EPS
Escola Politècnica Superior
UdG

# Projecte/Treball Fi de Carrera

**Estudi:** Eng. Tècn. Informàtica de Sistemes. Pla 2001

**Títol:** Segmentació de punts 3D rígids i no rígids obtinguts a partir d'un sistema estèreo

**Document:** Memòria

**Alumne:** Eduard Logroño Fontquerna

**Director/Tutor:** Xavier Lladó Bardera
**Departament:** Arquitectura i Tecnologia de Computadors
**Àrea:** Arquitectura i Tecnologia de Computadors

**Convocatòria** (mes/any): 09/2009

# Índex

# Capítol 1

# Introducció

## 1.1 Descripció general

Aquest projecte s'emmarca dins l'àmbit de la recerca de la Visió per Computador del departament d'Arquitectura i Tecnologia de Computadors de la Universitat de Girona, i més concretament, dins la temàtica de reconstrucció d'informació tridimensional mitjançant sistemes de visió.

En els últims anys, i degut a l'evolució de la informàtica i a l'ús d'informació tridimensional, la reconstrucció 3D i l'estudi de models deformables ha adquirit un gran protagonisme sobretot en camps com l'animació 3D, videojocs i en estudis mèdics sobre diferents parts del cos humà.

L'objectiu principal d'aquest PFC és el d'estudiar i implementar diferents tècniques que permetin, donada una evolució temporal d'un objecte/estructura 3D, segmentar els punts 3D rígids dels que són deformables tal i com es mostra a la figura 1.1 on podríem diferenciar per exemple que els punts rígids són els punts del nas i el front i els que estan propers a la boca són no rígids, d'aquesta manera podrem separar els punts rígids dels punts no rígids mitjançant mètodes que analitzarem, dissenyarem i implementarem. El motiu d'aquest estudi de separar punts rígids de punts no rígids es bàsicament per poder diferenciar i poder fer un seguiment dels punts deformables i/o poder-ne fer una reconstrucció 3D amb models deformables a partir de la identificació de punts rígids.

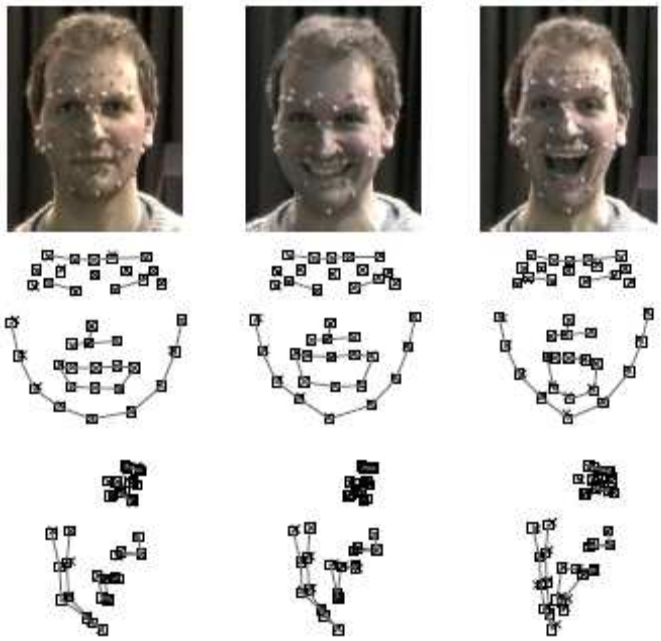


Figura 1.1 Exemple de reconstrucció 3D amb punts rígids i no rígids d'una cara [3]

Per avaluar aquests mètodes de segmentació de punts 3D, utilitzarem la informació tridimensional capturada a partir d'un sistema de visió estereoscòpica. Un sistema de visió estereoscòpic està format per dues càmeres les quals ens realitzaran captures reals des de dos punts de vistes diferents, on hi haurà sol·lapament de les dos imatges, per poder-ne extreure informació tridimensional amb la qual treballarem amb els algorismes que haurem analitzat, dissenyat i implementat. Això implicarà realitzar: 1) el desenvolupament i la calibració precisa d'un sistema de visió estèreo real; 2) la utilització del software necessari per realitzar la detecció i seguiment del punts d'interès en les imatges 2D capturades pel sistema, i per últim, 3) l'obtenció de les reconstruccions 3D a partir de la triangulació estèreo dels punts 2D. Les seqüències de reconstruccions 3D resultants, s'utilitzaran com a dades d'entrada per a la segmentació de punts rígids/deformables i la seva avaluació. On nosaltres ens centrarem específicament.

Per a assolir aquests objectius es treballarà bàsicament amb la plataforma de desenvolupament Matlab. Un entorn molt potent, eficient i molt utilitzat dins el món de la recerca.

Resumint, aquest projecte el dividirem en dos grans parts que serien la part de dissenyar, desenvolupar i implementar els mètodes de segmentació que ens serviran per separar els punts rígids dels punts no rígids/deformables. I l'altra

part seria la d'obtenir reconstruccions 3D a partir d'un sistema estèreo, passant per a la calibració de les càmeres del sistema, la realització de captures d'experiments reals, la generació de reconstruccions 3D per finalment posar a prova els mètodes desenvolupats en la part anterior. A l'esquema de la figura 1.2 s'explica gràficament el que realitzarem al llarg d'aquest PFC.

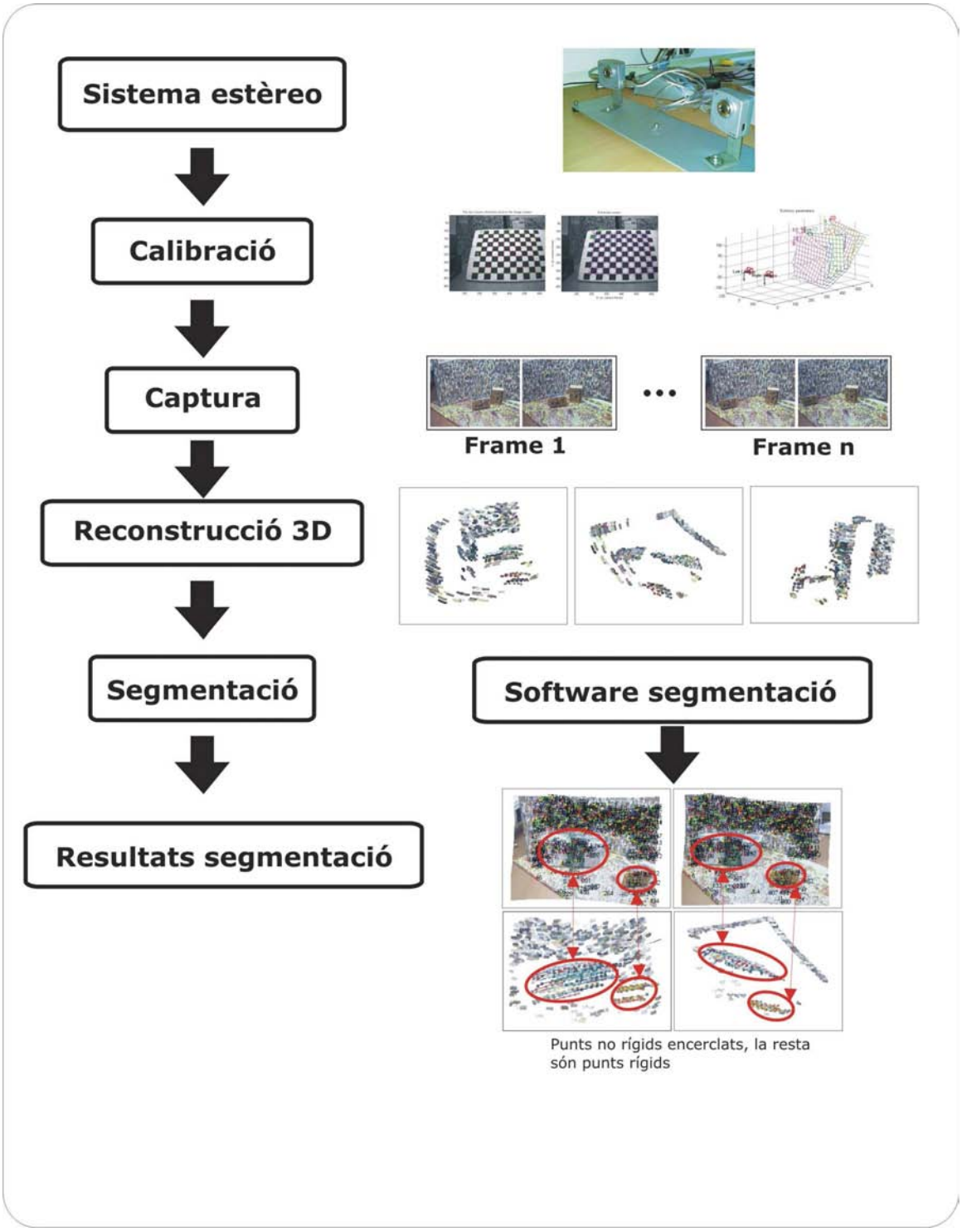



Figura 1.2. Esquema general del projecte

## 1.2 Objectius

Aquest projecte com s'ha dit anteriorment busca aprofundir en l'estudi de la reconstrucció 3D basat en objectes deformables i més específicament en la segmentació, per poder separar els punts 3D dels objectes que són rígids dels que són no rígids/deformables.

La següent llista d'objectius són els que es volen assolir al llarg d'aquest projecte:

- Introduir-nos en el món de la reconstrucció 3D.
- Aprendre a realitzar una bona calibració d'un sistema de visió estereoscòpic de manera precisa i efectiva amb la utilització de Toolbox específiques.
- Anàlisis i reducció dels errors de reprojecció en el mètode de calibració.
- Aplicar el mètode de la triangulació sobre dades reals per a obtenir la reconstrucció 3D a partir de seqüències d'imatges en 2D.
- Anàlisis, disseny i implementació dels mètodes de segmentació estudiats sobre dades reals i anàlisis de resultats de tals segmentacions.
- Anàlisis de resultats amb dades sintètiques. Gràcies a aquestes dades sintètiques ens serà més fàcil observar, provar i comparar diferents mètodes de segmentació de punts rígids i no rígids, degut a que tindrem el complet control de cadascun dels punts dels objectes sintètics creats.
- Utilització i aprenentatge el software STracker (Software dins del grup de visió per computador de la Universitat de Girona) per al "tracking" que és el procés de seguiment de punts característics per a les imatges 2D que ens generen cada una de les càmeres durant una seqüència.

## 1.3 Abast i sortides del projecte

Com a resultat del PFC es pretén desenvolupar una llibreria de matlab, així com generar tot un conjunt de dades tant sintètiques com reals (capturades pel sistema de visió estèreo) que puguin ser útils per a possibles futurs estudis dins d'aquest àmbit de la reconstrucció 3D. Per tant implicarà:

- Desenvolupament d'un sistema estèreo el qual serà una plataforma amb dues càmeres FireWire, les quals seran sotmeses a un rigorós procés de calibració.
- Desenvolupar un manual d'usuari per a realitzar la calibració de càmeres per al sistema estèreo el qual utilitzarem.
- Captació d'una seqüència real amb objectes que estaran sotmesos a una deformació, on finalment treballarem amb un conjunt d'imatges seleccionades de la captura realitzada.
- Implementació final dels mètodes desenvolupats en la primera part del projecte sobre les captures reals.
- Desenvolupament d'una completa Toolbox de segmentació que serà útil per a futures recerques sobre l'estudi de model deformables.

## 1.4. Planificació

La planificació d'aquest projecte es pot dividir en les següents etapes:

- **Especificacions del tema a tractar i aclariments sobre diferents temes relacionats amb el projecte.**

Primera fase del projecte en la qual ens proposarem els objectius a assolir i els temes que es tractaran al llarg d'aquest per tal d'estudiar i assolir els conceptes i coneixements necessaris.

- **Recerca d'informació i documentació dels temes a tractar.**

Ens documentarem de la teoria bàsica sobre el temari de la reconstrucció 3D i les seves bases teòriques a partir de documentació online i llibres.

- **Instal·lació del hardware i software necessari per el projecte.**

Començarem a entrar en matèria començant a instal·lar les càmeres FireWire junt amb l'aplicació amb la que realitzarem les captures de les seqüències d'imatges en format vídeo.

- **Muntatge del sistema estèreo fix**

Construirem una estructura per a que les càmeres es mantinguin completament fixes per a què així es trobin a una distància constant entre elles ja que una vegada calibrades, ajustats els paràmetres interns i externs de les càmeres, es mantinguin immòbils i així no hàgim de tornar a fer una calibració degut a algun moviment entre elles que ha de ser nul.

- **Creació de dades sintètiques per a l'estudi i la pràctica**

Com que treballar amb dades reals el procés de captura és complex i llarg, doncs per provar el funcionament i depurar diferents mètodes de segmentació ens serà més fàcil treballar amb dades sintètiques per poder comprovar el funcionament d'aquests mètodes i saber-ne el seu resultat per a la futura implementació real.

- **Calibració del sistema estèreo.**

Procés per a l'extracció dels paràmetres necessaris per a la calibració precisa del sistema estèreo, començant calibrant individualment càmera a càmera per acabar realitzant la calibració estèreo.

- **Estudi i reducció dels errors de reprojecció del procés de calibració.**

El procés de calibració mai és exempt d'errors, per a això haurem d'estudiar els errors amb els que ens trobem i intentar minimitzar-los per a millorar-ne la calibració que es fonamental per als passos que realitzarem a continuació.

- **Instal·lació, estudi i utilització del software per al procés de Tracking**

Per al seguiment de punts característics d'una seqüència de frames, utilitzarem el software que ens ha cedit en Jordi Ferrer Plana per a realitzar la tasca de *matching* i *tracking* per a una seqüència. El matching i el tracking seran el seguiment dels punts dels objectes al llarg d'una seqüència real on després en podrem fer la reconstrucció 3D per a cada instant de temps.

- **Captura de seqüències de frames (en vídeo) per a l'estudi dels moviments i/o deformacions d'objectes.**

Amb les càmeres FireWire gravarem seqüències en format vídeo per a després posar a prova els diferents algorismes i funcions que dissenyarem.

- **Estudiar i comprovar diferents mètodes de segmentació de punts rígids a no rígids sobre una seqüència real.**

En aquest cas analitzarem i provarem diferents mètodes per a la segmentació de punts rígids i deformables per extreure'n conclusions i fer diferents comparatives sobre les diferents segmentacions, eficiència i els resultats finals de dites segmentacions.

- **Anàlisis de resultats i extracció de conclusions.**

Després de posar a provar els diferents mètodes sobre experiments amb dades sintètiques i experiments amb dades reals en comprovarem els resultats i n'extraurem conclusions sobre tals i altres temes que hagin sortit al llarg d'aquests dos tipus d'experiments.

- **Redacció de la memòria.**

## 1.5 Estructura del document

Aquest document s'estructurarà en una sèrie de de capítols que a continuació resumim el que veurem al llarg d'aquests capítols:

- **Capítol 2: Bases teòriques:**

En aquest capítol ens centrarem a presentar els conceptes bàsics de la visió per computador que apareixeran al llarg d'aquest projecte, on la reconstrucció 3D serà el camp que serà més detallat. Parlarem sobre la part 2D de com es forma la imatge, explicarem els dos tipus de projeccions bàsics, i tot el temari que tocarem en quan a reconstrucció 3D.

- **Capítol 3: Anàlisis, disseny i implementació dels mètodes de segmentació.**

En el capítol 3 ens centrarem bàsicament en el tema de la segmentació, què és, com funciona i algunes diferents utilitats en el món de la visió per computador. Dissenyarem, desenvoluparem i implementarem els diferents mètodes de segmentació per a treballar sobre les dades que a nosaltres ens interessaran per

finalment implementar-los sobre dades sintètiques que crearem, per tal de veure'n el correcte funcionament de tals.

- **Capítol 4. Desenvolupament del sistema estèreo.**

Durant aquest capítol ens centrarem a desenvolupar el nostre sistema estèreo en el què utilitzarem dues càmeres FireWire, realitzarem tot el procés de calibració per a obtenir una rigorosa calibració, extraurem seqüències reals sobre objectes en moviment i deformables per a utilitzar el software per a fer el procés de seguiment de punts al llarg de la seqüència (tracking). El resultats que ens retornarà el software l'adaptarem per a poder finalment implementar els mètodes que hem desenvolupat en el capítol anterior.

- **Capítol 5. Conclusions**

Aquí valorarem l'abast d'aquest projecte, s'hi hem assolit els seus objectius i n'extraurem conclusions dels resultats als quals hem arribat al llarg dels anteriors capítols.

- **Annex:**

A l'annex hi trobarem tota la informació referent a la toolbox que hem desenvolupat per a poder treballar amb totes les dades, tant amb dades sintètiques com amb dades reals. Es detallaran l'ús i aplicació de cada una de les funcions que s'han desenvolupat al llarg del projecte.

# Capítol 2

# Bases teòriques

En aquest capítol bàsicament s'explicaran els fonaments i coneixements necessaris sobre la visió per computador fins a arribar el tema de la reconstrucció 3D, fent una pinzellada sobre les bases teòriques de tots aquests temes. Com que aquest projecte es troba dins l'àmbit de la visió per computador, començarem explicant que és i quines aplicacions té la visió per computador.

## 2.1 Visió per computador

La visió per computador, té el propòsit bàsic de desenvolupar un conjunt d'algorismes o programes informàtics per entendre, descriure o extreure característiques d'una escena a partir d'imatges.

La visió per computador estudia com poder aplicar la visió humana al camp de la informàtica. Per a fer això, un sistema de visió disposa de com a mínim una càmera que realitza la funció d'ull per a capturar escenes reals (imatges), llavors un algorismes o programa que fa que el computador tracti aquestes imatges de la manera desitjada per extreure'n tot tipus d'informació i/o característiques.

En els últims anys la visió per computador ha agafat molt de protagonisme en el camp industrial, mèdic i en la recerca per a trobar algorismes eficients i útils per a aplicacions com poden ser les següents:

- **Control de qualitat**: El control de qualitat és una de les aplicacions més utilitzades de la visió per computador. Bàsicament la tasca que realitza aquí un sistema de visió per computador seria per exemple el de controlar en una cadena de fabricació de peces que tals peces es mantinguin dintre una tolerància en les seves mides, no tinguin deformacions, siguin d'un color... Resumint, el sistema de visió controlaria amb una alta precisió i velocitat la qualitat de les peces.
- **Robòtica**: La visió per computador en el món de la robòtica bàsicament serveix per a donar-li visió a un robot, per a seguir trajectòries, evitar obstacles, és a dir, ajudar-lo a navegar.

- **Medicina**: El camp de la medicina és on la visió per computador els últims anys hi està agafant molt de protagonisme i on s'hi esta fent més recerca. En el camp de la medicina la visió per computador pot ajudar a tractar imatges generades per raigs X, ultrasons, ressonàncies magnètiques. Últimament també s'està estudiant molt en el tema de la reconstrucció 3D per a l'estudi de diferents parts del cos per a obtenir-ne informació tridimensional. Per exemple reconstrucció 3D del cervell, de la pròstata, etc..
- **Ús militar**: No podia faltar el camp militar, on en aquest camp també s'havien fet molts invents i descobriments, com els avions, els radars, internet. En aquest camp la visió per computador té diferents aplicacions, com estudiar terrenys, seguiments d'objectes, utilització en armes intel·ligents com podrien ser míssils, entre d'altres.
- **Reconeixement d'objectes**: El reconeixement d'objectes és un factor el qual la visió per computador és molt útil, un exemple fàcil i molt usat podria ser el reconeixement de caràcters a partir d'imatges, cercadors a través del contingut en les imatges...
- **Entreteniment**: En els últims anys la creació de videojocs en 3D s'ha basat bàsicament en la captura real d'escenes en 3D amb la utilització de vàries càmeres per obtenir informació tridimensional per utilitzar-ne les dades processades per a crear-ne animacions, pel·lícules, videojocs..
- **Altres**: No acaben aquí les aplicacions de la visió per computador, altres aplicacions podrien ser: Radars per al control de la velocitat, identificació com per exemple llegint les empremtes dactilars o bé pels ulls, etc..

## 2.2 Procés de creació d'una imatge

La geometria és una branca de les matemàtiques que s'ocupa de les propietats de l'espai, com són: punts, rectes, plans, polígons, políedres, corbes, superfícies, etc... Els seus orígens es remunten a la solució de problemes concrets relatius a mesures i és la justificació teòrica de molts instruments, on també hi apareix el tema de la reconstrucció 3D.

La reconstrucció 3D a partir de tècniques de visió per computador es basa en la utilització de diferents imatges en 2D, per això, començarem explicant la formació de les imatges 2D i de les bases geomètriques d'aquestes imatges abans d'atacar el tema de la reconstrucció 3D.

### 2.2.1 Projecció al pla imatge

La projecció d'un objecte és la manera que s'obté al dirigir totes les línies projectants de l'objecte cap a un pla, en el nostre cas ens interessa que es projecti sobre el pla imatge.

Els elements bàsics de la projecció són, el punt de vista o focus de projecció, el punt on es desitja projectar, la línia projectant i el pla sobre el que es projecte, anomenat pla imatge. Com podem veure a la figura 2.1 hi ha diferents tipus de projecció, on les línies projectants són paral·leles (projecció ortogràfica o ortogonal), o bé, quan les línies projectants intersecten totes en un punt fora del pla imatge (projecció perspectiva).

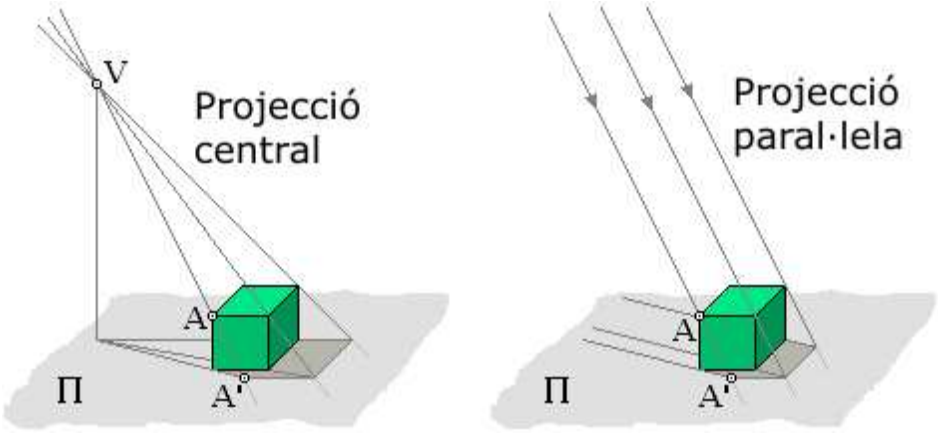



Figura 2.1. Imatges on es veu la diferència en les dos diferents projeccions, la projecció central (projecció perspectiva) i la projecció paral·lela (projecció ortogràfica).

En el pla imatge, doncs, hi trobem la imatge que no és més que un conjunt de píxels que omplen aquest pla. Els píxels contenen informació sobre la intensitat de la llum que s'ha projectat sobre el pla imatge. La informació d'aquests píxels s'han generat gràcies als sensors CCD, sensors sensibles a la llum, que disposa la càmera.  A la figura 2.2 podem veure una representació esquemàtica del que seria un punt 3D sobre el pla imatge, tal i com s'ha explicat.

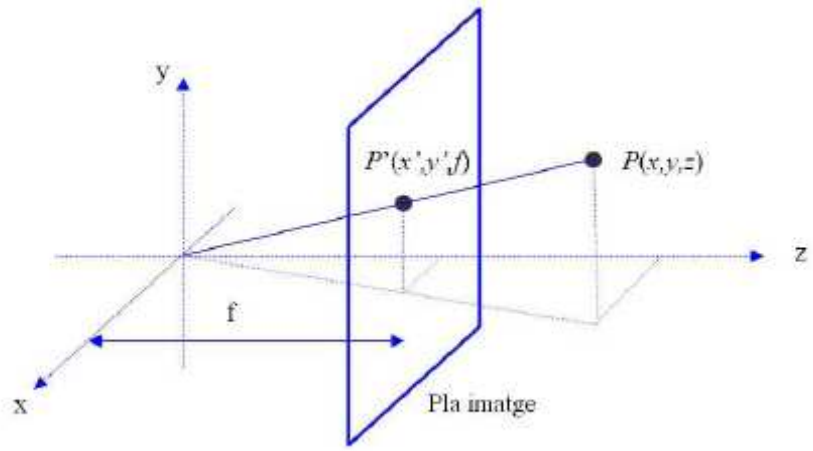



Figura 2.2. Esquema explicatiu de la projecció en el pla imatge d'un punt P(x,y,z) que es troba reflectit en el pla imatge P'(x',y',f).

Per a la generació d'aquestes imatges és necessari una càmera, on el model més simple per a l'adquisició d'aquestes es basa en el model de càmera pinhole que s'explica a continuació.

### 2.2.2 Càmera pinhole

El model de la càmera pinhole es tracta d'una càmera sense objectiu on la intensitat de la llum produeix una imatge que passaria a través d'un petit forat tal i com es mostra a la figura 2.3. En aquest forat hi passarien tots els raigs de llum, fa que aquests quedin invertits a la part posterior, després de passar aquest forat, llavors seria possible captar-los amb sensors fotosensibles, com els sensors CCD.

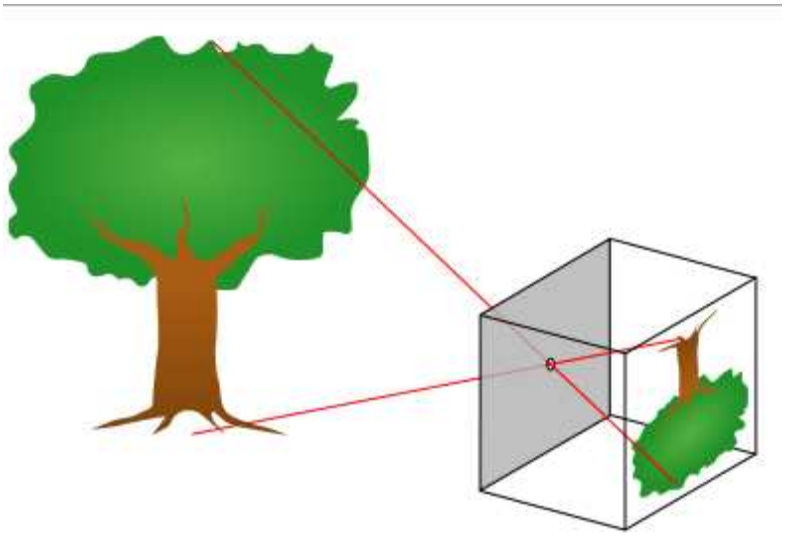


Figura 2.3. Exemple esquemàtic del model de càmera pinhole [4]

El model matemàtic de la càmera pinhole consisteix en un centre òptic O, on hi convergeixen totes les rectes de projecció, i un pla imatge P on la imatge és projectada. El pla imatge està situat a una distància focal f del centre òptic que és perpendicular a l'eix òptic.

El punt 3D real es projecta sobre el pla imatge, on passa a ser un punt 2D, el qual es defineix com la intersecció de la recta que prové del punt real amb el pla imatge.

A continuació s'explicarà dos tipus de transformacions que es poden produir sobre el pla imatge durant el procés de captura realitzada per una càmera: La projecció perspectiva i la projecció ortogràfica.

### 2.2.3 Projecció ortogràfica o ortogonal

A diferència de la projecció perspectiva, la projecció ortogràfica, té el punt de mira situat a l'infinit en la direcció a l'eix Z, és a dir, es prescindeix del punt focal i no és possible projectar-ho amb una càmera de model pinhole.

