2)$$

atau

$$d_r = K_1 r^3 + K_2 r^5 + K_3 r^7 + \ldots\ldots \qquad (2,3)$$

Dimana $K_1$, $K_2$, dan $K_3$ merupakan koofisien distorsi radial lensa, $r$ merupakan jarak radial dari pusat foto terkalibrasi yang didapat dari :

$$r = \sqrt{\bar{x}^2 + \bar{y}^2} \qquad (2.4)$$

Nilai $\bar{x}, \bar{y}$ didapat dari persamaan 2.4. Setelah nilai distorsi radial $\Delta x_r$ dan $\Delta y_r$ didapat, selanjutnya menghitung komponen koreksi yaitu $\delta_x$, $\delta_y$ sebagai berikut :

$$\frac{\Delta r}{r} = \frac{\delta x}{\bar{x}} = \frac{\delta y}{\bar{y}} \tag{2.5}$$

Atau dapat dijabarkan menjadi :

$$\delta x = \bar{x}\frac{\Delta r}{r}$$
$$\delta y = \bar{y}\frac{\Delta r}{r} \tag{2.6}$$

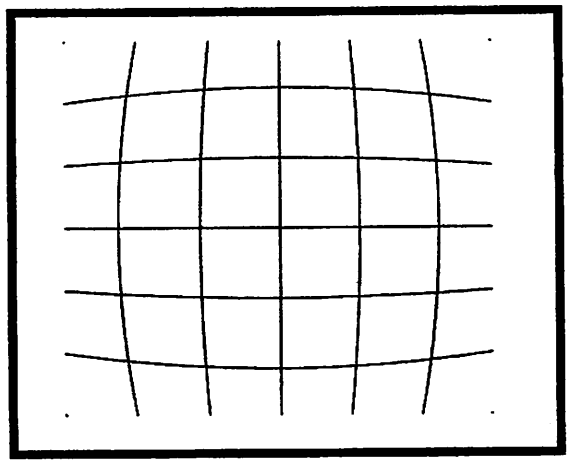
Posisi koordinat sebenarnya dapat dicari dengan menggunakan persamaan dibawah ini :

$$x = \bar{x} - \delta x$$
$$y = \bar{y} - \delta y \tag{2.7}$$

Ada dua jenis utama dari distorsi radial (Jedlička, Potůčková, 1999).

#### II.2.1.1 Distorsi *Barrel*

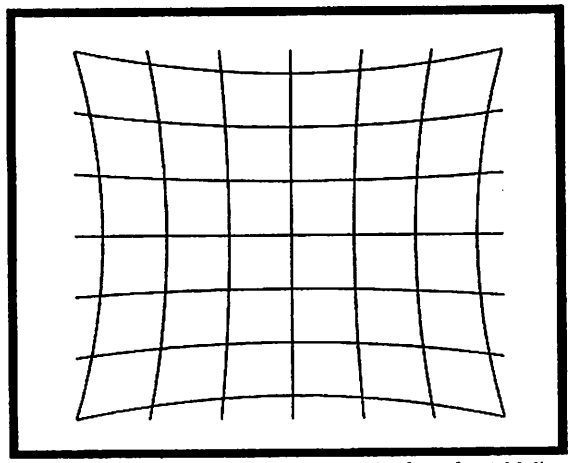
Kecenderungan garis menjadi melengkung keluar disebut distorsi *barrel*. Distorsi *barrel* terjadi saat titik berpindah dari posisi sebenarnya terhadap pusat foto. Distorsi *barrel* sering kita jumpai pada kamera yang berlensa sudut lebar atau *wide angle*, dimana foto yang penuh dengan garis tegas seperti garis arsitektur akan menjadi tidak alami, melengkung dan terkesan tidak professional. Tingkat kelengkungan ini berbeda-beda untuk tiap lensa, tentunya semakin baik lensa maka semakin kecil distorsi yang dihasilkan (Eternity 2009)



*Gambar 2.2 Distorsi Barrel(Wikipedia 2006)*

### II.2.1.2 Pincushion

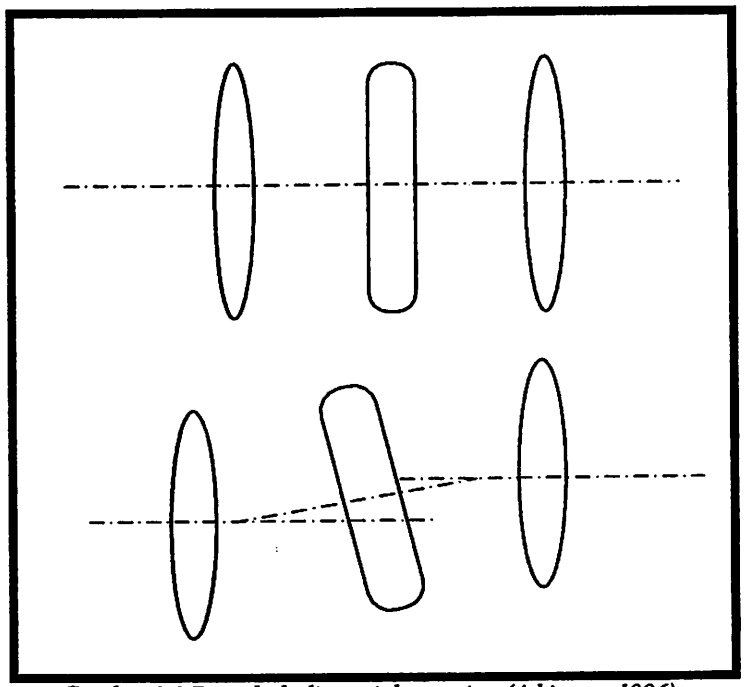
Tipe kedua distorsi radial adalah perpindahan positif, atau kecenderungan garis yang melengkung kedalam. Distorsi yang terjadi saat titik bergeser lebih jauh dari sumbu optik. Tipe ini disebut juga Distorsi *Pincushion*. Dan biasanya Distorsi *Pincushion* terjadi pada kamera yang berlensa sudut sempit. Perlu diketahui juga bahwa distorsi *pincushion* tidak terlalu signifikan jika dibandingkan dengan distorsi *barrel* (Perš, Kovačič, 2002).



*Gambar 2.2 Distorsi Pincushion(Wikipedia 2006)*

### II.2.2 Distorsi *decentring* (Distorsi tangensial)

Pada dasarnya semua elemen pada sistem lensa kamera seharusnya berada pada suatu garis lurus. Adanya pergeseran dan perputaran dari elemen lensa ini akan menyebabkan pergeseran geometri foto yang disebut sebagai distorsi *decentring* (Atkinson, 1996). Menurut Wolf, 1983 distorsi posisi gambar dengan arah tegak lurus terhadap garis radial dari titik utama (pada umumnya sangat kecil pengaruhnya dibandingkan dengan distorsi radial dan sering kali dapat diabaikan).



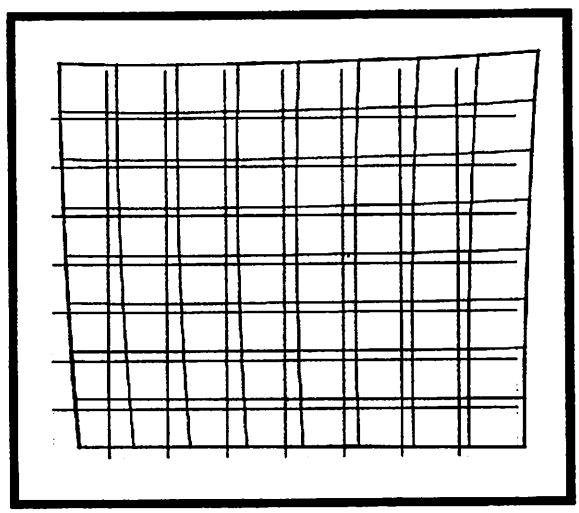
*Gambar 2.3 Penyebab distorsi decentring (Atkinson, 1996).*

Tidaklah mungkin mengurutkan elemen dari sistem lensa kolinier. Kekurangan ini mengakibatkan distorsi tangensial. Distorsi tangesial disebabkan kesalahan sentering elemen-elemen lensa dalam satu gabungan lensa (Atkinson, 1996). Dimana, titik pusat elemen-elemen lensa dalam gabuang lensa tersebut tidak terletak pada satu garis lurus. Pergeseran ini biasa dideskripsikan dengan 2 persamaan polonomial untuk pergeseran

pada arah x (dx) dan y (dy) (Nuraini, 2007). Dimana dua persamaan polonomial diuraikan seperti berikut:

$$\Delta x_d = P_1\left(3\bar{x}^2 + \bar{y}^2\right) + 2P_2\overline{xy}$$
$$\Delta y_d = 2P_1\overline{xy} + P_2\left(3\bar{y}^2 + \bar{x}^2\right) \quad (2.8)$$

Di mana $P_i$ merupakan koefisien dari distorsi tangensial. Karena kualitas sistem lensa yang tinggi kepentingan parameter ini sangatlah kecil. Pada umumnya parameter distorsi tangensial diuji bukan untuk kepentingan. Lagipula ada ketergantungan antara parameter distorsi tangensial dan titik utama, jadi suatu sisa distorsi akan diserap dari posisi titik pusat dan oleh karena itu parameter distorsi tangensial biasanya dihilangkan pada perataan (Dörstel et al, 2002).



*Gambar 2.4 Distorsi decentring (distorsi tangensial) (Atkinson, 1996).*

### II.2.3 Distorsi affinity

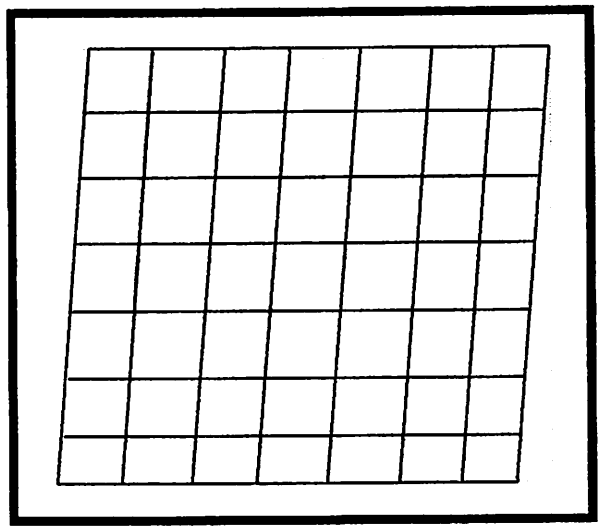
Parameter distorsi affinity impiris secara alami dan dimodelkan dalam bentuk persamaan polinomial. Untuk kamera CCTV jumlah parameter dikurangi dari 12 parameter menjadi 2 parameter.

Dimana parameter 'scale' b1 dan parameter 'scale' b2 (Dörstel et al, 2002).:

$$\Delta x_f = b_1 \bar{x} + b_2 \bar{y} \tag{2.9}$$

$$\Delta y_f = 0$$

Distorsi *affinity* ini terjadi akibat kurang sikunya bidang *CCD* atau *CMOS* yang digunakan untuk merekam bayangan obyek, sehingga *frame* dari foto tidak akan benar-benar berbentuk sebuah bujur sangkar ataupun persegi panjang, akan tetapi membentuk sebuah jajargenjang.



*Gambar 2.5 Distorsi Affinity*

## II.3 Metode Tsa'i

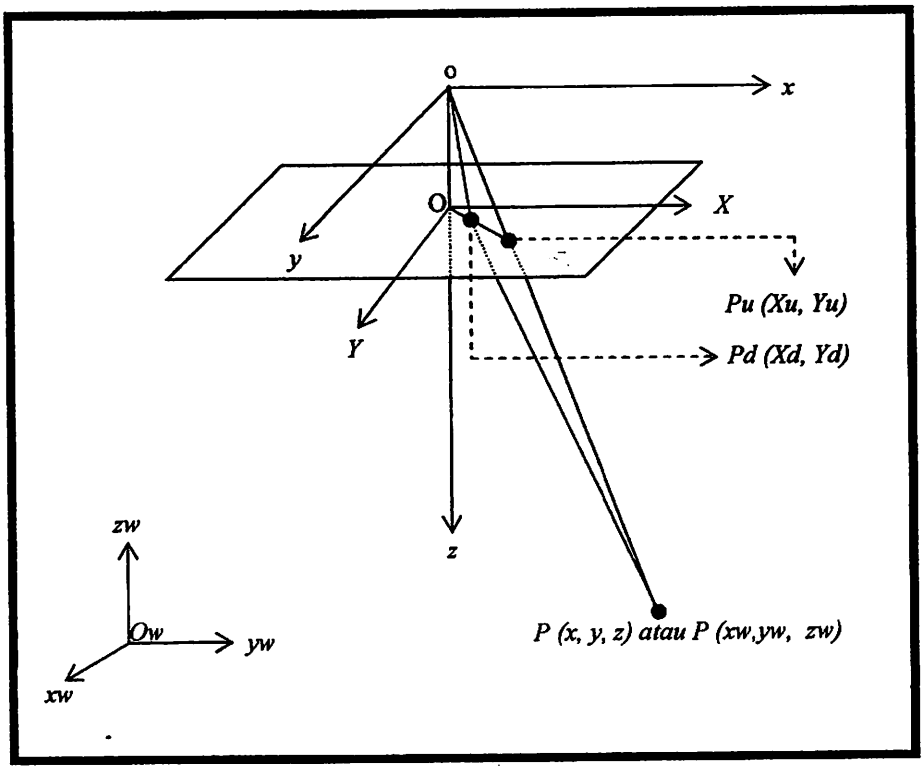
Metode non linear Faugeras-Toscani didasarkan pada penentuan dugaan awal tanpa mempertimbangkan distorsi lensa, model Tsai memang ekuivalen dengan model Faugeras-Toscani sebelumnya dengan distorsi (Armangu'e et al, 1999). Metode model Tsai juga merupakan distorsi lensa radial tapi mengasumsikan bahwa ada beberapa parameter kamera yang didistribusikan oleh pembuat (Salvi, Batlle, 1999). Kenyataan ini mengurangi jumlah

parameter yang berkalibrasi dalam langkah pertama. Akan tetapi, semuanya secara iterasi dioptimalkan dalam langkah terakhir (Armangu'e et al, 1999).

Kalibrasi kamera memerlukan penyelesaian untuk beberapa parameter kalibrasi, yang dihasilkan dalam pendekatan klasik yang memerlukan penelitian non linear skala besar (Tsai, 1987). Cara umum untuk mencegah penelitian non linear skala besar ini adalah dengan menggunakan pendekatan yang sama pada DLT (*Direct Linear Transformation*) (Tsai, 1987). Pendekatan kami adalah untuk mencari batasan, yang mana dimakksud disini adalah batasan baris radial (Tsai, 1987). Batasan ini hanyalah merupakan fungsi rotasi dan translasi relatif (kecuali untuk komponen z) antara kamera dan titik kalibrasi (Tsai, 1987). Dimana batasannya adalah fungsi non linear dari parameter kalibrasi yang dihitung dengan persamaan proyektif normal (Tsai, 1987).

### II.3.1 Model kamera

Bagian ini akan menjelaskan model kamera, defenisi dari parameter yang akan dikalibrasi. Model kamera itu sendiri pada dasarnya adalah sama seperti yang digunakan oleh setiap teknik kalibrasi kamera (Tsai, 1987)




*Gambar2.6 Geometri kamera dengan proyeksi perspektif dan distorsi lensa radial (Chiang, Boult, 1996)*

Gambar 2.6 menggambarkan geometri dasar model kamera. $(x_w, y_w, z_w)$adalah koordinat 3D dari objek *P* di dalam sistem koordinat dunia 3D (Chiang, Boult, 1996). $(x, y, z)$ adalah koordinat 3D dari titik obyek *P* di dalam sistem koordinat kamera, yang berpusat pada titik *o*, pusat optik, dengan sumbu *z* sama dengan sumbu optik (Chiang, Boult,1996). (*X, Y*) adalah sistem koordinat foto yang berpusat pada *O* dan sejajar dengan sumbu *x* dan *y* (Tsai, 1987). *f* adalah jarak antara bidang foto depan dan pusat optik (Chiang, Boult, 1996). ($X_u$, $Y_u$) adalah koordinat foto yang tidak terdistorsi atau ideal dari *(x, y, z)* (Chiang, Boult, 1996). ($X_d$, $Y_d$) adalah koordinat foto yang terdistorsi dari ($X_u$, $Y_u$) karena distorsi lensa radial (Tsai, 1987).

### II.3.1.1 Langkah-langkah transformasi dari koordinat bumi 3D kekoordinat kamera 3D

1) Transformasi *rigid body* dari sistem koordinat bumi 3D $(x_w, y_w, z_w)$ ke sistem koordinat kamera 3D $(x, y, z)$ (Tsai, 1987)

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \qquad (2.10)$$

Dimana, $R$ adalah matrik rotasi 3 × 3.

$$R \equiv \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}$$

Dan $T$ adalah vektor translasi

$$T \equiv \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

Parameter yang dikalibrasi adalah $R$ dan $T$ (Chiang, Boult,1996).

2) Transformasi dari koordinat kamera 3D *(x, y, z)* ke koordinat yang tidak didistorsikan *($X_u$, $Y_u$)* dengan menggunakan proyeksi perspektif dengan geometri lensa lubang jarum (Tsai, 1987)

$$X_u = f\frac{x}{z} \qquad (2.11)$$

$$Y_u = f\frac{y}{z} \qquad (2.12)$$

Parameter yang akan dikalibrasikan adalah panjang fokus efektif $f$(Chiang, Boult, 1996).

3) Distorsi lensa radial adalah

$$X_d + D_x = X_u \tag{2.13}$$

$$Y_d + D_y = Y_u \tag{2.14}$$

dimana ($X_d$, $Y_d$) adalah koordinat foto yang terdistorsikan pada bidang foto (Tsai, 1987)

$$D_x = X_d\left(k_1 r^2 + k_2 r^4 + \cdots\right) \tag{2.15}$$

$$D_y = Y_d\left(k_1 r^2 + k_2 r^4 + \cdots\right) \tag{2.16}$$

$$r = \sqrt{X_d^2 + Y_d^2} \tag{2.17}$$

Parameter yang akan dikalibrasikan adalah koefisien distorsi $k_i$ (Chiang, Boult, 1996). Untuk aplikasi industri hanya dibutuhkan koreksi terhadap distorsi radial (Manual Photogrammetry, 1980)

4) Transformasi koordinat foto yang terdistorsi ($X_d$, $Y_d$) ke kordinat foto yang terdapat di komputer ($X_f$, $Y_f$)

$$Xf = s_x d_x^{'-1} X_d + C_x \tag{2.18}$$

$$Y_f = s_y d_y^{-1} Y_d + C_y \tag{2.19}$$

dimana,

($X_f$, $Y_f$) : jumlah baris dan kolom dari *pixel* pada foto dalam komputer

($C_x$,$C_y$) : koordinat foto di komputer pada bidang foto asli

$$d_x' = d_x \frac{N_{cx}}{N_{fx}} \tag{2.20}$$

$d_x$ : jarak pusat ke pusat antara elemen sensor yang berdekatan dalam arah $X$ (garis scan)

$d_y$ : jarak pusat ke pusat antara sensor CCD yang berdekatan dalam arah $Y$

$N_{cx}$ : jumlah elemen sensor dalam arah $X$

$N_{fx}$ : jumlah *pixel* dalam garis seperti yang ditunjukkan oleh komputer

$s_x$ : faktor skala yang akan dikalibrasi.

Parameter yang akan dikalibrasi adalah ketidaktentuan faktor skala foto ($s_x$) dan $(C_x, C_y)$ (Chiang, Boult, 1996).

### II.3.2 Persamaan yang berhubungan dengan koordinat dunia 3D ke koordinat foto 2D

Dengan mengombinasikan tiga langkah terakhir, koordinat komputer ($X$, $Y$) dihubungkan dengan ($x$, $y$, $z$), koordinat 3D dari titik objek dalam sistem koordinat kamera, dapat dijelaskan dengan persamaan berikut (Tsai, 1987):

$$s_x^{-1} d_x' X + s_x^{-1} d_x' X k_1 r^2 = f \frac{x}{z} \tag{2.21}$$

$$d_y' Y + d_y Y k_1 r^2 = f \frac{y}{z} \tag{2.22}$$

dimana,

$$r = \sqrt{\left(s_x^{-1} d_x' X\right)^2 + \left(d_y + Y\right)^2} \tag{2.23}$$

Subtitusikan (1) kedalam persamaan (2.21) dan (2.22) menjadi:

$$s_x^{-1} d_x' X + s_x^{-1} d_x' X k_1 r^2 = f \frac{r_1 x_w + r_2 y_w + r_3 z_w + T_x}{r_7 x_w + r_8 y_w + r_9 z_w + T_z} \tag{2.24}$$

$$d_y' Y + d_y Y k_1 r^2 = f \frac{r_4 x_w + r_5 y_w + r_6 z_w + T_x}{r_7 x_w + r_8 y_w + r_9 z_w + T_z} \tag{2.25}$$

dimana,

$$\begin{aligned} X &= X_f - C_z \\ Y &= Y_f - C_y \end{aligned} \tag{2.26}$$

### II.3.2.1 Parameter ekstrinsik

Pada langkah 1 untuk transformasi dari sistem koordinat bumi objek 3D ke sistem koordinat 3D terpusat pada pusat optik yang disebut dengan parameter ekstrinsik (Tsai, 1987). Terdapat enam parameter: penyimpangan sudut *euler* $\theta$, hubungan $\phi$, dan kemiringan $\psi$ untuk rotasi (*R*), serta tiga komponen untuk vektor translasi *(T)*. Matrik rotasi *(R)* bisa ditunjukkan sebagai fungsi $\theta$, $\phi$, $\psi$ sebagai berikut (Tsai, 1987):

$$R = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ -\sin\psi\cos\phi + \cos\psi\sin\theta\cos\phi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \cos\theta\sin\phi \\ \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.27}$$

### II.3.2.2 Parameter intrinsik

*Interior Orientation (IO)* adalah hubungan antara koordinat pusat kamera dan koordinat foto (Horn, 2000). Sistem koordinat kamera mempunyai koordinat asli pada titik tengah kamera, yang sumbu z sepanjang sumbu optik dan sumbu x, y sejajar menuju sumbu x, y foto (Horn, 2000). Dalam Langkah 2-4 untuk

transformasi dari koordinat obyek 3D pada sistem koordinat kamera ke koordinat gambar komputer disebut sebagai parameter intrinsik (Tsai, 1987). Pada model Tsai terdapat lima parameter intrinsik (Tsai, 1986), yaitu:

$f$ : panjang fokus efektif atau jarak antara titik tengah kamera ke bidang foto

$k_1$ : koofesien distorsi radial

$s_x$ : ketidaksesuaian faktor skala

$(C_x, C_y)$ : pusat foto

### II.3.3 Defenisi masalah

Masalah kalibrasi kamera adalah menghitung parameter intrinsik dan ekstrinsik kamera berdasarkan pada jumlah titik yang koordinat obyeknya ada dalam sistem koordinat $(x_w, y_w, z_w)$ diketahui dan yang koordinat gambarnya $(X, Y)$ diukur (Tsai, 1987).

### II.3.4 Teknik kalibrasi kamera dua tingkat baru: motivasi

Dasar awal teknik baru adalah empat observasi sebagai berikut (Tsai, 1987):

#### II.3.4.1 Observasi I

Disini diasumsikan bahwa distorsi adalah radial, maka seberapa pun distorsinya, arah vektor $\overrightarrow{O_i P_d}$ diluar $O_i$ dalam bidang foto ke titik foto $(X_d, Y_d)$ masih tidak berubah dan secara radial diluruskan dengan vektor $\overrightarrow{P_{oz} P}$ di luar dari sumbu optik (atau, lebih tepatnya, titik $P_{oz}$ pada sumbu otpik yang koordinat $z$

nya sama dengan titik obyek ($x$, $y$, $z$)) ke titik obyek (x, y, z) (Tsai, 1987).

### II.3.4.2 Observasi II

Panjang fokus efektif ($f$) juga tidak mempengaruhi arah vektor $\overline{O_iP_d}$, karena skala $f$ koordinat foto $X_d$ dan $Y_d$ mempunyai tingkat yang sama (Tsai, 1987).

### II.3.4.3 Observasi III

Ketika sistem koordinat objek dipermukaan bumi dirotasi dan ditranslasikan ke dalam $x$ dan $y$ seperti dalam langkah 1 hingga $O_iP_d$ sejajar dengan $\overline{P_{oz}P}$ untuk semua titik, maka translasi dalam $z$ tidak akan mengubah arah $O_iP_d$ (ini dari kenyataan bahwa, menurut persamaan (2.11) dan (2.12), $z$ mengubah $X_u$ dan $Y_u$ dengan skala yang sama, sehingga $\overline{O_iP_u}$ // $\overline{O_iP_d}$ (Tsai, 1987).

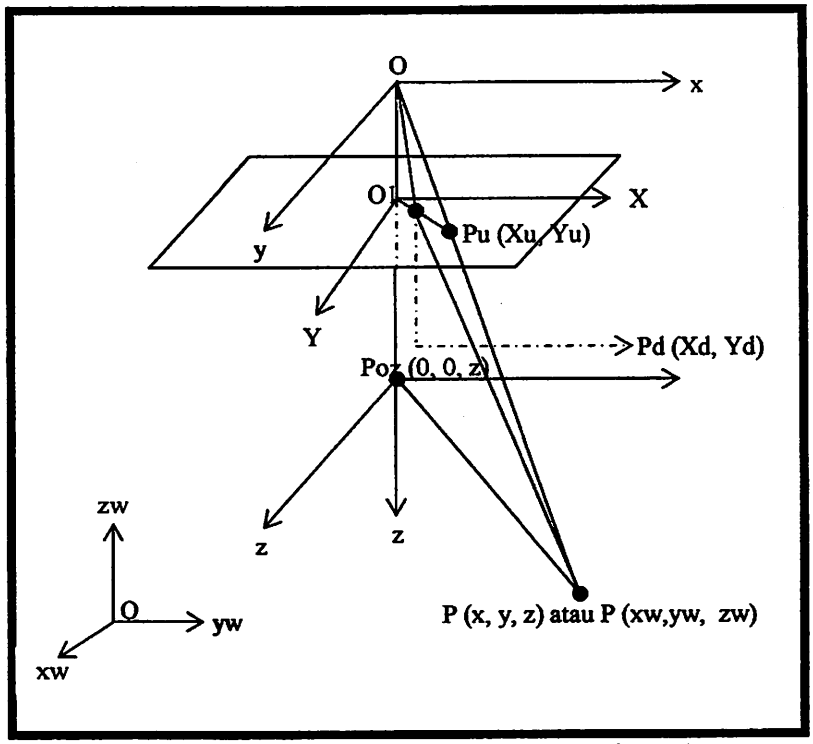
### II.3.4.4 Observasi IV

Batasan di mana $O_iP_d$ sejajar pada $\overline{P_{oz}P}$ untuk setiap titik, akan ditunjukkan sendiri dari koefisien distorsi radial ($k_1$, $k_2$), panjang fokus efektif ($f$), dan komponen $z$ translasi 3D vektor $T$, sebenarnya memadai untuk menentukan rotasi 3D komponen $R$, $X$, dan $Y$ pada translasi 3D dari sistem koordinat bumi ke sistem koordinat kamera, dan $s_x$ dalam komponen $X$ pada koordinat foto (Tsai, 1987).