Es consideren totes les línies de projecció perpendiculars al pla imatge, com es pot observar a la figura 2.4, així doncs, en aquest tipus de projecció els objectes

propers i llunyans tenen el mateix tamany en el pla imatge, per tant podem exposar les següents igualtats:

$$x' = x \qquad y' = y$$

El desavantatge d'aquesta projecció és que no hi ha percepció de la profunditat, ja que no succeeix el mateix que en la projecció perspectiva, on els objectes llunyans són més petits en el pla imatge que els objectes més propers.

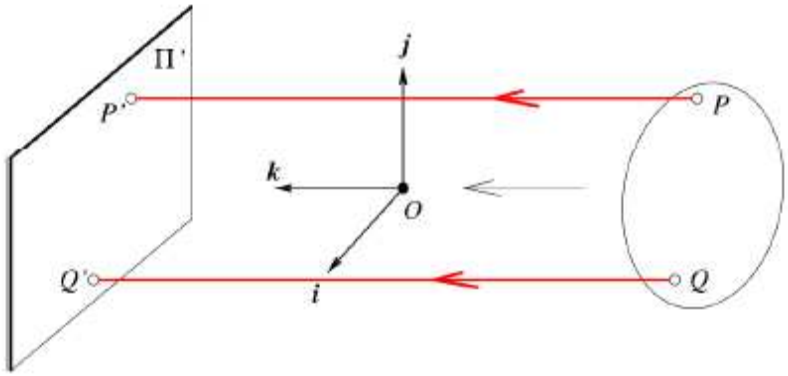



Figura 2.4. Esquema de la projecció de dos punts en projecció ortogràfica, com es pot apreciar aquí en el pla imatge no hi queda reflectida cap informació sobre la profunditat en què es troben els diferents punts [11].

## 2.2.4 Projecció perspectiva

El model més simple de càmera per a l'obtenció d'imatges és la càmera pinhole i la seva projecció és perspectiva, ara explicarem que és la projecció perspectiva. Les càmeres habituals tenen una lent  més gran que concentra la llum i permet millor les imatges. La projecció perspectiva proporciona les equacions que relacionen la imatge amb l'objecte. Un punt qualsevol d'un objecte, té una projecció a la imatge donada per la intersecció de la recta que passa pel punt d'aquest objecte i pel centre projecció (punt on es troba situat el forat de la càmera), i el pla imatge que és on es forma la imatge (veure figura 2.5).

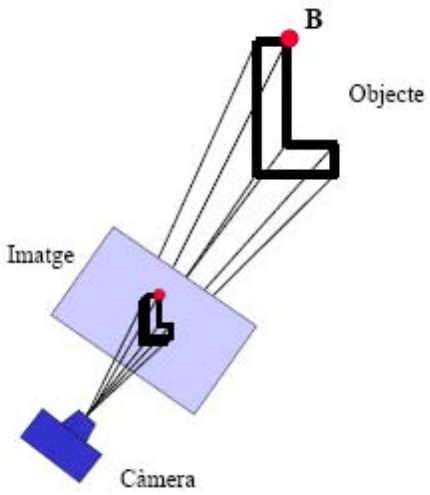



Figura 2.5. Exemple esquemàtic de la projecció perspectiva.

Definim l'eix òptic com la recta que passa pel punt del centre de la projecció i que és ortogonal al pla imatge. Anomenem punt principal O, al punt d'intersecció del pla imatge i l'eix òptic. La distància f del punt F al pla imatge es denomina distància focal. És equivalent col·locar el centre de projecció davant o darrera del pla imatge (veure figura 2.6).

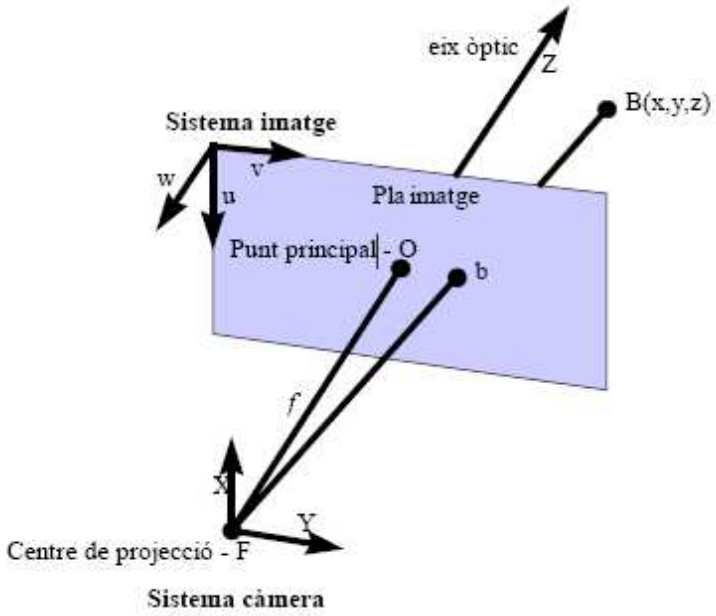



Figura 2.6. Un altre exemple de la projecció perspectiva en el pla imatge

Un punt B es projecte en el pla imatge mitjançant la recta que passa per B i F i que talla al pla en un punt b. Escollim un sistema de coordenada càmera: el pla X-Y, que és paral·lel al pla imatge, i l'eix Z és el mateix que l'eix òptic. L'origen d'aquest sistema es troba en F. Siguin (x,y,z) les coordenades del punt B, i (x',y',z') les de b, aleshores podem escriure les relacions següents per proporcionalitat de triangles semblants:

$$\frac{x}{x'} = \frac{y}{y'} = \frac{z}{z'}$$

D'on podem deduir el següent:

$$x' = \frac{f\,x}{z} \qquad y' = \frac{f\,y}{z} \qquad z' = \frac{f\,z}{z} \qquad z' = f$$

També es pot escriure en forma matricial si utilitzem les coordenades homogènies:

$$\begin{pmatrix} sx \\ sy \\ sz \\ s \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & x \\ 0 & 1 & 0 & 0 & y \\ 0 & 0 & 1 & 0 & z \\ 0 & 0 & 1/f & 0 & 1 \end{pmatrix}$$

Fent les operacions corresponents tenim:

$$s\,x' = x \qquad s\,y' = y \qquad sz' = z \qquad s = \frac{z}{f}$$

I finalment fent la substitució de s per el seu valor

$$x'\frac{z}{f} = x \qquad y'\frac{z}{f} = y \qquad z'\frac{z}{f} = z$$

Són equivalents a les tres relacions de la projecció perspectiva obtingudes anteriorment.

## 2.3. Reconstrucció 3D

La reconstrucció 3D és el procés en el qual s'obté informació tridimensional d'objectes a partir d'imatges 2D capturades per càmeres. Dintre la visió per computador existeixen moltes tècniques per realitzar una reconstrucció 3D, on el seu objectiu principal és obtenir un algorisme que sigui capaç de realitzar la connexió del conjunt de punts representatius de l'objecte.

En aquest projecte, el que tractarem de fer és trobar aquests punts 3D mitjançant el mètode de triangulació que explicarem amb més detall a continuació, a partir d'un sistema estèreo basant-nos en els fonaments de l'estereòscopia. Un sistema estèreo és un sistema on s'utilitzen dues càmeres les quals tenen sol·lapament d'una escena entre elles dos i amb aquest sistema som capaços d'extreure'n informació tridimensional mitjançant diferents mètodes de reconstrucció 3D. A continuació explicarem alguns temes bàsics per a entendre com funciona la reconstrucció 3D a partir d'un sistema estèreo.

### 2.3.1 L'estereocòpia

L'estereoscòpia o imatge 3D (imatge tridimensional) és qualsevol tècnica capaç de recollir informació visual tridimensional o de crear una il·lusió de profunditat d'una imatge. La il·lusió de la profunditat en una fotografia, pel·lícula o qualsevol imatge bidimensional és creada presentant una imatge lleugerament diferent per cada ull, tal com ens passa a nosaltres, els humans, alhora de recollir informació sobre la realitat.

L'estereoscòpia és útil per veure imatges amb informació 3D d'un conjunt de dades multi-dimensionals com les produïdes per dades experimentals. La imatge tridimensional de la indústria moderna pot utilitzar escànners 3D per detectar i guardar informació tridimensional. La informació tridimensional de la profunditat pot ser reconstruïda a partir de dos imatges utilitzant un computador que amb un algorisme sigui capaç de relacionar píxels que corresponguin a les imatges de l'esquerra i de la dreta, d'aquest tema se n'anomena el problema de la correspondència que se'n parlarà més endavant.

Amb l'obtenció d'aquests punts de correspondència més els paràmetres de la càmera, com veurem més endavant, serem capaços d'obtenir informació tridimensional precisa a partir de dos imatges.
Però prèviament, abans de tot això, s'haurà de realitzar una acurada calibració del sistema, tal com es detalla en la següent secció per trobar la geometria entre les dues càmeres.
La reconstrucció estereoscòpica tradicional consisteix en crear escenes 3D a partir d'imatges bidimensionals (2D), tal com el nostre cervell és capaç de percebre informació de la profunditat gràcies a la informació proporcionada per els ulls, ja que ens genera dos imatges diferents i que representen dos perspectives del mateix objecte, amb una petita separació entre ells.

### 2.3.2 Calibració

La calibració de les càmeres en la visió per computador és el procés per a determinar la geometria interna de la càmera i les seves característiques òptiques (paràmetres intrínsecs) i/o la posició 3D i l'orientació de la càmera respecte el sistema de coordenades del món, els quals s'anomenen paràmetres extrínsecs. A més a més amb la calibració podrem passar de píxels en imatges a mesures 3D reals com per exemple en mil·límetres que és com ho farem en aquest projecte.

Els paràmetres de la càmera, es divideixen en paràmetres intrínsecs i en paràmetres extrínsecs, els paràmetres de la càmera es representen en una matriu 3x4 anomenada matriu de la càmera

Sovint utilitzem $[x\ y\ 1]^\top$ per representar un punt 2D en les coordenades com a píxel. $[x_w\ y_w\ z_w]^\top$és utilitzat per representar un punt 3D al sistema de coordenades del món real.
Referent al model de la càmera pinhole, la matriu de la càmera que és utilitzada per denotar un mapeig de la projecció del món real a les coordenades de píxel seria la següent:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = A \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

**Paràmetres intrínsecs**

La matriu intrínseca conté 5 paràmetres. Aquests paràmetres són: la distància focal, format de la imatge i el punt principal. Els paràmetres $\alpha_x = f \cdot m_x$ i $\alpha_y = f \cdot m_y$ representen en termes la longitud focal en píxels, on mx i my són els factors d'escala que relacionen els píxels amb la distància.

També trobem paràmetres no lineals com és el cas de la distorsió de les lents que també és molt important perquè no s'han tingut en compte en la matriu dels paràmetres intrínsecs del model de la càmera lineal, però com que les càmeres no són perfectes tenen distorsió. Hi ha dos tipus de distorsions, la distorsió radial i la distorsió tangencial. A la figura 2.7 podem veure un exemple de distorsió radial on primer surt la imatge distorsionada i llavors la imatge rectificada, on eliminem la distorsió.

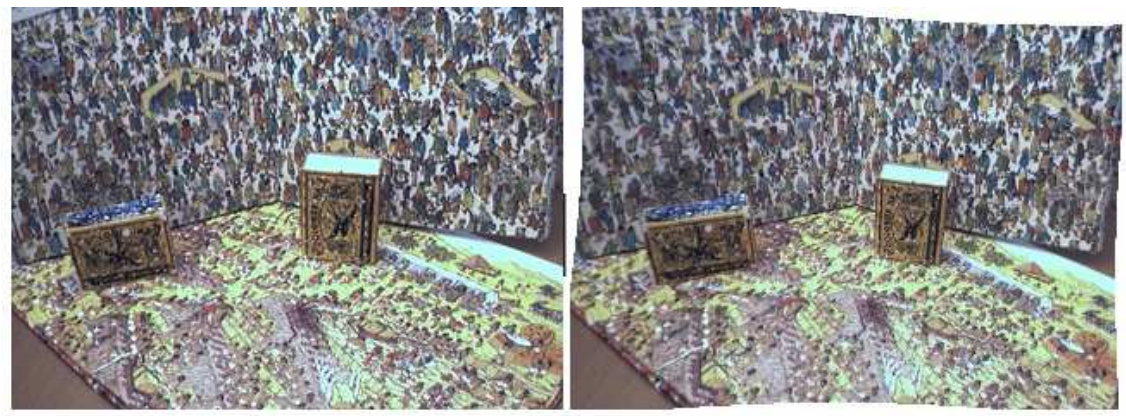


Figura 2.7. Exemple de distorsió radial, a l'esquerra tenim la imatge distorsionada i a la dreta la imatge corregida eliminant la distorsió radial.

Si projectem un punt sobre la imatge en forma de vector $\mathbf{X} = [\mathbf{X_c};\mathbf{Y_c};\mathbf{Z_c}]$ en el sistema de coordenades de la càmera.

$$p_d = \begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} \frac{P_x}{P_z} \\ \frac{P_y}{P_z} \end{pmatrix}$$

Denotem $r^2 = x_d^2 + y_d^2$. Després incloem el model de distorsió de les lents, on el punt sense distorsió es regeix per la següent igualtat:

$$p_u = \begin{pmatrix} x_u \\ y_u \end{pmatrix} = (1 + k_{c1} \cdot r^2 + k_{c2} \cdot r^4 + k_{c5} \cdot r^6) \cdot p_d + d_t$$

on *kc1, kc2.. kc5* són els coeficient de distorsió i on dt és el vector de la distorsió tangencial que es defineix:

$$d_t = \begin{pmatrix} 2 \cdot k_{c3} \cdot x_d \cdot y_d + k_{c4} \cdot (r^2 + 2 \cdot x_d^2) \\ k_{c3} \cdot (r^2 + 2 \cdot y_d^2) + 2 \cdot k_{c4} \cdot x_d \cdot y_d \end{pmatrix}$$

Els paràmetres kc1, kc2... kc5 són els coeficients de la distorsió no lineal. Després d'eliminar la distorsió, el punt (xu,yu,1)^T és projectat al pla imatge utilitzant la matriu K, de la següent manera:

$$\begin{cases} \mathbf{x_p} = \mathbf{fc(1)} \left( \mathbf{x_d(1)} + \mathbf{alpha\_c} * \mathbf{x_d(2)} \right) + \mathbf{cc(1)} \\ \mathbf{y_p} = \mathbf{fc(2)}\ \mathbf{x_d(2)} + \mathbf{cc(2)} \end{cases}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \cdot \begin{pmatrix} x_u \\ y_u \\ 1 \end{pmatrix}$$

Aquesta distorsió no lineal apareix degut a les imperfeccions de les lents de les càmeres, del centrat de la lent i defectes de fabricació entre d'altres [1].

**Paràmetres extrínsecs**

Els paràmetres extrínsecs ens diuen la relació entre el món i la càmera, a més a més, en sistemes multi-càmera (estèreo en el cas d'aquest projecte), els paràmetres extrínsecs són la relació geomètrica que hi ha entre les càmeres que formen el sistema, tant en rotació com en translació respecte els seus propis eixos de coordenades.
L'origen del sistema de coordenades és just en el centre de projecció de la càmera $(X_0, Y_0, Z_0)$ els eixos X i Y formen el pla imatge i l'eix Z és perpendicular al pla imatge el qual ens donarà informació de profunditat.

Com a paràmetres extrínsecs tenim la rotació i la translació. La rotació es tracta d'una matriu de rotació 3x3 i la translació es tracta d'un vector de 3x1. Com es pot veure la imatge en un procés de calibració situarem un sistema de coordenades sobre un objecte, en el cas de la calibració sovint utilitzarem un patró tal com es mostra a la figura 2.8.

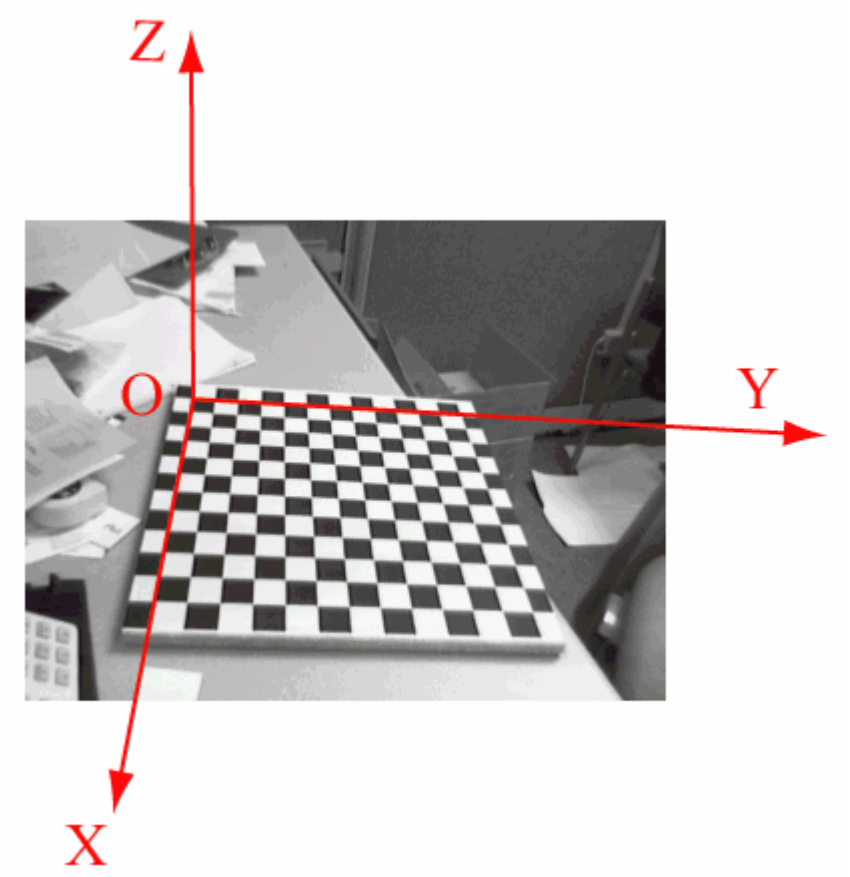



Figura 2.8. Exemple de l'eix de coordenades d'un punt el qual té una rotació i translació respecte l'origen del sistema de coordenades de la càmera [1].

Llavors com es pot veure a la figura 2.8, el sistema de coordenades dibuixat té una rotació i translació respecte el sistema de coordenades de la càmera. Bàsicament els paràmetres extrínsecs són aquests quan es tracta d'una sola càmera. En cas que hi hagi una segona càmera, suposant un sistema estèreo, els paràmetres extrínsecs per a calcular posicions 3D ens serà necessari per relacionar els sistema de coordenades de cada càmera, una respecte l'altra, d'això ja en parlarem més endavant.

Suposem $(XX_c)=(X_c Y_c Z_c)$ és el vector de coordenades de P respecte la càmera. Llavors si els sistemes de coordenades estan relacionats podem extreure'n la següent equació, tenint en compte el que s'ha explicat anteriorment.

$$(XX_c)=Rc_1 XX+Tc_1$$

El vector de translació $Tc_1$ és el vector de coordenades de l'origen de coordenades de la càmera a l'origen de coordenades de l'altre sistema, i llavors la matriu $Rc_1$ conté la rotació entre ambdós sistemes de coordenades.

Resumint, el procés de calibració com s'ha explicat és el procés de determinació dels paràmetres intrínsecs, les distorsions no lineals i els paràmetres extrínsecs (R,t) (la posició relativa de la càmera respecte un punt on hi ha un altre sistema de coordenades), per a realitzar aquesta calibració s'utilitzen un conjunt d'imatges d'un patró on els punts 3D i la distància entre punts ho coneixem. Quantes més imatges tinguem, més punts i més estimacions podrem fer per a extreure els esmentats paràmetres. Més endavant, en aquest projecte, veurem com es realitza el procés de calibració d'una càmera i d'un sistema estereoscòpic.

### 2.3.3 Geometria epipolar

La geometria epipolar es refereix a la geometria de la visió estèreo. Quan dues vistes diferents de dues càmeres en diferents posicions, tenen un nombre de relacions geomètriques entre els punts 3D reals i les projeccions en les dos imatges 2D que porten a restriccions entre les dos imatges tal i com es pot veure a la figura 2.9. Aquestes relacions es basen en el supòsit que les càmeres són similars a les del model pinhole.

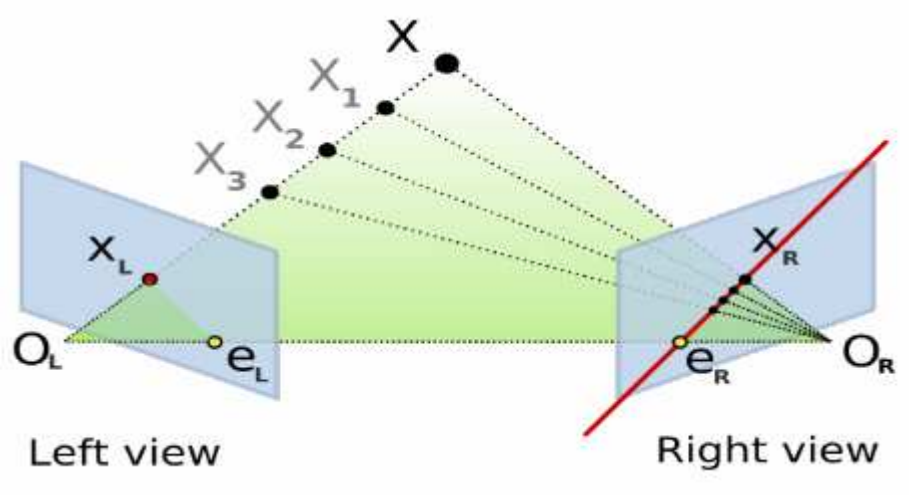



Figura 2.9. Esquema de la geometria epipolar

La imatge mostra dues càmeres pinhole buscant el punt X. En les càmeres reals, el pla imatge es troba darrera el punt focal, i això produeix una imatge rotada. Aquí apareix el problema de la projecció que és simplificat a través del pla imatge virtual davant del punt focal de cada càmera per produir imatges sense rotació. $\mathbf{O}_{\mathrm{L}}$ i $\mathbf{O}_{\mathrm{R}}$ representen el punt focal de cada càmera respectivament. X representa el punt d'interès d'ambdues càmeres. Els punts $\mathbf{x}_{\mathbf{L}}$ i $\mathbf{x}_{\mathrm{R}}$ i les projeccions del punt X en el seu pla imatge.

Cada càmera captura una imatge 2D del món real (3D). Aquesta conversió de 3D a 2D es basa en la projecció perspectiva, i tal com s'ha explicat anteriorment, és el que utilitza la càmera model pinhole. És normal que el punt a buscar passi pel punt focal d'ambdues càmeres. S'ha de tenir en compte que cada un de les línies correspon a un sol punt a la seva pròpia imatge.

Com que els dos punts focals de les càmeres són diferents, cada punt focal projecta un punt diferent a l'altre pla imatge. Aquests dos punts de les imatges es denoten com a $\mathbf{e}_{\mathrm{L}}$ i $\mathbf{e}_{\mathrm{R}}$ i s'anomenen els **punts epipolars**. Ambdós punts $\mathbf{e}_{\mathrm{L}}$ i $\mathbf{e}_{\mathrm{R}}$ estan en el seu pla respectiu i els dos punts focals $\mathbf{O}_{\mathrm{L}}$ i $\mathbf{O}_{\mathrm{R}}$ es troben en una línia 3D que les uneix.

La línia que va de $\mathbf{O}_{\mathrm{L}}$ a $\mathbf{x}$ és vista des de la càmera esquerra com un punt perquè passa directament pel seu punt focal, d'altra banda la càmera dreta, veu

aquest punt com a una línia en el seu pla imatge. Aquest línia ($\mathbf{e}_R$ **-** $\mathbf{x}_R$) de la càmera dreta s'anomena línia epipolar. Simètricament, succeeix el mateix per la línia $\mathbf{O}_R$ **-** $\mathbf{x}$ de la càmera dreta vist des del punt de vista de la càmera esquerra que s'observa la línia epipolar que va de $\mathbf{e}_L$ a $\mathbf{x}_L$.

Una línia epipolar és una funció d'un punt 3D, és a dir, hi ha un conjunt de línies epipolars a ambdues imatge si permetem que X variï sobre tots els punts 3D. Des de la línia 3D que va de $\mathbf{O}_L$ **a** $\mathbf{x}$ passa a través del punt focal de la càmera esquerra $\mathbf{O}_L$, la línia corresponent a la línia epipolar a la imatge dreta ha de passar a través de l'epipol $\mathbf{e}_R$ (i per a les línies epipolars de la imatge esquerra). Això significa que totes les línies epipolars d'una imatge han d'intersectar a un punt epipolar de la mateix imatge. De fet, qualsevol línia que intersecti amb un punt epipolar és una línia epipolar, ja que poden derivar d'algun punt 3D. D'aquesta manera ens facilitarà la localització d'un mateix punt 2D en l'altre imatge (problema de la correspondència).

**Pla epipolar**

Com una alternativa de visualització, considerant els punts $\mathbf{x}$, $\mathbf{O}_L$ i $\mathbf{O}_R$, aquests formen un pla anomenat el pla epipolar. El pla epipolar intersecta en cada pla imatge d'ambdues càmeres on es formen les línies epipolars. Totes les línies epipolars intersecten amb el pla independentment d'on es trobi X.

**Restricció epipolar i triangulació.**

Si es coneixen els paràmetres extrínsecs entre les dues càmeres (rotació i translació entre elles), la geometria epipolar corresponen porta a dos factors importants:

Si coneixem el punt de projecció xL, llavors la línia epipolar eR-xR també és coneguda i el punt X que projecta a la imatge de la dreta, en un punt xR que forma aquesta línia epipolar. Això proporciona una restricció epipolar corresponent a la imatge dels punts que ha de complir i significa que és possible posar-ho a provar si realment són els mateix punts 3D. Les restriccions epipolars poden descriure's a través de la matriu fonamental entre les dues càmeres, on aquesta matriu és la matriu que hem extret anteriorment de la calibració.

Si els punts $\mathbf{x}$, $\mathbf{x}_L$ i $\mathbf{x}_R$ són coneguts, llavors les línies de projecció són també conegudes. Si els dos punts de les imatges corresponen al mateix punts 3D (punt $\mathbf{x}$) les línies de projecció tenen que intersectar a $\mathbf{x}$. Això significa que X pot ser calculat a través de les coordinades dels dos punts de les imatges, a través de la **triangulació**.

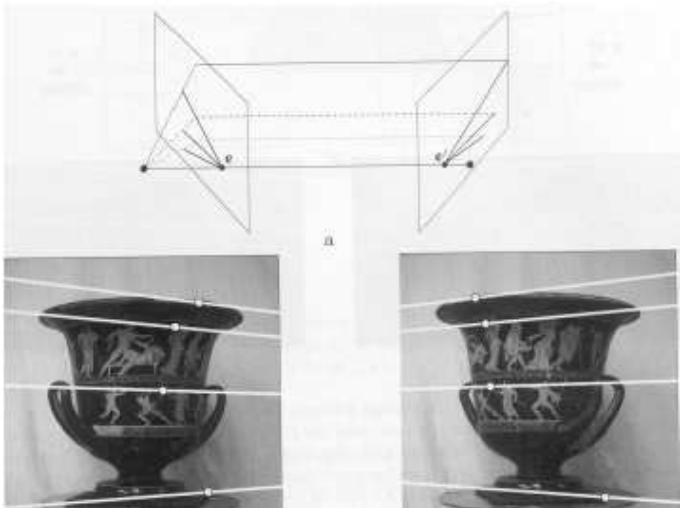


Figura 2.10. Exemple de les línies epipolars des de dos vistes diferents, vista esquerra i vista dreta [13]

A la figura 2.10 es mostra a la part superior les línies epipolars i l'epipol de les imatges en diferents posicions de la càmera respecte l'objecte. A la part superior de la figura es mostra la situació relativa de les càmeres. A la part inferior es mostren les imatges de cada càmera i les línies epipolars conjugades amb les correspondències de punts.

## 2.3.4 El problema de la correspondència

Donades dos o més imatges d'una mateixa escena en tres dimensions, capturades des de diferents punts de vista, el problema de la correspondència és trobar punts d'interès en una sola imatge que puguin ser identificats com els mateixos punts per a les altres imatges. Manualment, a simple vista podem resoldre aquest problema fàcilment i de manera ràpida, fins i tot quan les imatges contenen gran quantitat de soroll. En canvi, en la visió per computador, el problema de la correspondència s'estudia en el cas en què un computador ha de resoldre aquest problema de forma automàtica amb només les imatges com a

entrada. Una vegada que el problema de la correspondència és resolt, es diposita en una imatge el conjunt de punts que estan en correspondència amb totes les imatges. Hi ha diferents mètodes que es poden aplicar per resoldre aquest problema de la correspondència.

El problema de la correspondència apareix quan utilitzem dos imatges de la mateixa escena, en el cas d'aquest projecte, l'anomenem el problema de la correspondència d'un sistema estèreo. Tot i que aquest problema de la correspondència, pot trobar-se en N imatges d'una mateixa escena, ja que poden provenir de N càmeres diferents.

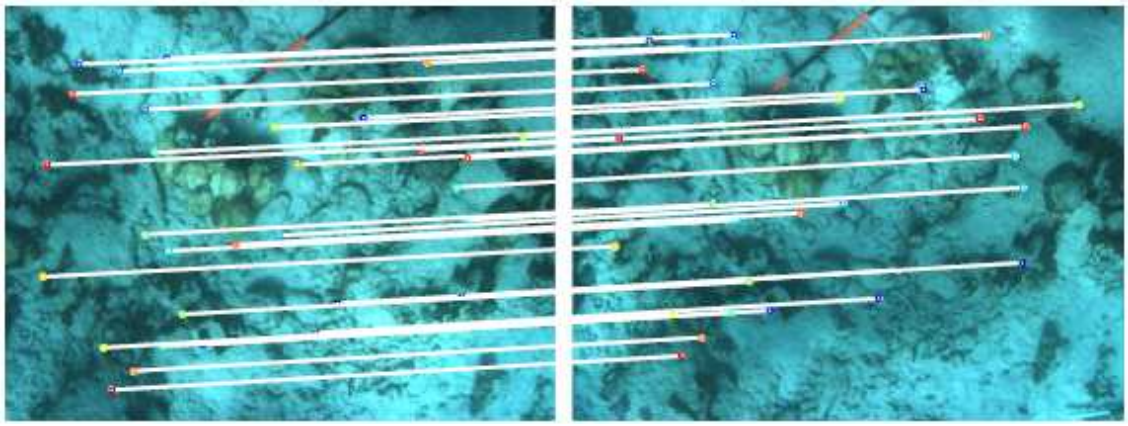


Figura 2.11. Exemple problema de correspondència del fons marí. Els punts que estan relacionats són els mateixos des de dos vistes diferents [5].

Una aplicació típica de la correspondència seria quan hi ha dues càmeres que estan a una petita distància l'una de l'altre (sistema estèreo) fixes i degudament calibrades. A la figura 2.11 podem veure un exemple del problema de la correspondència en imatges captades en un sistema estèreo.
Al conèixer els paràmetres de les càmeres (intrínsecs i extrínsecs) llavors si apliquem algun bon mètode per a resoldre el problema de la correspondència podrem reconstruir en 3D els diferents punts que compleixin amb la correspondència mitjançant el mètode de la triangulació.

Alguns bons mètodes per a resoldre el problema de la correspondència són el SIFT (Scale invariant feature transform) i el SURF (Speeded Up Robust Features) que ens extreuen característiques de les imatges, a més que són invariants en quan a rotació, translació i escala entre diferents imatges, d'aquesta manera podem trobar punts característics que siguin vistos per les càmeres des de

diferents posicions. Llavors es realitza el *matching* entre diferents imatges utilitzant altres algorismes.

### 2.3.5 Triangulació

En la visió per computador la triangulació es refereix al procés de determinació d'un punt 3D a partir de dos o més imatges amb diferents projeccions. Per aplicar aquest mètode és necessari conèixer tots els paràmetres de la calibració de cada una de les càmeres del sistema, gràcies a la matriu de la càmera que s'ha extret en el procés de calibració. A partir de la triangulació, doncs, podem crear una reconstrucció en 3D.

Si un parell de punts es correspon a les dos imatges, lògicament, corresponen el mateix punt a la realitat, essent així un punt comú 3D. Llavors el conjunt de línies de la imatge generada per punts d'intersecció amb el punt corresponent a les dos imatges i a la formulació algebraica de les coordenades d'aquest punt, després aquest punt 3D pot ser calculat de diferents maneres, d'aquesta manera resolem el problema de la triangulació.

A la pràctica, però, les coordenades reals no es poden mesurar sempre amb una precisió ben acurada, sempre poden aparèixer errors en les distàncies reals. D'altra banda també poden aparèixer punts que continguin soroll en una de les diferents imatges, per a això, hi ha mètodes per a seleccionar els punts òptims per a realitzar aquesta tasca.

A continuació s'explica un exemple de com funciona la triangulació: es suposa que la triangulació es realitza sobre punts de la imatge a partir de dos vistes diferents generades per càmeres del model pinhole.

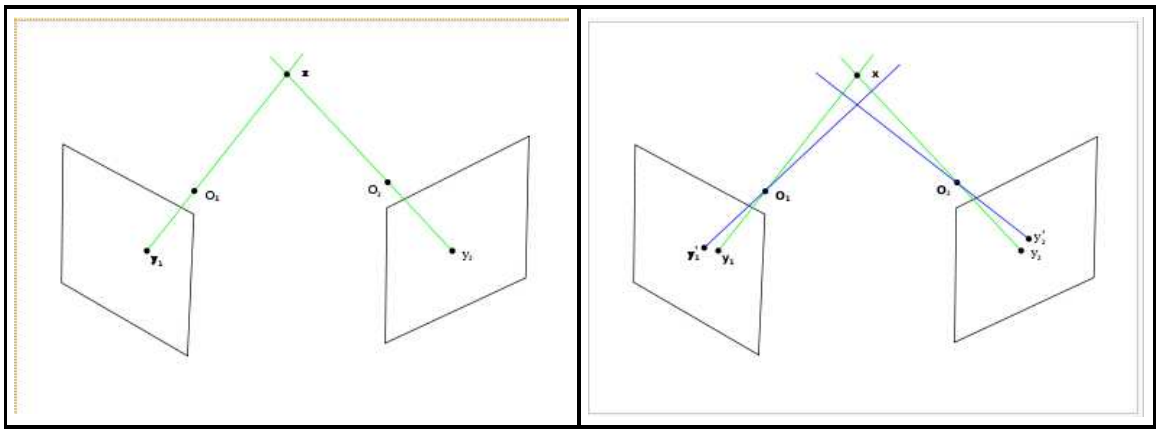


Figura 2.12. Exemple esquemàtic de l'error de triangulació

A la figura 2.12, la imatge de l'esquerra mostra la geometria epipolar d'un sistema estereoscòpic. Un punt x en l'espai 3D es projecta sobre els respectius plans imatge, tal com indica la línia verda que passa a través del punt focal de la càmera, *O1* i *O2*, donant lloc a la corresponent imatge de dos punts *y1* i *y2*. Si els paràmetres de calibració de la càmera són correctes, llavors amb els punts *y1* i *y2* acabaran intersectant a un punt x, tal com s'indica a la imatge. Utilitzant l'àlgebra lineal podem determinar aquest punt 3D on intersecten les dos projeccions.

La segona imatge mostra el cas real, és a dir, la posició dels punts de les imatges y1 i y2 no poden ser mesurats amb plena exactitud, degut a diferents factors com podrien ser:

- Distorsió geomètrica, com la distorsió de lents, tot i que amb una bona calibració sempre aconseguirem reduir la imprecisió esmentada.
- Les dues càmeres digitals poden generar diferents intensitats de llum per a les imatges capturades i al soroll.
- Els punts de la imatge utilitzats per a la triangulació es troben normalment a partir d'extractors de característiques, com podrien ser les cantonades d'alguns objectes. En aquests casos també hi ha un error de localització d'aquest punt per cada tipus d'extracció de característiques que es basa en mètodes d'operar amb els píxels veïns dels punts a analitzar.

A conseqüència d'això la imatge dels punts mesurats y1' i y2' en lloc de y1 i y2. Tot i això, les seves línies de projecció (línies blaves) no cal que s'entrecreuin en

l'espai 3D a acostar-se al punt x. De fet, aquestes línies s'intersecten només si y1' i y2' poden satisfer les propietats de la geometria epipolar. Donat un soroll a y1' i a y2' és possible, degut als problemes esmentats anteriorment, que la limitació epipolar no es compleixi i la projecció de línies no intersectin.

Aquests problemes es solucionen a la triangulació. Quin punt 3D és el millor per estimar x a través dels punts y1' i y2'? La resposta sovint es troba realitzant comprovacions reals, comprovant l'error entre el punt x' i el x, llavors podrem anar minimitzant aquest error. Tots els mètodes de triangulació proposen que x=x' en el cas que es compleixi també: y1=y1' i y2=y2'. És a dir, quan es compleixen les propietats de la geometria epipolar (a part de punts singulars).

A la figura 2.13 podem observar una explicació esquemàtica de com seria la triangulació 3D des dels sistemes de coordenades de dues càmeres per a un sistema estèreo.

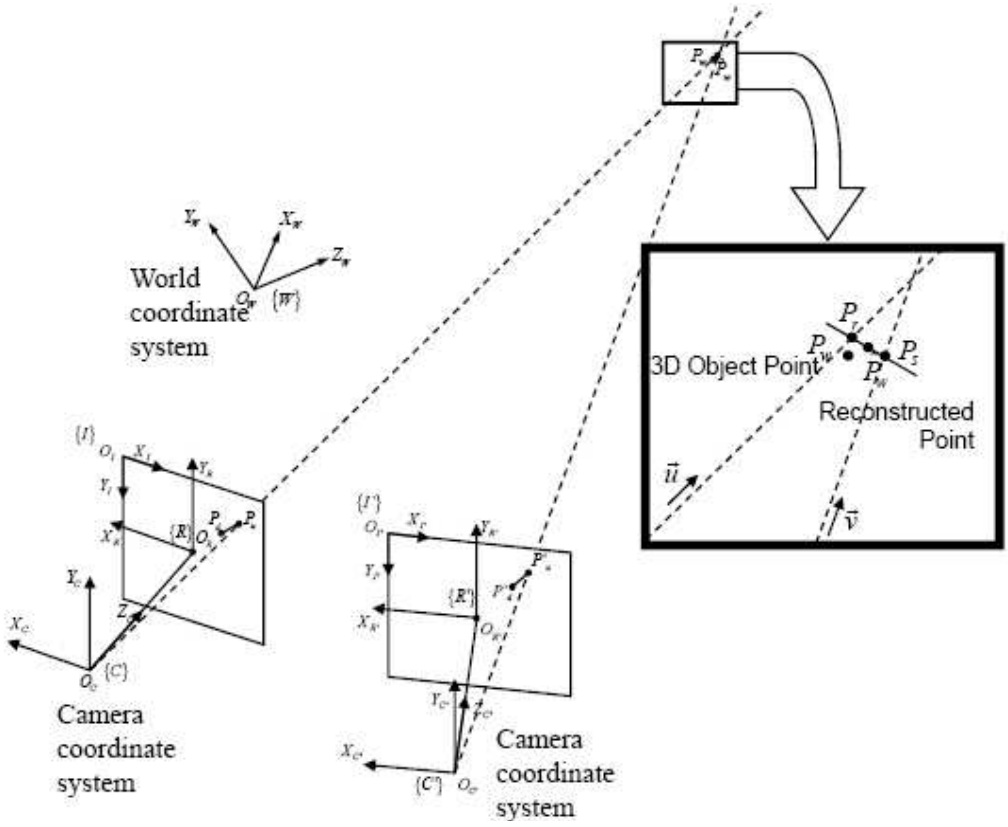



Figura 2.13. Esquema gràfic de triangulació on podem veure els sistemes de coordenades de les dues càmeres i com s'originaria la triangulació a partir d'aquests [11].

# Capítol 3

# Anàlisis, disseny i implementació d'algorismes de segmentació

## 3.1 Introducció

Els mètodes de segmentació són un grup d'algorismes que la seva finalitat és segmentar dades, és a dir separar objectes/punts d'una imatge que ens interessin de la resta en dos classes diferents. En el cas d'aquest projecte el que ens interessa és separar els punts rígids dels punts no rígids d'una estructura o objecte 3D. Amb l'ajuda de mètodes de llindarització podem decidir quins píxels/punts formen part dels objectes rígids o no rígids. Hi ha diferents tipus de segmentació, en el cas del processament d'imatges hi ha diferents mètodes per a crear imatges binàries (blanc i negre) on el procés s'anomena binarització, a la figura 3.1 podem veure un exemple, alguns d'aquests mètodes també es poden aplicar en el camp que estem estudiant.

Figura 3.1 Exemple de binarització, segmentació d'una imatge en blanc i negre

Com amb tots els mètodes de segmentació es tracta d'assignar un punt o píxel a un cert grup. El que s'ha de segmentar conté una quantitat de valors numèrics, en el cas de les imatges a les que es vol aplicar la binarització, tindrem valors que ens indiquen el nivell de gris dels píxels. La idea és trobar un llindar on el

seu valor realitzi la millor segmentació entre els píxels per obtenir així una completa imatge binària. Però aquest no és el cas d'aquest projecte. En aquest PFC ens interessarà poder segmentar els punts rígids dels que no són rígids, per a això, hem de seguir uns altres passos que es detallaran a continuació.

Per a poder segmentar punts rígids de punts no rígids en un entorn 3D, és necessari tenir una seqüència 3D on hi hagi una rotació i translació dels punts rígids i on també els punts no rígids hagin pogut desplaçar-se respecte els punts rígids. Amb la captura d'aquesta seqüència, llavors mitjançant diferents mètodes podem calcular l'error dels punts rotats i traslladats respecte el frame original. El mètode bàsic per a realitzar aquests passos és el RANSAC. El mètode RANSAC el que ens fa bàsicament és trobar una transformació en rotació i translació a mesura que avança la seqüència respecte el primer frame, el qual s'agafa com a punt de partida per al conjunt de punts 3D. Aquest model que ens dóna RANSAC ens servirà per aplicar la rotació i translació inversa a cada un dels punts del frame a tractar per a retornar-lo a la posició on es trobaven en el primer frame. Com que RANSAC utilitza un llindar per a detectar només els inliers (entenem que serien punts rígids) descartant els outliers (punts que es trobarien fora d'una certa distància i no formarien part del model), doncs amb el model retornat s'aplica la rotació i translació inversa a cada un dels punts per a retornar-los a l'origen, aquí en cas que el model fos perfecte, els punts rígids es trobarien sobre ells mateixos en el punt de partida de la seqüència, en canvi els punts no rígids, al haver sofert un desplaçament respecte els mateixos punts al inici de la seqüència, hi hauria un error de posició entre els punts del primer frame i els punts del frame a tractar. Aquest error nosaltres el mesurarem en mil·límetres, que serà el sumatori de cada un dels errors 3D frame a frame respecte el frame inicial. A més, a més. Amb aquesta segmentació ens podria ser útil per després recuperar models 3D deformables. A la figura 3.2 podem veure un exemple de segmentació entre diferents punts on tenen errors diferents.

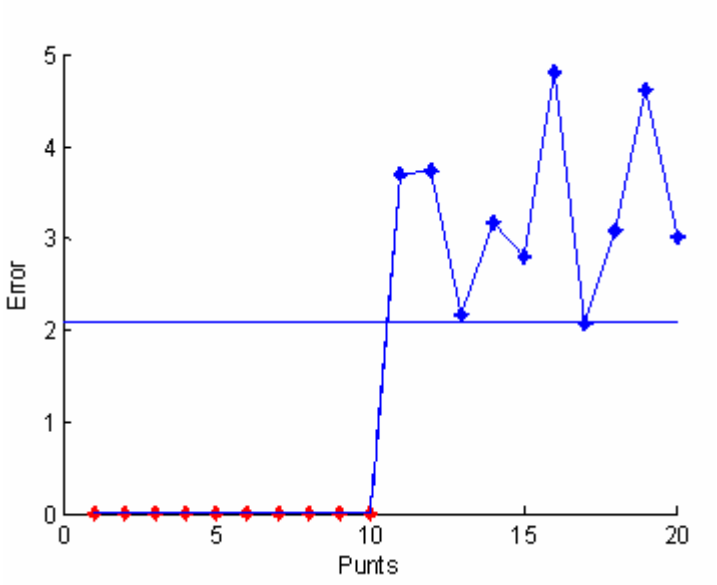



Figura 3.2 Exemple de segmentació entre punts rígids i punts no rígids a partir dels errors de posició.

Degut al soroll i a altres imprecisions, com els paràmetres de calibració de la càmera, l'error de posició per als punts rígids difícilment serà 0 però hi ha de ser proper. En canvi per als punts no rígids, com s'ha comentat en el paràgraf anterior, l'error sí que ha de ser diferent a 0 i més elevat respecte l'error dels punts rígids. És aquí doncs on amb la suma i/o mitjana d'errors de tots els punts per tots els frames disposarem de valors numèrics els quals podrem aplicar-hi algun mètode de segmentació per a poder distingir els punts rígids dels punts no rígids. A la figura 3.3 podem veure què passa quan hi ha soroll i els punts rígids i no rígids estan desordenats numèricament, a diferència de la figura 3.2.

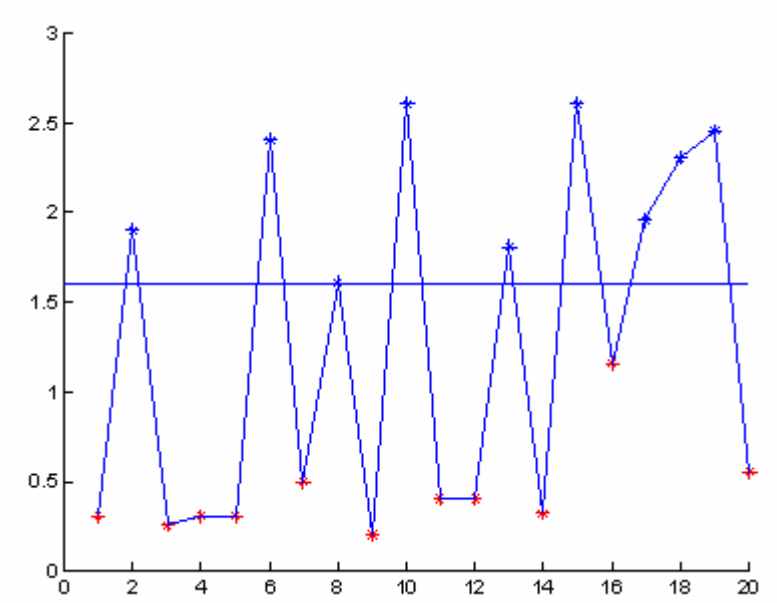



Figura 3.3 Exemple de segmentació amb soroll gaussià i punts desordenats, per sobre la línia de segmentació que es troba aproximadament a 1'6 s'hi troben els punts no rígids i per sota els punts rígids. El punt 16 possiblement s'hagi segmentat malament, s'ha considerat rígid en comptes de no rígid degut a l'excés de soroll que ha interferit en aquest procés de segmentació.

A continuació expliquem els diferents mètodes que posarem a prova al llarg d'aquest projecte. Els diferents mètodes no han estat pensats per la segmentació de punts rígids i no rígids, però que estudiarem i provarem com es comporten i com els podrem aplicar en aquest camp.

## 3.2 RANSAC

RANSAC és una abreviatura de "mostra aleatòria de consens" (Random Sample Consensus). Es tracta d'un mètode iteratiu per a calcular els paràmetres d'un model matemàtic d'un conjunt de dades observades, que conté una sèrie de punts "outliers". Es tracta d'un algorisme no determinista en el sentit que produeix un resultat raonable amb certa probabilitat, on aquesta probabilitat augmenta a mesura que es permeten més iteracions.

Un supòsit bàsic és que les dades es composen de "inliers", és a dir, la distribució de les dades s'explica per a un conjunt de paràmetres del model, i les dades també es componen de "outliers" que són les dades que no encaixen en el model. A més a més, les dades poden estar sotmeses a soroll.

Els outliers poden venir, per exemple, dels valors extrems de soroll, o bé, de mesures errònies o hipòtesis sobre la incorrecta interpretació de les dades. RANSAC també suposa que, tenint en compte una llista d'inliers, existeix un procediment que pot estimar els paràmetres d'un model que explica o s'adapta òptimament a aquestes dades.

Un exemple senzill podria ser la línia de regressió 2D sobre una llista de punts d'un conjunt d'observacions (veure figura 3.4). Suposant que aquest conjunt conté inliers, és a dir, un conjunt de punts que poden ser instal·lats sobre una línia, i outliers, els punts que no poden trobar-se sobre aquesta línia, un simple mètode dels mínims quadrats per a la línia de regressió es produeixen en general una línia mal ajustada als inliers. RANSAC, d'altra banda, pot produir un model que és a partir dels inliers, sempre que la probabilitat d'escollir només inliers en la selecció de dades sigui prou alta. No hi ha cap garantia d'aquesta situació però, hi ha una sèrie de paràmetres que han de ser degudament escollits per mantenir el nivell de probabilitat raonablement alt per l'algorisme.

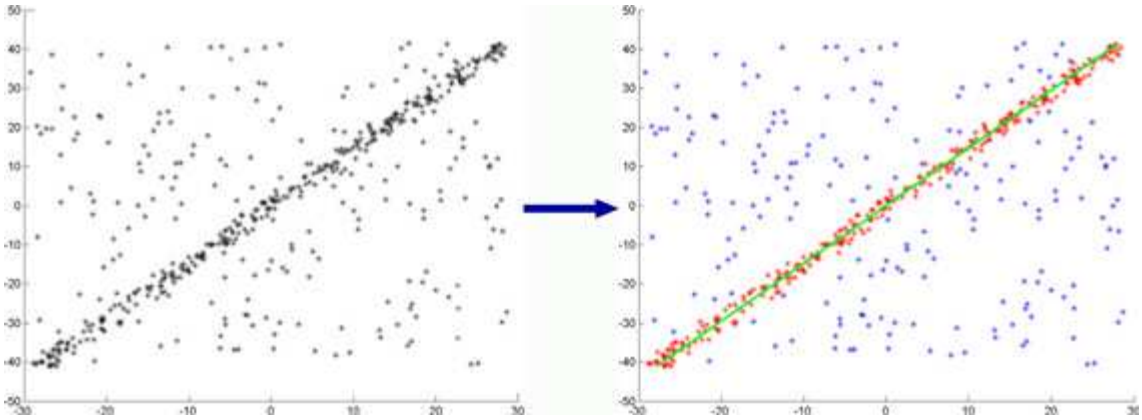


Figura 3.4. Exemple d'utilització de RANSAC per a trobar la línia de regressió, tots els punts que estan en vermell es consideren inliers, i els punts de color blau es consideren fora del model, per tant són outliers. [15]

L'entrada per l'algorisme RANSAC es tracta d'un conjunt de dades d'observacions, un model de paràmetres que poden explicar o ser situades les observacions, i alguns paràmetres de confiança per donar-hi cert marge a l'estimació (llindar).

RANSAC aconsegueix el seu objectiu gràcies a les iteracions d'una selecció d'un subconjunt aleatori de les dades originals.

Aquestes dades són hipotètics inliers i aquesta hipòtesi és prova de la següent manera:

1. Un model s'ajusta als hipotètics inliers, és a dir, prescindint de tots els paràmetres del model es reconstrueixen a partir de les dades establertes.

2. Totes les altres dades es sotmeten a la prova del model ajustat i si un punt s'adapta bé a l'estimació del model, també es considera com un hipotètic inlier.
3. L'estimació del model és raonablement bo si hi ha prou punts que han estat classificats com a hipotètics inliers.
4. El model és reestimat a partir de tots els hipotètics inliers, perquè només s'ha estimat a partir de la primera sèrie d'hipotètics inliers.
5. Finalment, el model és avaluat per l'estimació de l'error dels inliers en relació amb el model.

Aquest procediment es repeteix un nombre determinat de vegades, cada vegada, ja sigui la producció d'un model que és rebutjat per molts pocs punts es classifiquen com inliers, o bé, d'un refinat model juntament amb el corresponent error de mesura. En aquest últim cas, cal mantenir el model refinat si l'error és menor que l'anterior model.

En el cas de Ransac per a la utilització per aconseguir un model 3D, ens serà útil per a calcular les rotacions i translacions d'un mateix objecte traslladat i rotat on els punts per a calcular aquest model han de ser rígids, o bé, amb un petit error que no sobrepassi un llindar (distància euclidiana en aquest cas) que es passa com a paràmetre a la funció Ransac.

En aquest projecte, en altres paraules, Ransac ens serà útil per a comparar diferents frames amb el frame original després de fer la reconstrucció 3D. Aquesta comparació ens servirà per a trobar-ne deformacions, és a dir, al calcular el model matemàtic el qual conté la rotació i translació que ha sofert un frame respecte un altre frame (en el nostre utilitzarem el primer frame com a referència). Com que Ransac utilitza un llindar de confiança, aquest llindar començarà essent molt petit i anirà creixent a partir d'un bucle per tal de trobar una sèrie de punts els quals compleixin aquest interval de confiança per a

trobar-ne el model. Aquests punts seran els inliers, llavors la resta seran els outliers, és a dir, els punts fora del llindar de confiança. Amb aquests punts se'n calcula el model de la rotació i translació d'un frame a un altre on hi ha hagut moviment o no dels punts. Llavors aplicant el model de la rotació i translació, ho apliquem al frame a tractar respecte l'origen per tal de situar els punts sobre ells mateixos quan es trobaven en el frame original. Aleshores si hi ha punts els quals han sofert deformació (moviment extra respecte els del model), després d'aplicar-hi el model trobat per Ransac, hi haurà un error de posició per a aquests punts, els quals considerarem no rígids, en canvi els punts rígids es trobaran sobre si mateixos o molt propers depenent de la qualitat i el soroll dels frames de la seqüència (veure exemple figura 3.5 on tenim un cub el qual els seus 8 punts són rígids i al seu interior i voltant s'hi troben punts no rígids i alguns rígids).

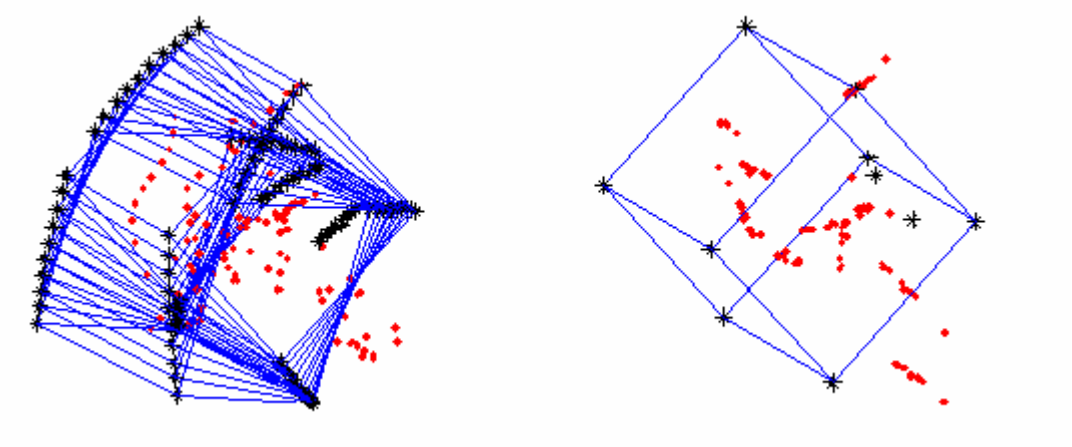


Figura 3.5. Exemple d'aplicació de Ransac quan no tenim soroll sobre dades sintètiques. En la primera imatge es veuen tots els punts del llarg de la seqüència de 10 frames, i en la segona imatge es veuen tots els punts restaurats a l'origen després de calcular el model amb Ransac. Els punts negres són els rígids, els quals no tenen error de posició nul, i en vermell es pot veure com els punts vermells són els que sí tenen error de posició.