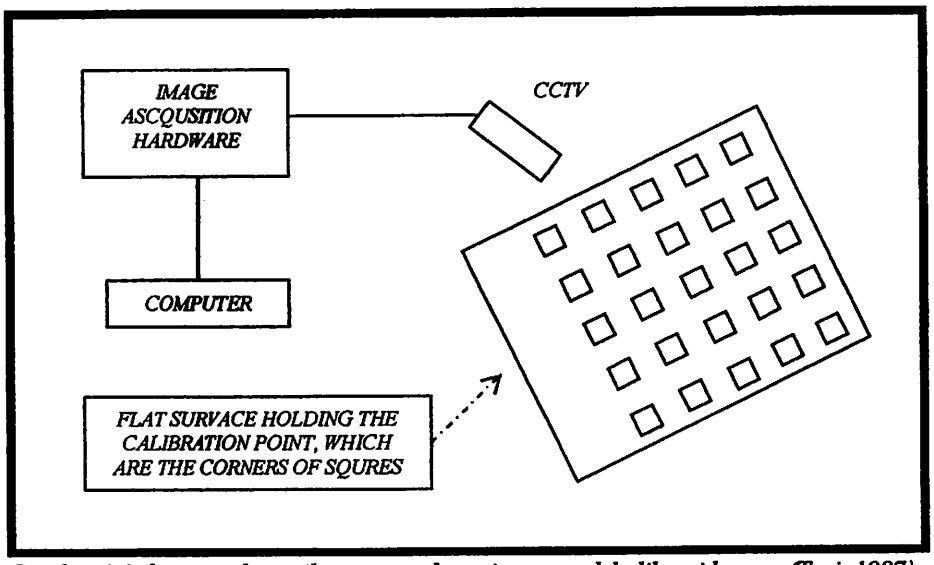
Secara normal dari empat obsevasi diatas hanya yang terakhir yang memerlukan optimasi non linear untuk menentukan parameter ekstrinsik (kecuali $T_z$) dan salah satu parameter intrinsik ($s_x$) (Tsai, 1986).

### II.3.5 Pengkalibrasian kamera dengan menggunakan titik coplanar

Metode diatas tidak dapat digunakan apabila targetnya adalah bidang datar (Horn, 2000). Karena target kita berupa sebuah bidang datar maka nilai $z_w$ diasumsikan = 0 untuk semua titik yang terdapat pada target tersebut (Tsai, 1987). Tujuan terakhir dari metode diatas adalah memastikan bahwa nilai $T_y$ adalah tidak selalu = 0 (Tsai, 1987).




*Gambar 2.7 ilustrasi garis distorsi radial(Chiang, Boult,1996).*




*Gambar 2.8 diagram skematik susunan eksperimen untuk kalibrasi kamera(Tsai, 1987).*

### II.3.5.1 Menghitung $R$, $T_x$, $T_y$

1) Menghitung koordinat foto terdistorsi $(X_d, Y_d)$

$$X_d^i = s_x^i d_x' \left(X_f^i - C_x\right)$$
$$Y_d^i = d_y \left(Y_f^i - C_y\right) \qquad (2.28)$$

2) Menghitung $r_1/T_y, r_2/T_y, T_x/T_y, r_4/T_y$, *dan* $r_5/T_y$ yang belum diketahui menggunakan persamaan linear (menghitung menggunakan point 1 diatas):

$$A_X = b \qquad (2.29)$$

Dimana

$$A \equiv \begin{bmatrix} Y_d^1 x_w^1 & Y_d^1 y_w^1 & Y_d^1 & -Y_d^1 x_w^1 & -Y_d^1 y_w^1 \\ Y_d^2 x_w^2 & Y_d^2 y_w^2 & Y_d^2 & -Y_d^2 x_w^2 & -Y_d^2 y_w^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_d^N x_w^N & Y_d^N y_w^N & Y_d^N & -Y_d^N x_w^N & -Y_d^N y_w^N \end{bmatrix},$$

$$x \equiv \begin{bmatrix} r_1/T_y \\ r_2/T_y \\ T_x/T_y \\ r_4/T_y \\ r_5/T_y \end{bmatrix}, \text{ dan } b \equiv \begin{bmatrix} Y_d^1 \\ Y_d^2 \\ \vdots \\ Y_d^N \end{bmatrix}$$

3) Menghitung *R*, *Tx*, dan $T_y$

a) Menghitung $|T_y|$ dari $r_1/T_y, r_2/T_y, T_x/T_y, r_4/T_y, dan\ r_5/T_y$

Diketahui matriks ${}_2[C]_2$ adalah sub matriks dari matriks *R*. Dimana, matriks *C* didefenisikan sebagai berikut

$$C \equiv \begin{bmatrix} r_1' & r_2' \\ r_4' & r_5' \end{bmatrix} \equiv \begin{bmatrix} r_1/T_y & r_2/T_y \\ r_4/T_y & r_5/T_y \end{bmatrix} \tag{2.30}$$

Jika (tidak semua baris dan kolom dari matriks *C* dihilangkan) maka kita dapat menghitung $T_y^2$ dengan persamaan berikut (Tsai, 1987):

$$T_y^2 = \frac{S_r - \left|S_r^2 - 4\left(r_1' r_5' - r_4' r_2'\right)^2\right|^{1/2}}{4\left(r_1' r_5' - r_4' r_2'\right)^2} \tag{2.31}$$

Dimana $S_r = r_1'^2 + r_2'^2 + r_4'^2 + r_5'^2$; ELSE (ini jarang tejadi, jika terjadi) kita dapat menghitung $T_y^2$ dengan persamaan berikut (Tsai, 1987);

$$T_y^2 = \left(r_i'^2 + r_j'^2\right)^{-1} \tag{2.32}$$

Dimana $r_i'^2$, $r_j'^2$ adalah unsur-unsur dalam baris dan kolom dari matriks *C* yang tidak dihilangkan (Tsai, 1987).

b) Menentukan tanda dari $T_y$

$$
\begin{aligned}
r_1 &= (r_1/T_y)\cdot T_y \\
r_2 &= (r_2/T_y)\cdot T_y \\
T_x &= (T_x/T_y)\cdot T_y \\
r_4 &= (r_4/T_y)\cdot T_y \\
r_5 &= (r_5/T_y)\cdot T_y \\
x &= r_1 x_w + r_2 y_w + r_3 \cdot 0 + T_x \\
y &= r_4 x_w + r_5 y_w + r_6 \cdot 0 + T_y
\end{aligned}
\qquad (2.33)
$$

Dimana $r_1/T_y, r_2/T_y, T_x/T_y, r_4/T_y$, *dan* $r_5/T_y$ jika (($x$ dan $X$ mempunyai tanda yang sama) dan ($y$ dan $Y$) mempunyai tanda yang sama)), maka tanda $Ty = +1$, selain itu tanda $Ty = -1$.

c) Menghitung matrik rotasi *(R)*

Hitung persamaan berikut:

$$
\begin{aligned}
r_1 &= (r_1/T_y)\cdot T_y \\
r_2 &= (r_2/T_y)\cdot T_y \\
T_x &= (T_x/T_y)\cdot T_y \\
r_4 &= (r_4/T_y)\cdot T_y \\
r_5 &= (r_5/T_y)\cdot T_y
\end{aligned}
\qquad (2.34)
$$

Dimana $r_1/T_y, r_2/T_y, T_x/T_y, r_4/T_y$, *dan* $r_5/T_y$ ditentukan dalam penentuan lima parameter yang belum diketahui di bagian sebelumnya (Tsai,1987).

Jika $f \geq 0$ maka matrik $R$ adalah :

$$R = \begin{bmatrix} r_1 & r_2 & \left(1 - r_1^2 - r_2^2\right)^{1/2} \\ r_4 & r_5 & s\left(1 - r_4^2 - r_5^2\right)^{1/2} \\ r_7 & r_8 & r_9 \end{bmatrix} \qquad (2.35)$$

Jika $f \leq 0$ maka matrik $R$ adalah :

$$R = \begin{bmatrix} r_1 & r_2 & -\left(1 - r_1^2 - r_2^2\right)^{1/2} \\ r_4 & r_5 & -s\left(1 - r_4^2 - r_5^2\right)^{1/2} \\ -r_7 & -r_8 & r_9 \end{bmatrix} \qquad (2.36)$$

Dimana $s = -sign(r_1 r_4 + r_2 r_5)$, dan $r_7$, $r_8$ *dan* $r_9$ ditentukan dengan *croos product* pada dua baris pertama menggunakan ortonormal dan kaedah tangan kanan dari matrik $R$ (Tsai, 1987)

### II.3.5.2 Menghitung $f$, $k_1$, $T_x$

a) Menghitung pendekatan dari $f$ dan $T_x$ dengan mengabaikan distorsi lensa radial

Untuk setiap titik kalibrasi $i$, ditentukan dengan persamaan linear untuk nilai pendekatan $f$ dan $T_x$ yang belum diketahui sebagai berikut (Chiang, Boult, 1996):

$$\begin{bmatrix} y_1 & -d_y Y_1 \\ y_2 & -d_y Y_2 \\ \vdots & \vdots \\ y_N & -d_y Y_N \end{bmatrix} \begin{bmatrix} f \\ T_z \end{bmatrix} = \begin{bmatrix} w_1 d_y Y_1 \\ w_2 d_y Y_2 \\ \vdots \\ w_N d_y Y_N \end{bmatrix} \qquad (2.37)$$

### II.3.6 Pengkalibrasian kamera dengan menggunakan titik noncoplanar

#### II.3.6.1 Menghitung $R$, $T_x$, $T_y$ dan $s_x$

1) Menghitung koordinat foto terdistorsi $(\overline{X_d}, \overline{Y_d})$, dimana $(\overline{X_d}, \overline{Y_d})$ didefinisikan sama persis sama dengan $(X_d, Y_d)$ dalam persamaan (6), kecuali $s_x$ yang diasumsikan =1 (Chiang, Boult, 1996).

$$\begin{aligned} X_d^i &= s_x^{-1} d_x' \left(X_f^i - C_x\right) \\ Y_d^i &= d_y' \left(Y_f^i - C_y\right) \end{aligned} \tag{2.38}$$

2) Menghitung $s_x r_1/T_y, s_x r_2/T_y, s_x T_x/T_y, r_4/T_y, r_5/T_y$, *dan* $r_6/T_y$, yang belum diketahui menggunakan persamaan linear (dihitung menggunakan point **1)** diatas);

$$A_X = b \tag{2.39}$$

Dimana

$$A \equiv \begin{bmatrix} \overline{Y}_d^1 x_w^1 & \overline{Y}_d^1 y_w^1 & \overline{Y}_d^1 & -\overline{X}_d^1 x_w^1 & -\overline{X}_d^1 y_w^1 & -\overline{X}_d^1 z_w^1 \\ \overline{Y}_d^2 x_w^2 & \overline{Y}_d^2 y_w^2 & \overline{Y}_d^2 & -\overline{X}_d^2 x_w^2 & -\overline{X}_d^2 y_w^2 & -\overline{X}_d^2 z_w^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \overline{Y}_d^N x_w^N & \overline{Y}_d^N y_w^N & \overline{Y}_d^N & -\overline{X}_d^N x_w^N & -\overline{X}_d^N y_w^N & -\overline{X}_d^N z_w^N \end{bmatrix}$$

$$x \equiv \begin{bmatrix} s_x r_1/T_y \\ s_x r_2/T_y \\ s_x r_3/T_y \\ s_x T_x/T_y \\ s_x r_4/T_y \\ s_x r_5/T_y \\ s_x r_3/T_y \end{bmatrix}, \text{ dan } b \equiv \begin{bmatrix} \overline{X}_d^1 \\ \overline{X}_d^2 \\ \vdots \\ \overline{X}_d^N \end{bmatrix}$$

3) Menghitung $R$, $T_x$, dan $T_y$

a) Menghitung $|T_y|$ dari

$s_x r_1/T_y, s_x r_2/T_y, s_x T_x/T_y, r_4/T_y, r_5/T_y$, *dan* $r_6/T_y$, maka untuk menghitung $|T_y|$ digunakan formula berikut:

$$|T_y| = 1/\sqrt{a_5^2 + a_6^2 + a_7^2} \tag{2.40}$$

b) Menentukan tanda dari $T_y$

$$\begin{aligned} r_1 &= (r_1/T_y)\cdot T_y \\ r_2 &= (r_2/T_y)\cdot T_y \\ r_3 &= (r_3/T_y)\cdot T_y \\ T_x &= (T_x/T_y)\cdot T_y \\ r_4 &= (r_4/T_y)\cdot T_y \\ r_5 &= (r_5/T_y)\cdot T_y \\ r_6 &= (r_6/T_y)\cdot T_y \\ x &= r_1 x_w + r_2 y_w + r_3 z_w + T_x \\ y &= r_4 x_w + r_5 y_w + r_6 z_w + T_y \end{aligned} \tag{2.41}$$

Dimana $s_x r_1/T_y, s_x r_2/T_y, s_x T_x/T_y, r_4/T_y, r_5/T_y$, *dan* $r_6/T_y$, ditentukan dari persamaan (22) diatas. Yang mana jika (($x$ dan $X$ mempunyai tanda yang sama) dan ($y$ dan $Y$)

mempunyai tanda yang sama)), maka tanda $Ty = +1$, selain itu tanda $Ty = -1$.

c) Menentukan $s_x$

Gunakan persamaan berikut untuk menghitung $s_x$

$$s_x = \left(a_1^2 + a_2^2 + a_3^2\right)^{1/2} \left|T_y\right| \tag{2.42}$$

d) Menghitung $R$ Matriks Rotasi

Menghitung $r_i, r = 1, \ldots, 6$, dan $T_x$ dengan menggunakan persamaan berikut:

$$\begin{aligned} r_1 &= a_1 \cdot T_y / s_x \\ r_2 &= a_2 \cdot T_y / s_x \\ r_3 &= a_3 \cdot T_y / s_x \\ T_x &= a_4 \cdot T_y / s_x \\ r_5 &= a_5 \cdot T_y \\ r_6 &= a_6 \cdot T_y \\ r_7 &= a_7 \cdot T_y \end{aligned} \tag{2.43}$$

Dimana $a_i, i = 1, \ldots, 7$ ditentukan dari point 3) diatas

Ingat $r_i, i = 1, \ldots, 6$, dan yang mana baris pertama dan kedua merupakan elemen dari matriks $R$, ditentukan dengan *croos product* pada dua baris pertama menggunakan ortonormal dan kaedah tangan kanan dari matrik $R$ (determinan dari $R$ =1, bukan -1) (Chiang, Boult, 1996).

### II.3.6.2 Menghitung $f$, $k_1$, $T_z$

a) Menghitung pendekatan dari $f$ dan $T_x$ dengan mengabaikan distorsi lensa radial

Untuk setiap titik kalibrasi $i$, ditentukan dengan persamaan linear untuk nilai pendekatan $f$ dan $T_z$ yang belum diketahui sebagai berikut (Chiang, Boult, 1996):

$$\begin{bmatrix} y_1 & -d_y Y_1 \\ y_2 & -d_y Y_2 \\ \vdots & \vdots \\ y_N & -d_y Y_N \end{bmatrix} \begin{bmatrix} f \\ T_z \end{bmatrix} = \begin{bmatrix} w_1 d_y Y_1 \\ w_2 d_y Y_2 \\ \vdots \\ w_N d_y Y_N \end{bmatrix} \tag{2.44}$$

Dimana $y_1 = r_4 x_w^i + r_5 y_w^i + r_6 z_w^i + T_y$ dan

$$w_1 = r_7 x_w^i + r_8 y_w^i + r_9 z_w^i$$

## II.4 Metode Zhang

Kalibrasi kamera merupakan langkah penting dalam visi 3D komputer untuk mengestrak informasi metrik dari gambar 2D (Zhang, 1998).

### II.4.1 Persamaan dasar

Sebuah titik 2D dinotasikan oleh $m = [u, v]^T$ (Zhang, 1998). Sebuah titik 3D dinotasikan oleh $M = [X, Y, Z]^T$ (Zhang, 1998). $\tilde{x}$ melambangkan vektor yang ditambahkan oleh penambahan 1 sebagai elemen terakhir, $\tilde{m} = [u, v, 1]^T$ dan $\tilde{M} = [X, Y, Z, 1]^T$ (Zhang, 1998). Sebuah kamera biasanya dimodelkan dengan lubang jarum: hubungan antara point 3D $M$ dan proyeksi foto $m$ diberikan oleh (Zhang, 1998):

$$s\tilde{m} = A[R \quad t]\tilde{M} \text{ dengan } A = \begin{bmatrix} \alpha & c & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.45}$$

Dimana, $s$ adalah sebuah faktor skala sembarang; ($R$, $t$), disebut sebagai parameter ekstrinsik adalah rotasi dan translasi yang mana dihubungkan dengan sistem koordinat bumi dan sistem koordinat foto; $A$ disebut matriks intrinsik foto, dan ($u_0$, $v_0$) koordinat untuk nilai dari *principle point*, $\alpha$ dan $\beta$ faktor skala dalam sumbu foto $u$ dan $v$, dan $c$ adalah parameter yang mendeskripsikan kemiringan dari dua sumbu foto (Zhang, 1998).

Disini digunakan singkatan

$$A^{-T} \text{ untuk } \left(A^{-1}\right)^T \text{ atau } \left(A^T\right)^{-1}$$

### II.4.2 Kesamaan bidang model dan foto

Tanpa menghilangkan persamaan umum, kita berasumsi bahwa bidang model pada Z=0 di sistem koordinat bumi (Zhang, 1998). Notasi $i^{th}$ : kolom dari matriks rotasi $R$ oleh $r_i$ dari (2.45)

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A[r_1 \quad r_2 \quad r_3 \quad t]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = A[r_1 \quad r_2 \quad t]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Kita masih menggunakan $M$ untuk mewakili nilai dari bidang datar, tapi $m = [X, Y]^T$ sejak Z adalah selalu sama dengan 0 (Zhang, 1998). Pada saat $\tilde{M} = [X, \ Y, \ 1]^T$. Oleh karna itu, bentuk persamaan untuk menotasikan hubungan antara $M$ model dan $m$ foto serta faktor skala atau yang kita kenal dengan nama matriks homography H (Zhang, 1998):

$$s\tilde{m} = H\tilde{M} \quad dengan\, H = A[r_1 \quad r_2 \quad t] \tag{2.46}$$

### II.4.3 Permasalahan parameter intrinsik

Mengingat gambar dari model bidang datar, maka sebuah homografi dapat ditentukan, notasi $H = [h_1 \quad h_2 \quad h_3]$ Dari persamaan (2.46 ), maka diperoleh

$$[h_1 \quad h_2 \quad h_3] = \lambda A [r_1 \quad r_2 \quad t]$$

Dimana λ adalah faktor skala sembarang, dengan diketahui bahwa $r_1$ dan $r_2$ orthonormal maka, kita mempunyai persamaan (Zhang, 1998):

$$h_1^T A^{-T} A^{-1} h_2 = 0 \tag{2.47}$$

$$h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 \tag{2.48}$$

Persamaan (2.47) dan (2.48) adalah persamaan dasar pada parameter intrinsik (Zhang, 1998). Sebuah homografi mempunyai 8 derajat kebebasan dan ada 6 parameter ekstrinsik yakni terdiri dari 3 parameter untuk rotasi dan 3 parameter untuk translasi (Zhang, 1998). Dimana, kita hanya dapat menggunakan 2 *constrains* pada parameter intrinsik (Loung, 1992).

### II.4.4 Memecahkan kalibrasi kamera

Pada bagian ini membahas secara detail bagaimana cara yang efektif guna menyelesaikan masalah kalibrasi kamera (Zhang, 1998).

### II.4.4.1 Solusi *close form*

$$B = A^{-T}A^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \tag{2.49}$$

$$= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0-u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2}+\frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma-u_0\beta)}{\alpha^2\beta^2}-\frac{v_0}{\beta^2} \\ \frac{v_0\gamma-u_0\beta}{\alpha^2\beta^2} & -\frac{\gamma(v_0\gamma-u_0\beta)}{\alpha^2\beta^2}-\frac{v_0}{\beta^2} & \frac{(v_0\gamma-u_0\beta)^2}{\alpha^2\beta^2}+\frac{v_0^2}{\beta^2}+1 \end{bmatrix}$$

Catatan bahwa $B$ adalah simetri, yang didefinisikan oleh vektor 6D

$$b = [B_{11},\ B_{12},\ B_{22},\ B_{13},\ B_{23},\ B_{33}]^T \tag{2.50}$$

Biarkan nilai $i$ pada kolom vektor dari $H$ menjadi $h_i = [h_{i1}\quad h_{i2}\quad h_{i3}]^T$. Maka, persamaan yang diperoleh adalah:

$$h_i^T B h_j = v_{ij}^T b \tag{2.51}$$

Dengan

$$v_{ij} = [h_{i1}h_{j1},\ h_{i1}h_{j2}+h_{i2}h_{j1},\ h_{i2}h_{j2},\ h_{i3}h_{j1}+h_{i1}h_{j3},\ h_{i3}h_{j2}+h_{i2}h_{j3},\ h_{i3}h_{j3}]^T$$

Oleh karna itu, dua *fundamental constrain* (2.47) dan (2.48) dari sebuah homografi yang diberikan, maka dapat ditulis sebagai 2 persamaan homogeneus yang berbeda dalam $b$ (Zhang, 1998):

$$\begin{bmatrix} v_{12}^T \\ (v_{11}-v_{22})^T \end{bmatrix} b = 0 \tag{2.52}$$

Jika foto $n$ pada model bidang datar diobservasi, dengan memasukan nilai $n$ pada persamaan (2.52) maka, sehingga persamaannya :

$$Vb = 0 \tag{2.53}$$

Dimana $V$ adalah sebuah matriks $2n \times 6$. Jika $n \geq 3$ maka kita akan memperoleh solusi yang unik sampai pada faktor skala. Jika $n = 2$, maka kita dapat menggunakan kemiringan *constrain* $\gamma = 0$ misalnya $[0,\ 1,\ 0,\ 0,\ 0,\ 0]$, $b = 0$, dimana ditambahkan sebuah tanda tambahan (Shimizu,et al 1998). Solusi ini ditambahkan sebagai *eigenvector* dari $V^T V$ yang digabungkan dengan *eigenvalue* (atau dengan kata lain, nilai matriks $V$ dari *eigenvector* digabungkan dengan nilai terkecil dari *eigenvalue*) (Zhang, 1998).

Setelah $b$ ditentukan, kita dapat menghitung matriks intrinsik kamera dengan menggunakan persamaan berikut (Zhang, 1998):

$$A = \begin{bmatrix} \alpha & c & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Dimana

$$v_0 = (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2)$$
$$\lambda = B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]/B_{11}$$
$$\alpha = \sqrt{\lambda / B_{11}}$$
$$\beta = \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)}$$
$$c = -B_{12}\alpha^2\beta/\lambda$$
$$u_0 = cv_0/\alpha - B_{13}\alpha^2/\lambda$$

Setelah $A$ diketahui, parameter ekstrinsik untuk masing-masing foto siap untuk dihitung (Zhang, 1998). Dari persamaan (2.46), kita memiliki persamaan seperti berikut:

$$\begin{aligned} r_1 &= \lambda A^{-1} h_1 \\ r_2 &= \lambda A^{-1} h_2 \\ r_3 &= r_1 \times r_2 \\ t &= \lambda A^{-1} h_3 \end{aligned} \tag{2.54}$$

Dengan $\lambda = 1/\left\|A^{-1}h_1\right\| = 1/\left\|A^{-1}h_2\right\|$

### II.4.5 Penentuan dengan distorsi radial

Suatu kamera digital biasanya mengalami distorsi lensa yang signifikan khususnya distorsi radial (Brown, 1971). Seuai dengan namanya distorsi radial, menyebabkan semua bagian gambar diubah letaknya menurut arah jari-jari, bermula dari sumbu optik (Wolf, 1983). Kejadian ini diakibatkan oleh adanya kesalahan dalam pengasahan lensa (Wolf, 1983).