Això que s'ha explicat es posarà a prova a l'apartat de resultats. Gràcies a Ransac podrem estimar l'error de posició de cada un dels punts que tractarem per a llavors a partir d'aquest error serem capaços d'aplicar-hi i estudiar diferents mètodes de segmentació per poder separar punts rígids de punts no rígids.

## 3.3 Mètodes

### 3.3.1 Mètode Otsu

El mètode Otsu [13], inventat per Nobuyuki Otsu l'any 1979 utilitza tècniques estadístiques per resoldre un problema de binarització d'imatges en escala de grisos. En concret, aquest mètode utilitza la variància, que és una mesura de la dispersió de valors, on el cas d'imatges en escales de grisos que es vulgui pretendre binaritzar (blanc i negre) utilitza la dispersió dels nivells de grisos, tal com es mostra en l'exemple de la figura 3.6, en el nostre cas utilitzarà la dispersió dels errors de posició.

Figura 3.6 Exemple de segmentació utilitzant el mètode Otsu, a l'esquerra tenim la imatge original, i a la dreta la imatge segmentada en blanc i negre (binaritzada) [16]

El mètode d'Otsu calcula el valor del llindar de forma que la dispersió dintre de cada segment sigui el més petita possible, però al mateix temps la dispersió sigui el més gran possible entre segments diferents. Per això, es calcula el quocient entre ambdues variàncies i es busca un valor pel llindar en què aquest quocient sigui el màxim.

Si ho mirem matemàticament, Com a punt de partida agafem dos segments de punts $(K_0(t)$ y $K_1(t))$, que seran definits a partir del valor del llindar t. t és la variable que busquem, i els dos segments són el resultat desitjat en la segmentació.

Sigui p(g) la probabilitat d'ocurrència del valor de gris $0 < g < G$ (G és el valor de gris màxim). Llavors la probabilitat d'ocurrència dels píxels en els dos segments és:

$K_0$: $P_0(t) = \sum_{g=0}^{t} p(g)$ y $K_1$: $P_1(t) = \sum_{g=t+1}^{G} p(g) = 1 - P_0(t)$

Si agafem dos segments ( o sigui un sol valor pel llindar) la suma d'aquestes dos probabilitat donarà evidentment 1.

Si g és la mitjana aritmètica dels valors de grisos de tota la imatge, i g0 i g1 els valors mitjos dintre de cada segment, llavors es poden calcular les variàncies dintre de cada segment com:

$$\sigma_0^2(t) = \sum_{g=0}^{t}(g - \overline{g_0})^2 p(g) \quad \text{y} \quad \sigma_1^2(t) = \sum_{g=t+1}^{G}(g - \overline{g_1})^2 p(g)$$

La finalitat és mantenir la variància dintre de cada segment i el més petita possible i aconseguir que la variància entre els dos segments sigui el més gran possible. Així obtenim:

$$Q(t) = \frac{\sigma_{zw}^2(t)}{\sigma_{in}^2(t)}$$

La variància entre els segments és:

$$\sigma_{zw}^2(t) = P_0(t) \cdot (\overline{g_0} - \overline{g})^2 + P_1(t) \cdot (\overline{g_1} - \overline{g})^2$$

La variància dintre dels segments s'obté de la suma de les dos:

$$\sigma_{in}^2(t) = P_0(t) \cdot \sigma_0^2(t) + P_1(t) \cdot \sigma_1^2(t)$$

El valor del llindar t es escollit de manera que el quocient Q(t) sigui màxim. Q(t) és per tant la mesura buscada. D'aquesta manera escollim un llindar que optimitza els dos segments en termes de variància.

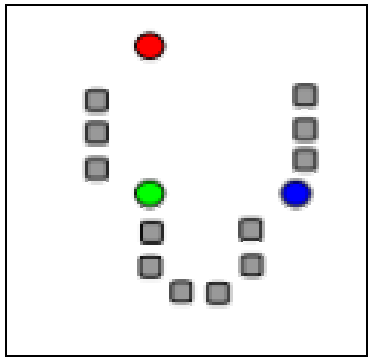
Hem explicat la idea bàsica amb per la qual es va inventar el mètode Otsu i com és utilitzat. Aquesta idea bàsica d'aplicar un mètode de segmentació/binarització d'imatges en escala de grisos, és també aplicable en el cas que estem estudiant,

la segmentació de punts rígids i punts no rígids. Simplement la diferència és que nosaltres no segmentarem nivells de grisos, sinó errors de posició entre els punts d'una seqüència on frame a frame anirem restaurant després de calcular la translació i rotació inversa gràcies el mètode Ransac, hi aplicarem aquesta rotació i translació a cada un dels punts del frame respecte l'origen i en calcularem l'error euclidià. Per tant, aquest mètode ens serà molt útil per a segmentar punts rígids de punts no rígids passant com a paràmetre d'entrada del mètode, doncs, els errors de posició en comptes dels nivells de grisos d'una imatge, així aconseguirem un llindar òptim seguint els passos que s'acaben d'explicar.

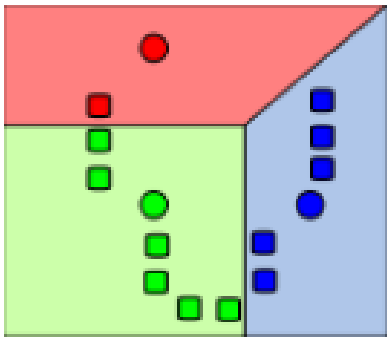
### 3.3.2 Mètode K-means

K-means [17], inventat per MacQueen a l'any 1967, és un dels algorismes més fàcils i útils per a resoldre el problema d'agrupació (clustering). El procediment tracta d'una manera senzilla de classificar un determinat conjunt de dades a través d'un cert nombre d'agrupacions (el nombre clusters que es desitgi) fixat a priori. L'idea principal és definir k "centroids", un per cada cluster (grup) . Aquests centroids inicialment haurien d'estar col·locats separats i aleatòriament degut a que els resultats han de ser diferents o bé es pot inicialitzar manualment si es té coneixement sobre les dades. Aleshores fem un recorregut per totes les dades assignant a quin cluster formen part cada una d'elles mirant quin centroid queda més proper de les dades. Quan ja hem fet el recorregut per totes les dades llavors necessitem recalcular els centroids, mirant les mitjanes dels clusters, una vegada hàgim recalculat els centroids tornarem a fer un recorregut per a totes les dades per a mirar a quin cluster pertanyen, es repetirà aquest procediment fins que no hi hagi cap canvi en les dades, és a dir, quan després de recalcular els centroids i fer un recorregut per totes les dades per a comprovar a quin cluster pertanyen no hi hagi cap canvi, llavors serà aquí quan es sortirà d'aquest bucle.

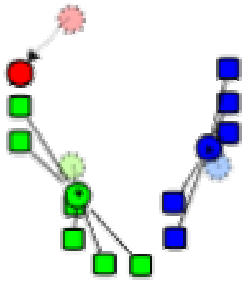
Una explicació gràfica d'aquest algorisme on volem separar dades en tres clusters diferents seria la següent:



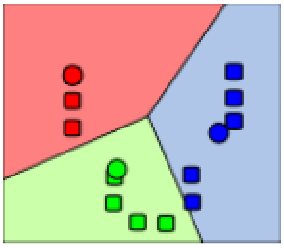
1) 3 centroids són seleccionats aleatòriament (mostrats els punts rodons el color).



2) 3 clusters són creats després d'associar cada una de les dades amb el centroid més proper. Les particions aquí es representen amb el diagrama de Voronoi, no hi entrarem en més detalls ja que nosaltres només treballarem en dos clusters.



3) El centroid per cada un dels clusters són recalculats a partir de la mitjana de cada un dels valors de les dades de cada cluster.



4) Repetim els passos 2 i 3 fins a aconseguir que no hi hagi més canvis.

A continuació s'adjunta el pseudocodi del mètode K-Means per a n_clusters.

```
algorisme K-Means(n_clusters,centreoids_inicials)

      //Triem aleatoriament els centres
      centres = calc_centres_inicials(n_clusters,
centroids_inicials)
      repetir
            //per totes les dades busquem quin centroid queda mes
proper
            per i de 1 fins màxim(valors)
                  canvis = calcula_cluster(i,centres)
            fper
            //recalculem tots els centroids
            centres = calcula_centres(n_clusters)
      fins no canvis
falgorisme

funcio calcula_cluster(i,centres)
      cluster_anterior = cluster_actual
      cluster_actual = busca_cluster_proper(i,centres)
      si cluster_anterior != cluster_actual llavors
            retorna cert
      altrament
            retorna fals
      fsi
ffuncio

funcio calcula_centres(n_clusters)
      per i de 1 fins num_clusters fer
            centres(i) = mitjana_cluster(i)
      fper
ffuncio
```

Aquest mètode de K-Means pot tenir diferents aplicacions en l'àmbit d'agrupar dades per a diferents finalitats, però a nosaltres el que ens interessa és poder agrupar aquestes dades en dos clusters. Aquests dos grups (clusters), lògicament, seran el grup de punts rígids, i el grup de punts no rígids. Més endavant a l'apartat de resultats veurem com es comporta aquest mètode com a mètode de segmentació de punts rígids i no rígids.

### 3.3.3 Mètode LDA (Linear Discriminant Analysis)

LDA o Linear Discriminant Analysis i el Fisher's linear discriminant són els mètodes utilitzats en les estadístiques d'aprenentatge automàtic per trobar la combinació lineal de característiques que millor separen dos o més classes d'objectes o esdeveniments. La combinació resultant pot ser utilitzat com un classificador lineal, o més freqüentment, per a la reducció de la dimensionalitat més tard o abans de la classificació.

LDA està estretament relacionat amb ANOVA (anàlisis de la variància) i l'anàlisi de la regressió, que també tracten d'expressar una variable dependent com a una combinació lineal d'altres característiques o d'amidaments. En els altres dos mètodes però, la variable dependent és una quantitat numèrica, mentre que per LDA és una categòrica variable (és a dir, la classe d'etiqueta).

LDA està també relacionada amb l'anàlisi de components principals (PCA) i l'anàlisi factorial en el qual ambdós busquen combinacions lineals de les variables que millor expliquen les dades. El model LDA explícitament intenta diferenciar les classes de dades. L'anàlisi discriminant és també diferent del factor d'anàlisis en el sentit que no és una tècnica d'interdependència: s'ha de fer una distinció entre les variables independents i dependents de les variables (també anomenades variables de criteri).

A nosaltres ens interessarà aplicar LDA per a dues classes, punts rígids i punts no rígids. Per tant, considerem en la possibilitat d'un conjunt de dades (errors de posició) per a cada mostra d'un punt amb una classe coneguda. Aquest conjunt de mostres s'anomena el conjunt de formació. Aleshores el problema és la classificació per trobar un bon predictor de la classe i de qualsevol mostra de la mateixa distribució (no necessàriament de la formació del conjunt) només d'una observació x.

LDA aborda el problema assumint que el condicional de la probabilitat són normalment distribuïts, en aquest supòsit, la solució òptima és predir els punts com de la segona classe, si la probabilitat és inferior a la ràtio d'un llindar T, de manera que sense cap mena d'hipòtesis, el classificador resultant es coneix com QDA (anàlisis discriminant quadràtic).

**Pseudocodi**

A continuació s'adjunta el pseudocodi d'aquest mètode, assumint que rep per entrada la llista d'errors de posició per cada un dels punts després de preparar les dades tal com s'ha explicat. Els altres dos paràmetres són dos paràmetres de probabilitat, P1 és la probabilitat que té un punt de ser de la classe 1 i P2 la probabilitat que té un punt de ser de la classe 2, lògicament la suma de les dos probabilitats hauria de ser igual a 1.

```
algorisme lda(llista_errors,P1,P2) // P1 i P2 probabilitat classe 1
i 2
  mitjana_all = mitjana(llista_errors)
  llindar = maxim(llista_errors)*0.99 //Agafem un valor lleugerament
                                      //inferior al més gran.
  Ratio_Maxim = 0
  decrementar = 0.001   //Valor  amb  el  que  anirem  reduirem  el
                        llindar, //quant més petit, més iteracions

  //Fem un recorregut fins que el llindar sigui 0
  mentre llindar>0 fer
    valors_c1 = trobar_valors_inferiors(llista_errors,llindar)
    valors_c2 = trobar_valors_superiors(llista_errors,llindar)

    //Calculem la mitjana de les dos classes>
    mitjana_c1 = mitjana(valors_classe1)
    mitjana_c2 = mitjana(valors_classe2)

    B    =    ((mitjana_c1-mitjana_all)^2)*P1    +    ((mitjana_c2-
mitjana_all)^2)*P2

    //Calculem les variàncies de les dos classes
    variancia_c1 = variancia(valors_c1)
    variancia_c2 = variancia(valors_c2)

    W = (P1*variancia_c1) + (P2*variancia_c2)

    Ratio_actual = B/W    //Calculem el ratio actual

    //Si el ratio actual es superior al màxim el guardem
    si Ratio_actual > Ratio_Maxim llavors
       Ratio_Maxim = Ratio_actual
    fsi

     llindar = llindar – decrementar
   fmentre
falgorisme
```

## 3.4 Avaluació dels mètodes amb dades sintètiques

En aquest capítol, posem a prova els diferents mètodes que hem desenvolupat amb dades sintètiques abans de posar-los a prova amb dades reals. D'aquesta manera podem avaluar els mètodes de segmentació on nosaltres tenim total control de l'entorn sintètic dissenyat. Aquestes proves sintètiques simulen un entorn real on a partir de dades que serien de diferents seqüències de parells d'imatges 2D d'un sistema estèreo amb els paràmetres de calibració de les càmeres i el sistema realitzem una reconstrucció 3D, obtenint així informació tridimensional sobre els diferents punts creats sintèticament. Per a fer-ho més real, com que a les dades reals ens trobarem amb soroll, aquí també hi apliquem diferents nivells de soroll. Aquest soroll ens generarà un desplaçament als diferents punts del conjunt de les dades sintètiques, el soroll serà de tipus gaussià amb nivells de 0, 0.5, 1, 1.5, 2.

**Objectes sintètics**

Els objectes sintètics que utilitzem, bàsicament és un cub el qual conté com a punts fixos els 8 vèrtexs. Llavors generem diferents dades sintètiques on hi generem més punts rígids i no rígids dintre i fora del cub. Per a què els punts no siguin rígids, lògicament, hem de generar una seqüència on els punts seran punts aleatoris que tindran deformacions també aleatòries.

És a dir aquesta seqüència es tractarà de donar-li una rotació i una translació al cub i a la resta dels punts, però als punts no rígids se'ls hi haurà de donar un desplaçament i/o rotació extra que com s'ha dit seran aleatoris per a diferenciar-los llavors dels punts que són realment rígids.

En total hem generat moltes més dades que en total són les següents:

- 10 punts rígids, 10 punts no rígids
- 10 punts rígids, 30 punts no rígids
- 10 punts rígids, 50 punts no rígids
- 10 punts rígids, 100 punts no rígids
- 30 punts rígids, 10 punts no rígids
- 30 punts rígids, 30 punts no rígids

- 30 punts rígids, 50 punts no rígids
- 30 punts rígids, 100 punts no rígids
- 50 punts rígids, 30 punts no rígids
- 50 punts rígids, 50 punts no rígids
- 50 punts rígids, 100 punts no rígids

Per a cada una d'aquestes dades, generem un total de 5 testos diferents. Aquests testos són diferents seqüències amb diferents rotacions i translacions per a les dades sintètiques creades amb el mateix soroll. A més a més, es realitzen aquests 5 testos per a 5 diferents tipus de soroll (0, 0.5, 1, 1.5 i 2 píxels). Amb tot aquest conjunt de dades generades disposarem també d'un enorme conjunt de resultats extrets a partir d'aquestes dades sintètiques. Així tindrem prou dades per a posar a prova els diferents mètodes de segmentació implementats.

Així doncs, en total tindrem 50 seqüències diferents per cada conjunts de punts. 5 testos per cada un dels 5 nivells de soroll que aquests 25 seran provats amb dos nivells de deformació dels punts no rígids (2 nivells de deformació x 5 nivells de soroll x 5 testos = 50 seqüències).

Les primera proves les realitzarem sobre un cub el qual té 10 punts rígids. Aquests punts rígids són els 8 vèrtexs més 2 punts rígids que es trobaran a dintre o al voltant del cub. Per aquests 10 punts variarem el nombre de punts no rígids que variaran en 10, 30, 50 i 100 punts. Com s'ha dit anteriorment, es realitzaran 50 seqüències per cada una de les combinacions dels 10 punts rígids amb la quantitat de punts no rígids comentats.

Com que 50 seqüències són moltes, nosaltres treballarem amb la mitjana d'error de les 5 seqüències amb el mateix soroll i mateix nivell de deformació.

Però abans de tot això explicarem com farem les diferents proves, per a això agafarem com exemple per explicar el procediment d'un cub amb els següents paràmetres: 10 punts rígids, 10 punts no rígids, nivell de soroll 0 i nivell de deformació elevat.

## Exemple 1: Cub amb 10 punts rígids i 10 de no rígids i sense soroll

A la figura 3.7 es poden veure tots els punts en projecció ortogràfica del primer frame de la seqüència. Els punts en color negre són els punts rígids i els punts en vermells són els no rígids que com veurem a les següents imatges tindran més desplaçament que els punts rígids al llarg de la seqüència.

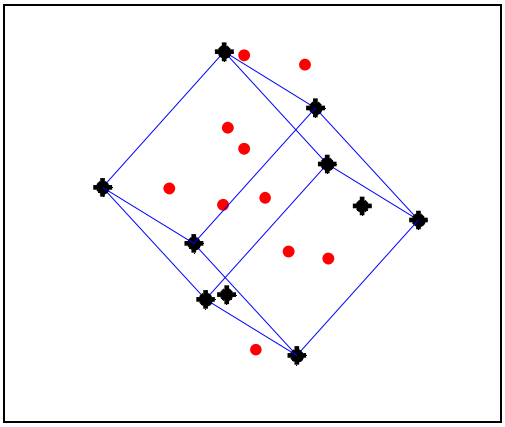


Figura 3.7. Frame inicial del cub amb 10 punts rígids i 10 punts no rígids.

Doncs agafem aquest frame com a punt de partida que ens servirà de referència al llarg de la seqüència al qual aplicarem Ransac a cada frame de la seqüència respecte aquest per tal de captura el model de la rotació i translació inversa i traslladar els punts a l'origen per finalment poder calcular l'error de posició. Recordem que en aquest exemple el soroll és 0 i per tant l'error de posició pels punts rígids ha de ser 0.

A continuació podem veure tota la seqüència a la figura 3.8. Aquesta seqüència l'hem posat tot en un sol frame per poder apreciar la rotació i translació de cada un dels punts. Si ens hi fixem podem veure que els punts rígids tenen una rotació i translació constants, en canvi els punts no rígids tenen un desplaçament extra respecte els rígids.

A mesura que es va generant la seqüència i anem tractant frame a frame, apliquem Ransac del frame a tractar respecte el frame original. Trobarem el model el qual aplicant la rotació i translació inversa els punts rígids haurien de trobar-se al mateix punt que en el primer frame. A més quan anem fent això

també anem guardant els errors de posició on en aquest exemple els punts rígids ha de ser 0.

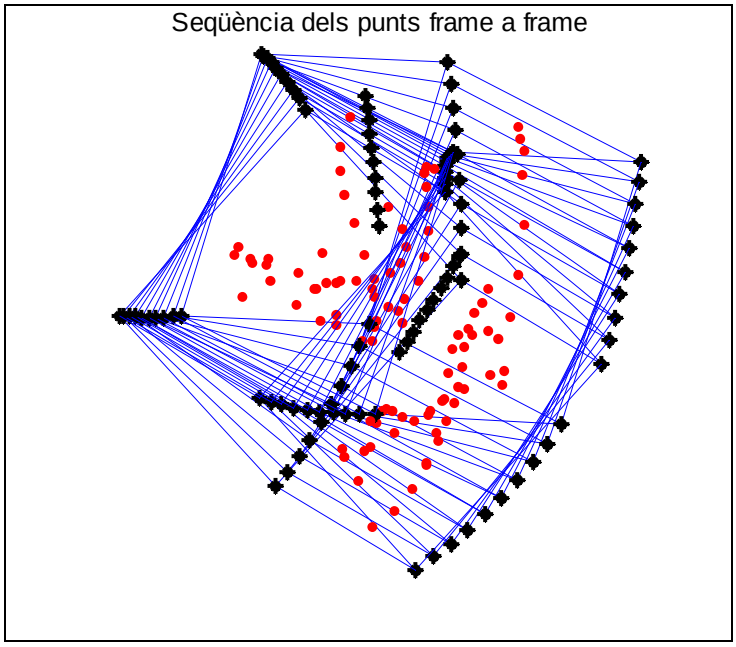



Figura 3.8. Aquesta gràfica representa tota la seqüència de 10 frames.

A continuació s'adjunta la figura 3.9 que representa tots els punts després d'aplicar el model trobat per Ransac.

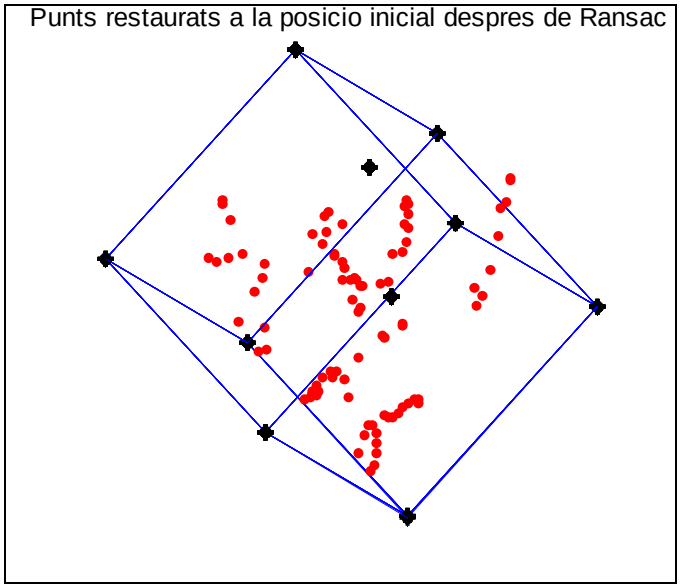



Figura 3.9. Punts després d'aplicar la matriu inversa trobada amb Ransac.

Doncs, aquí a la figura 3.9 després d'aplicar el model de rotació i translació trobat per Ransac podem veure que els punts negres sembla que només n'hi

hagin 10, però en realitat hi son tots els que apareixien a la matriu anterior, el que passa que com s'ha dit anteriorment l'error al no haver-hi soroll a la seqüència és 0. En canvi els punts vermells es pot veure que tenen un desplaçament (error de posició). Com que a mesura que hem anat tractant frame a frame hem anat guardant l'error del frame que es tractava respecte l'original, hem pogut generar tot un conjunt de dades per cada punt en cada frame donat on s'hi reflexa aquest error de posició, l'error que hi ha després d'aplicar el model trobat per Ransac i restaurant els punts sobre el frame inicial. Aquest error és el que ens servirà per segmentar.

Aquest error de posició és la suma de l'error de cada un dels frames respecte l'origen. S'ha triat fer aquest sumatori en comptes de la mitjana d'error per el motiu que a l'hora de segmentar les diferències d'errors són més elevades i la segmentació és més fàcil i fiable d'aquesta manera que no pas de l'altre. Això és degut a que la diferència dels errors entre els punts rígids i els punts no rígids seria molt menor i podrien haver-hi forces més ambigüitats.

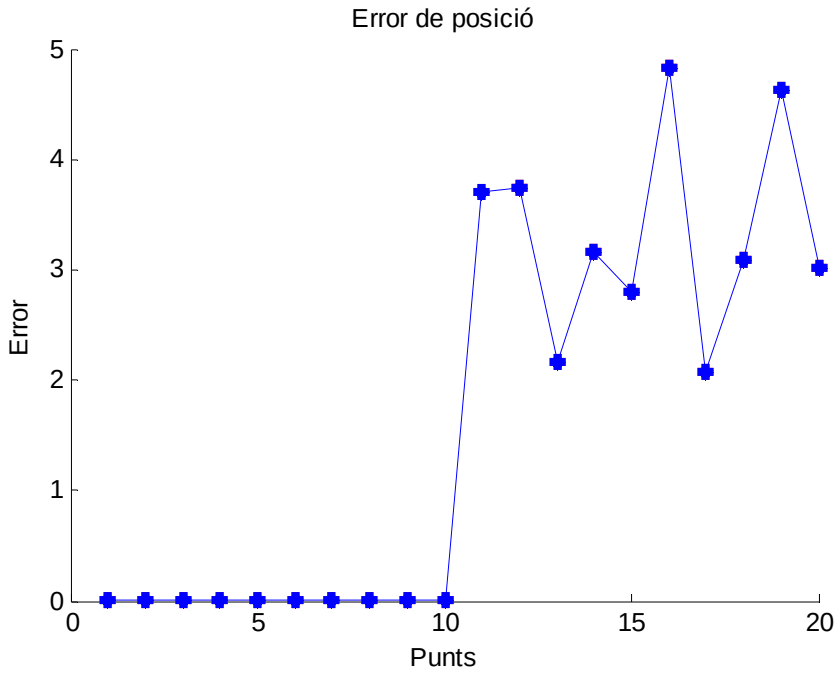



Figura 3.10. Gràfica que representa el sumatori d'errors de tots els punts al llarg de la seqüència

La figura 3.10 ens mostra aquest error de posició per a tots els punts. Com es pot observar entre el punt 0 i el punt 10 són els punts rígids, i al no haver-hi

soroll l'error és exactament, gràcies a la precisió del mètode Ransac. En canvi dels punts que van del 11 al 20 són no rígids i l'error que es representava a la figura 3 després d'haver estat restaurats és força elevat i va variant segons el punt, ja que no tots els punts no rígids pateixen la mateixa deformació.

Cal destacar que en aquests exemples utilitzem com a punts rígids tots els punts que van del 1 fins al número de punts rígids, per al motiu que d'aquesta manera és més fàcil poder il·lustrar el procés de segmentació, però que no té cap influència sobre el resultat final de segmentació ja que l'ordre no es té en compte sinó els valors de l'error independentment de l'ordre que es trobin.

Ara amb aquestes dades que tenim, ja podem procedir a aplicar els mètodes de segmentació que hem estat estudiant on els avaluarem, discutirem i en traurem conclusions sobre els resultats que ens donguin.

**Aplicació mètode segmentació Otsu**

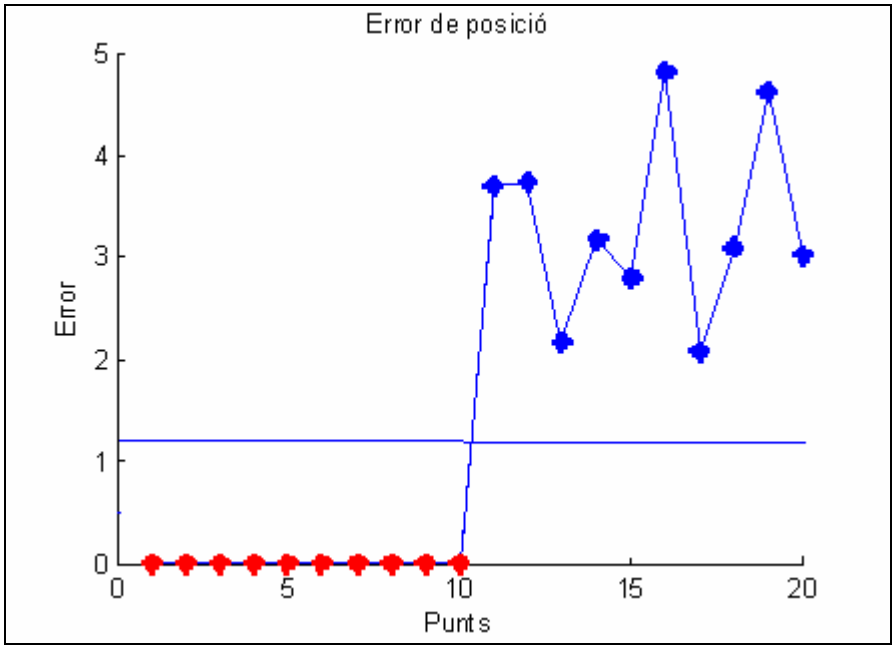



Figura 3.11. Segmentació utilitzant el mètode Otsu

Aquí hem posat a provar el mètode Otsu. Com es pot observar a la figura 3.11 no tenim cap error de segmentació, la segmentació s'ha fet correcta al 100%. Al no haver-hi soroll, i haver trobat error de posició 0 per als punts rígids ha facilitat molt trobar el llindar òptim per a que no succeís cap error, a més que

també ha facilitat que el nombre de punts rígids i el nombre de punts no rígids sigui exactament el mateix.

**Aplicació mètode segmentació K-Means**

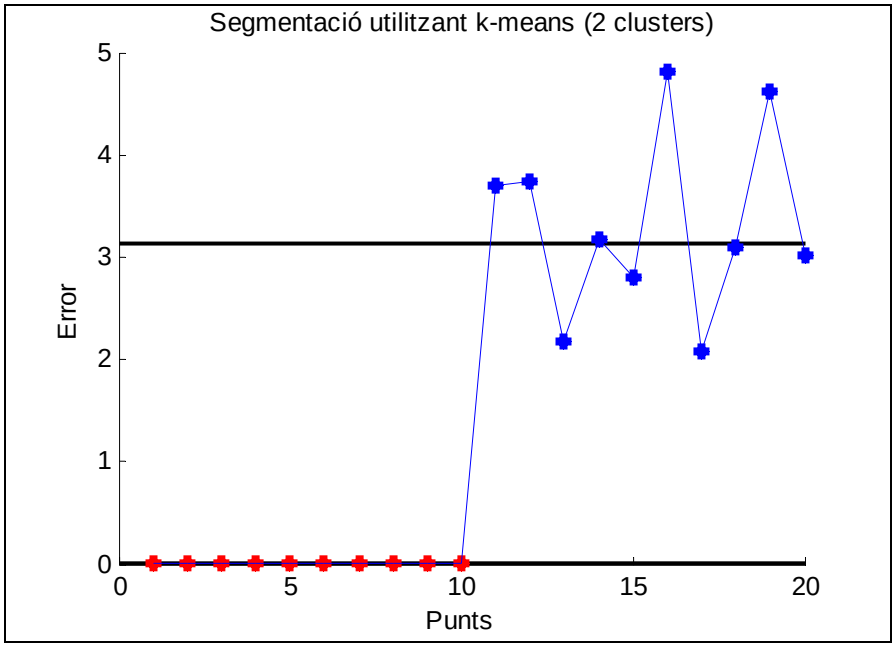



Figura 3.12. Segmentació utilitzant el mètode K-Means.

A la figura 3.12 podem veure la segmentació utilitzant el mètode K-Means amb dos clústers (grups), un grup és pels punts rígids, i l'altre pels punts no rígids. A diferència del mètode anterior, aquí no tenim un llindar. Aquí tenim dos "centroids" que és el resultat de fer la mitjana de tots els valors de cada clúster, llavors els punts formaran part del clúster que tinguin més proper. A sobre l'eix de les abscisses hi tenim el centroid del clúster dels punts rígids, que és 0, i aproximadament a l'error 3.1 hi tenim l'altre centroid per als punts no rígids.
Aquí igual que en el mètode anterior no hem tingut cap error de segmentació gràcies als mateixos factors que en l'anterior: no hi ha soroll i que hi hagi la mateixa quantitat de punts rígids que de no rígids.

**Aplicació mètode segmentació LDA**

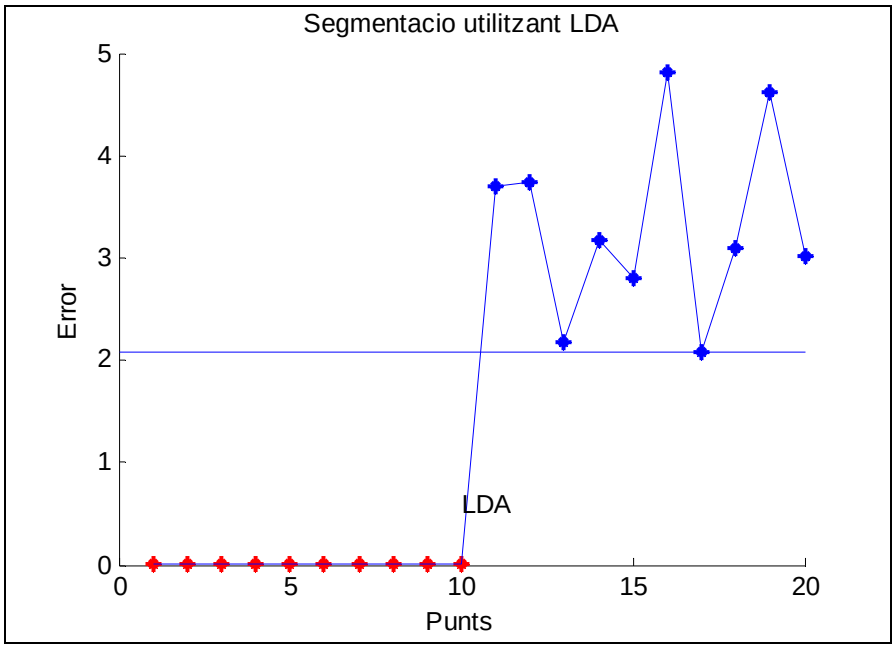



Figura 3.13. Segmentació utilitzant el mètode LDA

## Exemple 2: Cub amb 10 punts rígids, 10 punts no rígids, poca deformació i nivell de soroll 2

L'exemple 2 conté el mateix nombre de punts rígids com de no rígids, el que varia aquí és la deformació dels punts no rígids que és menor i que el nivell de soroll és de 2 píxels.

Aquest exemple és més real degut a l'aparició de soroll, ja que a la pràctica les càmeres no estan lliures de soroll i sempre ens n'hi trobarem, això ens complicarà les coses alhora de segmentar i ens apareixeran errors de segmentació on fàcilment ens poden aparèixer punts no rígids trobat com a rígids i viceversa, on també això dependrà del mètode de segmentació que aplicarem, ho veurem més endavant.

Comencem com en l'exemple anterior mostrant el frame inicial de la seqüència el qual ens basarem per aplicar Ransac i aconseguir el millor model possible. Aquest model al haver-hi soroll veurem que no està lliure d'error de posició.

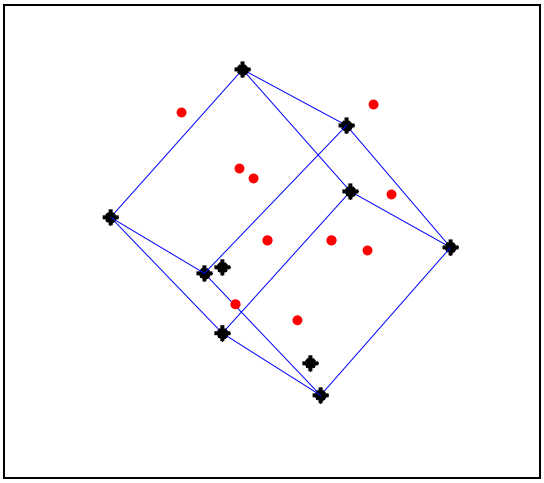


Figura 3.14. Frame inicial de la seqüència de l'exemple 2.

A la figura 3.14, com en l'exemple anterior, hi trobem punts aleatoris dintre i fora el cub, 2 d'aquests punts que no són els vèrtexs també són rígids com aquests últims. Llavors els punts vermells són els punts que sofriran deformacions/desplaçament extra.

A la figura 3.15, veiem tota la seqüència en el mateix gràfic, aquí hi podem notar el soroll lleugerament que s'hi ha afegit però si que podrem veure una mica millor la poca deformació que tenen els punts no rígids respecte els de la figura 2 de l'exemple anterior, on hi havia més nivell de deformació.

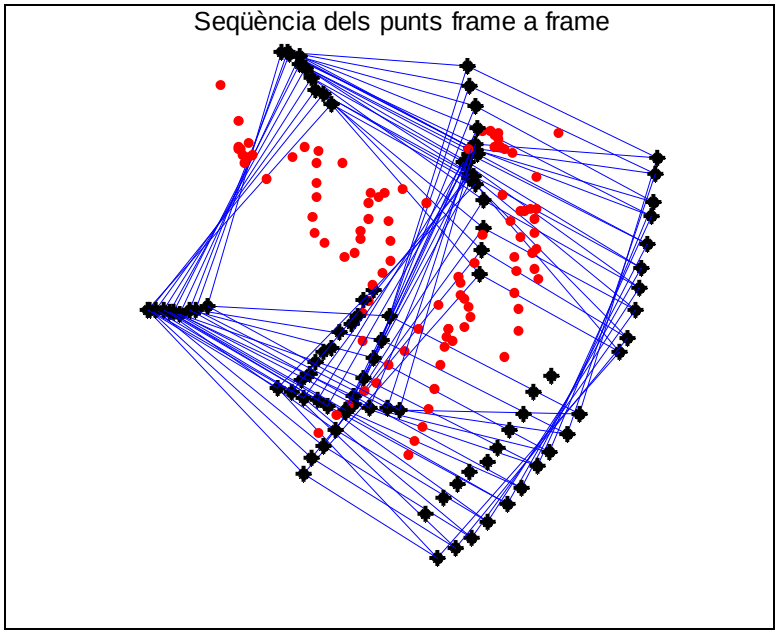



Figura 3.15. Gràfica on es mostra tota la seqüència de 10 frames de l'exemple 2.

Si frame a frame apliquem, com hem fet a l'exemple anterior, Ransac per a calcular el model tenint com a inliers els punts rígids, al haver-hi soroll veurem que el model que ens retorna Ransac conté forces errors, a més és possible degut al soroll que Ransac hagi considerat algun outlier com a inlier tot i utilitzar un llindar baix per a aconseguir el millor model possible. A més si provéssim de tornar a executar Ransac alguns cops més podríem tenir fàcilment resultats diferents, ja que com s'ha explicat en el tema de Ransac, aquest utilitza dades aleatòries de tot el conjunt de punts per a arribar a un model acceptable per això és un algorisme no determinista.

Després d'aconseguir el model gràcies a Ransac ja podem aplicar-hi la rotació i translació inversa per a situar-lo sobre el primer frame i extreure'n l'error de posició que uneix els punts corresponents entre el primer frame i el frame a tractar.

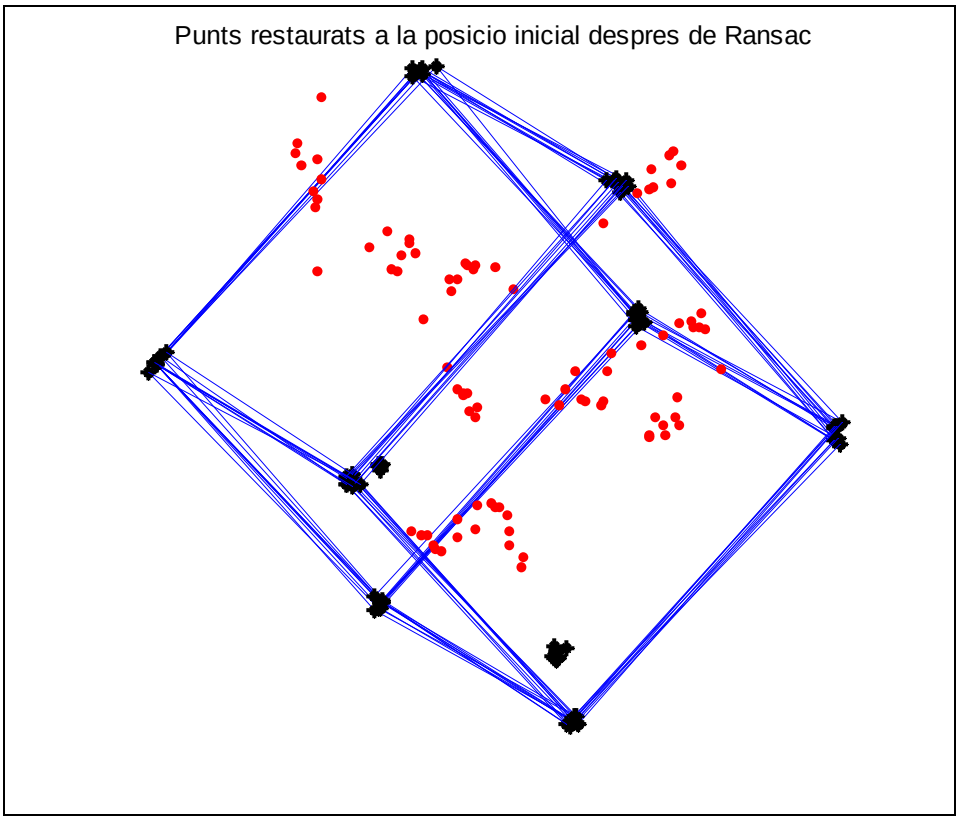



Figura 3.16. Gràfica on apareixen tots els frames restaurats al frame original després d'aplicar la rotació i translació inversa calculada pel mètode Ransac.

A la figura 3.16 es pot veure com per culpa del soroll Ransac no ha pogut calcular el model lliure d'error pels punts rígids, hi ha un cert desplaçament entre els punts rígids corresponents. Els punts no rígids també s'hi veuen afectats, a més per això podem trobar-nos més fàcilment amb el problema de confondre alguns punts rígids i punts no rígids ja que poden tenir errors de posició força semblants entre ambdós grups de punts.

Segons avançava la seqüència al anar guardant els errors de posició després d'aplicar la rotació i translació inversa podem generar la gràfica que ens mostra la figura 3.17 que ens indica l'error de posició

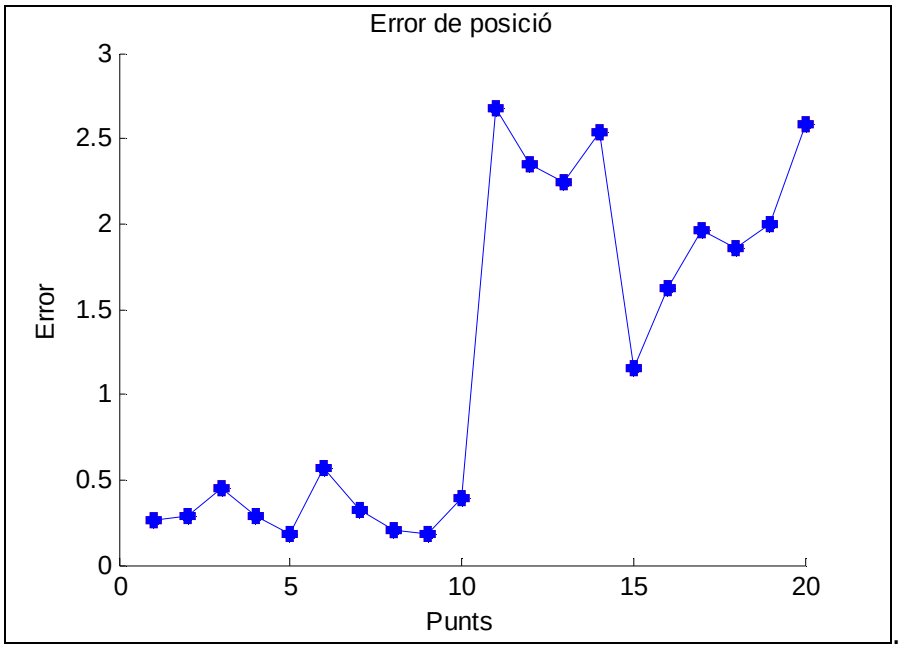



Figura 3.17. Sumatori de l'error de posició per a cada un dels punts de l'exemple 2.

Tal com es deia abans, aquí podem veure respecte la figura 3.7 de l'exemple 1 que l'error dels punts rígids no és 0, tot i que queda encara força separat de l'error dels punts no rígids.

Això ens dificultarà la segmentació i veurem que alguns mètodes són més bons que els altres alhora de segmentar aquests resultats, tot i que amb la mateixa quantitat de punts rígids que de no rígids l'error de segmentació (equivocar-se en dir que un punt no rígids és rígid) no és gaire alt, més endavant veurem també que passa quan hi ha més punts rígids que de no rígids i viceversa.

**Aplicació mètode segmentació Otsu**

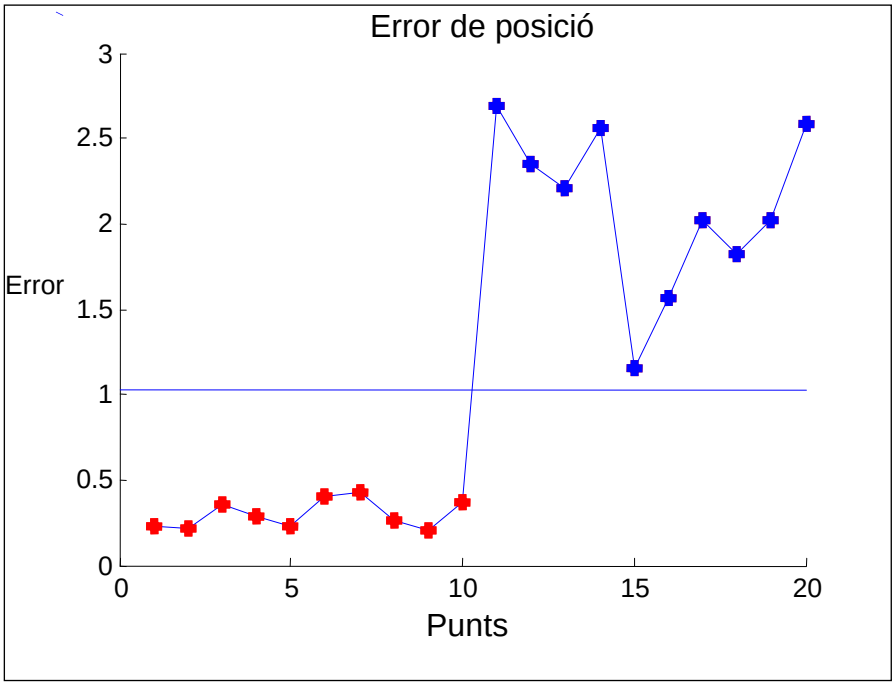



Figura 3.18. Segmentació de l'exemple 2 utilitzant el mètode Otsu

En aquest cas com que el nivell de deformació és menor i el soroll és més elevat el càlcul del llindar òptim utilitzant el mètode Otsu basat en el mètode Otsu, ha realitzat una segmentació 100% correcta, tot i que el llindar és més just que en el cas de no haver-hi soroll i on hi havia més deformació. Probablement si afegíssim encara més soroll, menys deformació o afegir més punts rígids o no rígids (eliminant així la igualtat en nombre de punts per a les dos classes) podríem tenir problemes en la segmentació i tenir-ne errors. Més endavant veurem que passa quan variem aquests paràmetres.

**Aplicació mètode segmentació K-Means**

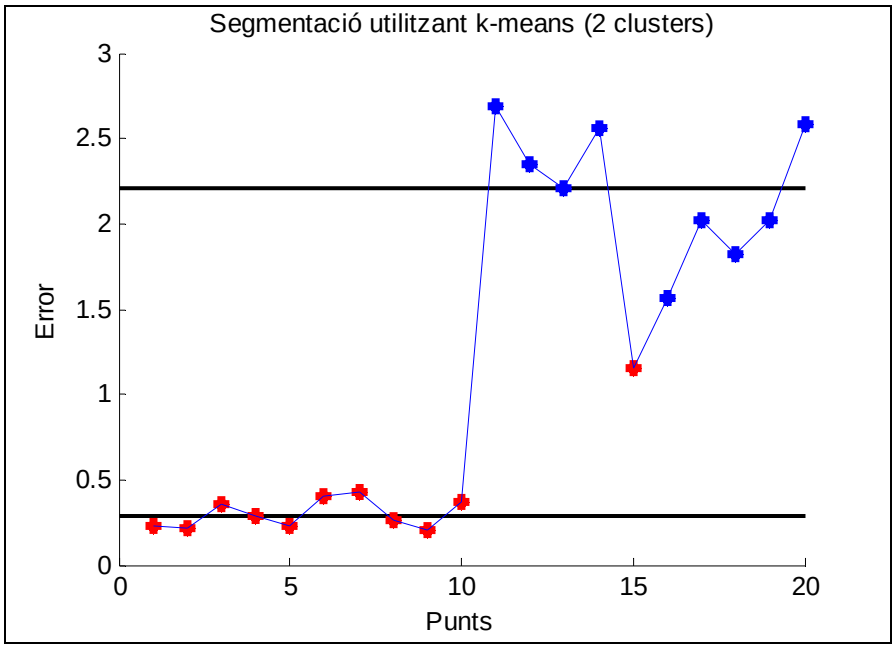



Figura 3.19. Segmentació de l'exemple 2 utilitzant el mètode K-Means.

A la figura 3.19 podem veure que passa quan en l'exemple 2 hi apliquem la segmentació utilitzant el mètode K-Means amb dos classes. D'entrada sembla bona la segmentació, però aquí ja hi tenim un error: el punt 15. Pels mateixos motius que es comentaven abans, l'error del punt 15 està més proper, per molt poc, del centroid dels punts rígids que dels punts no rígids. Per tant el considerem un error de segmentació.

**Aplicació mètode segmentació LDA**

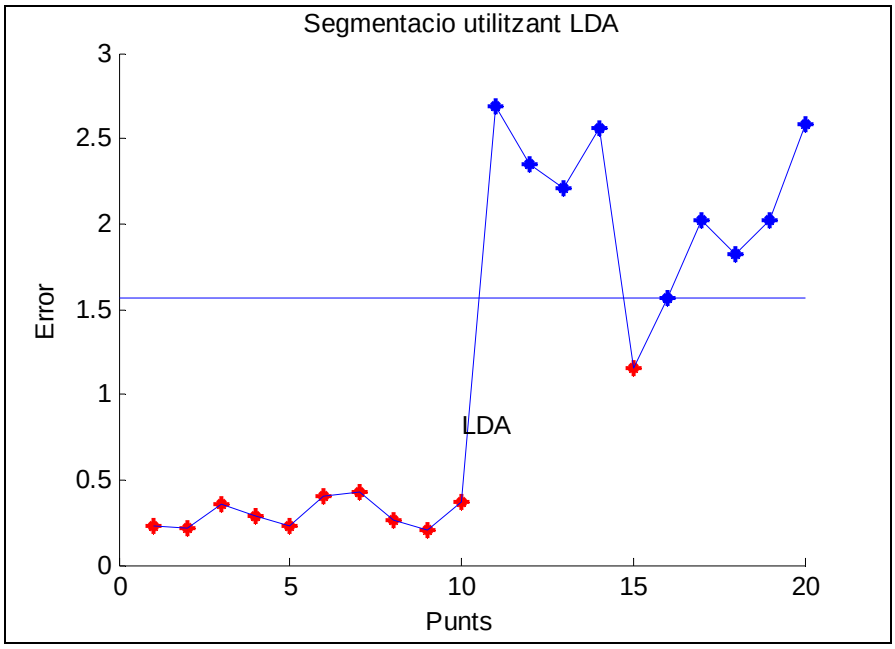



Figura 3.20. Segmentació de l'exemple 2 utilitzant el mètode LDA.

**Què passa si desordenem els punts ?**

En tots els casos d'aquestes dades sintètiques que hem creat, apareixen els punts ordenats, primer els punts rígids i llavors els punts no rígids. Per exemple en el cas que hem mostrat anteriorment, els 10 punts rígids són els punts que van del 1 al 10 i els punts no rígids els que van del punt 11 al punt 20. Aleshores què passa si desordenem aquests punts? Variarà el resultat de segmentació per als 3 mètodes?

La resposta la mostrem a continuació mostrant els 3 gràfics de segmentació de l'exemple anterior però mostrant els punts desordenats:

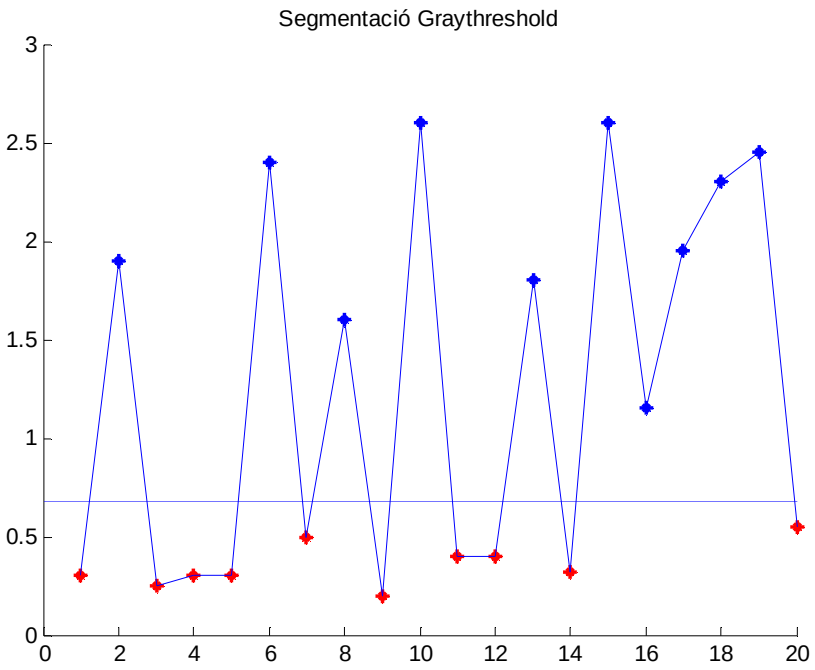



Figura 3.21. Segmentació utilitzant el mètode Otsu després de desordenar tots els punts.

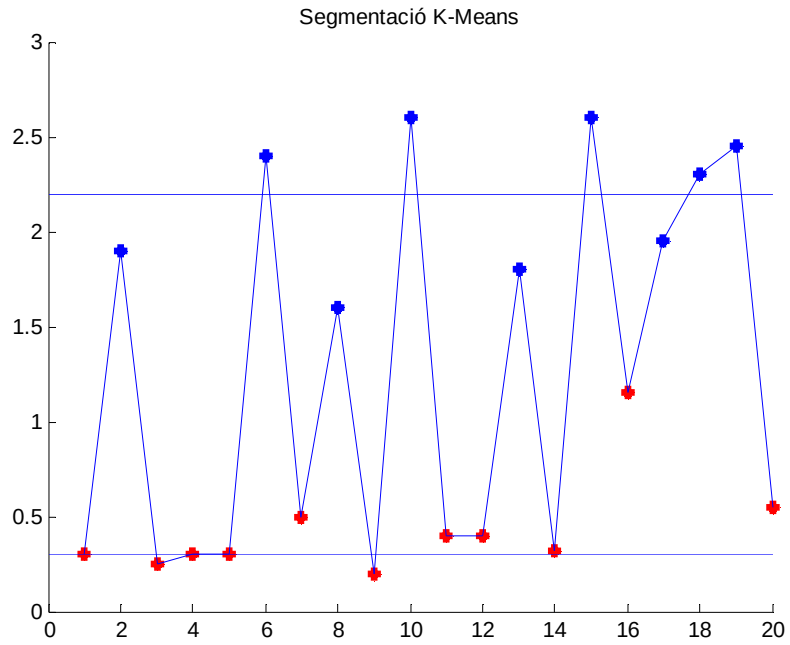



Figura 3.22. Segmentació utilitzant K-means després de desordenar aleatòriament tots els punts.