Koordinat piksel pada foto yang tidak terdistorsi $(u,v)$ dan koordinat foto yang terdistorsi $(\breve{x},\breve{y})$ dapat dinormalisasikan dengan persamaan berikut (Brown, 1971)

$$\breve{x} = x + x\left[k_1\left(x^2+y^2\right) + k_2\left(x^2+y^2\right)^2\right] \tag{2.55}$$

$$\breve{y} = y + y\left[k_1\left(x^2+y^2\right) + k_2\left(x^2+y^2\right)^2\right] \tag{2.56}$$

Dimana $k_1$ dan $k_2$ adalah koofesien dari dari radial distorsi. Pusat dari distorsi radial sama dengan titik *prisiple point* (Wei, Ma, 1994).

$$\text{Dari} \quad \breve{u} = u + (u-u_0)\left[k_1\left(x^2+y^2\right) + k_2\left(x^2+y^2\right)^2\right] \tag{2.57}$$

$$\breve{v} = v + (v - v_0)\left[k_1\left(x^2 + y^2\right) + k_2\left(x^2 + y^2\right)^2\right] \tag{2.58}$$

Kemudian, dari persamaan (2.57) dan (2.58) kita mempunyai dua persamaan untuk masing-masing titik di tiap foto

$$\begin{bmatrix} (u - u_0) + \left(x^2 + y^2\right) & (u - u_0)\left(x^2 + y^2\right)^2 \\ (v - v_0) + \left(x^2 + y^2\right) & (v - v_0)\left(x^2 + y^2\right)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \breve{u} - u \\ \breve{v} - v \end{bmatrix} \tag{2.59}$$

Solusi *least square* yang dapat digunakan untuk menyelesaikan persamaan diatas adalah:

$$k = \left(D^T D\right)^{-1} D^T d \tag{2.60}$$

Dimana $Dk = d$ dan $k = [k_1, k_2]^T$

## II.5 *Stereo* Kalibrasi

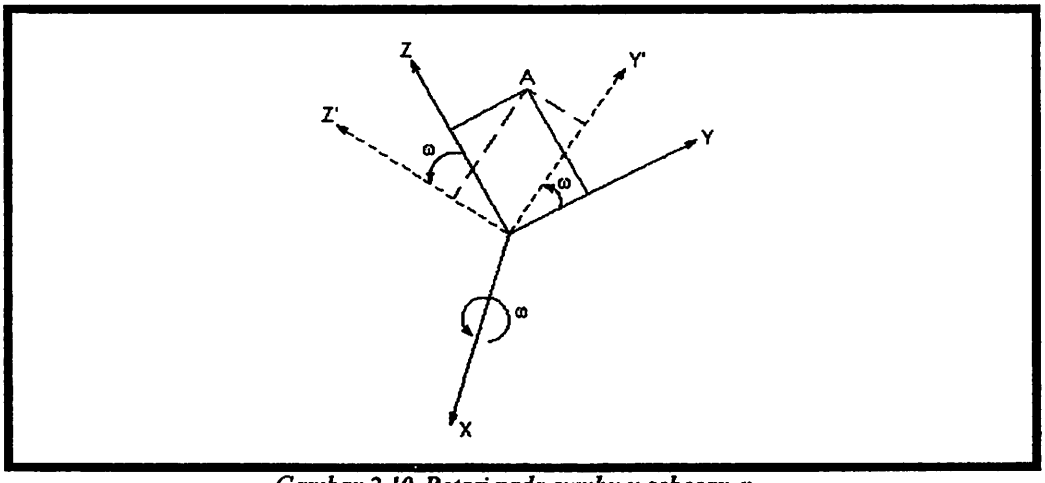
Stereo kalibrasi adalah proses menghitung hubungan geometris antara dua kamera (Bradski & Kaehler, 2008). Geometri dari dua kamera ini berkaitan dengan nilai informasi parameter eksterinsik dan parameter instrinsik untuk tiap kamera. Parameter interinsik biasanya dipakai sebagai parameter untuk mendefinisikan nilai geometri dari kamera, sedangkan parameter ekstrinsik digunakan untuk menjelaskan hubungan secara geometri posisi kamera dan objek dalam suatu sistem koordinat referensi. Parameter ekstrinsik terdiri dari rotasi dan translasi.

### II.5.1 Rotasi

Parameter posisi kamera didefinisikan dalam ruang tiga-dimensi yang merupakan posisi dari *perspektif center* (pusat kamera) $X_o, Y_o, Z_o$. Sedangkan parameter orientasi sudut berfungsi sebagai penghubung antara

sistem koordinat kamera $(x, y, z)$ dengan sistem koordinat referensi $(X, Y, Z)$. Sesuai dengan(*Elias, 2007;Fraser, 2006b;Mikhail et al., 2001;Shih, 1994;Wolf & Dewitt, 2000*)orientasi sudut dapat didefinisikan dalam sistem rotasi $\omega, \varphi, \kappa$ (*omega, phi, kappa*).

Dari masing-masing sistem rotasi, dapat dibangun sebuah matrik rotasi yang digunakan untuk menghubungkan kedua sistem koordinat. Sebagai contoh, akan dijelaskan proses penurunan sebuah persamaan pada matrik rotasi dalam sudut rotasi *omega* untuk rotasi sumbu *x*, *phi* untuk rotasi sumbu *y* dan *kappa* sebagai rotasi sumbu *z*. Dari ketiga sudut diatas didefinisikan dalam sistem rotasi tangan kanan, dimana apabila berlawanan dengan peutaran arah jarum jam akan bernilai negative dan sebaliknya.



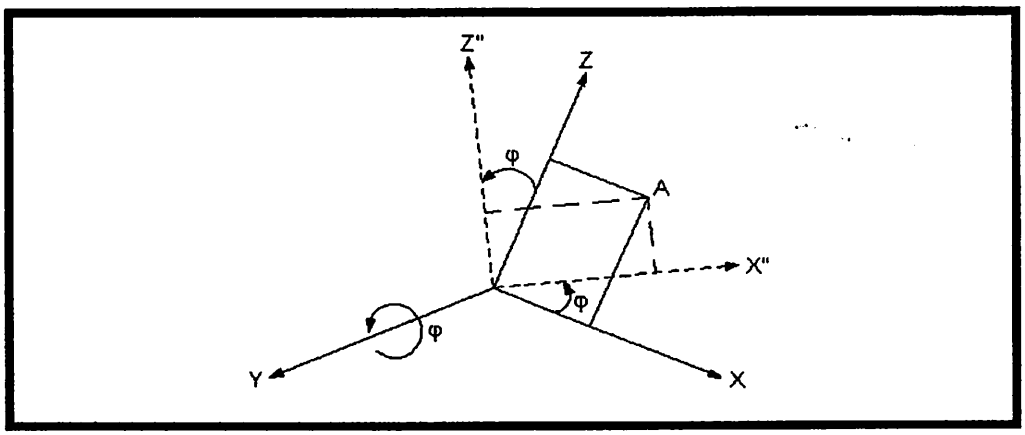

*Gambar 2.10 Rotasi pada sumbu x sebesar ω*

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

$$\begin{vmatrix} X' \\ Y' \\ Z' \end{vmatrix} = R_\omega \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \quad \text{.....(2.69)}$$

Dimana parameter $R_\omega$ didefinisikan sebagai :

$$R_\omega = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{vmatrix} \quad \text{.....(2.70)}$$



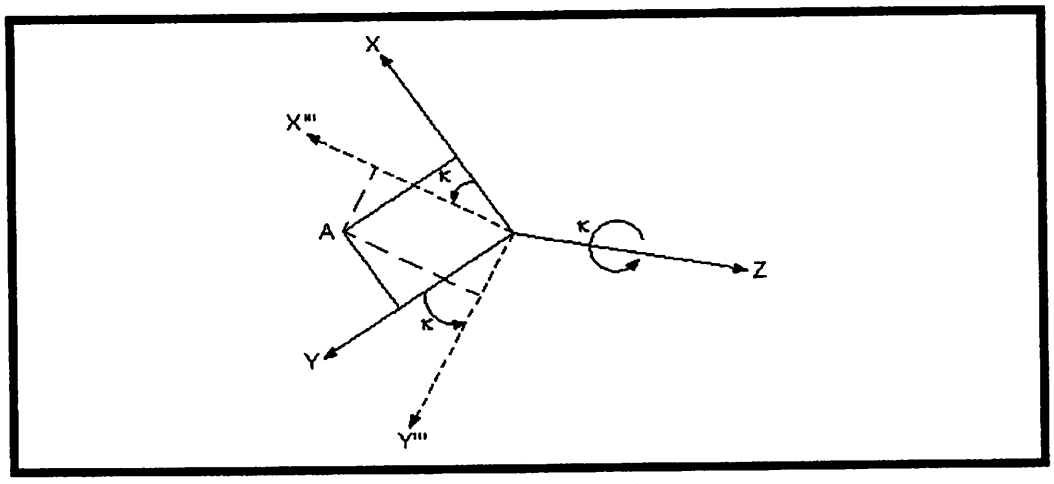

*Gambar 2.11 Rotasi sumbu y sebesar φ*

Selanjutnya, dilakukan rotasi terhadap sumbu *y* positif dengan arah rotasi positif searah dengan jarum jam seperti pada *gambar 2.7*. Dari hasil rotasi didapat nilai posisi titik *A* dalam sistem koordinat terotasi sebagai berikut.

$$\begin{vmatrix} X'' \\ Y'' \\ Z'' \end{vmatrix} = R_\varphi \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \quad \text{.....(2.71)}$$

Dengan parameter $R_\varphi$ didapat dari persamaan :

$$R_\varphi = \begin{vmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{vmatrix} \quad \text{.....(2.72)}$$




*Gambar 2.12 Rotasi sumbu z sebesar κ*

Dan yang terakhir, sumbu positif z dirotasi positif searah jarum jam, sebagaimana diilustrasikan pada *gambar 2.8* diatas. Sehingga posisi titik *A* pada sistem rotasi sumbu z dinyatakan dalam sebuah persamaan sebagai berikut :

$$\begin{vmatrix} X''' \\ Y''' \\ Z''' \end{vmatrix} = R_\kappa \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \quad \text{.....(2.73)}$$

Dimana parameter $R_\kappa$ didefinisikan sebagai :

$$R_\kappa = \begin{vmatrix} \cos\kappa & \sin\kappa & 0 \\ -\sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad \text{.....(2.74)}$$

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

$$R_{\omega\varphi\kappa} = \begin{vmatrix} \cos\varphi\cos\kappa & \sin\omega\sin\varphi\cos\kappa + \cos\omega\sin\kappa & -\cos\omega\sin\varphi\cos\kappa + \sin\omega\sin\kappa \\ -\cos\varphi\sin\kappa & -\sin\omega\sin\varphi\sin\kappa + \cos\omega\cos\kappa & \cos\omega\sin\varphi\sin\kappa + \sin\omega\cos\kappa \\ \sin\varphi & -\sin\omega\cos\varphi & \cos\omega\cos\varphi \end{vmatrix} \quad (2.75)$$

atau

$$R = \begin{vmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{vmatrix} \quad \text{.....(2.76)}$$

Dan apabila nilai matrik rotasi $R$ telah diketahui, parameter sudut rotasi berupa $\omega, \varphi$ dan $\kappa$ dapat ditentukan pula dengan menggunakan hubungan sebagai berikut (*Slabaugh, 2004; Wolf, 1993*) :

$$\sin\varphi = r_{31} \;\; ; \;\; \tan\omega = -r_{32}/r_{33} \;\; ; \;\; \tan\kappa = -r_{21}/r_{11} \quad \text{.....(2.77)}$$

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

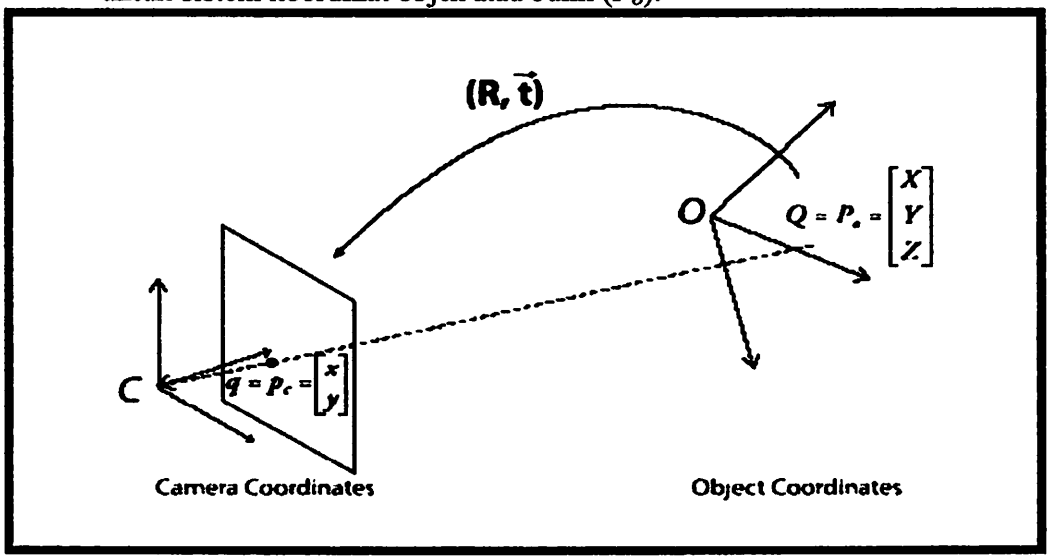
$$R^T R = RR^T = I \quad \text{.....(2.78)}$$

Kondisi keortogonalan matrik rotasi dapat dibuktikan juga dengan cara melakukan *invers* terhadap matrik tersebut. Apabila nilai *invers* dari matrik tersebut sama dengan nilai transposnya, maka matrik tersebut dapat dikatakan ortogonal sebagaimana yang dikemukakan oleh *Cooper & Robson, 2001* sebagai berikut :

$$R^{-1} = R^T \quad \text{.....(2.79)}$$

## II.5.2 Translasi

Vektor translasi adalah merupakan pergeseran dari suatu sistem koordinat ke suatu sistem koordinat sebenarnya yang dipindahkan ke lokasi lain atau dengan kata lain, vektor translasi hanya offset dari sistem koordinat sebenarnya yang pertama ke sistem koordinat sebenarnya yang kedua (Bradski & Kaehler, 2008). Jadi, untuk beralih dari pusat sistem koordinat objek ke suatu pusat kamera, vektor translasi yang tepat adalah $T = origin_{object} - origin_{camera}$. Dengan mengacu pada gambar 2.13 dimana terdapat sebuah titik q untuk sistem koordinat kamera ($p_c$) dan titik Q untuk sistem koordinat objek atau bumi ($P_o$):




*Gambar 2.13 Konversi dari Objek ke Sistem Koordinat Kamera:*
*Titik P pada Objek Terlihat Sebagai Titik p pada Bidang Foto (Bradski & Kaehler, 2008).*

Dari pembahasan diatas, sejauh ini terdapat enam parameter yang telah dihasilkan ($\omega, \varphi, \kappa$ dan $T_x, T_y, T_z$).

### II.5.3 Menghitung Panjang *Baseline*

Seperti yang telah dibahas sebelumnya dari proses Stereo kalibrasi adalah proses menghitung hubungan geometris antara dua kamera (Bradski & Kaehler, 2008). Geometri dari dua kamera ini berkaitan dengan nilai informasi parameter eksterinsik dan parameter instrinsik untuk tiap kamera. Dimana dari parameter ekstrinsik parameter ini menghasilkan rotasi serta vektor translasi. Nilai dari vektor translasi dapat digunakan untuk menghitung jarak basis antara kamera kiri dan kamera kanan. Tujuan dari menghitung jarak basis adalah untuk digunakan sebagai faktor skala pada proses interseksi. Adapun persamaan yang dapat digunakan untuk menghitung panjang basis menurut *OpenCV* adalah:

$$Baseline = \sqrt{T_x^2 + T_y^2 + T_z^2} \quad \text{.....(2.81)}$$

## II.6 *Object Oriented Programing (OPP)*

Istilah pemrograman memiliki pengertian sebagai teknik membuat suatu program (Andi 2008). Sedangkan program yang lebih dikenal dengan istilah software, merupakan satu atau beberapa *file*/berkas yang berisi kumpulan-kumpulan intruksi. Program dibuat dengan menggunakan bahasa pemrograman. Ada beberapa metode/teknik pemrograman, diantaranya adalah teknik pemrograman terstruktur, pemrograman prosedural, dan pemrograman berorientasi objek (*object-oriented programming* disingkat *OOP*) (Kurniawan, 2008).

Pemrograman berorientasi objek merupakan teknik membuat program dengan berdasarkan kepada objek. Suatu permasalahan yang akan kita carikan solusinya bisa kita lihat sebagai suatu objek. Objek merupakan suatu bentuk atau model yang tergantung dari cara pandang objek dimaksud (Andi 2008).

### II.5.1 Prinsip Dasar Pemograman Berorientasi Objek

Ada empat prinsip dasar pemograman berorientasi objek yaitu *Abstraction*

*Abstraction* adalah sebuah proses penyeleksian hal-hal yang perlu dan tidak perlu dari sebuah objek atau ide. Tujuan utama dari *abstraction* adalah memberikan sebuah bentuk objek yang benar-benar tepat dan efisien (Kurniawan, 2008).

a. *Encapstulation*

*Encapstulation* adalah proses penyatuan data dan *method* dalam satu "kapsul" *class*. Ada dua hal yang harus diperhatikan dalam *encapstulation*:

1. menggabungkan data dan fungsi dalam satu *entity*.
2. mengatur akses ke member dari *entity* tersebut.

b. *Inheritance*

*Inheritance* atau penurunan adalah kemampuan membuat suatu *class* anak dari suatu *class* induk, dimana seluruh *field* dan *method* diturunkan kepada *class* anak, kecuali *method* atau *field* dengan *access-modifier private*.

c. *Polimorphism*

*Polimorphism* adalah kemampuan *class-class* untuk menyediakan inplementasi yang berbeda-beda dari *method-method* yang dipanggil dengan nama yang sama. Polimorphism memperbolehkan sebuah *method* atau *class* untuk dipanggil tanpa memperhatiakan inplementasi spesifik apa yang ditetapkan.

### II.7.2 *Class* dan Objek

Objek merupakan dasar *intetitas runtime* dalam suatu sistem berorientasi objek sedangkan *Class* merupakan cetak *biru (blueprint)* atau *template* dari objek dengan kata lain *class* merupakan representasi abstrak sedangkan objek merupakan representasi nyata. Suatu *class* dapat berisi property, *field*, *method* dan *event* dari suatu objek yang

disebut *members* dari suatu *class*. Dalam .net semua *class* dasar dikelompokkan kedalam *Namespace* (Kurniawan, 2008).

- ✓ *Property* merupakan informasi (atribut) yang dapat disimpan dalam objek. *Property* dapat dideklarasikan dengan *Public Property* dan *Method Property* dapat mengontrol operasi *property*. Terdapat 2 bagian dalam *property* yaitu *Get* dan *Set*, *Get* hanya memperbolehkan akses data sedangkan *Set* memperbolehkan mengubah nilai data.
- ✓ *Field* merupakan informasi atau atribut yang terdapat didalam suatu objek. Bentuk *field* mirip dengan *variabel* yaitu dapat dibaca dan diset secara langsung.
- ✓ *Method* merupakan suatu tindakan yang dilakukan oleh objek.
- ✓ *Event* pemberitahuan yang diterima oleh objek.

### II.7.3 *Constructor*

Constructor adalah method yang digunakan untuk menciptakan sebuah objek berdasarkan kerangka *class* dan meletakannya pada kondisi yang tepat. Jika didefinisikansendiri maka *constructor* harus diberi nama sama dengan *class* karena *constructor* memang ditujukan untuk menciptakan *object* berdasarkan *design class* (Andi, 2008).

## II.7 *Emgu Computer Vision*

*Emgu computer vision* merupakan pengembangan dari *OpenCV (Open Computer Vision) dimana OpenCV merupakan* suatu *library* yang dikembangkan oleh *developer-developer Intel Corporation. Library* ini terdiri dari fungsi-fungsi Komputer *Vision* dan *API (Aplication Programming Interface)* untuk *image processing high level* maupun *low level* dan sebagai optimasi aplikasi *realtime* (Wikipedia, 2009) .

### *II.7.1.* Fitur-fitur pada *Library OpenCV*

Berikut ini adalah fitur-fitur pada *library OpenCV* (Wikipedia, 2009):

- ✓ Manipulasi data gambar (alokasi memori, melepaskan memori, kopi gambar, seting serta konversi gambar).
- ✓ *Image/Video I/O.*
- ✓ manipulasi matrix dan vektor serta terdapat juga *routines linear algebra (products, solvers, eigenvalues, SVD)* .
- ✓ *Image processing* dasar (*filtering, edge detection*, pendeteksian tepi, sampling dan interpolasi, konversi warna, operasi morfologi, *histograms, image pyramids*) .
- ✓ Analisis structural.
- ✓ Kalibrasi kamera.
- ✓ Pendeteksian grerak
- ✓ Pengenalan objek
- ✓ *Basic GUI* (*Display* gambar/video, *mouse/keyboard* kontrol, *scrollbar*).
- ✓ *Image Labelling (line, conic, polygon, text drawing).*

## II.8 *CSharp Matrix Library (CSML)*

*CSharp Matrix Library (CSML)* adalah suatu *class library* yang berbasis *.Net* yang menyediakan berbagai fungsi matriks untuk perhitungan aljabar liner. Komponen-komponen di dalam *CSharp Matrix Library* terdiri dari beberapa metode diantaranya adalah pengoprasian matriks (perkalian matriks, penjumlahan matriks, Eksponen dan sistem persamaan linear), manipulasi matriks (*Concatenation, insertion, transpose, inverse, flipping, simetrizing dan extraction*), dekomposisi matriks dan arimatika kompleks matriks (hanzzoid, 2007).

# BAB III

# PELAKSANAAN PENELITIAN

## III.1. Persiapan Pelaksanaan Penelitian

Pembuatan program kalibrasi foto stereo dilaksanakan dengan tujuan untuk melakukan otomatisasi proses perhitungan kalibrasi secara cepat tepat dan efisien tanpa meninggalkan ketelitian dari hitungan yang dibuat. Sebelum pembuatan program ini 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 yaitu literatur-literatur, perangkat keras *(hardwere)* dan perangkat lunak *(softwere)*.

## III.2. Bahan dan Alat Studi Penelitian

Tahap persiapan berupa bahan dan alat untuk studi penelitian ini merupakan tahap yang sangat penting dalam menunjang keberhasilan penelitian, karena tahap ini berisikan tentang bahan penelitian atau data-data yang diperlukan, persiapan alat studi serta literatur-literatur yang digunakan sebagai referensi dalam penelitian, adapun penjelasannya adalah sebagai berikut:

### III.2.1. Alat Penelitian

Adapun peralatan yang dipakai dalam penelitian ini adalah:

1. Perangkat lunak *(Softwere)* terdiri dari:
    - *Microsoft Windows XP Servis pack 2 Version 2002.*
    - *Microsoft Visual Studio 2008 C#.*
    - *Microsoft Library for Visual Studio 2008.*
    - *Librrary EMGU Computer Vision 2.10.*
    - *CSharp Matrix Library (CSML).*
    - *Microsoft Office Word 2003.*
    - *Microsoft Office Exel 2003.*
2. Perangkat keras *(Hardwere)* terdiri dari:
    - *CPU: Processor Intel(R) Core(TM)2 Duo 2.10GHz.*
    - *Hardisk 250 GB.*
    - *VGA 256 MB.*
    - *Monitor SVGA.*
    - *Mouse* dan *Keyboard*

### III.2.2. Bahan Penelitian

Adapuan bahan penelitian yang digunakan meliputi :

1. Spesifikasi kamera CCTV.
    a. Ukuran CCD 4.9 x 3.7
    b. Panjang fokus 12mm.
2. Panjang Baseline Bar Kamera 50cm.
3. Data pengukuran papan kalibrasi *(file .txt)*.