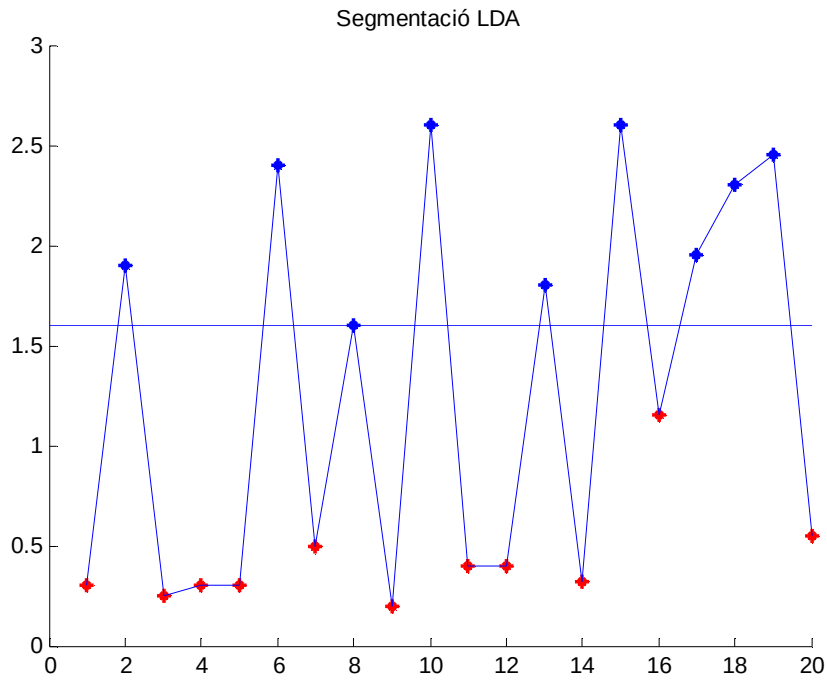



Figura 3.23. Segmentació utilitzant LDA després de desordenar aleatòriament tots els punts

Com podem veure, l'ordre dels punts no té res a veure alhora de la segmentació, sempre obtenim el mateix llindar de segmentació (mateixos centroids en el cas de la segmentació K-means). Per tant, no importa l'ordre amb el que es distribueixen els errors dels punts, ja que sempre obtindrem el mateix resultat de segmentació.

### 3.4.1 Resultats

Després d'explicar els procediments per a realitzar la segmentació i comprovar l'error de segmentació posarem a prova tots els experiments plantejats. A partir d'ara ja no mostrarem pas a pas el que fem per a cada una de les proves, que són moltes, sinó que en una taula mostrarem els diferents errors de segmentació per a cada un dels mètodes jugant amb els diferents paràmetres explicats.

Considerarem un error de segmentació, un punt que es consideri rígid quan en realitat és un punt no rígid. L'objectiu és trobar un bon conjunt de punts rígids. Estar segur de que ho són per ajudar a mètodes que permeten obtenir models deformables. Aquí en les proves sintètiques això ja ho sabem a priori, però quan provem en dades reals, tindrem el problema que no sabrem a priori quins són els punts no rígids els quals s'han considerat rígids, i aleshores serien errors de segmentació, però això ho veurem en més detall en el seu apartat corresponent.

No considerarem com a un error de segmentació en aquest projecte un punt rígid el qual s'hagi considerat no rígid, perquè bàsicament el que intentem és aconseguir el nombre màxim de punts rígids reals possibles sense equivocació. A continuació veurem que passa quan variem el nombre de punts (rígids i no rígids), variem el soroll i variem la deformació per a cada un dels mètodes de segmentació que estem estudiant.
Els valors que es donen és la mitjana d'error de segmentació de l'execució dels 5 testos que tenen cada un dels conjunt de paràmetres. Per exemple: en el cas que tenim 10 punts rígids i 30 punts no rígids amb deformació K3 i amb un nivell de soroll de 1, s'executen 5 testos diferents que contenen diferents translacions i rotacions junt amb diferents deformacions, doncs amb aquests 5 testos en fem la mitjana dels errors de segmentació que aquesta mitjana en serà el valor que mostrarem a les següents taules de resultats.

## 3.4.2 Avaluació sintètica: Resultats Otsu

| | | | Soroll | | | | |
|---|---|---|---|---|---|---|---|
| | | **Deformació** | **0** | **0.5** | **1** | **1.5** | **2** |
| **10 rígids** | **10 no rígids** | **K3** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | **K5** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | **30 no rígids** | **K3** | 0.20 | 0.20 | 0.20 | 0.20 | 0.40 |
| | | **K5** | 0.00 | 0.00 | 0.20 | 0.00 | 0.00 |
| | **50 no rígids** | **K3** | 0.40 | 0.60 | 1.00 | 1.80 | 1.80 |
| | | **K5** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **30 rígids** | **10 no rígids** | **K3** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | **K5** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | **30 no rígids** | **K3** | 1.20 | 1.20 | 1.60 | 1.60 | 1.40 |
| | | **K5** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | **50 no rígids** | **K3** | 1.00 | 1.20 | 3.20 | 2.40 | 4.20 |
| | | **K5** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **50 rígids** | **10 no rígids** | **K3** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | **K5** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | **30 no rígids** | **K3** | 0.80 | 1.00 | 0.80 | 1.00 | 1.40 |
| | | **K5** | 0.00 | 0.00 | 0.00 | 0.20 | 0.40 |
| | **50 no rígids** | **K3** | 1.40 | 2.20 | 2.00 | 4.60 | 3.60 |
| | | **K5** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

Aquest mètode de segmentació Otsu ens ha donat molt bons resultats, quan hem tingut força deformació (K5) els errors han sigut pràcticament nuls, en excepció de quan tenim 50 punts rígids i 30 no rígids i amb soroll 1.5 i 2 on l'error de segmentació és tan sols de 0.20 i 0.40. Per la resta amb aquest grau de deformació no ens hem equivocat, per tant podem dir que quan existeix força grau de deformació aquest mètode de segmentació ens dóna molt bons resultats.

Llavors quan hi ha menys grau de deformació (K3) la segmentació s'equivoca més, això passa degut a que l'error de posició després d'aplicar Ransac i restaurar els punts a l'origen després de trobar el model que ens ha retornat aquest error és més petit i aleshores al haver-hi menys marge d'error entre els punts rígids i els punts no rígids en el procés de segmentació tenim moltes més possibilitats d'equivocar-nos. Els pitjors resultats els tenim quan tenim 50 punts rígids i 50 punts no rígids, sembla doncs, que al augmentar el nombre de punts tant rígids com no rígids ens complica les coses, ja que si ens

fixem en quan tenim 10 punts rígids i 10 punts no rígids amb grau de deformació baix (K3) no hi tenim cap error, en canvi quan tenim 30 punts rígids i 30 punts no rígids això ja no passa, aquí ja ha aparegut error, i si finalment mirem quan tenim 50 punts rígids i 50 punts no rígids, les coses encara empitjoren.
En quant al soroll, a mesura que l'augmentem les coses també empitjoren lògicament, sobretot quan el grau de deformació és baix, això passa ja que si amb poca deformació ja tenim errors, a mesura que augmentem aquest soroll farà que els errors de posició siguin més inestables per als punts rígids i no rígids, i per la regla de tres diem que els errors de segmentació augmentaran, tal i com es pot veure en la taula.

Definitivament, podríem deduir que aquest algorisme ens dóna molt bons resultats quan tenim un grau de deformació alt i més errors quan aquest grau és més baix, en canvi a mesura que disminuïm el grau de deformació i n'augmentem també el soroll les coses es compliquen.

### 3.4.3 Avaluació sintètica: Resultats K-Means

Implementem el mètodes K-means utilitzant la inicialització dels clusters aleatòriament i en total realitzant 3 testos diferents.

| | | | Soroll | | | | |
|---|---|---|---|---|---|---|---|
| | | **Deformació** | **0** | **0.5** | **1** | **1.5** | **2** |
| **10 rígids** | **10 no rígids** | **K3** | 0.80 | 1.40 | 0.80 | 1.40 | 1.40 |
| | | **K5** | 0.80 | 0.80 | 0.80 | 1.00 | 0.80 |
| | **30 no rígids** | **K3** | 4.00 | 4.60 | 2.60 | 4.60 | 5.80 |
| | | **K5** | 5.00 | 4.20 | 5.20 | 0.00 | 0.00 |
| | **50 no rígids** | **K3** | 8.20 | 12.60 | 9.40 | 12.00 | 13.20 |
| | | **K5** | 12.00 | 13.00 | 15.40 | 13.00 | 12.20 |
| **30 rígids** | **10 no rígids** | **K3** | 0.60 | 0.60 | 0.60 | 1.00 | 0.60 |
| | | **K5** | 0.20 | 0.20 | 0.20 | 0.20 | 0.20 |
| | **30 no rígids** | **K3** | 2.80 | 3.40 | 3.20 | 3.60 | 3.60 |
| | | **K5** | 1.00 | 1.00 | 1.60 | 1.40 | 1.80 |
| | **50 no rígids** | **K3** | 1.80 | 2.40 | 2.80 | 3.20 | 6.20 |
| | | **K5** | 2.00 | 2.40 | 3.40 | 3.40 | 4.60 |
| **50 rígids** | **10 no rígids** | **K3** | 0.20 | 0.60 | 0.40 | 0.20 | 1.20 |
| | | **K5** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | **30 no rígids** | **K3** | 0.80 | 1.40 | 1.40 | 1.40 | 2.20 |
| | | **K5** | 0.00 | 0.00 | 0.20 | 0.00 | 0.00 |
| | **50 no rígids** | **K3** | 1.80 | 2.20 | 3.00 | 4.00 | 4.20 |
| | | **K5** | 1.40 | 1.40 | 1.80 | 2.20 | 2.80 |

Després d'implementar el mètode de segmentació K-means per a tot el conjunt de dades hem obtingut aquests resultats. Aquests resultats a simple vista es veuen molt pitjors que el mètode anterior. Aquí veiem que els errors són molt més elevats que en el cas anterior. On hi tenim més errors de segmentació és quan tenim 10 punts rígids i 50 punts no rígids i quan tenim més deformació que quan en tenim menys. En aquest cas l'explicació seria la següent: en els 10 punts rígids l'error de posició és idèntic o semblant quan hi ha més soroll entre aquests, en canvi com hem vist en gràfiques anteriors, els punts no rígids l'error és molt variable i per tant entre un error de posició d'un punt no rígid i un punt rígid pot variar molt. Aleshores com que aquest algorisme busca les millors mitjanes per separar dos classes, al ser tan irregular aquest error, molts punts no rígids s'acosten més al clúster dels punts rígids que el clúster dels punts no rígids.

Això es pot comprovar en els resultats de quan tenim 50 punts rígids i 10 punts no rígids, es pot veure que tenim molts menys errors de segmentació

comparant-ho amb l'altre cas degut a això mateix que s'ha explicat, els punts rígids tenen un error de posició molt semblant, i al haver menys punts no rígids, el clúster de la classe dels no rígids es trobarà molt més proper d'aquest conjunt de punts. L'error de segmentació quan tenim 50 punts rígids i 10 punts no rígids és nul encara que hi tinguem molt de soroll quan hi existeix força deformació (deformació = K5), en canvi si hi tenim menys deformació si que ja hi apareixen errors de segmentació.
Per la resta de casos succeeix el mateix, tot i que no hi ha tant de contrast com aquest cas que hem explicat.

En quant al soroll veiem que lògicament també augmenten els errors de segmentació tot i que tampoc augmenten excessivament fins hi tot en algun cas a mesura que augmentem el soroll milloren els resultats de segmentació (10 punts rígids, 30 no rígids).

Per tant, aquest mètode de segmentació podríem deduir que ens seria útil quan tenim molts punts rígids i pocs punts no rígids, i quant més deformació menys possibilitats d'equivocar-nos tindríem. En casos que el nombre de punts rígids i no rígids sigui semblant, no seria molt aconsellable utilitzar-lo, seria més útil utilitzar el mètode anterior.

### 3.4.4 Resultats LDA

| | | | Soroll | | | | |
|---|---|---|---|---|---|---|---|
| | | **Deformació** | **0** | **0.5** | **1** | **1.5** | **2** |
| **10 rígids** | **10 no rígids** | **K3** | 2.20 | 2.20 | 2.20 | 2.20 | 2.20 |
| | | **K5** | 2.40 | 2.80 | 2.60 | 2.60 | 2.80 |
| | **30 no rígids** | **K3** | 5.80 | 6.0 | 5.80 | 11.80 | 11.80 |
| | | **K5** | 6.60 | 6.80 | 6.60 | 7.00 | 7.00 |
| | **50 no rígids** | **K3** | 0.00 | 0.00 | 0.00 | 1.00 | 10.40 |
| | | **K5** | 0.00 | 0.00 | 0.80 | 0.20 | 0.80 |
| **30 rígids** | **10 no rígids** | **K3** | 3.40 | 3.40 | 3.80 | 4.40 | 4.60 |
| | | **K5** | 6.00 | 6.00 | 7.00 | 6.00 | 6.00 |
| | **30 no rígids** | **K3** | 3.00 | 8.20 | 9.80 | 10.40 | 19.00 |
| | | **K5** | 6.40 | 6.40 | 6.80 | 15.80 | 16.00 |
| | **50 no rígids** | **K3** | 8.20 | 8.40 | 9.20 | 10.40 | 21.20 |
| | | **K5** | 1.00 | 1.20 | 1.40 | 1.60 | 20.00 |
| **50 rígids** | **10 no rígids** | **K3** | 4.40 | 4.40 | 4.20 | 4.60 | 4.40 |
| | | **K5** | 3.80 | 3.80 | 3.80 | 3.80 | 3.60 |
| | **30 no rígids** | **K3** | 8.20 | 8.40 | 12.40 | 14.80 | 20.40 |
| | | **K5** | 14.00 | 14.60 | 14.20 | 14.40 | 14.20 |
| | **50 no rígids** | **K3** | 10.60 | 11.40 | 12.20 | 30.00 | 30.20 |
| | | **K5** | 10.60 | 10.60 | 30.20 | 30.80 | 31.00 |

Aquest últim mètode de segmentació sembla que és el que pitjors resultats ha donat. Veiem que és molt més sensible al soroll que els altres mètodes, a mesura que l'augmentem l'error de segmentació puja proporcionalment. A on tenim els millors resultats és quan tenim 10 punts rígids i 50 punts no rígids encara que l'error de segmentació puja lleugerament en aquest cas a mesura que augmentem el soroll. Cal destacar que aquest mètode juga amb probabilitats i que les probabilitats d'entrada per a aquest mètode s'han considerat com a 0.5 de probabilitat per a la classe 1 (punts rígids) i 0.5 de probabilitat per a la classe 2. Encara que hi hagi el mateix nombre de punts rígids que de punts no rígids, els resultats no es que siguin molt bons. En aquests experiments sintètics podríem ajustar millor les probabilitats, però com que amb les dades reals a priori no sabríem la proporció de punts, doncs aquí juguem amb aquestes probabilitats.

Potser que aquest mètode en realitat no tingui una bona aplicació en la segmentació de punts rígids i punts no rígids en el món de la reconstrucció 3D, per tant diríem que és el pitjor mètode dels tres degut als mals resultats que ens ha donat.

# Capítol 4

# Obtenció de dades reals

## 4.1 Introducció

En aquest capítol veurem el comportament dels tres mètodes que hem explicat en el capítol anterior sobre dades reals. A més a més, també hi podrem veure les bases teòriques que hem estudiat posades en pràctica.

Primer de tot en aquest capítol veurem com calibrem el sistema estèreo començant calibrant individualment les càmeres (estimem els paràmetres intrínsecs) per a realitzar finalment la calibració estèreo del sistema (paràmetres extrínsecs). Amb una bona calibració ja estarem llestos per fer captures reals per a treballar amb les dades que extraurem utilitzant un software que ens servirà per a realitzar el procés de Matching & Tracking (correspondència i seguiments de punts 3D) en una seqüència d'imatges que seran capturades amb el sistema estèreo.

Per a realitzar el procés de captura posarem a gravar una seqüència de vídeo les dues càmeres. Amb els fitxers de vídeo que haurem generat llavors els passarem a seqüències d'imatges, per això, amb l'ajuda d'un làser que haurem posat a l'inici de la gravació el qual serà apagat ens ajudarà a realitzar la sincronització entre les dues càmeres, és a dir, els frames per a cada càmera han d'estar capturats al mateix instant de temps, d'aquesta manera el frame inicial i la resta per ambdues seqüències seran exactament en el mateix instant de temps.

Amb les corresponents seqüències d'imatges per a cada càmera llavors utilitzarem el software creat per en Jordi Ferrer Plana (Laboratori de Visió per Computador de la Universitat de Girona) per a realitzar el procés de Matching & Tracking que ens retornarà un conjunt de punts 3D que apareixen al llarg de tota la seqüència que vulguem tractar. Amb aquests punts 3D podrem generar també tot el procés de segmentació tal i com hem fet al capítol anterior. D'aquesta manera podrem provar sobre dades reals els tres mètodes que hem estudiat i

desenvolupat per a la segmentació de punts 3D en una seqüència amb el sistema correctament calibrat.

## 4.2 Calibració del sistema

A continuació s'explicarà com s'ha realitzat el procés de calibració del sistema estèreo. La calibració d'una càmera, tal com s'ha explicat en capítols anteriors, és el procés el qual determina els paràmetres intrínsecs i extrínsecs de la càmera. Per a calibrar un sistema estèreo, cal calibrar individualment cada una de les dues càmeres per determinar, primer de tot, els paràmetres de la geometria interna de la càmera (paràmetres intrínsecs). Llavors una vegada calibrades les càmeres individualment coneixent els seus paràmetres intrínsecs, ja podrem procedir a determinar els paràmetres extrínsecs del sistema estèreo. Cal destacar que el sistema està muntat i que les càmeres estaran en una posició fixa durant tot el procés de calibració, és a dir, les càmeres romandran immòbils durant tot el procés i lògicament durant el realitzament de les captures necessàries que es faran més endavant. El motiu, és que una calibració només és vàlida quan les càmeres estan a una posició constant l'una de l'altre, tan en rotació com en la distància entre les dos a més de tenir els paràmetres intrínsecs invariables durant dits processos.

Els paràmetres extrínsecs del sistema estèreo bàsicament són la rotació i translació que té una càmera respecte l'altre. A continuació es profunditzarà més cada un dels passos del procés de calibració el qual serà realitzat amb la ***Camera Calibration Toolbox for Matlab*** [1].

Figura 4.1. Sistema estèreo utilitzat amb un parell de càmeres Unibrain Fire-i amb resolució 640x480 píxels.

### 4.2.1 Entorn de treball

Tal com s'acaba d'enunciar, per a calibrar les càmeres utilitzarem una Toolbox de Matlab anomenada "Camera Calibration Toolbox for Matlab", aquesta eina creada per *Jean-Yves Bouguet* és una gran utilitat per a realitzar calibracions precises de càmeres i de sistemes estèreo. Disposa d'una gran quantitat de funcions per a determinar tots els paràmetres necessaris per a obtenir els paràmetres intrínsecs i extrínsecs d'una càmera, més endavant, aquesta eina també ens serà útil per determinar els paràmetres extrínsecs del sistema estèreo que tenim muntat.

A la següent secció veurem com es calibra una individualment una càmera utilitzant aquesta potent eina.

### 4.2.2 Calibració d'una càmera

La calibració individual de les càmeres amb la *"Camera Calibration Toolbox for Matlab"* es realitza a través d'un gran conjunt de captures d'imatges fetes per la càmera a calibrar sobre un patró de calibració. Quantes més captures hàgim fet, més punts tindrem per analitzar i extreure'n els paràmetres intrínsecs, tot i que

poden haver-hi imatges que no ens interessin degut a la imprecisió amb la que s'han fet les estimacions, i que hi hagi un error relativament alt respecte les altres imatges. Com que les càmeres que utilitzem no disposen d'una gran qualitat d'imatge i tenen molta distorsió radial ens veiem obligats a realitzar una gran quantitat de captures per a una millor estimació del model de distorsió que serà el factor més important i complex en el procés d'obtenció dels paràmetres intrínsecs. La resolució de les imatges és de 640x480 píxels.

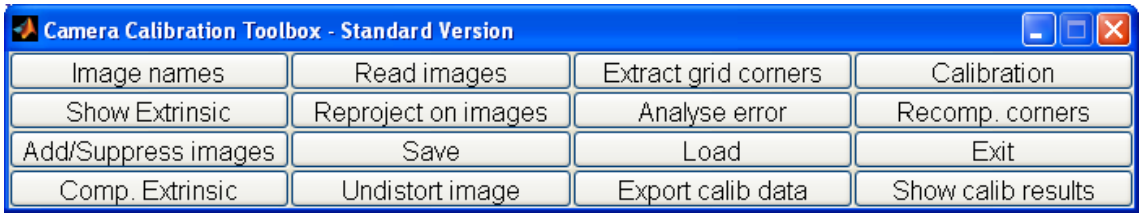



Figura 4.2. *Camera calibration toolbox* – Menú principal per a la calibració d'una càmera.

Per a les captures es recomana utilitzar un patró semblant a un tauler d'escacs, el qual pugui ser fàcil de determinar punts característics, tals com els còrners (cantonades) de cada un dels quadres que formen tot el patró. Durant les diferents captures aquest patró s'ha tingut que rotar i traslladar al llarg de l'escena, és a dir, les captures han de reflexar el patró en diferents posicions per tota l'àrea de treball. Una vegada s'hagin realitzat tota la sèrie de captures ja es poden carregar les imatges des de la "Camera Calibration Toolbox for Matlab".

A la figura 4.3 es poden veure en un mosaic el conjunt de captures fetes tal com s'ha comentat anteriorment.

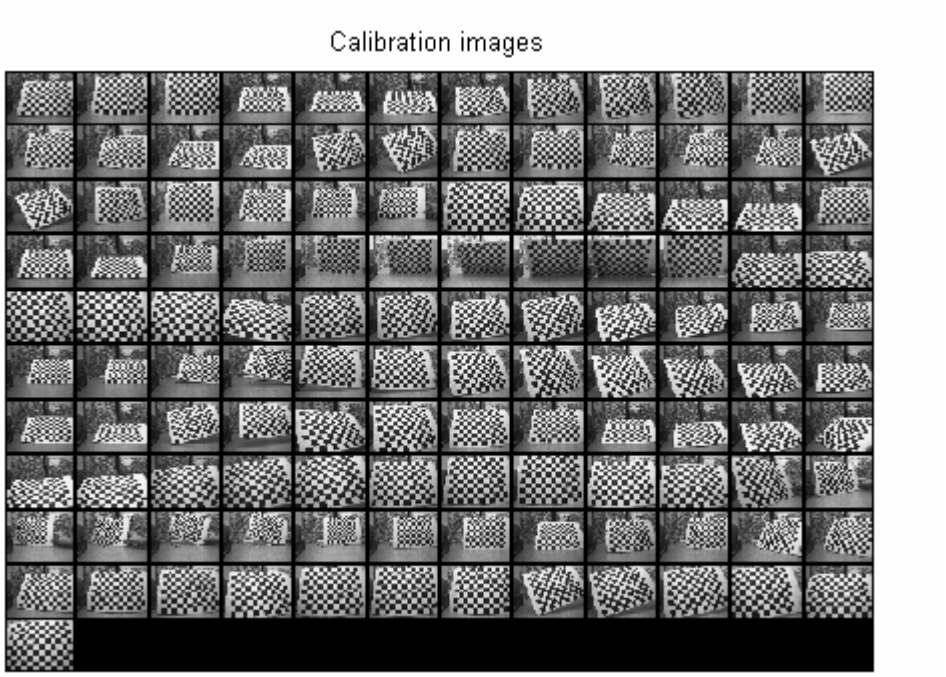



Figura 4.3. Mosaic amb totes les captures fetes amb la càmera esquerra en total 121 de les quals n'acabarem eliminant algunes.

Ara caldrà analitzar imatge a imatge extraient cada un dels còrners de tots els quadres que formen el patró, tal com es mostra a la figura 4.4. Per a extreure aquests còrners, la *"Camera Calibration Toolbox"* és capaç de reconèixer quants quadrats hi ha dintre el patró només seleccionant manualment els quatre extrems d'aquest, això gràcies a un "corner finder" o detector de còrners el qual li entrem la mida de la finestra a aquest *corner finder* que estimarà on es trobarà el còrner d'un quadrat acabarà tractant píxel a píxel per decidir quin és el millor píxel dintre aquesta finestra que millor s'adapta a les característiques de ser un còrner Si per culpa de la distorsió o bé perquè està proper el patró el còrner surt de rang del *corner finder* llavors aquest és incapaç de trobar-lo ja que es troba fora de la matriu, aquest problema es pot solucionar llavors augmentant o disminuint (si és el cas que el patró està molt lluny i es confon de còrner) el tamany del còrner per a què el còrner finder pugui trobar el seu còrner corresponent.

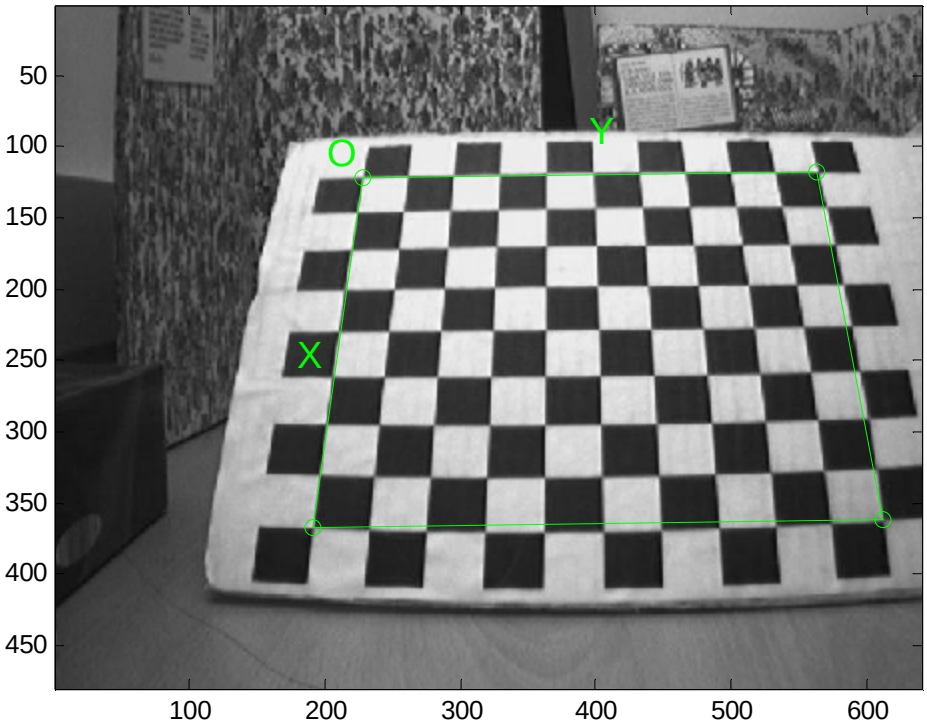



Figura 4.4 . Aquí clickem el quatre còrners principals

Després de seleccionar els quatre còrners principals del patró, la toolbox crida una sèrie de funcions que estimen on es trobaran tots els còrners restants dels quadrats que formen tot el patró, (veure figura 4.5) tot estimant la distorsió que s'ha anat trobant a les imatges anteriors. En cas que no s'hagi estimat bé el coeficient de distorsió i els corners finders es trobin uns quants o la majoria fora dels seus còrners corresponents, llavors es té la possibilitat d'entrar un coeficient de distorsió manualment o augmentar el tamany de la finestra del *corner finder*.

A mesura que augmenta el nombre d'imatges que anem processant l'estimació del model de la distorsió anirà acostant-se més al coeficient no lineal real i reduint la desviació estàndard del punt focal (principal point) i distància focal (focal length).

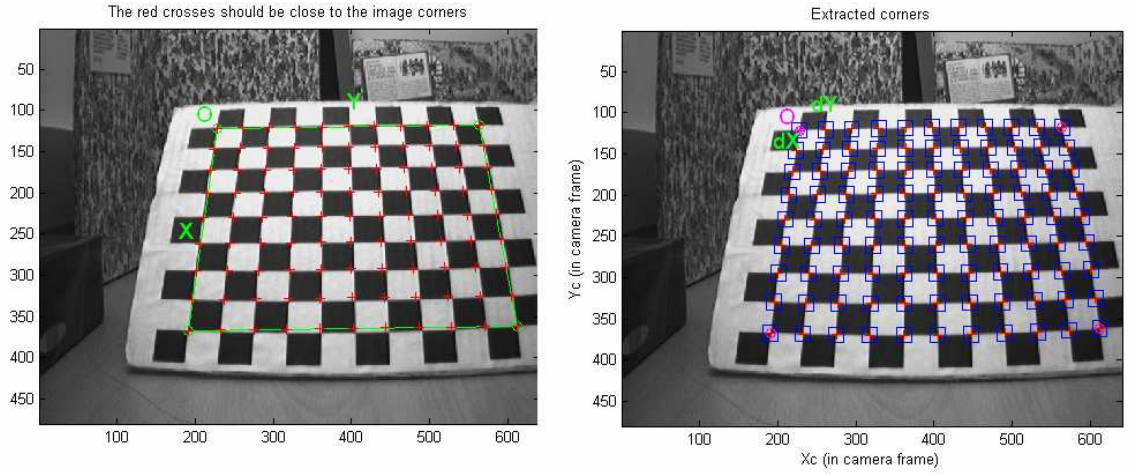



Figura 4.5. A l'esquerra els còrners sense calcular el coeficient de distorsió i a la dreta es pot veure com s'ha estimat els còrners del patró d'acord al coeficient de distorsió estimat per la Toolbox.

La Toolbox ens preguntarà la distància dels costats dels quadrats que formen tot el patró. La distància dels costats és molt important per a determinar els paràmetres extrínsecs de la càmera, gràcies a que realitza una sèrie de càlculs i transformacions de píxels a mil·límetres, però en això no hi entrarem en més detall. En el nostre cas el patró que utilitzem cada un dels quadres mesura 20mm, tant en **dX** com en **dY**.

Tot aquest procés que s'ha explicat per a una imatge, caldrà fer-ho per tota la resta d'imatges que hem carregat a la Toolbox, al tenir força soroll i molta distorsió amb aquestes càmeres ens veiem obligats a utilitzar una gran quantitat d'imatges per a reduir la desviació estàndard del punt focal i de la distància focal ja que amb el software que utilitzarem a la secció d'experiments reals necessitarem tenir els valors d'aquesta desviació per sota d'1 píxel.

Una vegada realitzat aquest procés per a totes les imatges, ja podrem executar la funció "Calibration" que ens realitzarà tots els càlculs per a determinar els paràmetres intrínsecs a partir de tots els punts de cada una de les imatges. Doncs quan ha finalitzat l'execució d'aquesta funció ens retorna una sèrie de paràmetres els quals són tots els que ens interessen, tal com es mostra a la figura 4.6

```
Aspect ratio not optimized (est_aspect_ratio = 0) -> fc(1)=fc(2). Set
est_aspect_ratio to 1 for estimating aspect ratio.
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point,
set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
     Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .

Main calibration optimization procedure - Number of images: 93
Gradient descent iterations:
1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19..
.done
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 766.60424   766.60424 ] ± [ 2.94323   2.94323 ]
Principal point:       cc = [ 323.45197   218.15493 ] ± [ 3.44629   3.01490 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.42628   0.25083   0.00104   -0.00021  0.00000 ] ± [
0.01002   0.04569   0.00087   0.00057  0.00000 ]
Pixel error:          err = [ 0.68069   0.70608 ]

Note: The numerical errors are approximately three times the standard deviations (for
reference).


Pixel error:          err = [ 0.68069   0.70608] (all active images)
```

Figura 4.6. Valors després de la primera calibració amb 121 imatges, aquí de moment els resultats de la primera calibració són força dolents, degut a que tenen una variància propera a 3 tant per la longitud focal i el punt principal, l'ideal seria tenir-la inferior o propera a 1.

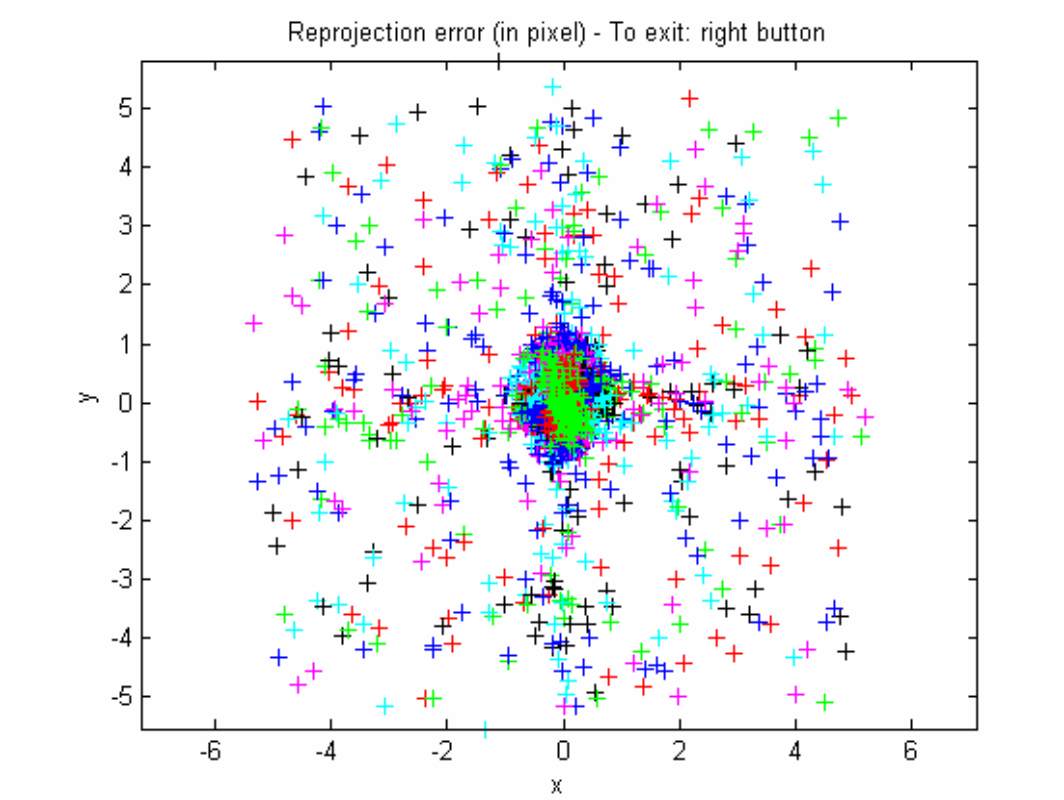



Figura 4.7. Error de reprojecció en píxels. Aquest és l'error entre el punt real on es troba i l'estimació que ha fet la Toolbox. Cada un dels píxels error hauria de ser inferior al píxel i la mitjana propera al [0.25 0.25].

Acabem de fer la primera calibració de la càmera amb les imatges que hem utilitzat. Els valors d'aquesta primera calibració no són gaire bons si mirem el píxel error i la variància de la distància focal (fc) i del punt focal (cc). Primer de tot el que hem de fer és intentar disminuir l'error de píxel. Aquest error de píxel és l'error que hi ha entre un punt estimant on hi ha el còrner i on realment s'ha de trobar. Això pot ser degut, a part de la falta de qualitat o excés de soroll de les càmeres, a que el corner finder és incapaç de trobar el còrner dintre els seus límits. Per això, com s'ha dit, caldrà augmentar el tamany del *corner finder* per a les imatges corresponents per tal que el còrner es trobi dintre el rang del *corner finder*, d'aquesta manera si fem això per totes les imatges les quals algun dels seus còrners surti de rang del seu corresponent còrner finder reduirem molt aquest error de reprojecció. En el procés de calibració no ens hem trobat que hàgim tingut que fer el contrari, és a dir, reduir el corner finder perquè aquest contenia dos còrners dintre ell mateix.

Després d'augmentar el tamany dels còrners finders per a les captures que ho requerien, ja podrem tornar a procedir a una altra calibració, de manera que els

errors que es mostraven a la figura 4.7 quedarien enormement reduïts com es mostra a la figura 4.8.

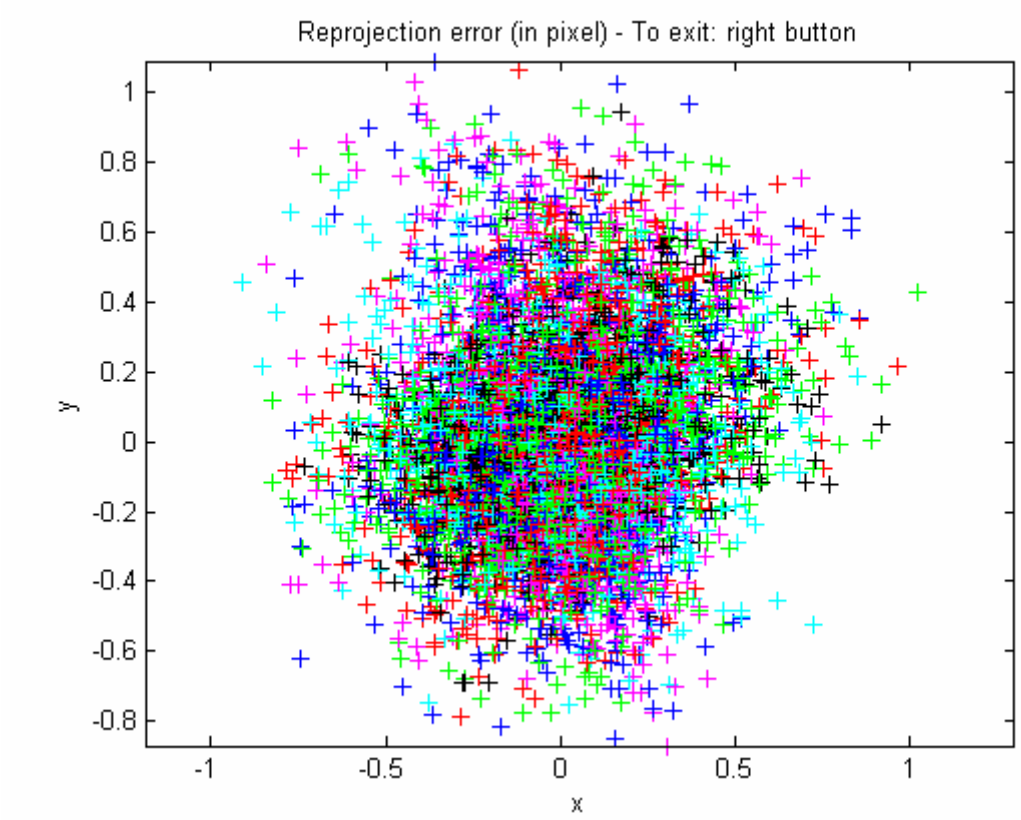



Figura 4.8. Errors de reprojecció després d'augmentar alguns corners finders per a algunes imatges, eliminar les que no podem disminuir l'error per sota del píxel i tornar a executar la calibració.

A la figura 4.9 podem veure la reprojecció dels punts sobre una imatge en concret després d'estimar els paràmetres intrínsecs de la càmera, és a dir, ens mostra cada un dels punts que ha estimat segons els paràmetres intrínsecs i on es troba el punt real. Amb una fletxa ens indica cap a on es trobaria el punt real, i quant més llarga sigui la fletxa ens indicaria que l'error de reprojecció és més elevat. A la figura 4.10 hi podem veure una imatge on els errors de reprojecció són molt més grans en comparació amb la figura anterior. I a la figura 4.11 podem veure-hi els resultats de la calibració després d'haver ajustat segons les imatges el còrner finder.

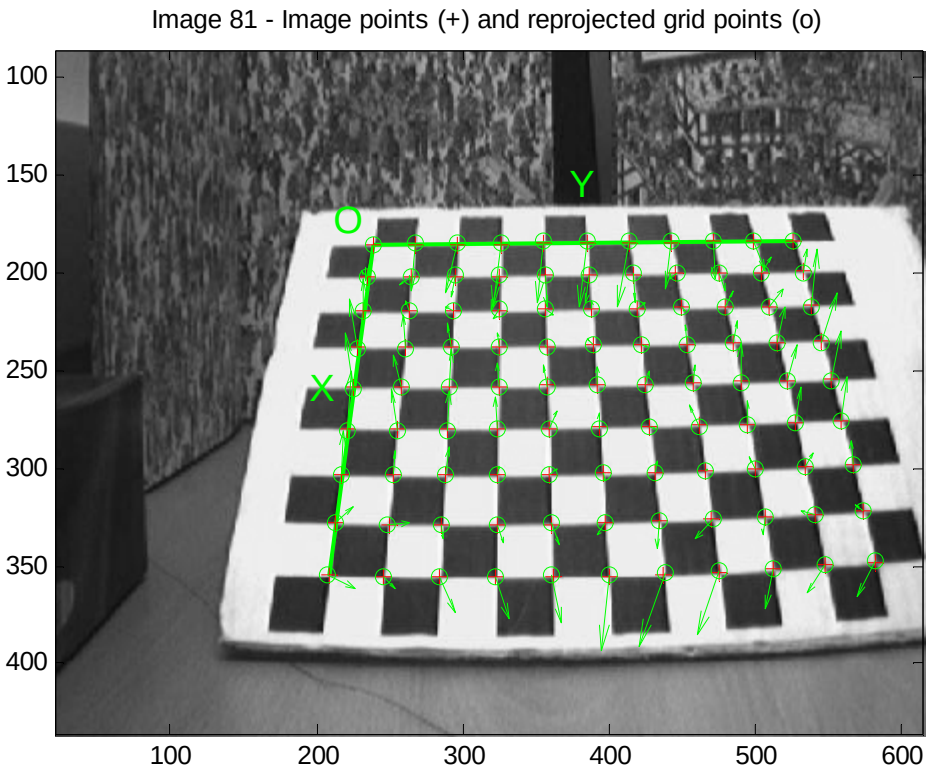



Figura 4.9. Reprojecció dels punts estimats sobre la imatge on hi podem veure per cada còrner la finestra que busca el millor píxel per considerar-lo cantonada (corner finder). La fletxa ens indica cap a quina direcció es troba el punt real, i quant més llarga és aquesta indica que més gran és l'error de reprojecció sobre el punt.

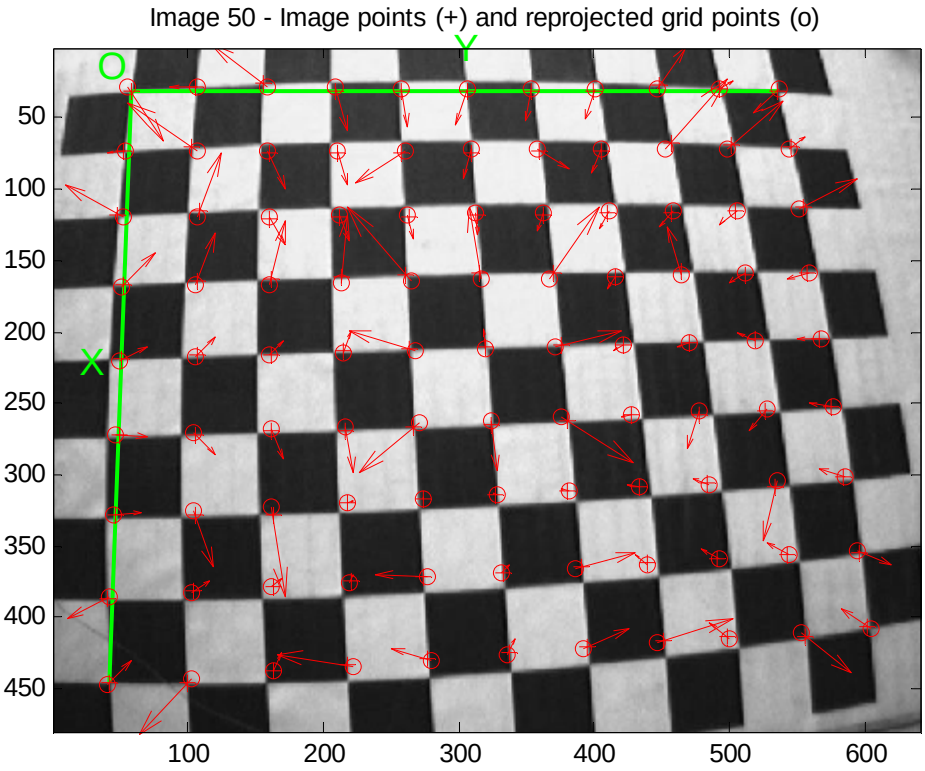



Figura 4.10. Reprojecció dels punts estimats sobre una imatge la qual s'han estimat erròniament els punts que es consideren còrners, per tant en aquesta imatge hi tindrem un error de reprojecció molt elevat.

```
Aspect ratio optimized (est_aspect_ratio = 1) -> both components of fc are estimated
(DEFAULT).
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point,
set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
     Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .

Main calibration optimization procedure - Number of images: 94
Gradient descent iterations:
1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19..
.20...done
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 761.38482   762.85971 ] ± [ 1.32793   1.29738 ]
Principal point:       cc = [ 317.47851   221.55350 ] ± [ 1.24635   1.43820 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.40358   0.17067   0.00014   0.00031  0.00000 ] ± [
0.00398   0.01756   0.00055   0.00020  0.00000 ]
Pixel error:          err = [ 0.21929   0.32664 ]
```

Figura 4.11. Resultats dels paràmetres intrínsecs després d'ajustar bé el còrner finder per a algunes imatges i eliminar les imatges amb les quals no hem pogut reduir l'error de píxel per sota del píxel.

Com que les càmeres que utilitzem tenen els píxels quadrats podem considerar que la distància focal és igual tant en x com en y, per això posem a 0 la variable est_aspect_ratio = 0, amb això reduirem la complexitat del model de distorsió a calcular. A més a més si mirem el tercer i cinquè coeficient de distorsió i la seva corresponent variància, veiem que és molt semblant, això ens crearà problemes al calcular-ne el model, almenys la variància hauria de ser uns 10 cops més petita que el coeficient. Així doncs, posarem a 0 el coeficient 3 i 5 que són sobre la distorsió radial. Com que el primer coeficient és més de 100 vegades més gran que la seva variància i el segon coeficient és d'unes 20 vegades més gran que la seva respectiva variància també utilitzarem aquests dos per a simplificar la complexitat del model de distorsió.

A continuació, a la figura 4.12, s'adjunten els resultats després d'haver fet les anteriors modificacions.

```
Aspect ratio not optimized (est_aspect_ratio = 0) -> fc(1)=fc(2). Set
est_aspect_ratio to 1 for estimating aspect ratio.
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point,
set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
     Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .
     Tangential distortion not estimated (est_dist(3:4)~=[1;1]).

Main calibration optimization procedure - Number of images: 93
Gradient descent iterations: 1...done
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 763.98353   763.98353 ] ± [ 0.76912   0.76912 ]
Principal point:       cc = [ 321.53211   220.08907 ] ± [ 0.96560   0.81668 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.42568   0.25360   0.00000   -0.00000  0.00000 ] ± [
0.00359   0.01629   0.00000   0.00000  0.00000 ]
Pixel error:          err = [ 0.22968   0.27697 ]
```

Figura 4.12. Resultats finals de calibració per a la càmera esquerra, com es pot apreciar la desviació estàndar de la longitud focal (focal length) i el punt focal (principal point) és inferior al píxel (<1) i la mitjana del píxel error està rondant els 0.25.

A més a més, podem visualitzar gràficament els paràmetres extrínsecs de 1 sola càmera (la posició de la càmera respecte les imatges utilitzades durant el procés de calibració del patró) obtinguts des de dos vistes diferents: una vista tenint com a referència la càmera on el patró surt en diferents posicions (figura 4.13) ,

i l'altra vista seria on el patró apareixeria fix i la càmera sortiria desplaçada, és a dir, tenint com a referència el món (figura 4.14).

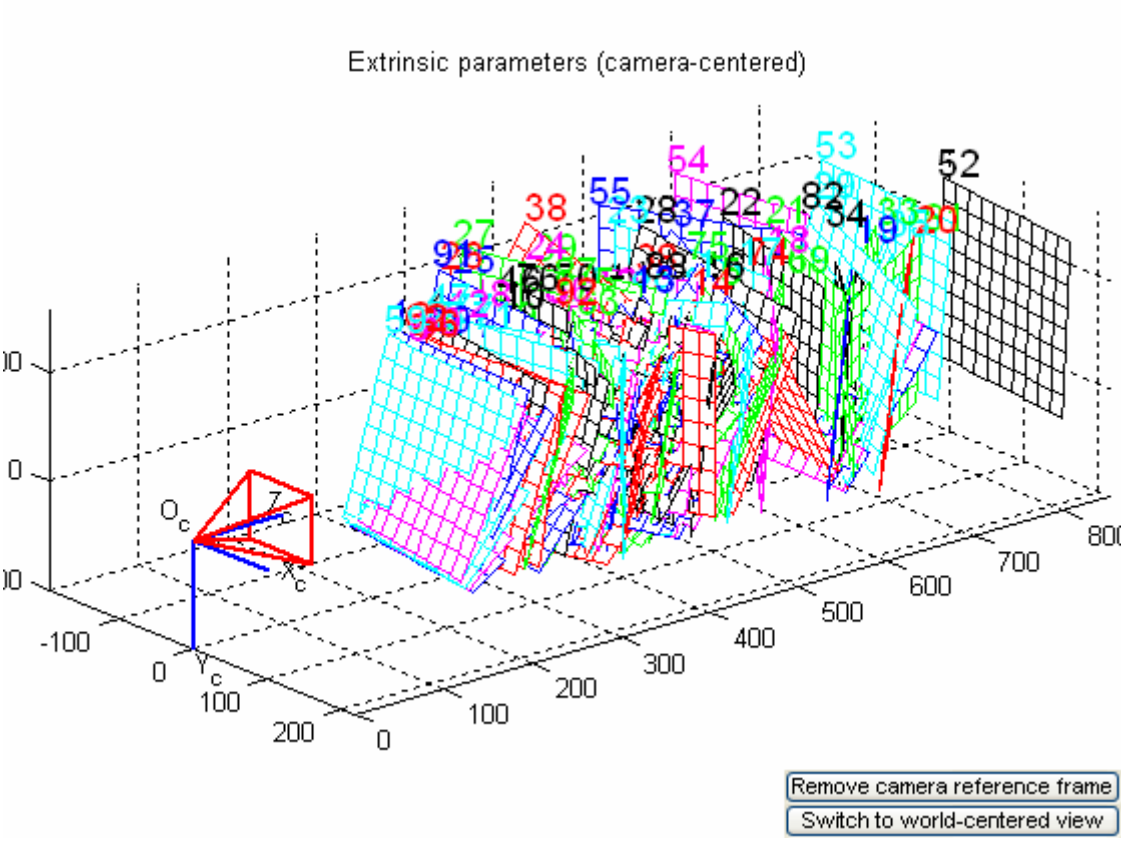



Figura 4.13. Visualització dels paràmetres extrínsecs utilitzant la càmera com a referència.

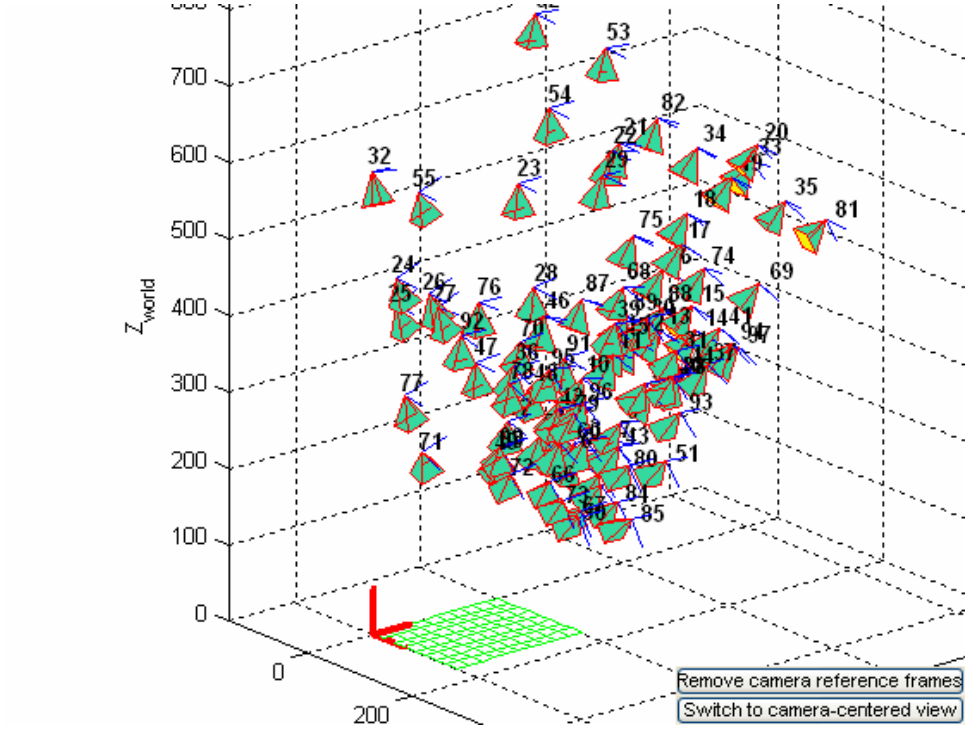



Figura 4.14. Visualització dels paràmetres extrínsecs utilitzant el món real com a referència.