4. Empat pasang foto stereo papan kalibrasi dengan posisi yang papan kalibrasi yang berbeda-beda.

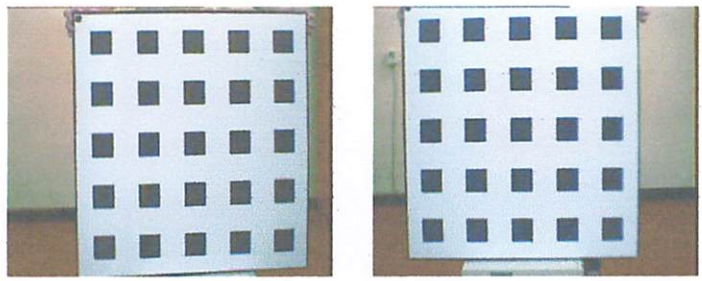


*Foto Kiri 1*



*Foto Kanan 1*

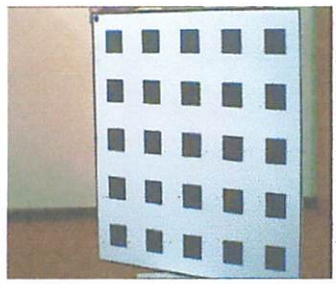


*Foto Kiri 2*

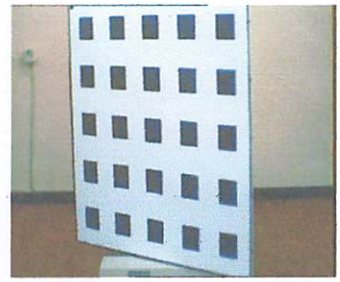


*Foto Kanan 2*

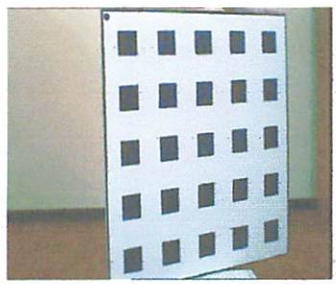


*Foto Kiri 3*

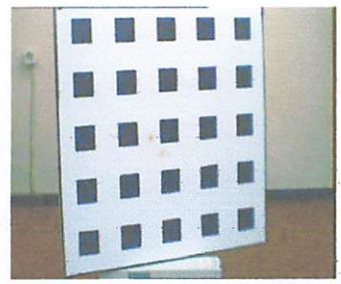


*Foto Kanan 3*

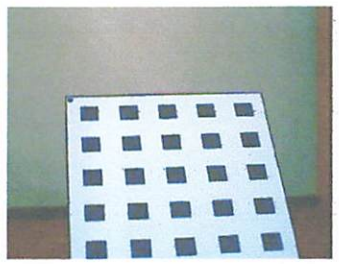


*Foto Kiri 4*

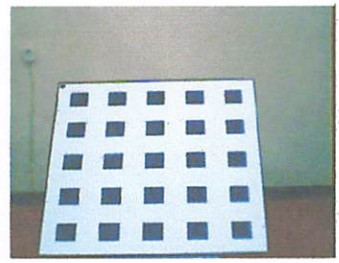
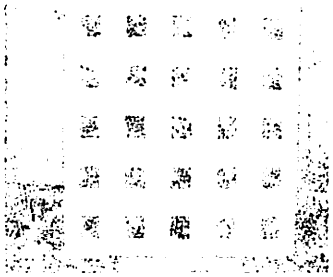
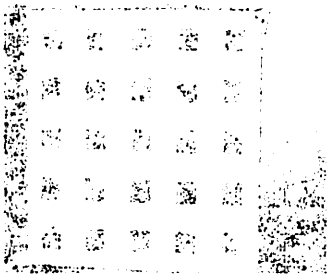
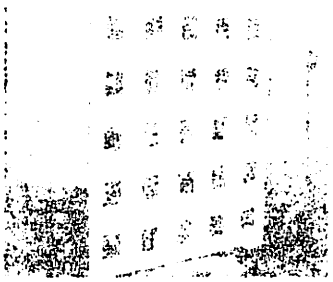


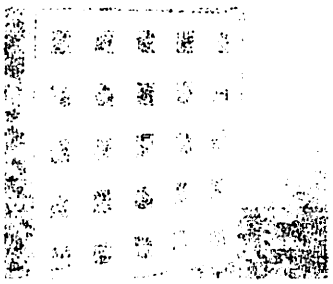
*Foto Kanan 4*

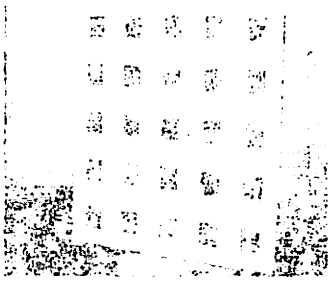
*Gambar 3.1 Foto Stereo papan kalibrasi*

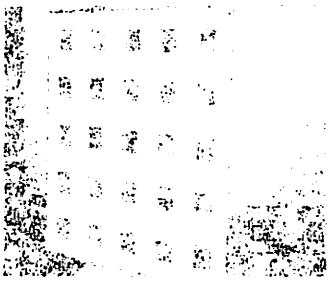


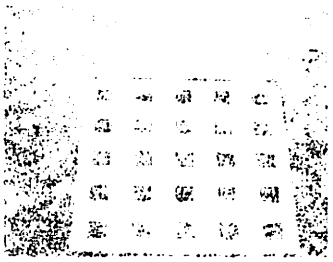


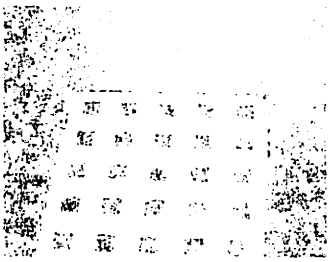






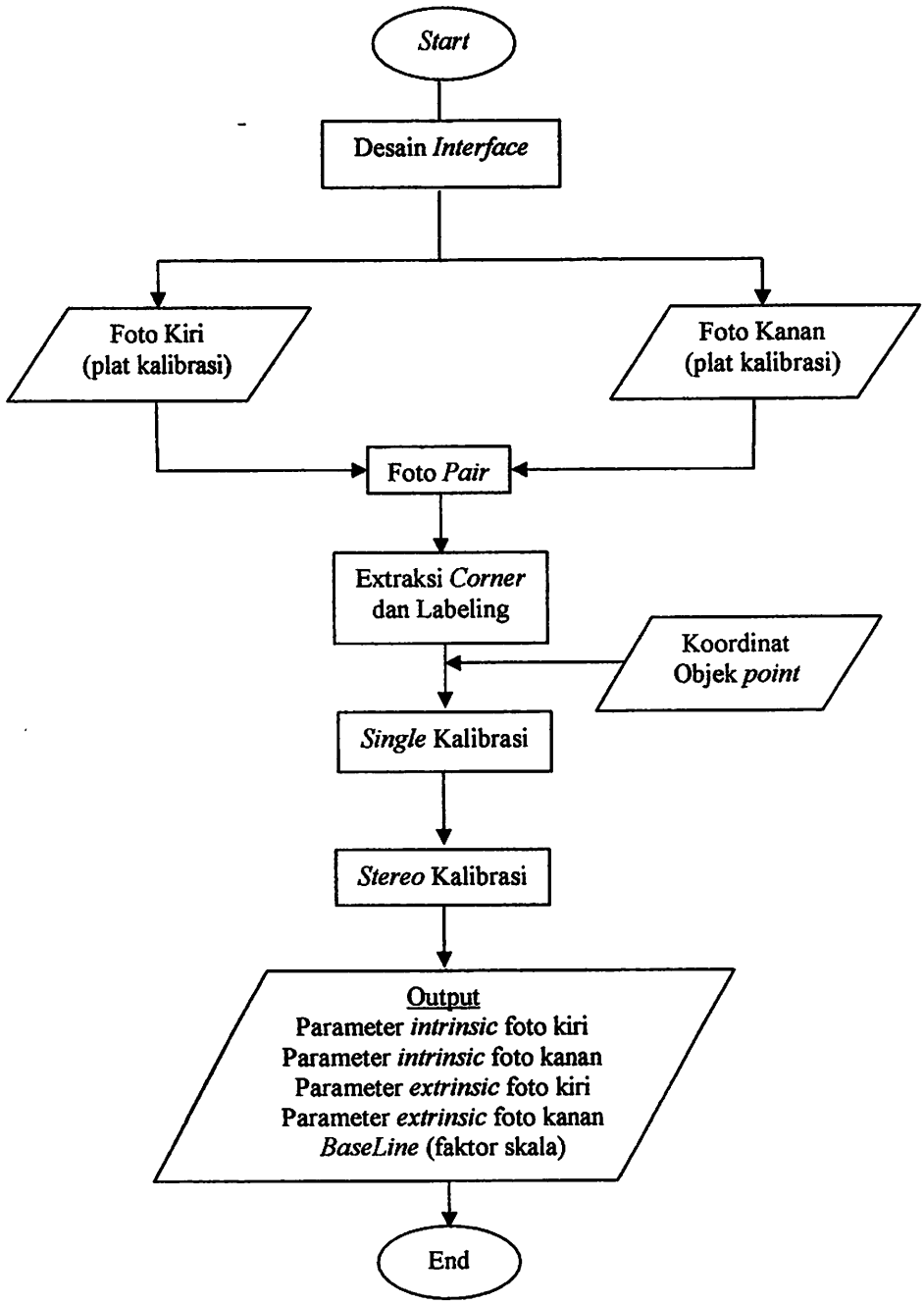






## III.3. Diagram alir penelitian

Secara umum pelaksanaan penelitian digambarkan dalam bagan alir sebagai berikut:




*Gambar 3.2 Diagram alir penelitian*

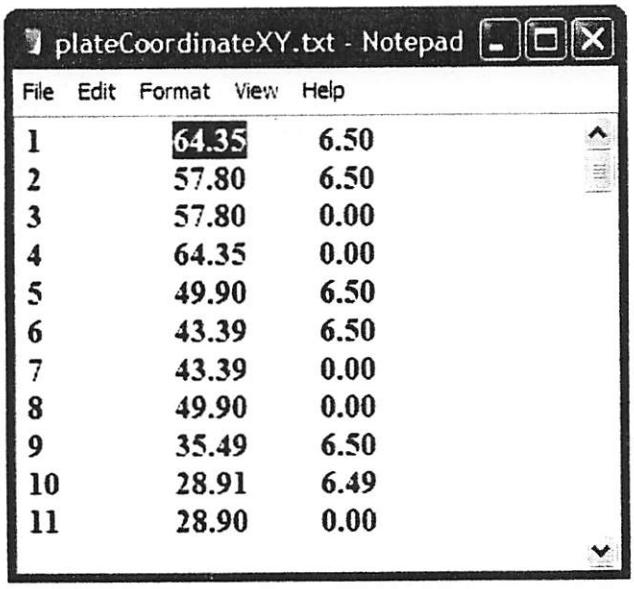
Keterangan diagram alir penelitian:

1. Desain *Interface* merupakan proses mendesain tampilan serta hubungan antara program dan objek (foto) yang akan di olah, dimana dalam desain interface ini terdiri dari beberapa tampilan *form* dan *class-class* pendukung pada aplikasi program ini yang terdiri dari:
   a. *Form StereoMapp*
   b. *Form StereoCalibration*
   c. *Form PhotoPair*
   d. *Form Text Dialog*
   e. *Form CameraProperties*
   f. *Class Cross*
   g. *Class Crosses*
   h. *Class Image2ImageBoxScaleConvertion*
   i. *Class ImagePair*
   j. *Class StereoCalib*
   k. *Class ImageBoxExt*
   l. *Class LSM*
   m. *Class StereoParams*
2. Proses penginputan beberapa pasangan foto stereo papan kalibrasi dimana foto stereo ini merupakan hasil *capture* dari dua kamera *CCTV* yang sama spesifikasinya yang dipasang diatas sebauah bar.

3. Proses foto *Pair* adalah proses penggabungan *name file* foto antara *name file* foto kiri papan kalibrasi dan foto kanan papan kalibrasi dan dijadikan satu *file name.*

4. Extraksi *corner* merupakan proses pendeteksian sudut pada kotak-kotak yang ada pada papan kalibrasi di foto kiri dan foto kanan yang ditandai dengan munculnya tanda *cross* (+), dimana setiap *cross* mempunyai nilai koordinat piksel *(x', y')* kemudian di beri label berupa nomor-nomor 1-100 (proses labelling harus disesuaikan dengan data koordinat objek).

5. Penginputan koordinat objek, koordinat objek (X, Y) merupakan koordinat yang diambil dari papan kalibrasi (ukuran kotak-kotak yang berwarna hitam) dengan cara dihitung secara manual menggunakan milimeter blok dan disimpan dengan format *file* *.txt.

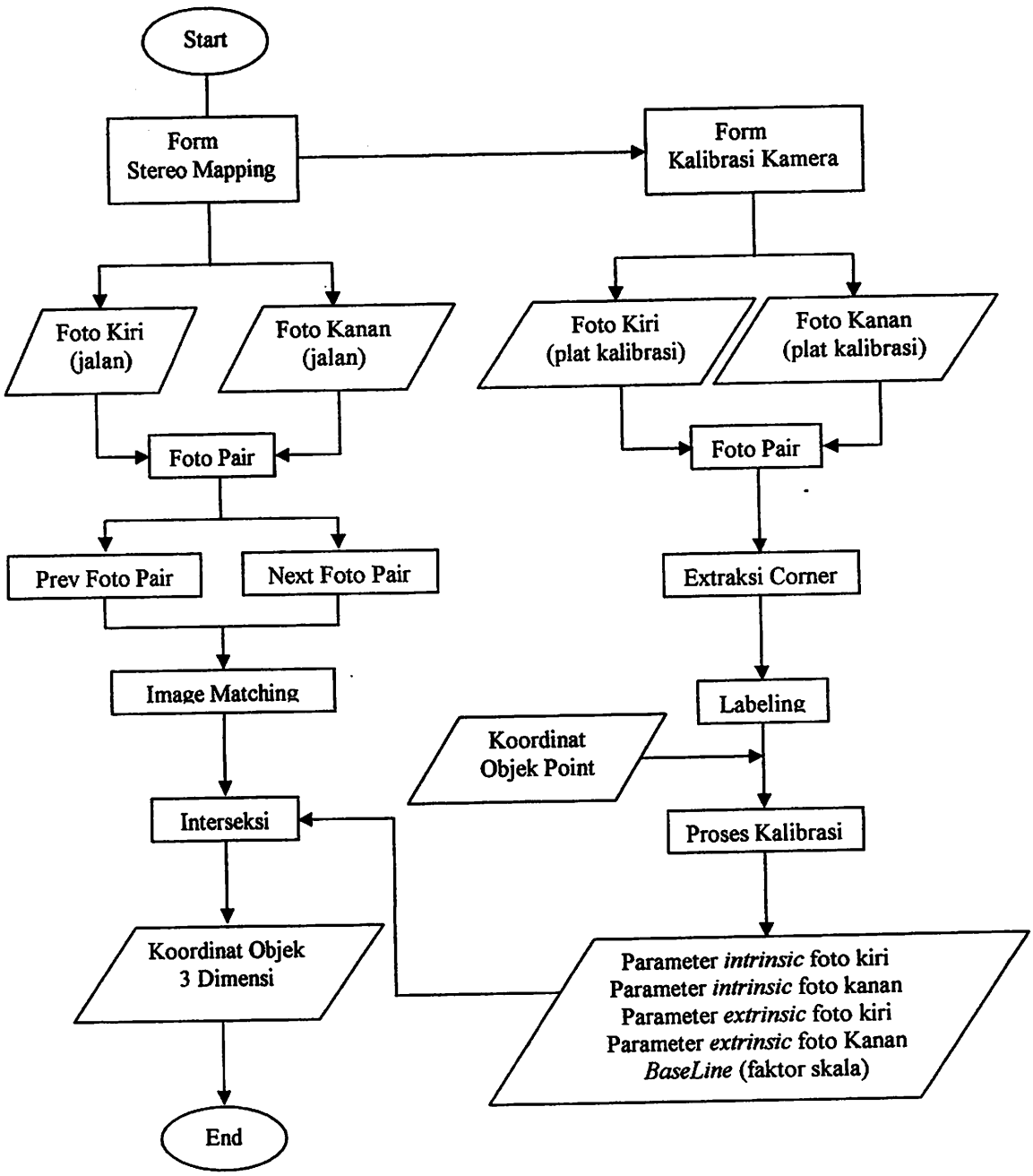



*Gambar 3.3 Data koordinat objek*

6. Proses *Single* kalibrasi merupakan proses penentuan parameter *exterior orientation* pada satu kamera baik kamera kiri dan kamera kanan, hasil dari proses single kalibrasi berupa *field of view (fovx, fovx)*, panjang fokus *(focal leght)*, aspek rasio dan *principal point (x, y)*.
7. Proses *Stereo* kalibrasi merupakan proses perhitungan *rotasi* (R) dan *translasi* (T) dari kamera kanan ke kamera kiri dimana hasil dari proses ini berupa parameter *extrinsic*, *foudamental* matriks dan *esensial* matriks.
8. Output dari program merupakan parameter-parameter kalibrasi yang selanjutnya akan di pakai untuk program *stereo mapping* diantaranya adalah:
    a. Parameter *Extrinsic* foto kiri *(fovx, fovy, focal leght, aspec rasio* dan *prinspical point)*.
    b. Parameter *Extrinsic* foto kanan *(fovx, fovy, focal leght, aspec rasio* dan *prinspical point)*.
    c. Parameter *Intrinsic* foto kiri (k1, k2, k3, p1 dan p2).
    d. Parameter *Intrinsic* foto kanan (k1, k2, k3, p1 dan p2).
    e. Faktor skala *(baseline)*.

## III.4. Diagram Alir Program

Berikut ini merupakan diagram alir program yang digambarkan secara umum:




*Gambar 3.4 Diagram alir program*

Dari diagram alir program diatas secara umum dapat disimpulkan sebagai berikut:

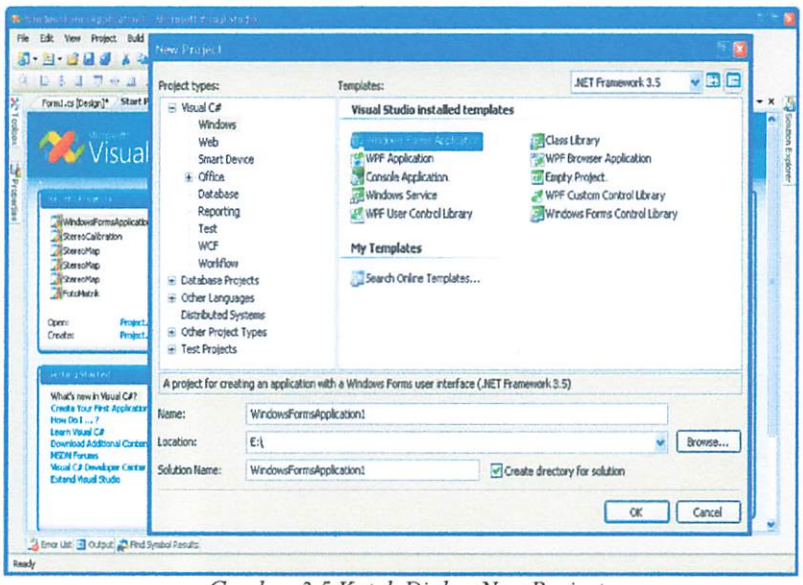
1. Program ini mempunyai dua project yaitu *project StereoMapp* dan *project StereoCalibration.*
2. Project *StereoMapp* adalah *project* penentuan koordinat objek tiga dimensi (X, Y, Z) dari pasangan foto *stereo* (objek foto jalan) sedangakan *project StereoCalibration* merupakan *project* kalibrasi kamera *stereo* (objek foto Papan kalibrasi).
3. Sebelum memulai *project StereoMapp* terlebih dahulu menjalankan project *StereoCalibration* karena *output* dari *project StereoCalibration* berupa parameter-parameter kalibrasi kamera yang akan dipakai sebagai data awal pendekatan pada proses yang berada pada *project StereoMapp* untuk menghasilkan koordinat objek tiga dimensi.

## III.5. Pembuatan Program

Pembuatan program untuk stereo kalibrasi kamera CCTV ini menggunakan bahasa pemograman *Microsoft Visual Studio C# 2008.* Program ini memiliki tampilan menu utama yang terdiri dari *Input, Proscess, Output,* dan *Exit.* Untuk menjalankan program *Microsoft Visual Studio C# 2008* maka dapat dilakukan dengan cara memilih *Start* ➔ *All Programs* ➔ *Microsoft Visual Studio 2008.*

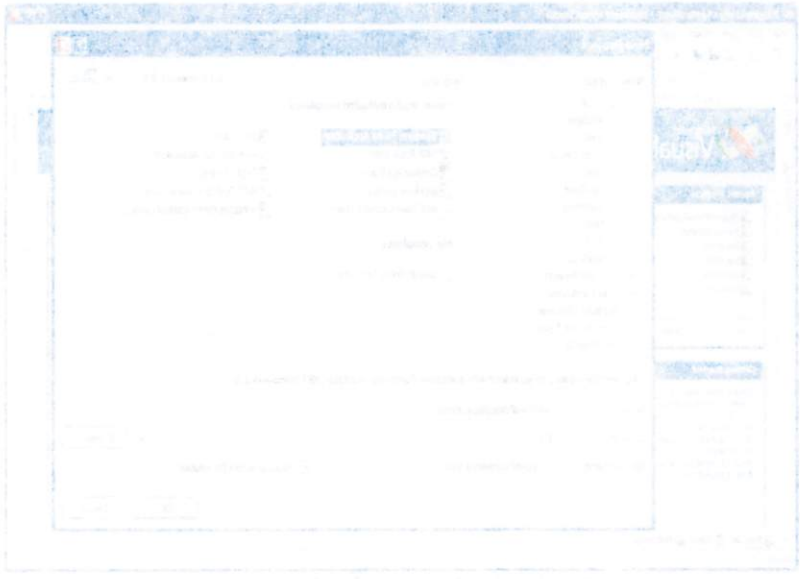
### III.5.1. Membuat *Project* Baru

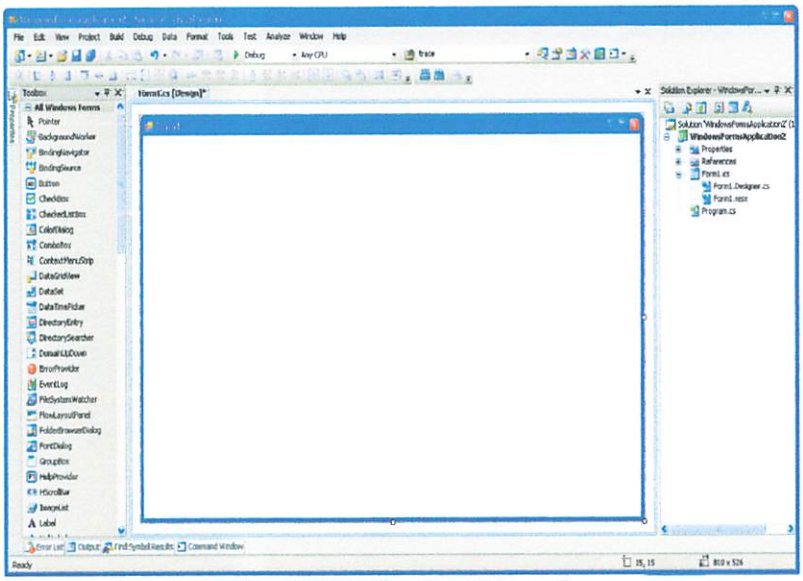
Untuk membuat project baru pada *Microsoft Visual Studio* 2008 maka pada menu *file* ➔ *new* ➔ *project*. Maka kotak dialog *New Project* akan muncul seperti gambar dibawah ini:




*Gambar 3.5 Kotak Dialog New Project*

Pada kotak dialog dialog *New Project*, pilih *Windows Form C#*, berhubung program yang akan dibuat berbentuk *Windows Form*. Maka akan muncul tampilan *Form* kosong yang merupakan tempat untuk mendesain tampilan program seperti gambar dibawah ini :

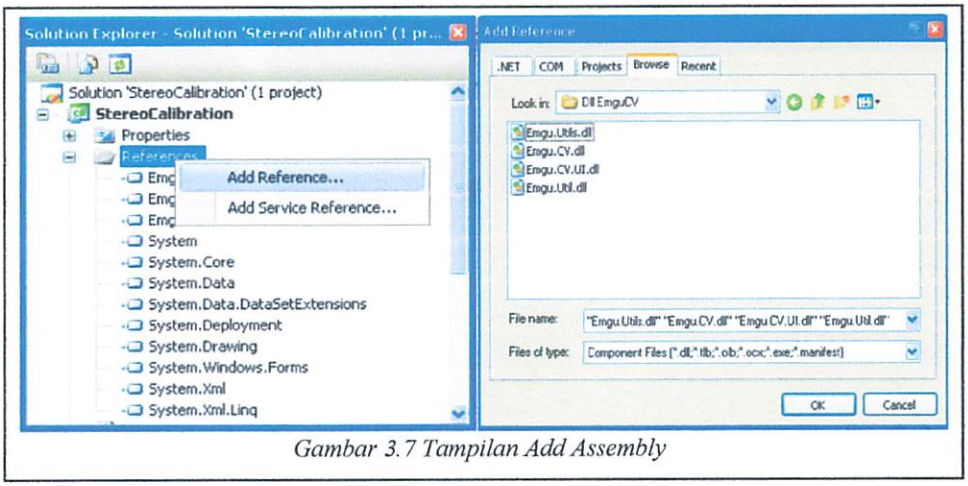




*Gambar 3.6 Tampilan Form Utama Program*

### III.5.2. Menambahkan *Assembly*

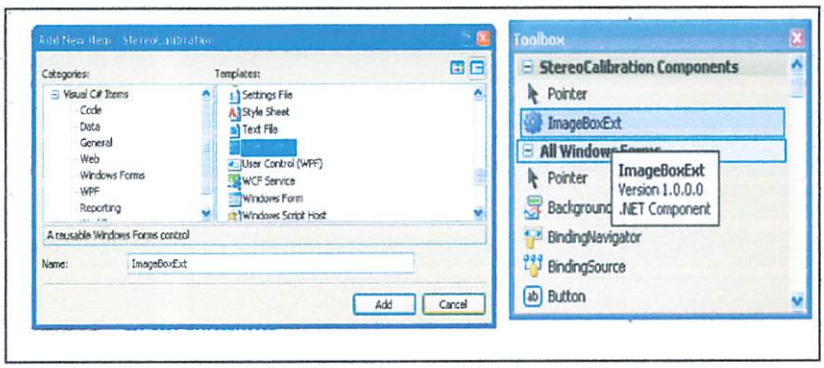
Pada panel *Solution Explorer*, klik kanan pada *References* ➔*Add References*. Kemudian akan muncul kotak dialog *Add References*. Pada menu *Browse*, masukan referensi *Emgu.CV.dll, Emgu.CV.Util.dll, Emgu.CV.Utils.dll, Emgu.CV.UI.dll* kemudian klik *OK*.




*Gambar 3.7 Tampilan Add Assembly*

### III.5.3. Menambahkan *User Control* Pada *Toolbox*

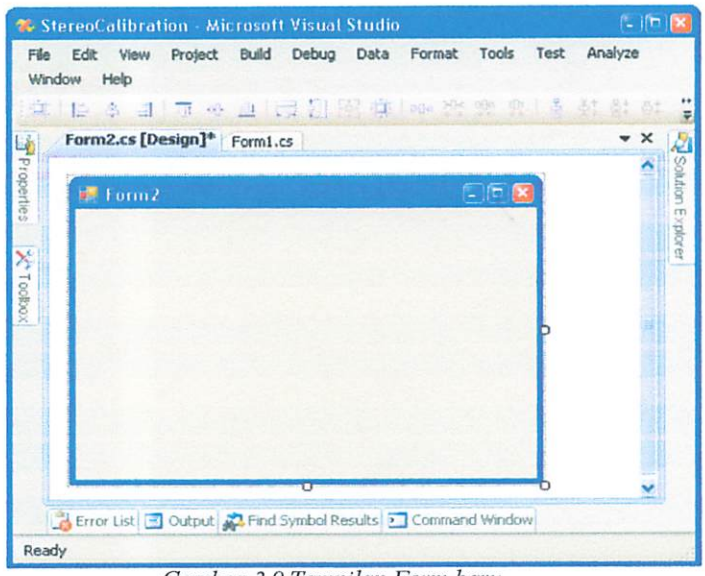
*User Control* berfungsi untuk manampilkan foto diatas *form*, dimana untuk menambahkan *User Control* pada *Toolbox*, maka klik *Project* pada *Toolbar* ➔ *Add User Control* maka akan muncul kotak dialog *Add New Item*, pilih *User Control* kemudian diberi nama *ImageBoxExt*. Klik *Add*, selanjutnya klik *Build* ➔ *Build Solution* untuk menapilkan *User Control* pada *Toolbox*.




*Gambar 3.8 Tampilan Add User Control*

### III.5.4. Menambahkan *Form* pada *Project*

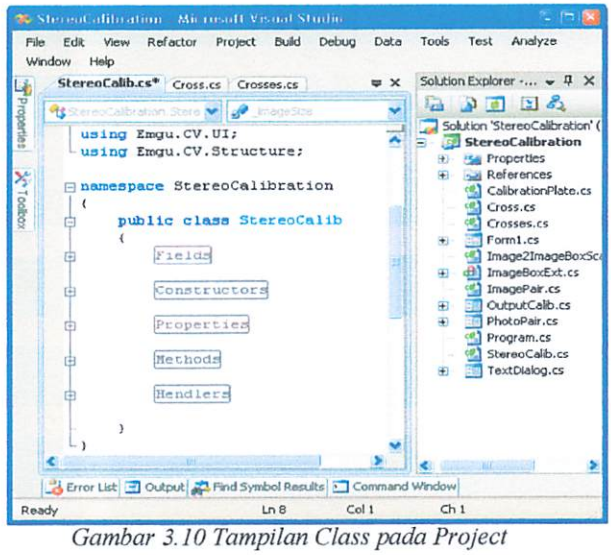
Untuk menambahkan Form baru pada *Projetc* program ini dengan klik *Projetc* ➔ *Add Windows Form* maka akan muncul tampilan kotak dialog *Add New Item (Gambar 3.8)*, kemudian beri nama *Form* pada kolom *Name*.




*Gambar 3.9 Tampilan Form baru*

### III.5.5. Menambahkan *Class*

Untuk menambahkan *Class* baru pada *Projetc* program ini dengan klik *Project* ➔ *Add Class* maka akan muncul tampilan kotak dialog *Add New Item Item (lihat Gambar 3.8),* kemudian isi *Name Class* tersebut pada kolom *Name.*




*Gambar 3.10 Tampilan Class pada Project*

Adapun *Class-class* yang mendukung program ini program ini terdiri dari:

1. *Class ImagePair* adalah *class* yang menggabungkan nama *file* foto dari *class* foto pair antara foto kiri dan foto kanan menjadi satu nama *file (String Pair Name).*

```
public ImagePair(string pairName,string
FullPathLeftPhoto, string fullPathRightPhoto)
```

2. *Class Cross* merupakan *class drawing*, dimana fungsi di dalam *class* ini untuk menggambarkan tanda *cross* (+) diatas foto serta merekam posisi koordinat piksel pada foto tersebut.

```
private void DrawCrossSign(Graphics g, float
x, float y, Pen pen)
```

3. *Class Crosess* merupakan yang sama seperti *class cross* tetapi didalam *class* ini mengoleksi poin-poin *cross* diatas gambar.

```
public void DrawCrossOnImageBox(Graphics g)
```

4. *Class CalibrationPlate* adalah *class* yang mengurutkan *point* koordinat dan *label* dari *file* inputan papan kalibrasi.

```
public PlateCalibrationPoint(PointF pointf,
String label)
```

5. *Class Image2ImageBoxScaleConvertion* merupakan *class* yang mengkonversi atau menyamakan antara ukuran *image* ke *image box* dan *image box* ke *image*.

```
public void SetScale(int imageWidth, int
imageHeight, int imageBoxWidth, int  imageBoxHeight)
```

6. *Class StereoCalib* merupakan *class* inti pembuatan program ini dimana didalam *class* ini terdapat fungsi-fungsi untuk proses kalibrasi kamera.

```
public StereoCalib(CalibrationPlate plateCoordinate,
Size imgSize, List<ImagePair> pairList)
```

### III.5.6. Penggunaan Fungsi Emgu

Berikut ini merupakan beberapa fungsi yang digunakan dalam proses pembuatan program ini adalah:

1. Fungsi *Good Features To Track*

```
public PointF[][] GoodFeaturesToTrack(
        int maxFeaturesPerChannel,
        double qualityLevel,
        double minDistance,
        int blockSize)
```

2. Fungsi *FindCornerSubPix*

```
public void FindCornerSubPix(PointF[][]corners,
        Size win,
        Size zeroZone,
        MCvTermCriteria criteria)
```

3. Fungsi *Camera Calibration*

```
public static void CalibrateCamera(
        MCvPoint3D32f[][] objectPoints,
        PointF[][] imagePoints,
        Size imageSize,
        IntrinsicCameraParameters intrinsicParam,
        CALIB_TYPE flags,
        out ExtrinsicCameraParameters
        [] extrinsicParams)
```

4. *Fungsi Stereo Calibration*

```
public static void StereoCalibrate(
        MCvPoint3D32f[][] objectPoints,
        PointF[][] imagePoints1,
        PointF[][] imagePoints2,
        IntrinsicCameraParameters
        intrinsicParam1,
        IntrinsicCameraParameters intrinsicParam2,
        Size imageSize,
        CALIB_TYPE flags,
        MCvTermCriteria termCrit, out
        ExtrinsicCameraParameters extrinsicParams,
        out Matrix<double> foundamentalMatrix,
        out Matrix<double> essentialMatrix)
```

5. Fungsi *Intrinsic Camera Parameters*

```
[SerializableAttribute]
public class IntrinsicCameraParameters :
IEquatable<IntrinsicCameraParameters>
```

6. Fungsi *ExtrinsicCameraParameters*

```
[SerializableAttribute]
public class ExtrinsicCameraParameters :
IEquatable<ExtrinsicCameraParameters>
```

7. cvCalibrationMatrixValues

```
public static void cvCalibrationMatrixValues(
        IntPtr calibMatr,
        int imgWidth,
        int imgHeight,
        double apertureWidth,
        double apertureHeight,
        ref double fovx,
        ref double fovy,
        ref double focalLength,
        ref MCvPoint2D64f principalPoint,
        ref double pixelAspectRatio)
```

# BAB IV

# HASIL DAN PEMBAHASAN

## IV.1 Hasil desain *interface*

Berikut ini merupakan tampilan *form* utama dari program stereo kalibrasi yang terdiri dari:

1. *Menu File*

   Pada *menu file* terdiri dari beberapa *menu item* diantaranya adalah:

   - ✓ *Open Project*
   - ✓ *Save Project*
   - ✓ *Add Pairs*
   - ✓ *Exit*

2. *Tab Control*

   - ✓ *Tab Stereo Calibrate Camera*
   - ✓ *Stereo Mapp*

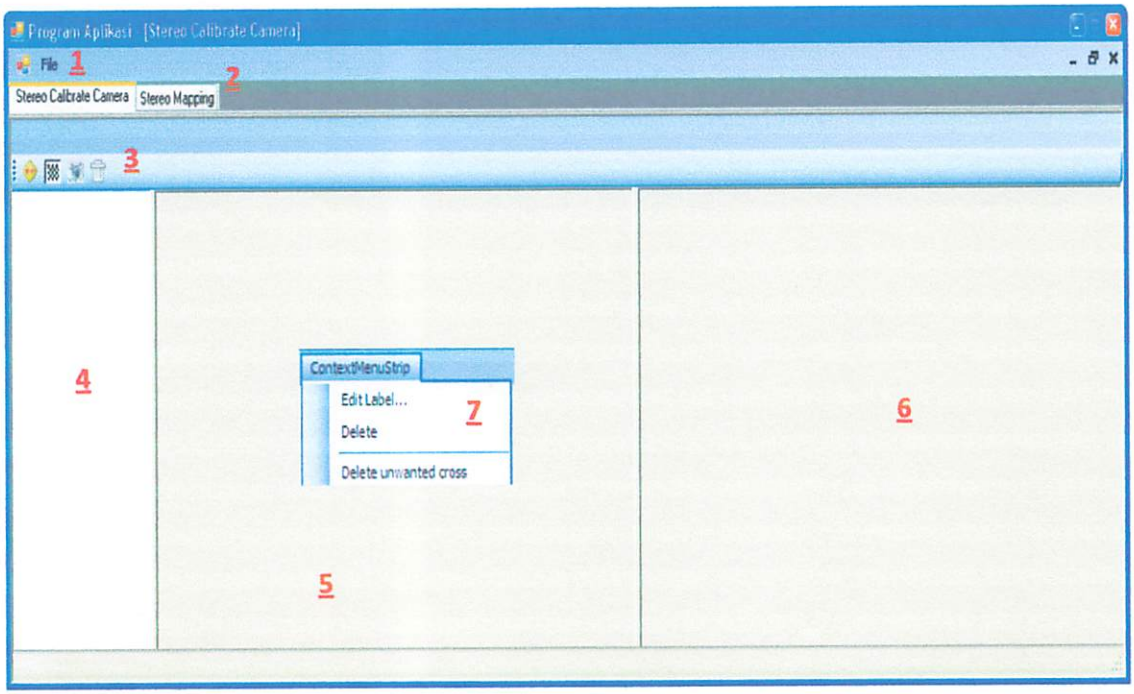
3. *Button tools strip*

   *Pada button tools strip* terdiri dari beberapa *button* diantaranya adalah:

   - ✓ *Extract Corrner*

- ✓ *Plate Point*
- ✓ *Calibration Tools*
- ✓ *DeleteUnwanted Cross*

4. *List Box*
5. *Image Box* kiri
6. *Image Box* kanan
7. *Context Menu Strip*
   - ✓ *Edit label*
   - ✓ *Delete*
   - ✓ *DeleteUnwanted Cross*



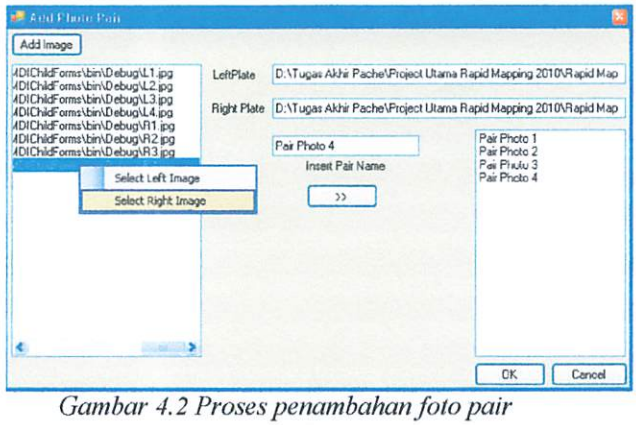

*Gambar 4.1 Tampilan Interface program*

## IV.2 Proses kerja program

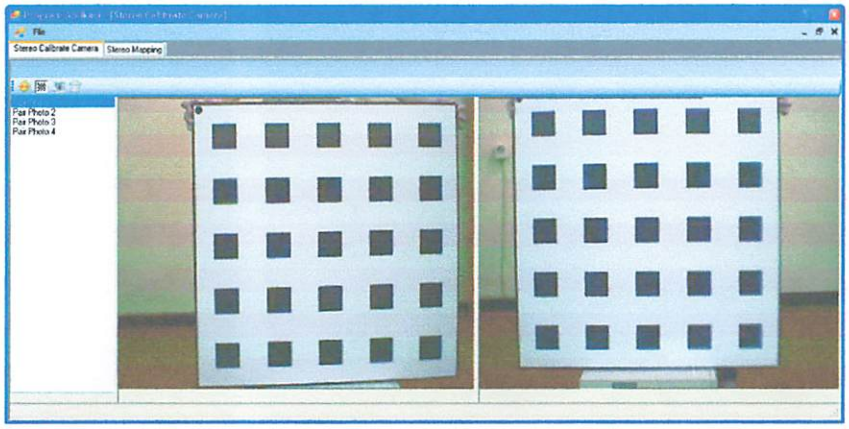
Berikut ini merupakan proses atau langkah penggunaan program kalibrasi foto stereo meliputi:

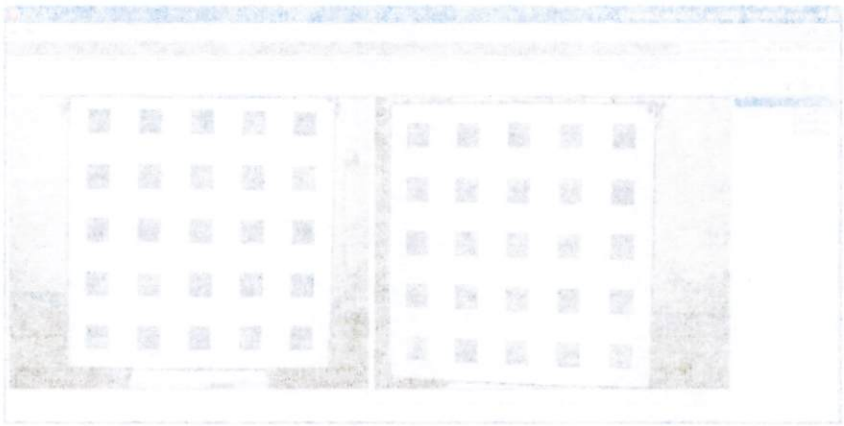
### IV.2.1 Menambahkan foto pair

Penambahan foto *pair* merupakan proses penggabungan name *file* antara foto *name file* foto kiri dan foto kanan di jadikan satu *name file* kemudian di inputkan ke *form* utama, adapun cara untuk membuat foto pair dengan mengklik *Menu* ➔ *Add Photo Pair* maka akan muncul tampilan seperti Gambar 4.2, kemudian *klik button Add Image* untuk menambahkan foto.




*Gambar 4.2 Proses penambahan foto pair*





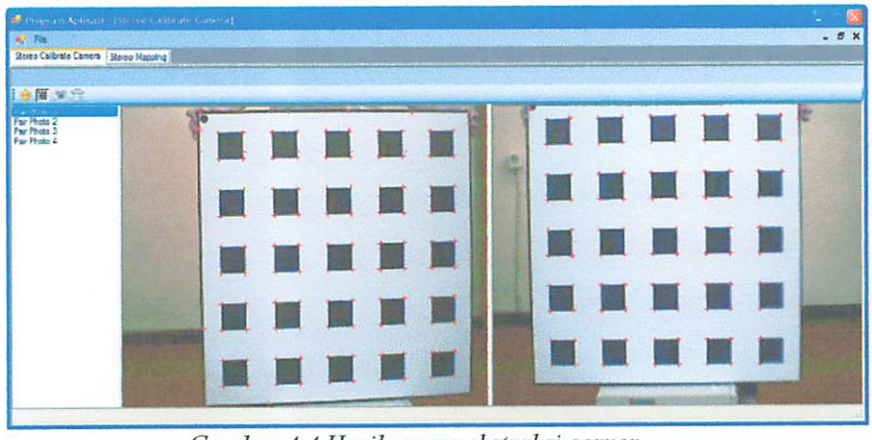
### IV.2.1.1 *Source code* penambahan foto pair

```
private void AddImageButton_Click(object sender, EventArgs e)
{
    using (OpenFileDialog dlg = new OpenFileDialog())
    {
        dlg.Title = "Add Image(s) to Project";
        dlg.Multiselect = true;
        dlg.Filter = "Image Files (JPEG, TIF, GIF, BMP, etc.)|"
           + "*.jpg;*.jpeg;*.gif;*.bmp;"
           + "*.tif;*.tiff;*.png|"
           + "JPEG files (*.jpg;*.jpeg)|*.jpg;*.jpeg|"
           + "GIF files (*.gif)|*.gif|"
           + "BMP files (*.bmp)|*.bmp|"
           + "TIFF files (*.tif;*.tiff)|*.tif;*.tiff|"
           + "PNG files (*.png)|*.png|"
           + "All files (*.*)|*.*";

        dlg.InitialDirectory = Environment.CurrentDirectory;
        dlg.RestoreDirectory = true;

        if (dlg.ShowDialog() == DialogResult.OK)
        {
            string[] files = dlg.FileNames;
            IEnumerable<string> sortAscendingQuery = from filename in files
                                                     orderby filename
                                                     select filename;

            foreach (string s in sortAscendingQuery)
            {

                listBox1.Items.Add(s);
            }
        }
    }
}
```

### IV.2.2 Extraksi *corner* dan labeling

Proses ekstraksi *corner* merupakan proses pendeteksian sudut-sudut tiap kotak-kotak pada foto papan kalibrasi, untuk mengekstrak *corner* pada foto papan kalibrasi yakni dengan mengklik *button* ekstraksi *corner* yang berada pada *butoon tools strip* maka hasil dari ekstraksi *corner* berupa tanda *cross* (+) berikut ini merupakan gambar hasil ekstraksi *corner* pada foto papan kalibrasi.

tanda *cross* (+) berikut ini merupakan gambar hasil ekstraksi *corner* pada foto papan kalibrasi.

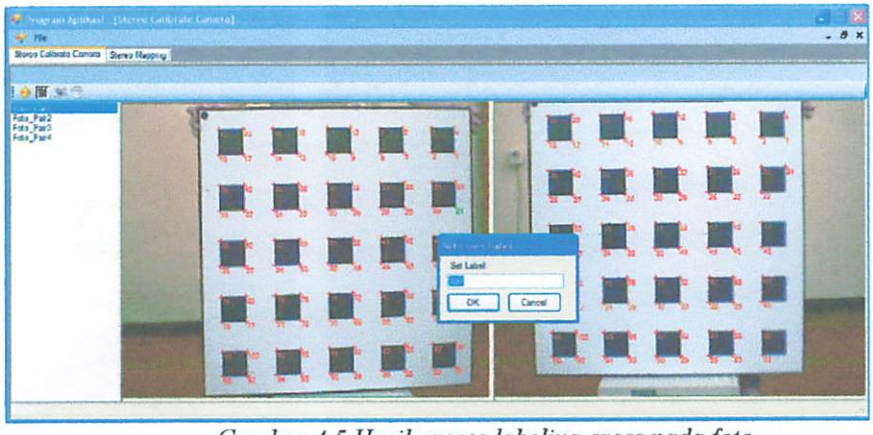


*Gambar 4.4 Hasil proses ekstraksi corner*

### IV.2.2.1 *Source code* ekstraksi *corner*

```
private void ExtractCorner(bool leftImage)
        {
            Image<Gray,Byte> temp = null;
            if (leftImage)  //true for left image
                temp = _leftPhoto.Convert<Gray, Byte>();
            else    //for right image
                temp = _rightPhoto.Convert<Gray, Byte>();

             PointF[][] corners = temp.GoodFeaturesToTrack(1000, 0.01, 1,
                                  2, true, 0.04);

            //find sub-pixel accuracy
              temp.FindCornerSubPix(corners, new Size(10, 10), new Size(-
                                  1, -1), new MCvTermCriteria(0.05));
            foreach (PointF[] p in corners)
            {
                foreach (PointF pp in p)
                {
                    Cross pt = new Cross(pp, "xx");
                    if (leftImage)

               _selectedPair.CrossesLeftPhoto.CrossPoints.Add(pt);
               else
               _selectedPair.CrossesRightPhoto.CrossPoints.Add(pt);

                }
            }
            if (leftImage)
            {
                imageBoxExtLeft.Invalidate();
                imageBoxExtLeft.Update();
            }
            else
            {
                imageBoxExtRight.Invalidate();
                imageBoxExtRight.Update();
            }
            _lastSaved = false;
        }
```

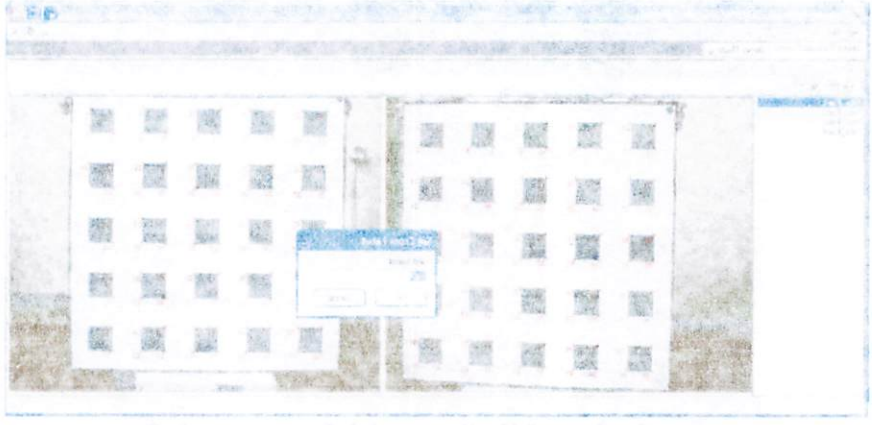
Setelah proses ektraksi *corner* maka selanjutnya proses labeling pada tanda *cross* (+) yang berada pada foto, proses editing labeling ini harus sesuai di sesuaikan dengan urutan data koordinat objek, berikut ini merupakan gambar hasil prosesing labeling.




*Gambar 4.5 Hasil proses labeling cross pada foto*

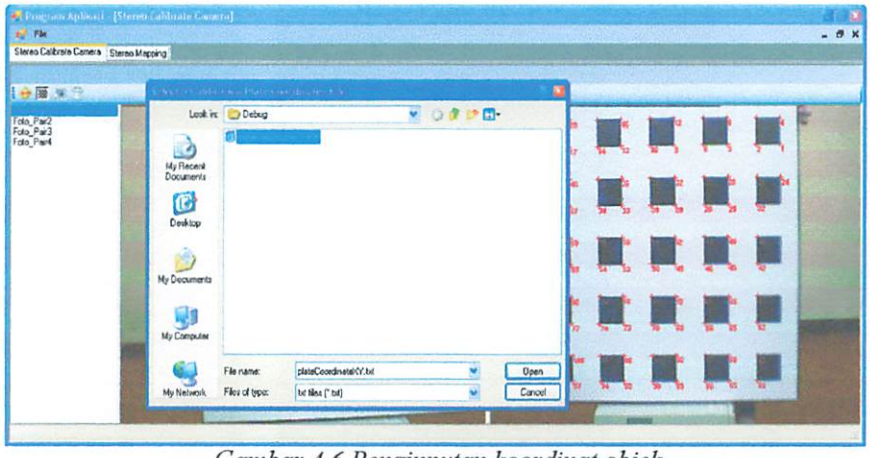
### IV.2.2.2 *Source code* labeling *corner*

```
private void editLabelContextMenuItem_Click(object sender, EventArgs e)
        {
            if (String.IsNullOrEmpty(_highlightedCross.LabelName) ||
               _highlightedCross.LabelName == "xx")
                _labelDlg.Label = _currentTempLabel;
            else
                _labelDlg.Label = _highlightedCross.LabelName;
            if (_labelDlg.ShowDialog() == DialogResult.OK)
            {
                _currentTempLabel = _labelDlg.Label;
                Trace.WriteLine("Current Label: " + _currentTempLabel);
                _highlightedCross.LabelName = _currentTempLabel;
                if (_activeLeftImageBox)
                {
                imageBoxExtLeft.Invalidate(RectangleOnImageBox(_highlightedCross.
                                                        BoundingRectangle));
                imageBoxExtLeft.Update();
                _currentTempLabel = IncrementCurrentTempLabel(_currentTempLabel);
                }
                else
                {
                imageBoxExtRight.Invalidate(RectangleOnImageBox(_highlightedCross.
                                        BoundingRectangle));
                imageBoxExtRight.Update();
                _currentTempLabel = IncrementCurrentTempLabel(_currentTempLabel);
                }
                _lastSaved = false;
                imageBoxExtLeft.Invalidate();
                imageBoxExtRight.Invalidate();
            }
        }
```

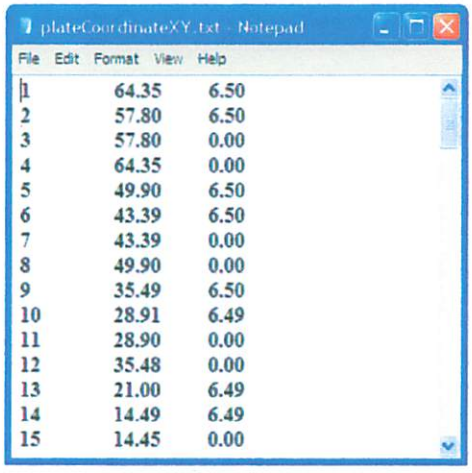


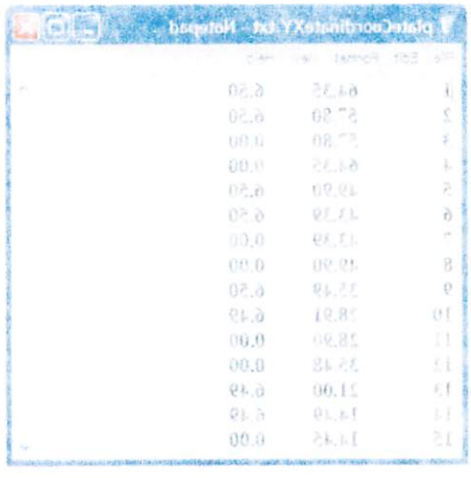
### IV.2.3 Input koordinat objek

Proses penginputan koordinat objek point merupakan proses sebelum melakukan proses kalibrasi kamera, data input koordinat objek merupakan hasil ukuran papan kalibrasi yang diukur secara manual kemudian di simpan dalam bentuk *extension file* *txt dengan dengan format urutan label, koordinat X dan koordinat Y, untuk menambahkan koordinat objek yaitu dengan mengklik *button open calibration strip*  .




*Gambar 4.6 Penginputan koordinat objek*





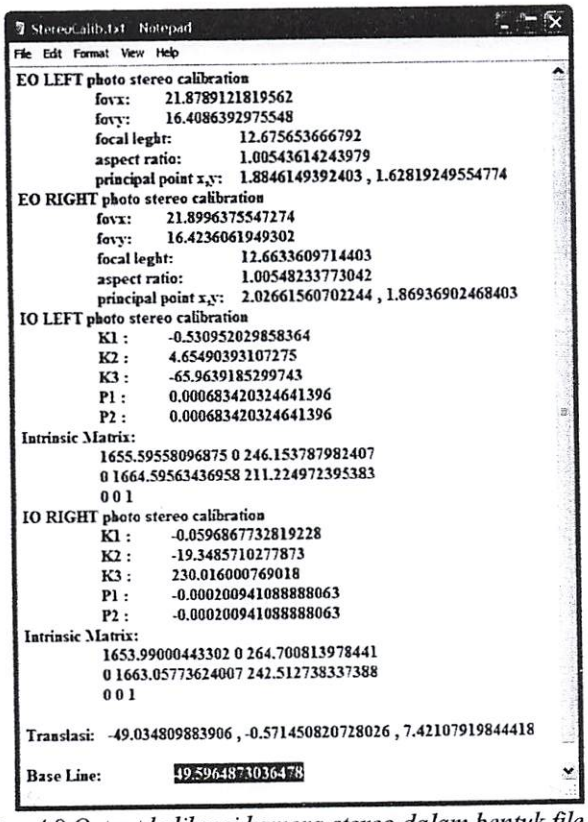
### IV.2.3.1 *Source code* input koordinat objek

```
private void platePointToolStripButton_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog dlg = new OpenFileDialog())
            {
                dlg.Title = "Select a Calibration Plate Coordinates File";
                dlg.Multiselect = false;
                dlg.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
                dlg.InitialDirectory = Environment.CurrentDirectory;

                dlg.RestoreDirectory = true;
                if (dlg.ShowDialog() == DialogResult.OK)
                {
                    PlateCalibrationPoint plates = null;
                    StreamReader sr = null;
                    sr = new StreamReader(dlg.FileName);
                    string file = sr.ReadToEnd();
                    string[] files = file.Split(new char[] { '\n' });
                    foreach (string line in files)
                    {
                       string[] coords = line.Split(new char[] { ' ', '\t' });
                       PointF pt = new PointF(float.Parse(coords[1]),
                                              float.Parse(coords[2]));
                        plates = new PlateCalibrationPoint(pt, coords[0]);
                        _platePoints.PlatePoints.Add(plates);
                       string txt = String.Format("label: {0}, x: {1}, y: {2}",
                       coords[0], coords[1], coords[2]);
                        Trace.WriteLine(txt);
                    }
                    Trace.WriteLine(_platePoints.ToString());
                    _lastSaved = false;
                    _isPlatePointsAvailable = true;
                }
            }
        }
```

### IV.2.4 Proses dan output kalibrasi kamera *stereo*

Proses ini merupakan proses akhir dari pengoprasian program kalibrasi kamera stereo dimana data-data yang di butuhkan untuk proses kalibrasi telah terkumpul seperti koordinat foto dari hasil ekstraksi *corner* dan koordinat objek, selanjutnya untuk proses kalibrasi pada program ini dengan mengklik *button strip camera camera calibration*  , maka program akan

memproses data-data tersebut dan *output* dari program ini berupa *file *.txt* seperti gambar dibawah ini.




*Gambar 4.8 Output kalibrasi kamera stereo dalam bentuk file *.txt*

### IV.2.4.1 *Source code* kalibrasi kamera stereo

```
private void CalibrateStereoPair()
        {
            //Calibrate left Camera
            ExtrinsicCameraParameters[] eoLeftPhoto;
            CameraCalibration.CalibrateCamera(_objectPoints, _imagePointsLeft,
                                              _imageSize,
                                              _ioLeftPhoto,
                                              Emgu.CV.CvEnum.CALIB_TYPE.DEFAULT,
                                              out eoLeftPhoto);

            double fovx = 0.0;
            double fovy = 0.0;
            double focal = 0.0;
            MCvPoint2D64f pp = new MCvPoint2D64f();
            double aspect = 0.0;
```

```
CvInvoke.cvCalibrationMatrixValues(_ioLeftPhoto.IntrinsicMatrix,
                    _imageSize.Width, _imageSize.Height, 4.9, 3.7,
                    ref fovx, ref fovy, ref focal, ref pp, ref aspect);


//Calibrate right Camera
ExtrinsicCameraParameters[] eoRightPhoto;
CameraCalibration.CalibrateCamera(_objectPoints, _imagePointsRight,
                                  _imageSize,
                                  _ioRightPhoto,
                                  Emgu.CV.CvEnum.CALIB_TYPE.DEFAULT,
                                  out eoRightPhoto);

CvInvoke.cvCalibrationMatrixValues(_ioLeftPhoto.IntrinsicMatrix,
                    _imageSize.Width, imageSize.Height, 4.9, 3.7,
                    ref fovx, ref fovy, ref focal, ref pp, ref aspect);

Trace.WriteLine("IO Right photo    before stereo calibration: ");
DisplayIOParams(_ioRightPhoto);

//Calibrate a stereopair by using their intrinsic params
CameraCalibration.StereoCalibrate(_objectPoints,
                                  _imagePointsLeft,
                                  _imagePointsRight,
                                  _ioLeftPhoto,
                                  _ioRightPhoto,
                                  _imageSize,

   Emgu.CV.CvEnum.CALIB_TYPE.CV_CALIB_USE_INTRINSIC_GUESS,
                                  new MCvTermCriteria(0.0001),
                                  out _eoParams,
                                  out _fundamentalMatrix,
                                  out _essentialMatrix);

   Trace.WriteLine("Translasi" +
                   _eoParams.TranslationVector[0, 0].ToString() + " , " +
                   _eoParams.TranslationVector[1, 0].ToString() + " , " +
                   _eoParams.TranslationVector[2, 0].ToString());

   Trace.WriteLine("IO left photo after stereo calibration: ");
   DisplayIOParams(_ioLeftPhoto);
   Trace.WriteLine("IO Right photo after stereo calibration: ");
   DisplayIOParams(_ioRightPhoto);

   double BaseLine = Math.Sqrt(Math.Pow(_eoParams.TranslationVector
   [0, 0], 2) + Math.Pow(_eoParams.TranslationVector[1, 0], 2) +
                Math.Pow(_eoParams.TranslationVector[2, 0], 2));

    Trace.WriteLine("BaseLine: " + BaseLine.ToString());
```

## IV.3 Hasil Program

Dari hasil perhitungan program kalibrasi foto *stereo* diatas, hasil perhitungan dari program tersebut perlu dibandingkan dengan data pembanding diantaranya sebagai berikut:

1. Data lapangan:
   - ✓ Data spesifikasi kamera CCTV dengan panjang fokus 12mm.
   - ✓ Data jarak antara kamera kiri dan kamera kanan pada bar kalibrasi yang diukur secara manual menggunakan penggaris baja (panjang *baseline*) adalah 50cm.
2. Data hasil program:

   Data hasil program tercantum dalam table berikut ini:

| **Data Kalibrasi Foto Kiri** | | |
|---|---|---|
| *fovx* | | 21.8789121819562 |
| *fovy* | | 16.4086392975548 |
| *Aspec Ratio* | | 1.00543614243979 |
| *Principle Point x, y* | | 1.8846149392403 , 1.62819249554774 |
| *Facal Leght* | | 12.675653666792 |
| *Matrix Intrinsic* | | |
| Parameter Distorsi | K1 | -0.530952029858364 |
| | K2 | 4.65490393107275 |
| | K3 | -65.9639185299743 |

| | | |
|---|---|---|
| | P1 | 0.000683420324641396 |
| | P2 | 0.000683420324641396 |
| **Data Kalibrasi Foto Kanan** | | |
| *fovx* | | 21.8996375547274 |
| *fovy* | | 16.4236061949302 |
| *Aspec Ratio* | | 1.00548233773042 |
| *Principle Point x, y* | | 2.02661560702244, 1.86936902468403 |
| *Facal Leght* | | 12.6633609714403 |
| *Matrix Intrinsic* | | 1653.99000443302 0 264.700813978441<br>0 1663.05773624007 242.512738337388<br>0 0 1 |
| Parameter Distorsi | K1 | -0.0596867732819228 |
| | K2 | -19.3485710277873 |
| | K3 | 230.016000769018 |
| | P1 | -0.000200941088888063 |
| | P2 | 0.00356828670410506 |
| *Translasi :* -49.034809883906 , -0.571450820728026 , 7.42107919844418 | | |
| *Baseline :* 49.5964873036478 | | |

*Tabel 4.1 Tabel output kalibrasi foto stereo*

# BAB V

# KESIMPULAN DAN SARAN

## V.1 Kesimpulan

Kesimpulan yang dapat diambil dari hasil penelitian mengenai pembuatan paket program kalibrasi foto *stereo* dengan menggunakan plat datar dan bahasa pemograman *Microsoft Visual Studio C#* 2008 telah memberi alternatif lain dalam proses perhitungan kalibrasi kamera stereo sebagai berikut:

1. Program kalibrasi foto *stereo* dengan menggunakan plat datar merupakan program pengolahan data citra digital dimana foto sebagai objek utama pada paket program ini.
2. Output dari program kalibrasi foto *stereo* merupakan parameter-parameter kalibrasi diantaranya adalah:
   - ✓ *field of view (fovx, fovy)* untuk foto kiri dan foto kanan.
   - ✓ *Aspec Ratio* untuk foto kiri dan foto kanan.
   - ✓ *Principal Point* untuk foto kiri dan foto kanan.
   - ✓ *Intrinsic* dan *Extrinsic Matrix.*
   - ✓ Parameter distorsi (k1, k2, k3, p1, p2).
   - ✓ *BaseLine.*
   3. Selisih panjang *baseline* yang di hasilkan oleh program ini antara kamera kiri dan kamera kanan adalah 0.403513

mm, nilai selisih tersebut sudah cukup kecil sekitar 0.00807 % dari ukuran bar kalibrasi.

4. Penggunaan funsi-fungsi *Emgu Library* yang di sediakan oleh *Open Computer Vision (OpenCV)* sangat membantu dalam proses pengolahan image didalam program ini.
5. Pembuatan program ini di tujukan untuk mengkalibrasi kamera stereo non metrik.

## V.2 Saran

Saran yang dapat diberikan dari hasil penelitian yang dilakukan selama ini, yang menyangkut pembuatan paket program kalibrasi foto *stereo* sebagai berikut:

1. Sebaikanya menggunakan kamera yang beresolusi tinggi.
2. Program ini masih sebatas kalibrasi strereo kamera CCTV dengan spesifikasi tertentu, dan di harapkan kedepannya program ini dapat lebih fleksibel untuk semua tipe kamera.
3. Perlunya penataan *interface* dari program ini agar dalam proses pengolahan program ini tidak terlalu memakan banyak memori pada Komputer.
4. Diharapakan bagi yang ingin memperdalam ilmu Fotogrametri Dijital dalam bidang pengolahan citra dijital lebih banyak mempelajari konsep-konsep pemograman komputer dan pengolahan data citra dijital.

## DAFTAR PUSTAKA


Andi, P. (2008). Belajar Pemograman C#. Yogyakarta.

Bradski, G. and A. Kaebler (2008). Learning OpenCV Computer Vision with the OpenCV Library. Gravenstein Highway North, Sebastopol.

Cooper, M. A. R. and S. Robson (2001). Close Range Photogrammetry and Computer Vision : Theory of Close Range Photogrammetry. Scotland, Whittles Publishing Services.

Dörstel, C., K. Jacobsen, et al. (2002). "DMC-Photogrammetric Accuracy-Calibration Aspects and Generation of Synthetic DMC Images." IAPRS.

Elias, R. (2007). "Enhancing Accuracy of Camera Rotation Angles Detected by Inaccurate Sensors and Expressing them in Different Systems for Wide Baseline Stereo." Computer Science and Engineering Department.

Eternity, M. (2009). "Perbaiki distorsi dan perspektif ", from http://majesty-eternity.blogspot.com/2010/02/perbaiki-distorsi-dan-perspektif-dengan.html.

Fraser, C. S. (2006). "Interior Orientation and Network Design." Lecture Notes 2 **University of Melbourne**.

Fryer, J. G. (1981). Camera Calibration In Non-Topographic Photogrammetry.

Fryer, J. G. (1983). "Lens Distortion For Close Range Photogrammetry."

Fryer, J. G. (1983). "Lens Distortion for Close Range Photogrammetry." Depertment of Civil Eingineering and Surveying.

George E. Karras, D. M. (2001). "Simple Calibration Techniques for Non-Metric Cameras." CIPA International Symposium, Potsdam(Department of Surveying, National Techical ).

hanzzoid ( 2007). CSML Documentation. **v1.1**.

Horn, B. K. P. (2000). "Tsai's Camera Calibration Method Revisited."

Itong. (2008). "Belajar C# - Pemrograman Berorientasi Objek - itong."

J.Jedlička and M.Potůčková (1999). "CORRECTION OF RADIAL DISTORTION IN DIGITAL IMAGES." Charles University in Prague Faculty of Science.

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

Lucas Teixeira, M. G., and Manuel Fernandez (2006). "Zhang's Camera Calibration: Step By Step."

Mikhail, E. M., J. S. Bethel, et al. (2001). Introduction to Modern Photogrammetry. Brisbane, John Wiley & Sons Inc.

Nuraini, R. (2007). "Sekilas tentang Close Range Photogrammetry." from http://geodesy.gd.itb.ac.id/nrahmah/?p=9.

Shih, T. Y. (1994). "RLT : A Closed Form Solution for Relative Orientation." IAPRS: 357-363.

Slabaugh, G. G. (2004). "Computing Euler Angels from a Rotation Matrix." 1-6.

Stefanovic, P. (1973). "Relative Orientation - a New Approach." ITC Journal: 417-448.

Thompson, E. H. (1959). "A Rational Algebraic Formulation of the Problem of Relative Orientation." Photogrammetric Record **Vol III**.

Tjahjadi, E. (2009). Precision Feature Extraction from Unmanned Aerial Platforms, Jurusan Teknik Geodesi, Fakultas Teknik Sipil dan Perencanaan.

Trucco, E. and A. Verri (1998). Introductory Techniques for 3D Computer Vision. Edinburgh, Prentice Hall.

Tsai, R. Y. (1987). "A Versatile Camera Calibration Techinique for High-Accuracy 3D Machine Vision Metrology using Off-the-Shelf TV Cameras and Lenses." IEEE(Jurnal of Robotics and Automatic).

Wikipedia (2006). "Distortion."

Wikipedia (2009). "Emgu Computer Vision."

Wikipedia. (2008). "Pinhole Camera Models." 2010.

Wolf, P. R. (1993). Elemen Fotogrametri. Yogyakarta, Gajah Mada University Press.

Wolf, P. R. and B. A. Dewitt (2000). Elements of Photogrammetry : with application in GIS 3rd Edition. New York, McGraw-Hill Companies.

Zhang, Z. (1998). "Flexible Camera Calibration By Viewing a Plane From Uknown Orientation." IEEE.

## LAMPIRAN A

## TABEL KOORDINAT OBJEK

| TITIK | KOORDINAT | | TITIK | KOORDINAT | |
|---|---|---|---|---|---|
| | X(cm) | Y(cm) | | X(cm) | Y(cm) |
| 1 | 64.350 | 6.500 | 51 | 28.920 | 28.500 |
| 2 | 57.800 | 6.500 | 52 | 35.500 | 28.500 |
| 3 | 57.800 | 0.000 | 53 | 21.000 | 34.990 |
| 4 | 64.350 | 0.000 | 54 | 14.500 | 35.000 |
| 5 | 49.900 | 6.500 | 55 | 14.490 | 28.500 |
| 6 | 43.390 | 6.500 | 56 | 21.000 | 28.500 |
| 7 | 43.390 | 0.000 | 57 | 6.550 | 35.000 |
| 8 | 49.900 | 0.000 | 58 | 0.000 | 35.000 |
| 9 | 35.490 | 6.500 | 59 | 0.000 | 28.510 |
| 10 | 28.910 | 6.490 | 60 | 6.550 | 28.510 |
| 11 | 28.900 | 0.000 | 61 | 64.390 | 49.290 |
| 12 | 35.480 | 0.000 | 62 | 57.810 | 49.250 |
| 13 | 21.000 | 6.490 | 63 | 57.810 | 42.800 |
| 14 | 14.490 | 6.490 | 64 | 64.380 | 42.800 |
| 15 | 14.450 | 0.000 | 65 | 49.910 | 49.290 |
| 16 | 21.000 | 0.000 | 66 | 43.400 | 49.250 |
| 17 | 6.540 | 6.500 | 67 | 43.400 | 42.790 |
| 18 | 0.000 | 6.500 | 68 | 49.900 | 42.800 |
| 19 | 0.000 | 0.000 | 69 | 35.500 | 49.290 |
| 20 | 6.550 | 0.000 | 70 | 28.990 | 49.250 |
| 21 | 64.350 | 20.700 | 71 | 28.950 | 42.790 |
| 22 | 57.800 | 20.700 | 72 | 35.500 | 42.800 |
| 23 | 57.800 | 14.300 | 73 | 21.200 | 49.250 |
| 24 | 64.350 | 14.290 | 74 | 14.500 | 49.250 |
| 25 | 49.900 | 20.700 | 75 | 14.500 | 42.800 |
| 26 | 43.400 | 20.700 | 76 | 21.000 | 42.800 |
| 27 | 43.400 | 14.290 | 77 | 6.550 | 49.290 |
| 28 | 49.900 | 14.300 | 78 | 0.000 | 49.290 |
| 29 | 35.500 | 20.700 | 79 | 0.000 | 42.800 |
| 30 | 28.950 | 20.700 | 80 | 6.550 | 42.800 |
| 31 | 28.910 | 14.280 | 81 | 64.400 | 63.500 |
| 32 | 35.500 | 14.290 | 82 | 57.850 | 63.500 |
| 33 | 21.000 | 20.700 | 83 | 57.830 | 57.050 |
| 34 | 14.490 | 20.700 | 84 | 64.400 | 57.090 |
| 35 | 14.490 | 14.290 | 85 | 49.970 | 63.500 |
| 36 | 21.000 | 14.290 | 86 | 43.410 | 63.500 |
| 37 | 6.550 | 20.700 | 87 | 43.400 | 57.050 |

| TITIK | KOORDINAT | | TITIK | KOORDINAT | |
|---|---|---|---|---|---|
| | X(cm) | Y(cm) | | X(cm) | Y(cm) |
| 38 | 0.000 | 20.710 | 88 | 49.950 | 57.090 |
| 39 | 0.000 | 14.300 | 89 | 35.500 | 63.500 |
| 40 | 6.550 | 14.300 | 90 | 28.990 | 63.500 |
| 41 | 64.350 | 35.000 | 91 | 28.990 | 57.050 |
| 42 | 57.800 | 35.000 | 92 | 35.500 | 57.090 |
| 43 | 57.800 | 28.500 | 93 | 21.200 | 63.500 |
| 44 | 64.350 | 28.500 | 94 | 14.500 | 63.500 |
| 45 | 49.900 | 34.990 | 95 | 14.500 | 57.090 |
| 46 | 43.400 | 34.990 | 96 | 21.200 | 57.090 |
| 47 | 43.400 | 28.500 | 97 | 6.550 | 63.500 |
| 48 | 49.900 | 28.500 | 98 | 0.000 | 63.500 |
| 49 | 35.500 | 34.990 | 99 | 0.000 | 57.090 |
| 50 | 28.930 | 34.990 | 100 | 6.560 | 57.010 |

# LAMPIRAN B

## FOTO PAPAN KALIBRASI

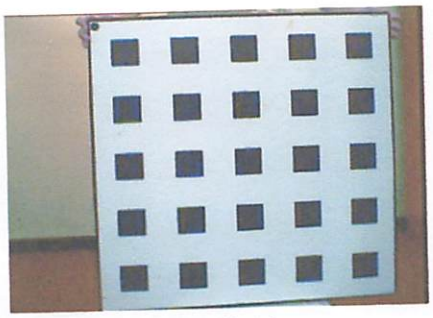


*Foto Kiri 1*

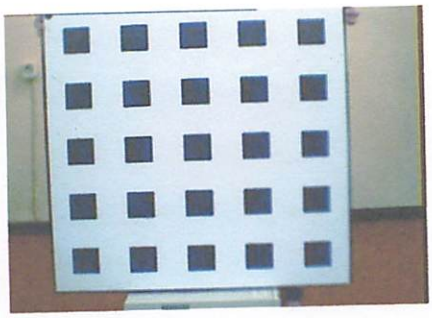


*Foto Kanan 1*

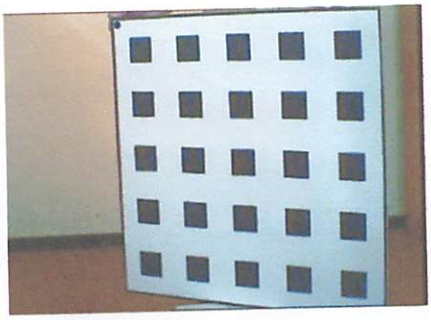


*Foto Kiri 2*

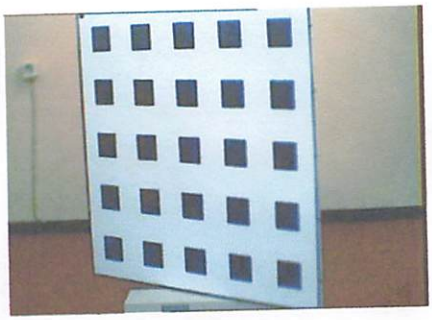


*Foto Kanan 2*

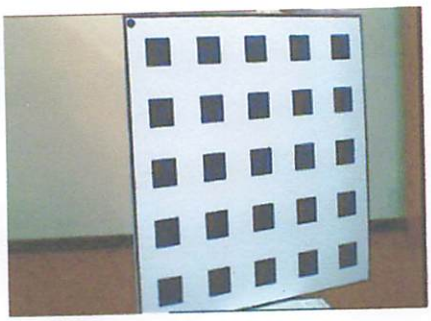


*Foto Kiri 3*

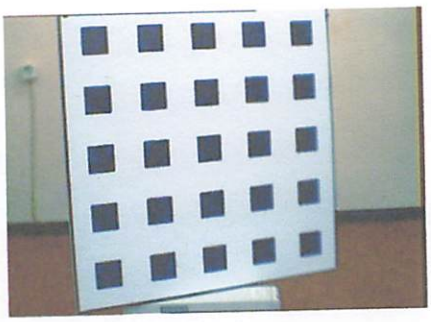


*Foto Kanan 3*

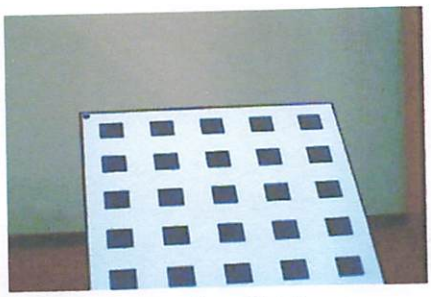


*Foto Kiri 4*

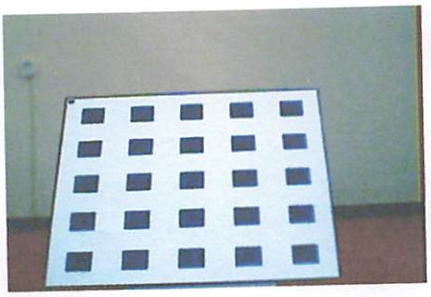
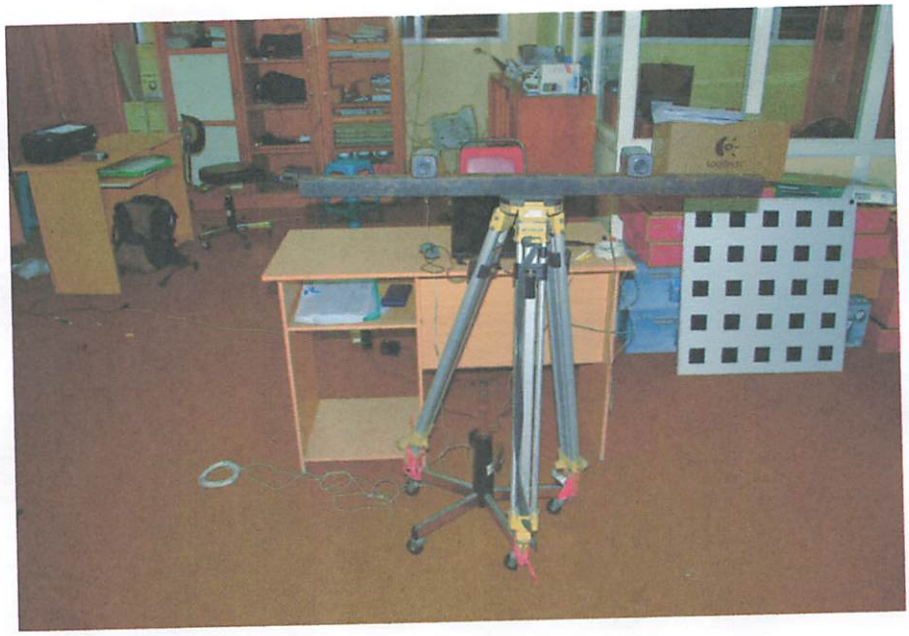
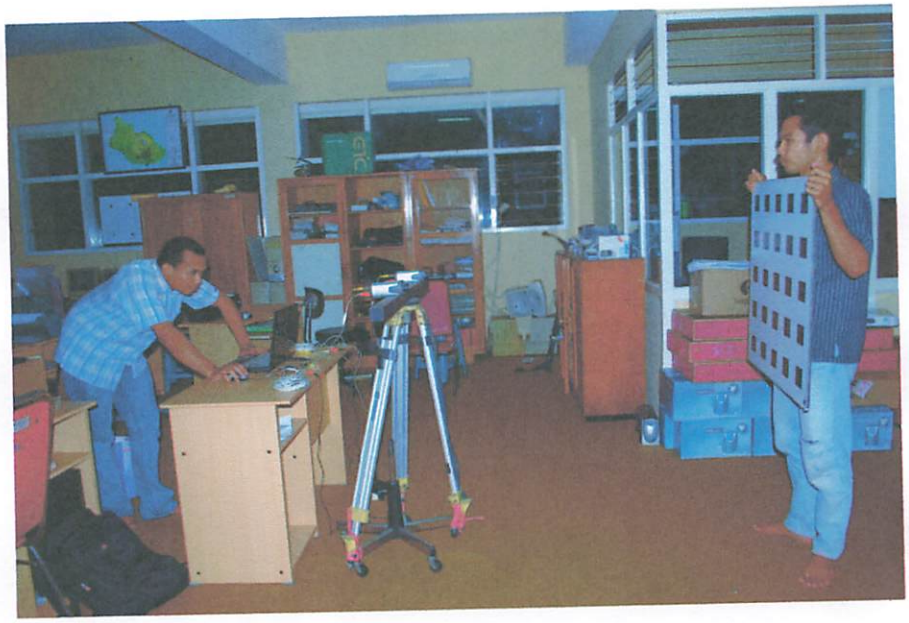


*Foto Kanan 4*

## LAMPIRAN C

## FOTO PROSES PENGAMBILAN DATA KALIBRASI KAMERA STEREO





# LAMPIRAN D
# SOURCE PROGRAM

C:\Users\pache\Desktop\Rapid Mapping Aplication\StereoCalibrationF\StereoCalibration\Form1.cs 1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using System.IO;
using Emgu.CV;
using Emgu.Util;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using System.Threading;

namespace StereoCalibration
{
    public partial class Form1 : Form
    {
        #region Fields
        List<ImagePair> _photoPairList;
        Image<Bgr, Byte> _leftPhoto;
        Image<Bgr, Byte> _rightPhoto;
        private String _projectName = String.Empty;
        private bool _lastSaved = true;
        ImagePair _selectedPair;
        private TextDialog _labelDlg;
        private String _currentTempLabel;

        //KeyValuePair<String, Crosses> _selected_leftPhoto_leftCrosses;
        //KeyValuePair<String, Crosses> _selected_rightPhoto_rightCrosses;
        bool _drawing;
        private Image2ImageBoxScaleConversion _image2imageBoxScale;
        private Cross _crossPoint;      //temporary cross point
        bool _activeLeftImageBox = true; //whether imagebox left focus
        protected Cross _highlightedCross;


        private CalibrationPlate _platePoints;
        private bool _isPlatePointsAvailable;



        #endregion


        #region Constructors
        public Form1()
        {
            InitializeComponent();
            _photoPairList = new List<ImagePair>();
            _selectedPair = null;

            _drawing = false;
            _image2imageBoxScale = new Image2ImageBoxScaleConversion();
            _highlightedCross = null;

            _labelDlg = new TextDialog();
```

```
            _currentTempLabel = "1";

            //calibration plate
            _platePoints = new CalibrationPlate();
            _isPlatePointsAvailable = false;




        }
        #endregion

        #region Handlers
        private void addPairsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            using (PhotoPair app = new PhotoPair())
            {
                if (app.ShowDialog() == DialogResult.OK)
                {
                    if (app.PhotoPairList.Count > 0)
                    {
                        foreach (ImagePair p in app.PhotoPairList)
                        {
                            _photoPairList.Add(p);
                        }
                    }
                }
            }

            //Fill ListBox with pairName
            if (_photoPairList.Count > 0)
            {
                if (listBox1.Items.Count > 0)
                    listBox1.Items.Clear();
                foreach (ImagePair pair in _photoPairList)
                {
                    listBox1.Items.Add(pair.PhotoPairName);
                }
            }

        }

        private void listBox1_SelectedValueChanged(object sender, EventArgs e)
        {
            if (_photoPairList.Count > 0)
            {
                foreach (ImagePair pair in _photoPairList)
                {
                    if ((string)listBox1.SelectedItem == pair.PhotoPairName)
                    {
                        _leftPhoto = new Image<Bgr, Byte>((string)pair.LeftPhoto);
                        _rightPhoto = new Image<Bgr, byte>((string)pair.RightPhoto);
                        imageBoxExtLeft.Image = _leftPhoto;
                        imageBoxExtLeft.PhotoName = pair.LeftPhoto;
                        imageBoxExtRight.Image = _rightPhoto;
                        imageBoxExtRight.PhotoName = pair.RightPhoto;

                        _selectedPair = pair;
```

```
                    //set left imageBox scale
                    _image2imageBoxScale.SetScale(imageBoxExtLeft.Image.Bitmap.Width,
                                                  imageBoxExtLeft.Image.Bitmap.Height,
                                                  imageBoxExtLeft.Width,
                                                  imageBoxExtLeft.Height);
                    _selectedPair.CrossesLeftPhoto.SetScaleForDisplay(
                                    _image2imageBoxScale.HorisontalScaleImage2ImageBox,
                                    _image2imageBoxScale.VerticalScaleImage2ImageBox,
                                    _image2imageBoxScale.HorisontalScaleImageBox2Image,
                                    _image2imageBoxScale.VerticalScaleImageBox2Image);
                    //set right imageBox scale
                    _image2imageBoxScale.SetScale(imageBoxExtRight.Image.Bitmap.Width,
                                                  imageBoxExtRight.Image.Bitmap.Height,
                                                  imageBoxExtRight.Width,
                                                  imageBoxExtRight.Height);
                    _selectedPair.CrossesRightPhoto.SetScaleForDisplay(
                                    _image2imageBoxScale.HorisontalScaleImage2ImageBox,
                                    _image2imageBoxScale.VerticalScaleImage2ImageBox,
                                    _image2imageBoxScale.HorisontalScaleImageBox2Image,
                                    _image2imageBoxScale.VerticalScaleImageBox2Image);
                }
            }
        }
    }

    private void openProjectToolStripMenuItem_Click(object sender, EventArgs e)
    {
        using (OpenFileDialog dlg = new OpenFileDialog())
        {
            dlg.Title = "Open Project";
            dlg.DefaultExt = "txt";
            dlg.Filter = "Album files (*.txt)|*.txt|"
                + "All files|*.*";
            dlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.
Personal);
            dlg.RestoreDirectory = true;
            dlg.Multiselect = false;

            if (dlg.ShowDialog() == DialogResult.OK)
            {
                _projectName = dlg.FileName;
                ReadProject();
                _lastSaved = false;
                //_isPlatePointsAvailable = false;

                Trace.WriteLine("Test Open Project");
                Trace.WriteLine("Number of ImagePair: " + _photoPairList.Count.ToString())
;
                foreach(ImagePair p in _photoPairList)
                {
                    Trace.WriteLine(p.PhotoPairName);
                    Trace.WriteLine(p.LeftPhoto);
                    Trace.WriteLine(p.RightPhoto);
                    Trace.WriteLine("cross on left photo" + p.CrossesLeftPhoto.CrossPoints
.Count.ToString());
                    Trace.WriteLine("cross on right photo" + p.CrossesRightPhoto.
CrossPoints.Count.ToString());
                }
            }
```

```
            }
        }

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

                    if (dlg.ShowDialog() == DialogResult.OK)
                    {
                        _projectName = dlg.FileName;
                        if (SaveProject())
                            _lastSaved = true;
                        else
                            _lastSaved = false;
                    }
                }
            }
            else if (!_lastSaved && !String.IsNullOrEmpty(_projectName))
            {
                if (SaveProject())
                    _lastSaved = true;
                else
                    _lastSaved = false;
            }
        }

        private void imageBoxExtLeft_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left && _leftPhoto != null)
            {
                _activeLeftImageBox = true;
                _drawing = true;
                //cross from mouse click, done in image coordinates => convert it to image
coord
                PointF temp = ImageBox2Image(e.Location);
                _crossPoint = new Cross(temp, "xx");
                Trace.WriteLine("crossPoint from left Photo Available: " + _crossPoint.
ToString());
            }

            if (e.Button == MouseButtons.Right && _leftPhoto != null)
            {
                _activeLeftImageBox = true;
            }
        }

        private void imageBoxExtRight_MouseDown(object sender, MouseEventArgs e)
        {
```

```
        if (e.Button == MouseButtons.Left && _rightPhoto != null)
        {
            _activeLeftImageBox = false;
            _drawing = true;
            //cross from mouse click, done in image coordinates => convert it to image coord
            PointF temp = ImageBox2Image(e.Location);
            _crossPoint = new Cross(temp, "xx");
            Trace.WriteLine("crossPoint from Right Photo Available: " + _crossPoint.ToString());
        }
        if (e.Button == MouseButtons.Right && _rightPhoto != null)
        {
            _activeLeftImageBox = false;
        }
    }

    private void imageBoxExtLeft_MouseMove(object sender, MouseEventArgs e)
    {
        _activeLeftImageBox = true;
        if (_leftPhoto != null)
        {
            if (_drawing)
            { }
            else
            {
                Cross cross = _selectedPair.GetCross(_activeLeftImageBox, ImageBox2Images(e.Location));
                if (_highlightedCross == cross)
                    return;
                //reset any existing highlighted element
                if (_highlightedCross != null)
                {
                    imageBoxExtLeft.Invalidate(RectangleOnImageBox(_highlightedCross.BoundingRectangle));
                    _highlightedCross.Highlighted = false;
                    _highlightedCross = null;
                }
                // Find and set new highlighted element, if any
                if (cross != null)
                {
                    _highlightedCross = cross;
                    _highlightedCross.Highlighted = true;
                    imageBoxExtLeft.Invalidate(RectangleOnImageBox(_highlightedCross.BoundingRectangle));
                }
                imageBoxExtLeft.Update();
            }
        }
    }

    private void imageBoxExtRight_MouseMove(object sender, MouseEventArgs e)
    {
        _activeLeftImageBox = false;
        if (_rightPhoto != null)
        {
            if (_drawing)
            { }
            else
```

```
            {
                Cross cross = _selectedPair.GetCross(_activeLeftImageBox, ImageBox2Images
(e.Location));
                if (_highlightedCross == cross)
                    return;
                //reset any existing highlighted element
                if (_highlightedCross != null)
                {
                    imageBoxExtRight.Invalidate(RectangleOnImageBox(_highlightedCross.
BoundingRectangle));
                    _highlightedCross.Highlighted = false;
                    _highlightedCross = null;
                }
                // Find and set new highlighted element, if any
                if (cross != null)
                {
                    _highlightedCross = cross;
                    _highlightedCross.Highlighted = true;
                    imageBoxExtRight.Invalidate(RectangleOnImageBox(_highlightedCross.
BoundingRectangle));
                }
                imageBoxExtRight.Update();
            }
        }
    }

    private void imageBoxExtLeft_MouseUp(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left && _leftPhoto != null)
        {
            if (!_drawing)
                return;
            _drawing = false;

            if (_crossPoint != null)
            {
                _selectedPair.CrossesLeftPhoto.CrossPoints.Add(_crossPoint);
                Trace.WriteLine("Number of cross points on left image: " + _selectedPair.
CrossesLeftPhoto.CrossPoints.Count.ToString());
                //_crossesOnEachImage.GetCrosses(_currentSelectedImageName).AddCross
(_crossPoint);
                imageBoxExtLeft.Invalidate(RectangleOnImageBox(_crossPoint.
BoundingRectangle));
                imageBoxExtLeft.Update();
                _crossPoint = null;
            }
            _lastSaved = false;
        }
    }

    private void imageBoxExtRight_MouseUp(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left && _rightPhoto != null)
        {
            if (!_drawing)
                return;
            _drawing = false;

            if (_crossPoint != null)
```

```
            {
                _selectedPair.CrossesRightPhoto.CrossPoints.Add(_crossPoint);
                Trace.WriteLine("Number of cross points on right image: " + _selectedPair.
CrossesRightPhoto.CrossPoints.Count.ToString());
                //_crossesOnEachImage.GetCrosses(_currentSelectedImageName).AddCross
(_crossPoint);
                imageBoxExtRight.Invalidate(RectangleOnImageBox(_crossPoint.
BoundingRectangle));
                imageBoxExtRight.Update();
                _crossPoint = null;
            }
            _lastSaved = false;
        }
    }

    private void imageBoxExtLeft_Paint(object sender, PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        //Draw Cross
        if (_leftPhoto != null && _selectedPair.CrossesLeftPhoto.CrossPoints.Count > 0)
        {
            _selectedPair.CrossesLeftPhoto.DrawCrossOnImageBox(g);
        }

    }

    private void imageBoxExtRight_Paint(object sender, PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        //Draw Cross
        if (_rightPhoto != null && _selectedPair.CrossesRightPhoto.CrossPoints.Count > 0)
        {
            _selectedPair.CrossesRightPhoto.DrawCrossOnImageBox(g);
        }
    }

    private void contextMenuStrip1_Opening(object sender, CancelEventArgs e)
    {
        contextMenuStrip1.Items.Clear();
        if (_highlightedCross != null)
        {
            contextMenuStrip1.Items.Add(editLabelContextMenuItem);
            contextMenuStrip1.Items.Add(deleteContextMenuItem);
            //contextMenuStrip1.Items.Add(sendToBackContextMenuItem);
        }
        else
        {
            contextMenuStrip1.Items.Add(deleteUnwantedCrossToolStripMenuItem);
            //contextMenuStrip1.Items.Add(lineContextMenuItem);
            //contextMenuStrip1.Items.Add(crossContextMenuItem);
        }
    }

    private void editLabelContextMenuItem_Click(object sender, EventArgs e)
    {

        if (String.IsNullOrEmpty(_highlightedCross.LabelName) || _highlightedCross.
LabelName == "xx")
            _labelDlg.Label = _currentTempLabel;
```

```
            else
                _labelDlg.Label = _highlightedCross.LabelName;

            if (_labelDlg.ShowDialog() == DialogResult.OK)
            {
                _currentTempLabel = _labelDlg.Label;
                Trace.WriteLine("Current Label: " + _currentTempLabel);
                _highlightedCross.LabelName = _currentTempLabel;
                if (_activeLeftImageBox)
                {
                    imageBoxExtLeft.Invalidate(RectangleOnImageBox(_highlightedCross.
    BoundingRectangle));
                    imageBoxExtLeft.Update();
                    _currentTempLabel = IncrementCurrentTempLabel(_currentTempLabel);
                }
                else
                {
                    imageBoxExtRight.Invalidate(RectangleOnImageBox(_highlightedCross.
    BoundingRectangle));
                    imageBoxExtRight.Update();
                    _currentTempLabel = IncrementCurrentTempLabel(_currentTempLabel);
                }
                _lastSaved = false;
                imageBoxExtLeft.Invalidate();
                imageBoxExtRight.Invalidate();
            }
        }

        private void deleteContextMenuItem_Click(object sender, EventArgs e)
        {
            if (_highlightedCross != null)
            {
                if (_activeLeftImageBox)
                {
                    Trace.WriteLine("Before delete: ");
                    Trace.WriteLine(_selectedPair.CrossesLeftPhoto.CrossPoints.Count.ToString
    ());

                    _selectedPair.CrossesLeftPhoto.Delete(_highlightedCross);
                    imageBoxExtLeft.Invalidate(RectangleOnImageBox(_highlightedCross.
    BoundingRectangle));
                    _highlightedCross = null;
                    imageBoxExtLeft.Update();

                    Trace.WriteLine("After delete: ");
                    Trace.WriteLine(_selectedPair.CrossesLeftPhoto.CrossPoints.Count.ToString
    ());

                }
                else
                {
                    Trace.WriteLine("Before delete: ");
                    Trace.WriteLine(_selectedPair.CrossesRightPhoto.CrossPoints.Count.ToString
    ());

                    _selectedPair.CrossesRightPhoto.Delete(_highlightedCross);
                    imageBoxExtRight.Invalidate(RectangleOnImageBox(_highlightedCross.
    BoundingRectangle));
                    _highlightedCross = null;
```

```
                imageBoxExtRight.Update();

                Trace.WriteLine("After delete: ");
                Trace.WriteLine(_selectedPair.CrossesRightPhoto.CrossPoints.Count.ToString
());
            }
            _lastSaved = false;
        }
    }

    private void deleteUnwantedCrossToolStripMenuItem_Click(object sender, EventArgs e)
    {
        DeleteUnwantedCross(true);
        DeleteUnwantedCross(false);
        _lastSaved = false;

    }

    private void extractCornerToolStripButton_Click(object sender, EventArgs e)
    {
        ExtractCorner(true);        //extrack corner for left image
        ExtractCorner(false);       //extract corner for right image
    }

    private void platePointToolStripButton_Click(object sender, EventArgs e)
    {

        using (OpenFileDialog dlg = new OpenFileDialog())
        {
            dlg.Title = "Select a Calibration Plate Coordinates File";
            dlg.Multiselect = false;
            dlg.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
            dlg.InitialDirectory = Environment.CurrentDirectory;

            dlg.RestoreDirectory = true;
            if (dlg.ShowDialog() == DialogResult.OK)
            {
                PlateCalibrationPoint plates = null;
                StreamReader sr = null;
                sr = new StreamReader(dlg.FileName);
                string file = sr.ReadToEnd();
                string[] files = file.Split(new char[] { '\n' });
                foreach (string line in files)
                {
                    string[] coords = line.Split(new char[] { ' ', '\t' });
                    PointF pt = new PointF(float.Parse(coords[1]), float.Parse(coords[2]))
;
                    plates = new PlateCalibrationPoint(pt, coords[0]);
                    _platePoints.PlatePoints.Add(plates);
                    string txt = String.Format("label: {0}, x: {1}, y: {2}", coords[0],
coords[1], coords[2]);
                    Trace.WriteLine(txt);
                }
                Trace.WriteLine(_platePoints.ToString());
                _lastSaved = false;
                _isPlatePointsAvailable = true;
            }
        }
    }
```

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



            if (!_isPlatePointsAvailable)
            {
                MessageBox.Show("Plate Coordinates are not available, load this file
before continouing calibration",
                    "Calibration Data Error");
            }
            this.backgroundWorker1.RunWorkerAsync();
            if (_photoPairList.Count > 0)
            {

                StereoCalib stereocalib = new StereoCalib(_platePoints, _leftPhoto.Bitmap.
Size, _photoPairList);

            }


            else
            {

                MessageBox.Show("The project must have photo pairs", "Stereo Calibration
Error");



            //MessageBox.Show(" Stereo Calibration Succes, \n chek in file >>> \n @C:-
Documents and Settings-Administrator-My Documents-Output Stereo Calibration-''StereoCalib.
txt''", "Calibration Camera");

        }
    }
    private void toolStripButton1_Click(object sender, EventArgs e)
    {

        deleteUnwantedCrossToolStripMenuItem_Click(sender, e);

    }

    #endregion Handlers

    #region Methods
    private void DeleteUnwantedCross(bool leftImage)
    {
        Trace.WriteLine("Before Delate left image: "+ _selectedPair.CrossesLeftPhoto.
CrossPoints.Count.ToString());
        Trace.WriteLine("Before Delate right image: " + _selectedPair.CrossesRightPhoto.
CrossPoints.Count.ToString());

        IEnumerable<Cross> unselectedcross = null;
        if (leftImage)
        {
            unselectedcross = from cross in _selectedPair.CrossesLeftPhoto.CrossPoints
                              where cross.LabelName != "xx"
```

```
                            select cross;
            }
            else
            {
                unselectedcross = from cross in _selectedPair.CrossesRightPhoto.CrossPoints
                            where cross.LabelName != "xx"
                            select cross;
            }

            List<Cross> t = unselectedcross.ToList<Cross>();
            if (leftImage)
            {
                _selectedPair.CrossesLeftPhoto.CrossPoints = t;
                imageBoxExtLeft.Invalidate();
                imageBoxExtLeft.Update();
            }
            else
            {
                _selectedPair.CrossesRightPhoto.CrossPoints = t;
                imageBoxExtRight.Invalidate();
                imageBoxExtRight.Update();
            }

            Trace.WriteLine("After Delate left image: " + _selectedPair.CrossesLeftPhoto.
CrossPoints.Count.ToString());
            Trace.WriteLine("After Delate right image: " + _selectedPair.CrossesRightPhoto.
CrossPoints.Count.ToString());
        }

        private void ExtractCorner(bool leftImage)
        {
            Image<Gray,Byte> temp = null;
            if (leftImage)  //true for left image
                temp = _leftPhoto.Convert<Gray, Byte>();
            else   //for right image
                temp = _rightPhoto.Convert<Gray, Byte>();

            PointF[][] corners = temp.GoodFeaturesToTrack(1000, 0.01, 1, 2, true, 0.04);

            //find sub-pixel accuracy
            temp.FindCornerSubPix(corners, new Size(10, 10), new Size(-1, -1), new
MCvTermCriteria(0.05));
            foreach (PointF[] p in corners)
            {
                foreach (PointF pp in p)
                {
                    Cross pt = new Cross(pp, "xx");
                    if (leftImage)
                        _selectedPair.CrossesLeftPhoto.CrossPoints.Add(pt);
                    else
                        _selectedPair.CrossesRightPhoto.CrossPoints.Add(pt);

                }
            }
            if (leftImage)
            {
                imageBoxExtLeft.Invalidate();
                imageBoxExtLeft.Update();
            }
```

```
            else
            {
                imageBoxExtRight.Invalidate();
                imageBoxExtRight.Update();
            }
            _lastSaved = false;
        }


        private void ReadCross(StreamReader sr,ref ImagePair p, bool leftPhoto)
        {
            //read the number of crosses
            int count = int.Parse(sr.ReadLine());
            for (int i = 0; i < count; i++)
            {
                string line = sr.ReadLine();
                if (leftPhoto)  //true
                {
                    if (line != null && line.Length != 0)
                    {
                        Trace.WriteLine(line);
                        string[] fields = line.Split(new char[] { ' ', '\t' });
                        Trace.WriteLine(fields[2].ToString());
                        PointF pt = new PointF(float.Parse(fields[1]), float.Parse(fields[2]))↙
;
                        Trace.WriteLine(fields[0] + " " + fields[1] + " " + fields[2]);
                        Cross cross = new Cross(pt, fields[0]);
                        p.CrossesLeftPhoto.CrossPoints.Add(cross);

                    }
                }
                else
                {
                    if (line != null && line.Length != 0)
                    {
                        Trace.WriteLine(line);
                        string[] fields = line.Split(new char[] { ' ', '\t' });
                        Trace.WriteLine(fields[2].ToString());
                        PointF pt = new PointF(float.Parse(fields[1]), float.Parse(fields[2]))↙
;
                        Trace.WriteLine(fields[0] + " " + fields[1] + " " + fields[2]);
                        Cross cross = new Cross(pt, fields[0]);
                        p.CrossesRightPhoto.CrossPoints.Add(cross);

                    }
                }
            }
        }

        private void ReadPhotoPairAndCrosses(int pair, StreamReader sr)
        {
            if (listBox1.Items.Count > 0)
                listBox1.Items.Clear();
            for (int i = 0; i < pair; i++)
            {
                ImagePair pairItem = new ImagePair();
                pairItem.PhotoPairName = sr.ReadLine();
                pairItem.LeftPhoto = sr.ReadLine();
                pairItem.RightPhoto = sr.ReadLine();
```

```
            ReadCross(sr, ref pairItem,true);   //true for left photo
            ReadCross(sr, ref pairItem, false);
            _photoPairList.Add(pairItem);
            listBox1.Items.Add(pairItem.PhotoPairName);
        }
    }

    private void ReadProject()
    {
        StreamReader sr = null;
        if (_photoPairList != null)
        {
            _photoPairList = null;
            _photoPairList = new List<ImagePair>();
        }
        try
        {
            sr = new StreamReader(_projectName);
            //read the number of pair
            int count = int.Parse(sr.ReadLine());
            ReadPhotoPairAndCrosses(count, sr);

        }
        catch (UnauthorizedAccessException uax)
        {
            throw new UnauthorizedAccessException("Unable to acces project" + _projectName↙
, uax);
        }
        catch (ArgumentException ax)
        {
            throw new ArgumentException("Unable to acces project doe to invalid argumen" +↙
 _projectName, ax);
        }
        catch (FileNotFoundException fnx)
        {
            throw new FileNotFoundException("Unable to read project" + _projectName, fnx);
        }
        catch (Exception e)
        {
            throw new Exception("Unable to read a file" + _projectName, e);
        }
        finally
        {
            if (sr != null)
                sr.Close();
        }
    }

    private bool SaveProject()
    {
        StreamWriter sw = null;
        try
        {
            sw = new StreamWriter(_projectName, false);
            //save number of pair in the project
            sw.WriteLine(_photoPairList.Count);
            foreach (ImagePair pair in _photoPairList)
            {
                sw.WriteLine(pair.PhotoPairName);
```

```
            sw.WriteLine(pair.LeftPhoto);
            sw.WriteLine(pair.RightPhoto);
            //save left-photo crosses
            SaveCrosses(sw, pair.CrossesLeftPhoto);
            //save right-photo crosses
            SaveCrosses(sw, pair.CrossesRightPhoto);
            //sw.WriteLine(filename);
            //SaveCrosses(sw, filename);
        }
    }
    catch (UnauthorizedAccessException uax)
    {
        throw new UnauthorizedAccessException("Unable to access album " + _projectName↙
ax);
    }
    catch (ArgumentException ax)
    {
        throw new ArgumentException("Unable to access album due to invalid argument" +↙
ojectName, ax);
    }
    catch (Exception e)
    {

        throw new Exception("Unable to create file" + _projectName, e);
    }
    finally
    {
        if (sw != null)
            sw.Close();
    }
    return true;
}

private void SaveCrosses(StreamWriter sw, Crosses crosses)
{
    //write the number of cross
    sw.WriteLine(crosses.CrossPoints.Count);
    //int lastIndex = crosses.CrossPoints.Count - 1;
    //int count = 0;
    foreach (Cross cross in crosses.CrossPoints)
    {
        String txt = String.Format("{0} {1} {2}", cross.LabelName, cross.Point2D.X,    ↙
ss.Point2D.Y);
        //if (lastIndex != crosses.CrossPoints.Count)
            sw.WriteLine(txt);
        //else
        //    sw.Write(txt);
    }
}

private String IncrementCurrentTempLabel(String currentTempLabel)
{
    int temp = int.Parse(currentTempLabel);
    String txt = String.Format("{0}", ++temp);
    return txt;
}

private PointF ImageBox2Image(Point pt)
{
```

```
        Point p = new Point();
        p.X = (int)((double)pt.X * hzScale);
        p.Y = (int)((double)pt.Y * vScale);

        return p;
    }

    //Convert Rectangle on Image to ImageBox
    private Rectangle RectangleOnImageBox(Rectangle r)
    {
        double hzScale = 0.0;
        double vScale =0.0;
        if (_activeLeftImageBox) //left image box
        {
            hzScale = _selectedPair.CrossesLeftPhoto.HzScaleImage2ImageBox;
            vScale = _selectedPair.CrossesLeftPhoto.VScaleImage2ImageBox;
        }
        else
        {
            hzScale = _selectedPair.CrossesRightPhoto.HzScaleImage2ImageBox;
            vScale = _selectedPair.CrossesRightPhoto.VScaleImage2ImageBox;
        }
        Rectangle temp = new Rectangle();
        temp.X = (int)((double)r.X * hzScale);
        temp.Y = (int)((double)r.Y * vScale);
        temp.Width = (int)((double)r.Width * hzScale);
        temp.Height = (int)((double)r.Height * vScale);

        return temp;
    }

    #endregion Methods

    private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
    {
        e.Result = this.LongRunningMethod(sender as BackgroundWorker, e);
    }

    private long LongRunningMethod(BackgroundWorker instance, DoWorkEventArgs e)
    {

        for (int i = 0; i < 100; i++)
        {
            if (instance.CancellationPending)
            {
                e.Cancel = true;
                break;
            }
            else
            {

                System.Threading.Thread.Sleep(100);
                 instance.ReportProgress(i);
            }


        }
        MessageBox.Show(" Stereo Calibration Success, \n chek in file >>> \n @C:-Documents↙
```

```
     and Settings-Administrator-My Documents-Output Stereo Calibration-''StereoCalib.txt''",  ↙
    "Calibration Camera");

            return 1L;
        }

        public void backgroundWorker1_ProgressChanged(object sender, ProgressChangedEventArgs ↙
    e)
        {
            this.progressBar1.Value = e.ProgressPercentage;
        }

        public void backgroundWorker1_RunWorkerCompleted(object sender,                       ↙
    RunWorkerCompletedEventArgs e)
        {
            this.CalibrationToolStripButton.Enabled = true;
            this.progressBar1.Value = 0;
        }


    }
}
```

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

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

namespace StereoCalibration
{
    public partial class ImageBoxExt : ImageBox
    {
        #region Fields
        String _photoName;
        #endregion

        #region Constructor
        public ImageBoxExt()
        {
            InitializeComponent();
        }
        #endregion

        #region Properties
        public String PhotoName
        {
            get { return _photoName; }
            set { _photoName = value; }
        }
        #endregion
    }
}
```

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

namespace StereoCalibration
{
    public class Image2ImageBoxScaleConversion
    {
        private double _hzScaleImage2ImageBox;
        private double _vScaleImage2ImageBox;
        private double _hzScaleImageBox2Image;
        private double _vScaleImageBox2Image;

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

        public double HorisontalScaleImage2ImageBox
        {
            get { return _hzScaleImage2ImageBox; }
        }

        public double VerticalScaleImage2ImageBox
        {
            get { return _vScaleImage2ImageBox; }
        }

        public double HorisontalScaleImageBox2Image
        {
            get { return _hzScaleImageBox2Image; }
        }

        public double VerticalScaleImageBox2Image
        {
            get { return _vScaleImageBox2Image; }
        }

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

    }
}
```

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

namespace StereoCalibration
{
    public partial class PhotoPair : Form
    {
        #region Fields
        string _pairName;
        string _left;
        string _right;
        //PlatePair m_platePair;
        List<ImagePair> _photoPair;
        //String _p = "";
        #endregion

        public PhotoPair()
        {
            InitializeComponent();
            _photoPair = new List<ImagePair>();
        }

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

        private void AddImageButton_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog dlg = new OpenFileDialog())
            {
                dlg.Title = "Add Image(s) to Project";
                dlg.Multiselect = true;
                dlg.Filter = "Image Files (JPEG, TIF, GIF, BMP, etc.)|"
                   + "*.jpg;*.jpeg;*.gif;*.bmp;"
                   + "*.tif;*.tiff;*.png|"
                   + "JPEG files (*.jpg;*.jpeg)|*.jpg;*.jpeg|"
                   + "GIF files (*.gif)|*.gif|"
                   + "BMP files (*.bmp)|*.bmp|"
                   + "TIFF files (*.tif;*.tiff)|*.tif;*.tiff|"
                   + "PNG files (*.png)|*.png|"
                   + "All files (*.*)|*.*";

                dlg.InitialDirectory = Environment.CurrentDirectory;
                dlg.RestoreDirectory = true;

                if (dlg.ShowDialog() == DialogResult.OK)
                {
                    string[] files = dlg.FileNames;
                    IEnumerable<string> sortAscendingQuery = from filename in files
orderby filename
select filename;
```

```
                    foreach (string s in sortAscendingQuery)
                    {
                        //_p = Path.GetFileNameWithoutExtension(s);
                        //_p = Path.GetFileName(s);
                        listBox1.Items.Add(s);
                    }
                }
            }
        }

        private void displayLeftImageToolStripMenuItem_Click(object sender, EventArgs e)
        {
            leftImageTextBox.Text = listBox1.SelectedItem.ToString();
            _left = listBox1.SelectedItem.ToString();
        }

        private void displayRightImageToolStripMenuItem_Click(object sender, EventArgs e)
        {
            rightImageTextBox.Text = listBox1.SelectedItem.ToString();
            _right = listBox1.SelectedItem.ToString();
        }

        private void btnAddPair_Click(object sender, EventArgs e)
        {
            _pairName = pairNameTextBox.Text;
            if (!String.IsNullOrEmpty(_pairName))
            {
                ImagePair pair = new ImagePair(_pairName, _left, _right);
                _photoPair.Add(pair);
                listBox2.Items.Add(pair.PhotoPairName);

                leftImageTextBox.Clear();
                rightImageTextBox.Clear();
            }
            else
                return;
        }
    }
}
```

C:\Users\pache\Desktop\Rapid Mapping Aplication\StereoCalibrationF\StereoCalibration\Cross.cs 1

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

namespace StereoCalibration
{
    public class Cross
    {
        #region Fields
        //centre point of the cross
        //ALL DONE IN IMAGE COORDINATES!!!!!
        private PointF _position;
        protected Rectangle _boundRect;
        private const int _radius = 3;
        //Label stuff
        private String _label;

        protected bool _highlighted;
        Color _highlightedColor;
        #endregion

        #region Constructor
        public Cross(PointF centre, String label)
        {
            _position = centre;
            _highlighted = false;
            _highlightedColor = Color.Green;
            //Label stuff
            _label = label;

            this._boundRect = new Rectangle(
            (int)(_position.X + 0.5f) - _radius,
            (int)(_position.Y + 0.5f) - _radius,
            (int)(10 * _radius),
            (int)(10 * _radius));
            if (_boundRect.Width < 2) _boundRect.Width = 2;
            if (_boundRect.Height < 2) _boundRect.Height = 2;
        }
        #endregion

        #region Properties

        public PointF Point2D
        {
            get { return _position; }
            set { _position = value; }
        }

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

```
            set
            {
                _boundRect = value;
            }
        }

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

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


        #endregion

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

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

            DrawCrossSign(g, x, y, pen);

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

        }

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

```
            g.DrawLine(pen, 0, 0, 0, _radius);
            g.ResetTransform();
        }

        private void DrawLabel(Graphics g, float x, float y, Pen pen, Font font, SolidBrush
    brush)
        {
            g.TranslateTransform(x, y);
            String temp = String.Empty;
            if (_label == "xx")
                g.DrawString(temp, font, brush, 0.0f, 0.0f);
            else
                g.DrawString(_label, font, brush, 0.0f, 0.0f);
            g.ResetTransform();
            //Trace.WriteLine("Draw Label");
        }

        //public override string ToString()
        //{
        //    String txt = String.Format("{0},{1},{2}", _label, _position.X, _position.Y);
        //        return txt;
        //}

        #endregion
    }
}
```

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

namespace StereoCalibration
{
    public class Crosses
    {
        #region Fields
        private Pen _pen;
        //protected Color _highlightedColor;


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

        private List<Cross> _crossPoints;

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

        #endregion

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

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

            _crossPoints = new List<Cross>();
        }
        #endregion

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



        public double HzScaleImage2ImageBox
        {
            get { return _hzScaleImage2ImageBox; }
        }

        public double VScaleImage2ImageBox
```

```
        {
            get { return _vScaleImage2ImageBox; }
        }

        public double HzScaleImageBox2Image
        {
            get { return _hzScaleImageBox2Image; }
        }

        public double VScaleImageBox2Image
        {
            get { return _vScaleImageBox2Image; }
        }
        #endregion

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

        }

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


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

            return null;
        }

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

            return element;
        }

         public void AddCross(Cross c)
         {
             _crossPoints.Add(c);
         }
```

```
        #endregion
    }
}
```

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

using Emgu.CV.Structure;

namespace StereoCalibration
{
    [Serializable]
    public class PlateCalibrationPoint
    {
        private PointF _point;
        private  String _label;

        public PlateCalibrationPoint()
        { }

        public PlateCalibrationPoint(PointF pointf, String label)
        {
            _point = pointf;



            _label = label;
        }

        public PointF Point2D
        {
            get { return _point; }
            set { _point = value; }
        }

        public String Label
        {
            get { return _label; }
            set { _label = value; }
        }
    }

    [Serializable]
    public class CalibrationPlate
    {
        private List<PlateCalibrationPoint> _platePoints;

        public CalibrationPlate()
        {
            _platePoints = new List<PlateCalibrationPoint>();
        }

        public List<PlateCalibrationPoint> PlatePoints
        {
            get { return _platePoints; }
            set { _platePoints = value; }
        }
    }
}
```

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

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

namespace StereoCalibration
{
    public class StereoCalib
    {
        #region Fields
        //input
        //private CalibrationPlate _plateCoord;
        private Size _imageSize;
        private int _pairNumber;
        MCvPoint3D32f[][] _objectPoints;
        PointF[][] _imagePointsLeft;
        PointF[][] _imagePointsRight;


        //output
        private IntrinsicCameraParameters _ioLeftPhoto;
        private IntrinsicCameraParameters _ioRightPhoto;
        private ExtrinsicCameraParameters _eoParams;

        private Matrix<double> _fundamentalMatrix;
        private Matrix<double> _essentialMatrix;

        //output to file *.txt

        public Double _fovXleft = 0.0;
        public Double _fovYleft = 0.0;
        public Double _focalLeft = 0.0;

        public Double _fovXright = 0.0;
        public Double _fovYright = 0.0;
        public Double _focalRight = 0.0;

        public Double _aspectratioleft = 0.0;
        public Double _aspectratioright = 0.0;

        public Double _K1left = 0.0;
        public Double _K2left = 0.0;
        public Double _P1left = 0.0;
        public Double _P2left = 0.0;
        public Double _K3left = 0.0;

        public Double _K1right = 0.0;
        public Double _K2right = 0.0;
        public Double _P1right = 0.0;
        public Double _P2right = 0.0;
```

```
        public Double _K3right = 0.0;

        public Double _EoLeftPhoto;
        public Double _EoRightPhoto;

        private Double BaseLine;
        #endregion Fields

        #region Constructors
        public StereoCalib(CalibrationPlate plateCoordinate, Size imgSize, List<ImagePair>
    pairList)
        {
            Trace.WriteLine("StereoCalib");
            //input
            _imageSize = imgSize;
            _pairNumber = pairList.Count;
            _objectPoints = new MCvPoint3D32f[_pairNumber][];
            _imagePointsLeft = new PointF[_pairNumber][];
            _imagePointsRight = new PointF[_pairNumber][];

            //output
            _ioLeftPhoto = new IntrinsicCameraParameters();
            _ioRightPhoto = new IntrinsicCameraParameters();
            _eoParams = new ExtrinsicCameraParameters();
            _fundamentalMatrix = null;
            _essentialMatrix = null;

            //Arrange inputs
            ArrangeInputsForAllPairs(plateCoordinate, pairList);
            //DisplayPoints();

            //Calibrate StereoPair
        }

        #endregion Constructors

        #region Properties
        public Double _baseline
        {
            get { return BaseLine; }
            set { BaseLine = value; }
        }
        public Double FovxLeft
        {
            get { return _fovXleft; }
            set { _fovXleft = value; }
        }
        public Double FovyLeft
        {
            get { return _fovYleft; }
            set { _fovYleft = value; }
        }
        public Double Focalleft
        {
            get { return _focalLeft; }
            set { _focalLeft = value; }
        }
        public Double AspectRatioLeft
        {
```

```
        get { return _aspectratioleft; }
        set { _aspectratioleft = value; }
    }
    public Double FovXright
    {
        get { return _fovXright; }
        set { _fovXright = value; }
    }
    public Double FovYright
    {
        get { return _fovYright; }
        set { _fovYright = value; }
    }
    public Double FocalRight
    {
        get { return _focalRight; }
        set { _focalRight = value; }
    }
    public Double AspectRatioRight
    {
        get { return _aspectratioright; }
        set { _aspectratioright = value; }
    }
    public Double K1_Left
    {
        get { return _K1left; }
        set { _K1left = value; }
    }
    public Double K2_Left
    {
        get { return _K2left; }
        set { _K2left = value; }
    }
    public Double P1_Left
    {
        get { return _P1left; }
        set { _P1left = value; }
    }
    public Double P2_Left
    {
        get { return _P2left; }
        set { _P2left = value; }
    }
    public Double K3_Left
    {
        get { return _K3left; }
        set { _K3left = value; }
    }

    public Double K1_Right
    {
        get { return _K1right; }
        set { _K1right = value; }
    }
    public Double K2_Right
    {
        get { return _K2right; }
        set { _K2right = value; }
    }
```

```
        public Double P1_Right
        {
            get { return _P1right; }
            set { _P1right = value; }
        }
        public Double P2_Right
        {
            get { return _P2right; }
            set { _P2right = value; }
        }
        public Double K3_Right
        {
            get { return _K2right; }
            set { _K2right = value; }
        }
        public Matrix<double> EssentialMatrix
        {
            get { return _essentialMatrix; }
            set { _essentialMatrix = value; }
        }

        #endregion Properties

        #region Methods
        private void ArrangeInputsForAllPairs(CalibrationPlate plateCoord, List<ImagePair> ↙
    pairList)
        {
            bool hasLeftHaveFewerPoints = true;
            int index = 0;
            foreach (ImagePair pair in pairList)
            {
                List<Cross> minCrossesNumber = new List<Cross>();
                List<Cross> otherCross = new List<Cross>();
                hasLeftHaveFewerPoints =
                    (pair.CrossesLeftPhoto.CrossPoints.Count <= pair.CrossesRightPhoto. ↙
    CrossPoints.Count) ? true : false;
                if (hasLeftHaveFewerPoints)  //left has fewer crosses
                {
                    minCrossesNumber = pair.CrossesLeftPhoto.CrossPoints;
                    otherCross = pair.CrossesRightPhoto.CrossPoints;
                }
                else
                {
                    minCrossesNumber = pair.CrossesRightPhoto.CrossPoints;
                    otherCross = pair.CrossesLeftPhoto.CrossPoints;
                }

                IEnumerable<Cross> sortedminCrossesNumber =
                                                from pt in minCrossesNumber
                                                orderby (int.Parse(pt.LabelName))
                                                select pt;

                //temporaryPoints
                List<MCvPoint3D32f> tempObjPts = new List<MCvPoint3D32f>();
                List<PointF> otherTempPts = new List<PointF>();
                List<PointF> minTempPts = new List<PointF>();

                Trace.WriteLine("Pair index" + index.ToString());
```

```
            foreach (Cross p in sortedminCrossesNumber)
            {
                //set min number of point list
                String label = p.LabelName;
                PointF minPt = new PointF(p.Point2D.X,p.Point2D.Y);
                minTempPts.Add(minPt);
                //Find point on the other photo
                foreach (Cross other in otherCross)
                {
                    if (label == other.LabelName)
                    {
                        PointF pt = new PointF(other.Point2D.X, other.Point2D.Y);
                        otherTempPts.Add(pt);
                        break;
                    }
                }
                //Find object point
                foreach (PlateCalibrationPoint pt in plateCoord.PlatePoints)
                {
                    if (label == pt.Label)
                    {
                        MCvPoint3D32f point = new MCvPoint3D32f(pt.Point2D.X, pt.Point2D.Y↙
, 0.0f);
                        tempObjPts.Add(point);
                        break;
                    }
                }
                //Trace.WriteLine(p.LabelName.ToString() + " , " + p.Point2D.X.ToString() ↙
+ " , " + p.Point2D.Y.ToString());
            }
            //Set object point
            _objectPoints[index] = tempObjPts.ToArray();

            //Set cross points on the left and right image
            if (hasLeftHaveFewerPoints)
            {
                _imagePointsLeft[index] = minTempPts.ToArray();
                _imagePointsRight[index] = otherTempPts.ToArray();
            }
            else
            {
                _imagePointsRight[index] = minTempPts.ToArray();
                _imagePointsLeft[index] = otherTempPts.ToArray();
            }

            Trace.WriteLine("object left right point count: " + tempObjPts.Count.ToString ↙
() + " , " +
                minTempPts.Count.ToString() + " , " + otherTempPts.Count.ToString());

            index++;
        }
        CalibrateStereoPair();
    }

    private void CalibrateStereoPair()
    {

        //Calibrate left Camera
        ExtrinsicCameraParameters[] eoLeftPhoto;
```

```
        CameraCalibration.CalibrateCamera(_objectPoints, _imagePointsLeft,
                                          _imageSize,
                                          _ioLeftPhoto,
                                          Emgu.CV.CvEnum.CALIB_TYPE.DEFAULT,
                                          out eoLeftPhoto);
        double fovx = 0.0;
        double fovy = 0.0;
        double focal = 0.0;
        MCvPoint2D64f pp = new MCvPoint2D64f();
        double aspect = 0.0;

        //CvInvoke.cvCalibrationMatrixValues(_ioLeftPhoto.IntrinsicMatrix, _imageSize.
Width, _imageSize.Height,
        //    4.9, 3.7, ref fovx, ref fovy, ref focal, ref pp, ref aspect);   //CCTV
        CvInvoke.cvCalibrationMatrixValues(_ioLeftPhoto.IntrinsicMatrix, _imageSize.Width,
 _imageSize.Height,
          23.6, 15.8, ref fovx, ref fovy, ref focal, ref pp, ref aspect);   //DSLR D60

        TextWriter _tw = new StreamWriter(@"C:\Documents and Settings\Administrator\My
Documents\Output Stereo Calibration\StereoCalib.txt");
        _tw.WriteLine("EO LEFT photo stereo calibration ");
        _tw.WriteLine("\t fovx: " + "\t" + fovx.ToString());
        _tw.WriteLine("\t fovy: " + "\t" + fovy.ToString());
        _tw.WriteLine("\t focal leght: " + "\t" + focal.ToString());
        _tw.WriteLine("\t aspect ratio: " + "\t" + aspect.ToString());
        _tw.WriteLine("\t principal point x,y: " + "\t" + pp.x.ToString() + " , " + pp.y.
ToString());
        _tw.WriteLine("");

        Trace.WriteLine("EO LEFT camera.....................................................");
        Trace.WriteLine("fovx = " + fovx.ToString());
        Trace.WriteLine("fovy = " + fovy.ToString());
        Trace.WriteLine("focal = " + focal.ToString());
        Trace.WriteLine("aspect ratio = " + aspect.ToString());
        Trace.WriteLine("principal point x,y: " + pp.x.ToString() + " , " + pp.y.ToString
());
        Trace.WriteLine("IO left photo before stereo calibration: ");
        DisplayIOParams(_ioLeftPhoto);

        //Calibrate right Camera
        ExtrinsicCameraParameters[] eoRightPhoto;
        CameraCalibration.CalibrateCamera(_objectPoints, _imagePointsRight,
                                          _imageSize,
                                          _ioRightPhoto,
                                          Emgu.CV.CvEnum.CALIB_TYPE.DEFAULT,
                                          out eoRightPhoto);

        //CvInvoke.cvCalibrationMatrixValues(_ioRightPhoto.IntrinsicMatrix, _imageSize.
Width, _imageSize.Height,
        //   4.9, 3.7, ref fovx, ref fovy, ref focal, ref pp, ref aspect);   //CCTV
        CvInvoke.cvCalibrationMatrixValues(_ioLeftPhoto.IntrinsicMatrix, _imageSize.Width,
 _imageSize.Height,
         23.6, 15.8, ref fovx, ref fovy, ref focal, ref pp, ref aspect);   //DSLR D60

        _tw.WriteLine("EO RIGHT photo stereo calibration ");
        _tw.WriteLine("\t fovx: " + "\t" + fovx.ToString());
        _tw.WriteLine("\t fovy: " + "\t" + fovy.ToString());
        _tw.WriteLine("\t focal leght: " + "\t" + focal.ToString());
        _tw.WriteLine("\t aspect ratio: " + "\t" + aspect.ToString());
```

```
        _tw.WriteLine("\t principal point x,y: " + "\t" + pp.x.ToString() + " , " + pp.y.ToString());
        _tw.WriteLine("");

        Trace.WriteLine("EO Right stereo calibration: ----------------------------------: ");
        Trace.WriteLine("fovx = " + fovx.ToString());
        Trace.WriteLine("fovy = " + fovy.ToString());
        Trace.WriteLine("focal = " + focal.ToString());
        Trace.WriteLine("aspect ratio = " + aspect.ToString());
        Trace.WriteLine("principal point x,y: " + pp.x.ToString() + " , " + pp.y.ToString());
        Trace.WriteLine("IO Right photo    before stereo calibration: ");
        DisplayIOParams(_ioRightPhoto);

        //Calibrate a stereopair by using their intrinsic params
        CameraCalibration.StereoCalibrate(_objectPoints,
                                          _imagePointsLeft,
                                          _imagePointsRight,
                                          _ioLeftPhoto,
                                          _ioRightPhoto,
                                          _imageSize,
                                          Emgu.CV.CvEnum.CALIB_TYPE.CV_CALIB_USE_INTRINSIC_GUESS,
                                          new MCvTermCriteria(0.0001),
                                          out _eoParams,
                                          out _fundamentalMatrix,
                                          out _essentialMatrix);
        Trace.WriteLine("Translasi " + "\t" + _eoParams.TranslationVector[0, 0].ToString() + " , " +
                        _eoParams.TranslationVector[1, 0].ToString() + " , " +
                        _eoParams.TranslationVector[2, 0].ToString());


        _K1left = _ioLeftPhoto.DistortionCoeffs[0, 0];
        _K2left = _ioLeftPhoto.DistortionCoeffs[1, 0];
        _P1left = _ioLeftPhoto.DistortionCoeffs[2, 0];
        _P2left = _ioLeftPhoto.DistortionCoeffs[3, 0];
        _K3left = _ioLeftPhoto.DistortionCoeffs[4, 0];

        _K1right = _ioRightPhoto.DistortionCoeffs[0, 0];
        _K2right = _ioRightPhoto.DistortionCoeffs[1, 0];
        _P1right = _ioRightPhoto.DistortionCoeffs[2, 0];
        _P2right = _ioRightPhoto.DistortionCoeffs[3, 0];
        _K3right = _ioRightPhoto.DistortionCoeffs[4, 0];

        Trace.WriteLine("IO left photo after stereo calibration: ");
        DisplayIOParams(_ioLeftPhoto);
        Trace.WriteLine("IO Right photo after stereo calibration: ");
        DisplayIOParams(_ioRightPhoto);

        double BaseLine = Math.Sqrt(Math.Pow(_eoParams.TranslationVector[0, 0], 2) +
                                    Math.Pow(_eoParams.TranslationVector[1, 0], 2) +
                                    Math.Pow(_eoParams.TranslationVector[2, 0], 2));

        Trace.WriteLine("BaseLine: " + BaseLine.ToString());

        _tw.WriteLine("IO LEFT photo stereo calibration ");
        _tw.WriteLine("\t K1 :" + "\t" + this._K1left.ToString());
```

```
            _tw.WriteLine("\t K2 :" + "\t" + this._K2left.ToString());
            _tw.WriteLine("\t K3 :" + "\t" + this._K3left.ToString());
            _tw.WriteLine("\t P1 :" + "\t" + this._P2left.ToString());
            _tw.WriteLine("\t P2 :" + "\t" + this._P2left.ToString());
            _tw.WriteLine("");
            _tw.WriteLine("Intrinsic Matrix:");
            _tw.WriteLine(String.Format("\t {0} {1} {2}", _ioLeftPhoto.IntrinsicMatrix[0, 0], ↙
    _ioLeftPhoto.IntrinsicMatrix[0, 1],
                                                    _ioLeftPhoto.IntrinsicMatrix[0, 2]));
            _tw.WriteLine(String.Format("\t {0} {1} {2}", _ioLeftPhoto.IntrinsicMatrix[1, 0], ↙
    _ioLeftPhoto.IntrinsicMatrix[1, 1],
                                                    _ioLeftPhoto.IntrinsicMatrix[1, 2]));
            _tw.WriteLine(String.Format("\t {0} {1} {2}", _ioLeftPhoto.IntrinsicMatrix[2, 0], ↙
    _ioLeftPhoto.IntrinsicMatrix[2, 1],
                                                    _ioLeftPhoto.IntrinsicMatrix[2, 2]));

            _tw.WriteLine("");
            _tw.WriteLine("IO RIGHT photo stereo calibration ");
            _tw.WriteLine("\t K1 :" + "\t" + this._K1right.ToString());
            _tw.WriteLine("\t K2 :" + "\t" + this._K2right.ToString());
            _tw.WriteLine("\t K3 :" + "\t" + this._K3right.ToString());
            _tw.WriteLine("\t P1 :" + "\t" + this._P1right.ToString());
            _tw.WriteLine("\t P2 :" + "\t" + this._P1right.ToString());
            _tw.WriteLine("");
            _tw.WriteLine("Intrinsic Matrix:");
            _tw.WriteLine(String.Format("\t {0} {1} {2}", _ioRightPhoto.IntrinsicMatrix[0, 0],↙
     _ioRightPhoto.IntrinsicMatrix[0, 1],
                                                    _ioRightPhoto.IntrinsicMatrix[0, 2]));
            _tw.WriteLine(String.Format("\t {0} {1} {2}", _ioRightPhoto.IntrinsicMatrix[1, 0],↙
     _ioRightPhoto.IntrinsicMatrix[1, 1],
                                                    _ioRightPhoto.IntrinsicMatrix[1, 2]));
            _tw.WriteLine(String.Format("\t {0} {1} {2}", _ioRightPhoto.IntrinsicMatrix[2, 0],↙
     _ioRightPhoto.IntrinsicMatrix[2, 1],
                                                    _ioRightPhoto.IntrinsicMatrix[2, 2]));
            _tw.WriteLine("");

            _tw.WriteLine("Translasi: " + "\t" + _eoParams.TranslationVector[0, 0].ToString() ↙
    + " , " +
                          _eoParams.TranslationVector[1, 0].ToString() + " , " +
                          _eoParams.TranslationVector[2, 0].ToString());

            _tw.WriteLine("");
            _tw.WriteLine("Base Line: " + "\t" + BaseLine.ToString());
            _tw.Close();

        }

        #endregion Methods

        #region Hendlers
        private void DisplayPoints(MCvPoint3D32f[] p)
        {

        }
        private void DisplayIOParams(IntrinsicCameraParameters io)
        {
            Trace.WriteLine("Intrinsic Matrix:");
            String line1 = String.Format("{0} {1} {2}", io.IntrinsicMatrix[0, 0], io.        ↙
    IntrinsicMatrix[0, 1],
```

```
                                                    io.IntrinsicMatrix[0, 2]);
            String line2 = String.Format("{0} {1} {2}", io.IntrinsicMatrix[1, 0], io.
    IntrinsicMatrix[1, 1],
                                                    io.IntrinsicMatrix[1, 2]);
            String line3 = String.Format("{0} {1} {2}", io.IntrinsicMatrix[2, 0], io.
    IntrinsicMatrix[2, 1],
                                                    io.IntrinsicMatrix[2, 2]);

            Trace.WriteLine(line1);
            Trace.WriteLine(line2);
            Trace.WriteLine(line3);

            Trace.WriteLine("k1 " + io.DistortionCoeffs[0, 0].ToString());
            Trace.WriteLine("k2 " + io.DistortionCoeffs[1, 0].ToString());
            Trace.WriteLine("p1 " + io.DistortionCoeffs[2, 0].ToString());
            Trace.WriteLine("p2 " + io.DistortionCoeffs[3, 0].ToString());
            Trace.WriteLine("k3 " + io.DistortionCoeffs[4, 0].ToString());
        }

        #endregion Hendlers

    }
}
```