Finalment, després d'haver reduït al mínim possible els errors de píxel i les variàncies del punt focal i de la distància focal, podem donar per bons els paràmetres intrínsecs, ja que la variància per la longitud focal i del punt focal és inferior al píxel (<1). Hem necessitat una gran quantitat d'imatges (93) per a arribar a aquests valors que a continuació s'adjunten. Si disposéssim d'una càmera amb poc soroll i millor qualitat d'imatge probablement amb forces menys imatges n'haguéssim fet suficient per a arribar a aquests valors.

```
fc = [ 763.98353   763.98353 ] ± [ 0.76912   0.76912 ]
Principal point:      cc = [ 321.53211   220.08907 ] ± [ 0.96560   0.81668 ]
Skew:               alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.42568   0.25360   0.00000   -0.00000  0.00000 ] ± [
0.00359   0.01629   0.00000   0.00000  0.00000 ]
```

A la taula següent es mostren els paràmetres intrínsecs calculats:

| | |
|---|---|
| Longitud focal | fc = [ 763.98353 763.98353 ] ± [ 0.76912 0.76912 ] |
| Punt focal | cc = [ 321.53211 220.08907 ] ± [ 0.96560 0.81668 ] |
| Skew | alpha_c = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.00000 ± 0.00000 degrees |
| Coeficients de distorsió | kc = [ -0.42568 0.25360 0.00000 -0.00000 0.00000 ] ± [ 0.00359 0.01629 0.00000 0.00000 0.00000 ] |

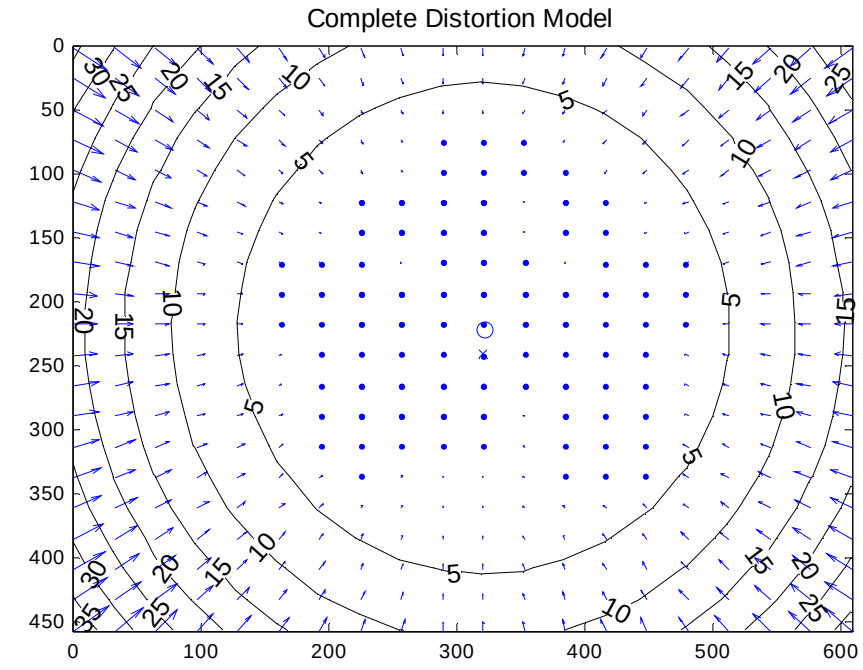
Amb aquests paràmetres intrínsecs ja podem obtenir la matriu dels paràmetres extrínsecs, sabent que:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = KK \begin{bmatrix} x_d(1) \\ x_d(2) \\ 1 \end{bmatrix} \qquad KK = \begin{bmatrix} fc(1) & alpha\_c * fc(1) & cc(1) \\ 0 & fc(2) & cc(2) \\ 0 & 0 & 1 \end{bmatrix}$$

Coneixent els paràmetres intrínsecs podem substituir els valors de l'equació anterior:

```
K = [763.98353   0  33.49493; 0  321.53211     220.08907; 0     0   1]
```

Tornant als paràmetres intrínsecs, també tenim la possibilitat de poder mostrar gràficament les diferents distorsions que s'han rectificat gràcies als càlculs per a l'obtenció dels coeficients per corregir aquesta distorsió (veure figura 4.15).




| | | |
|---|---|---|
| Pixel error | = [0.2297, 0.277] | |
| Focal Length | = (763.984, 763.984) | +/- [0.7691, 0.7691] |
| Principal Point | = (321.532, 220.089) | +/- [0.9656, 0.8167] |
| Skew | = 0 | +/- 0 |
| Radial coefficients | = (-0.4257, 0.2536, 0) | +/- [0.003585, 0.01629, 0] |
| Tangential coefficients | = (0, -0) | +/- [0, 0] |

Figura 4.15. Gràfica de la representació del model distorsió complet.

A la figura 4.15 es mostra l'impacte complet del model de distorsió calculat, aquest model complet de distorsió conté l'error radial i l'error tangencial de distorsió per cada píxel de la imatge. Cada fletxa de color blau representa el desplaçament d'un píxel induït per la distorsió de les lents. Si s'observa aquests punts que són dels còrners de la imatge, alguns surten desplaçats forces píxels. La figura 4.16 ens mostra l'impacte de la component tangencial de la distorsió. Per últim, la figura 4.17 mostra l'impacte de la component radial de la de la distorsió. A aquesta darrera gràfica és molt semblant a la primera figura (al model complet de distorsió), degut a que la component tangencial pot ser quasi nul·la, per tant pràcticament no interfereix en el model de distorsió complet. A aquestes figueres el cercle que es troba al centre ens indica el punt focal.

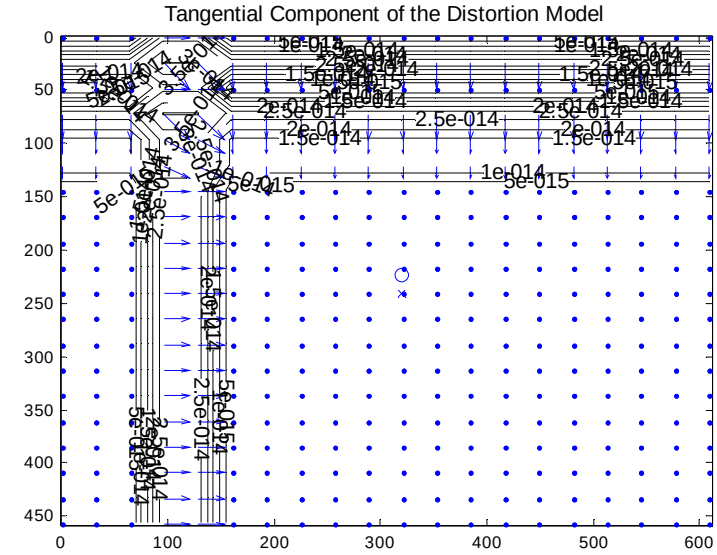



Figura 4.16. Representació gràfica del model de distorsió tangencial, com que no l'hem estimat per què hem donat per nul·la aquesta distorsió així aconseguíem reduir la complexitat de l'estimació dels coeficients de distorsió.

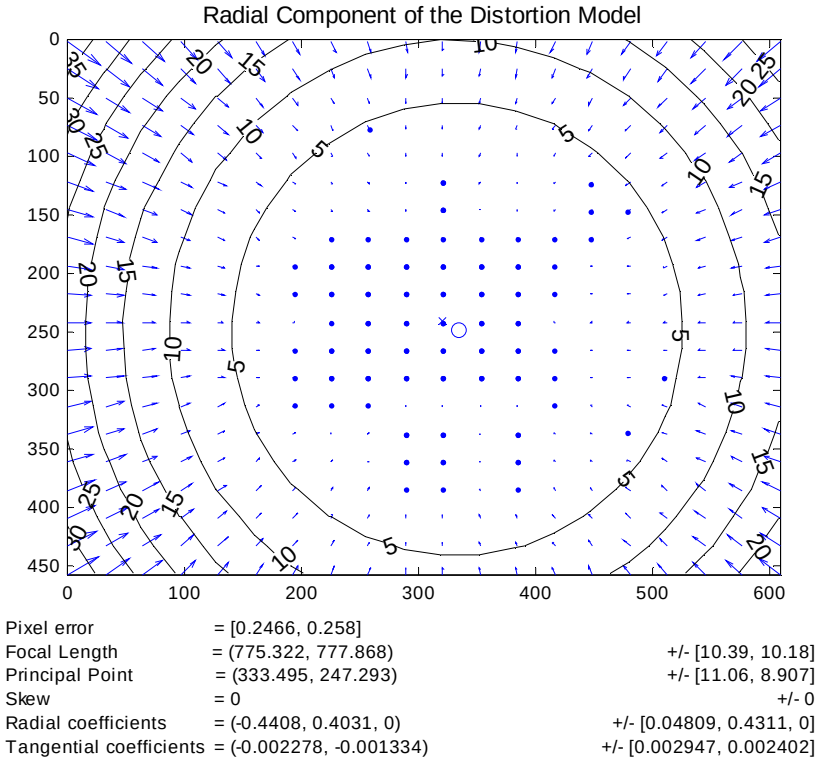



Figura 4.17. Representació gràfica del model de distorsió radial, al ser nul·la la distorsió tangencial és el mateix que el model de distorsió complet

També tenim la possibilitat de computar els paràmetres extrínsecs una vegada calibrada la càmera, per a això, podem agafar una imatge qualsevol del patró realitzada amb la mateixa càmera, amb la funció "Comp. Extrinsic", on haurem de fer com el procés de calibració, però només seleccionant els quatre extrems del patró o donant-li la mida dels quadres interns (amplada i altura). Llavors la Toolbox a través dels paràmetres de la càmera realitzarà una sèrie de càlculs que ens retornarà la rotació i translació del patró respecte l'origen del sistema de coordenades de la càmera.

```
Extrinsic parameters:

Translation vector: Tc_ext = [ -54.095576     -60.337877     443.118722 ]
Rotation vector:   omc_ext = [ 1.863667       1.894657       -0.704608 ]
Rotation matrix:    Rc_ext = [ -0.040051      0.996681       -0.070876
                               0.800799       -0.010406      -0.598842
                               -0.597592      -0.080742      -0.797725 ]
Pixel error:           err = [ 0.18961        0.34476 ]
```

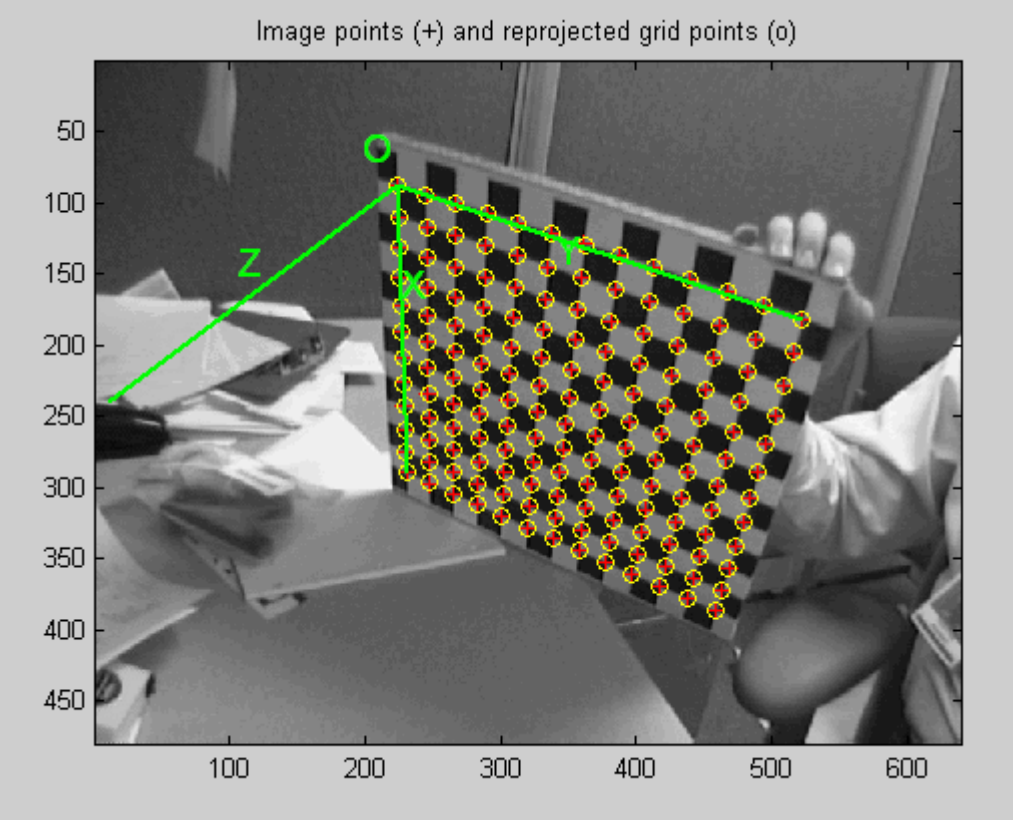



Figura 4.18. Reprojecció d'una captura amb els punts i el sistema de coordenades propi del patró de calibració.

Si donem per bona la calibració de la càmera, cal guardar-la i llavors ja es pot anar calibrar la segona càmera del sistema estèreo, la qual ha de seguir els mateixos passos i utilitzant les mateixes captures del patró, però com és lògic, des de la seva pròpia posició. Cal recalcar que per a una correcta calibració estèreo el sistema ha d'estar totalment fix, és a dir, les càmeres a una posició fixa una de l'altre.

Si en el procés de calibració de la primera càmera s'han eliminat certes imatges, també cal eliminar les captures corresponents a la segona càmera.

### 4.2.3 Calibració sistema estèreo

A aquesta secció explicarem com es calibra el sistema estèreo després d'haver calibrat individualment les dues càmeres i donar-les per bones. Per a calibrar el sistema estèreo es segueix utilitzant la "Camera Calibration Toolbox for Matlab" la qual disposa també d'una sèrie de funcions per realitzar els càlculs per estimar els paràmetres extrínsecs entre ambdues càmeres.

Per a computar els paràmetres extrínsecs del sistema estèreo, primer es carreguen les calibracions de les dues càmeres indicant quina és la càmera esquerra i quina és la càmera dreta, mitjançant el nom dels fitxers. Quan s'han carregat correctament, cal cridar a la funció "Stereo calibration" que realitzarà tota la sèrie de càlculs per a realitzar una estimació dels paràmetres extrínsecs reals, també realitza una sèrie de càlculs per a optimitzar els paràmetres extrínsecs de les dues càmeres i així minimitzar els errors de reprojecció a ambdues càmeres.

```
Stereo calibration parameters after optimization:


Intrinsic parameters of left camera:

Focal Length:          fc_left = [ 758.44666   758.44666 ] ± [ 0.00000   0.00000 ]
Principal point:       cc_left = [ 318.12388   221.92275 ] ± [ 0.00000   0.00000 ]
Skew:             alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes
= 90.00000 ± 0.00000 degrees
Distortion:            kc_left = [ -0.40287   0.17918   0.00000   0.00000  0.00000 ]
± [ 0.00000   0.00000   0.00000   0.00000  0.00000 ]


Intrinsic parameters of right camera:

Focal Length:          fc_right = [ 757.48186   757.48186 ] ± [ 0.00000   0.00000 ]
Principal point:       cc_right = [ 348.35718   201.30341 ] ± [ 0.00000   0.00000 ]
Skew:             alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes
= 90.00000 ± 0.00000 degrees
Distortion:            kc_right = [ -0.40583   0.19850   0.00000   0.00000  0.00000 ]
± [ 0.00000   0.00000   0.00000   0.00000  0.00000 ]


Extrinsic parameters (position of right camera wrt left camera):

Rotation vector:             om = [ 0.00883   0.08996  0.00028 ] ± [ 0.00029
0.00034  0.00017 ]
Translation vector:           T = [ -132.00361   6.92330  9.14420 ] ± [ 0.15683
0.13165  0.08492 ]


Note: The numerical errors are approximately three times the standard deviations (for
reference).
```

Al final d'aquestes dades apareixen els paràmetres extrínsecs del sistema estèreo que és la posició de la càmera dreta respecta la càmera esquerra. A la figura 4.19 es pot veure la representació 3D del sistema estèreo reprojectant-lo

gràficament amb les diferents posicions del patró que apareix al llarg de les captures que s'han utilitzat en el procés de calibració.

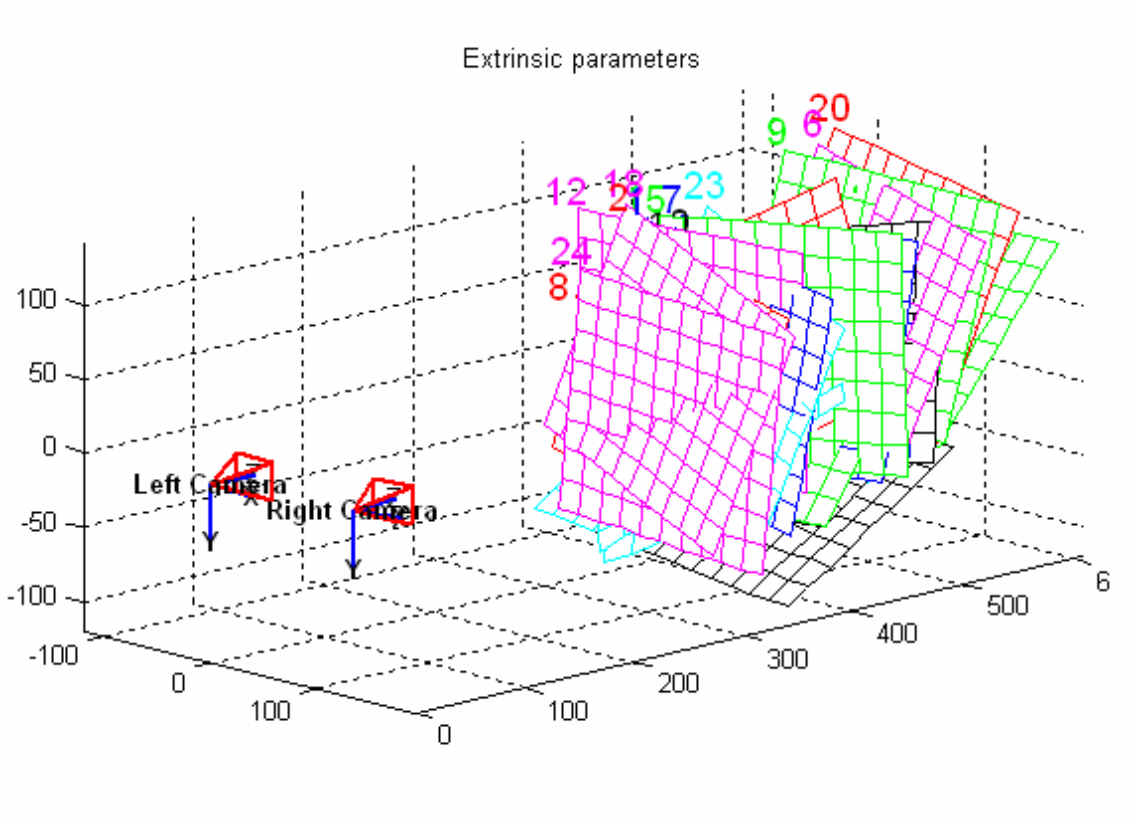



Figura 4.19. Representació gràfica dels paràmetres extrínsecs del sistema estèreo.

Per a comprovar si l'estimació dels paràmetres extrínsecs de la càmera és més o menys correcta, podem mesurar físicament la distància entre les dos òptiques de les dues càmeres i llavors comparar-ho amb la distància euclidiana de la translació que ens ha retornat la Toolbox.

Ara, si es dóna per bona la calibració després de realitzar una sèrie de proves, com l'esmentada en el paràgraf anterior, cal guardar-la ja que ens serà útil per quan vulguem treballar amb dades reals i/o triangular punts com veurem més endavant per obtenir-ne una bona informació 3D sobre tals punts.

## 4.3 Experiments reals

### 4.3.1 Introducció

En aquesta secció provarem els diferents mètodes que hem implementat en aquest PFC però aquesta vegada ho farem sobre dades reals. Per a això, primer de tot, utilitzarem el software específic desenvolupat per Jordi Ferrer Plana del grup VICOROB de la Universitat de Girona per a realitzar un "tracking" (seguiment) de diferents punts reals en una seqüència de vídeo que convertirem en una seqüència d'imatges. Aquesta seqüència d'imatges serà amb la que realitzarem un seguiment dels punts deformables i dels punts rígids per més endavant aplicar els mètodes de segmentació. Els punts amb els quals treballarem han de ser punts que apareguin al llarg de tota la seqüència, per exemple: en una seqüència de 15 frames, no ens interessa treballar amb punts que en aquesta seqüència de 15 frames hagin aparegut 14 o menys cops, ja que llavors no podríem aplicar els mètodes de segmentació i preprocessat que hem elaborat ja en la part d'experiments sintètics. A més a més, al treballar amb punts que han aparegut al llarg de la seqüència aconseguirem eliminar falses correspondències, sempre i quan la seqüència tingui almenys més d'uns 4 o 5 frames aproximadament.

Aquest software per a realitzar el Tracking el que fa és buscar correspondències de punts 2D utilitzant l'algorisme SURF (Speeded Up Robust Features) i a través de les línies epipolars donant-li una certa tolerància buscarà correspondències entre el mateix frame per a la imatge de la càmera esquerra i de la imatge de la càmera dreta. Llavors al trobar una possible correspondència comprovarà que si en el següent frame es pot trobar el mateix punt en ambdues imatges. I realitzarà aquest procés fins a arribar a l'últim frame que nosaltres li indiquem. Com he dit en el paràgraf anterior, només ens interessarà trobar punts que apareixen en tota la seqüència que estem tractant, d'aquesta manera serà molt més restrictiu i perdrem molts de punts que són rígids i/o estan en moviment. Ho fem així per què per l'objectiu de tenir models deformables necessitem visibilitat de tots els punts al llarg de tota la seqüència.

Cal destacar també que el software el farem treballar amb les imatges rectificades, és a dir, com que hem estimat l'equació de distorsió de les lents per cada càmera, llavors el software s'encarregarà de corregir aquestes imatges per

llavors buscar-ne els punts característics i les seves corresponents correspondències.

Primer de tot, haurem de trobar un bon entorn on hi hagi molta textura per a poder extreure'n punts característics fàcilment, d'aquesta manera podrem aconseguir més quantitat de punts amb el software que utilitzarem per trobar-los i fer-ne el seguiment d'aquests. És molt important tant en l'entorn com en els objectes que tinguin textura, sinó ens serà molt difícil poder treballar amb dades reals.

Els paràmetres de calibració per a totes les captures reals que farem són els següents:

**Paràmetres intrínsecs càmera esquerra**

```
Focal Length:          fc_left = [ 758.44666   758.44666 ] ± [ 0.76103   0.76103 ]
Principal point:       cc_left = [ 318.12388   221.92275 ] ± [ 1.02256   0.79868 ]
Skew:             alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes
= 90.00000 ± 0.00000 degrees
Distortion:            kc_left = [ -0.40287   0.17918   0.00000   0.00000  0.00000 ]
± [ 0.00381   0.01698   0.00000   0.00000  0.00000 ]
```

**Paràmetres intrínsecs càmera dreta**

```
Focal Length:          fc_right = [ 757.48186   757.48186 ] ± [ 0.65853   0.65853 ]
Principal point:       cc_right = [ 348.35718   201.30341 ] ± [ 0.94729   0.78938 ]
Skew:             alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes
= 90.00000 ± 0.00000 degrees
Distortion:            kc_right = [ -0.40583   0.19850   0.00000   0.00000  0.00000 ]
± [ 0.00311   0.01279   0.00000   0.00000  0.00000 ]
```

**Paràmetres extrínsecs (posició de la càmera dreta respecta la càmera esquerra).**

```
Rotation vector:             om = [ 0.00861   0.07449  -0.00130 ]
Translation vector:           T = [ -131.57873   6.43231  5.99690 ]
```

Les càmeres per a la captura real que utilitzarem són càmeres *FireWire Unibrain Fire-i Digital* connectades en sèrie i alimentades per un repetidor que aquest va connectat al ordinador. La resolució és el màxim que ens permeten aquestes càmeres que és de 640x480, el format del píxel és de RGB 24-Bit i el Frame rate

per a la captura de vídeo és de 15 frames per segon. A la figura 4.20 podem veure el sistema estèreo utilitzat.

Figura 4.20. Càmeres Unibrain Fire-i que utilitzarem al llarg d'aquest capítol per a fer les captures adients.

### 4.3.2 Entorn de treball (Setup)

Per a poder treballar amb el software del Tracking que hem comentat, primer de tot, és essencial tenir un entorn al qual li puguem extreure gran quantitat de punts característics per a què ens ajudin a trobar els punts 3D dels objectes els quals mourem/deformarem. Per a això es va provar amb diferents objectes per a posar-los de fons ocupant pràcticament tota l'àrea de sol·lapament entre les dues càmeres. Després de provar de posar diferents peces de roba que tinguin força contrast i no treure'n resultats prou bons, al final vam utilitzar dos llibres oberts els quals tenen molts dibuixos en color i on el software ens va fer el tracking i matching que ha donat bons resultats. Els dos llibres són de «On està en Wally ?» oberts, com ja se sap, és "difícil" trobar a en Wally degut a que hi ha gran quantitat de dibuixos, que equivaldria a tenir molta textura en el tema de la Visió per Computador en el tractament d'imatges.

Com que també ens interessa tenir un entorn el qual puguem apreciar la profunditat i que el fons no és pla, hem posat un llibre de base i llavors un llibre dret obert aproximadament uns 120º tal com es mostra a la imatge. Llavors el sistema estèreo l'hem elevat aproximadament uns 20 centímetres i l'hem inclinat

uns 30º. D'aquesta manera podrem obtenir millor informació 3D gràcies a que les càmeres podran veure els objectes (i el fons) en totes les seves dimensions: alçada, amplada i profunditat. I finalment, per il·luminar l'escena utilitzarem llum difusa (veure figura 4.20).


Figura 4.21. Entorn de treball on hi farem moure objectes i simularem deformacions.

Si posem a prova el software per tan sols aconseguir punts 3D sobre l'escena en una seqüència de 2 frames, on no hi ha cap moviment, podrem veure si és prou bo l'entorn de treball. Tal com es pot veure en el quadruplet de la figura 4.22, es troben moltíssimes correspondències. En total, després d'una primera execució per trobar-ne punts 3D de l'entorn de treball sense cap objecte, obtenim un total de 856 punts 3D.

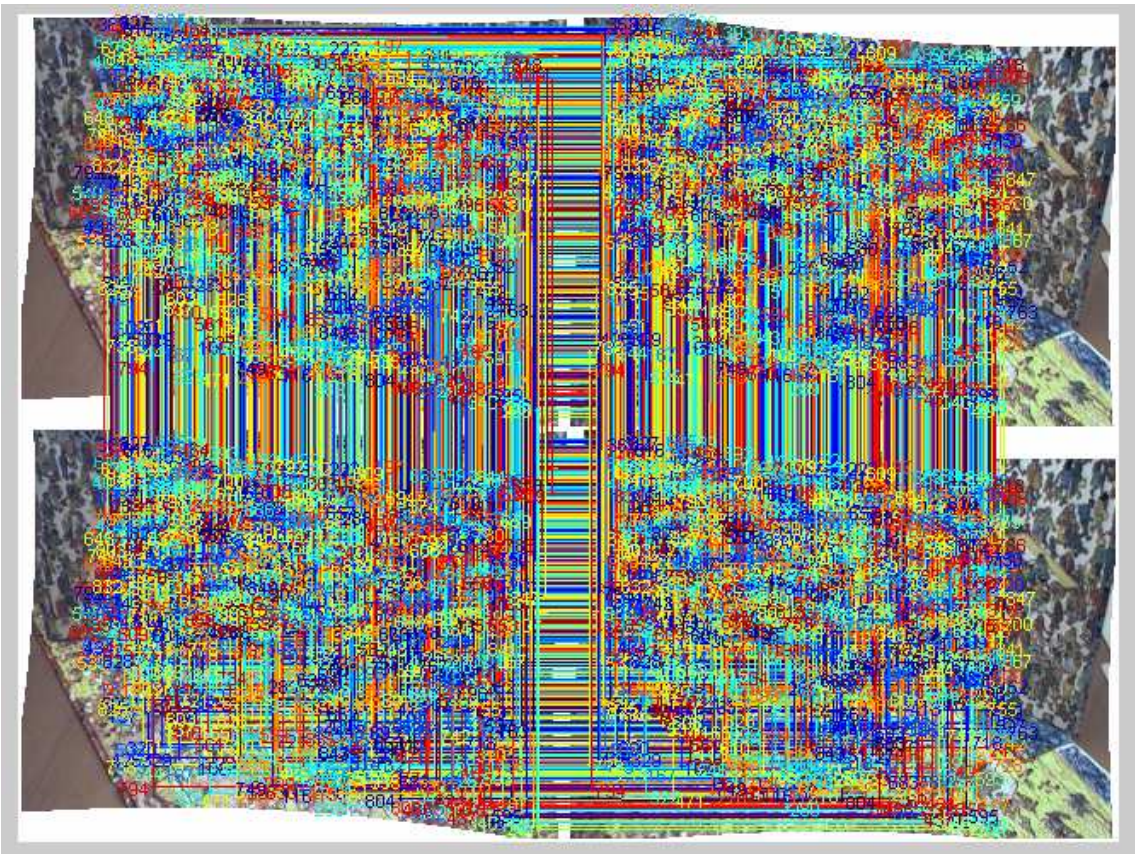


Figura 4.22 Quadruplet on apareixen les correspondències esquerra-dreta i frame n-frame n+1

Amb els punts 3D que hem obtingut amb el STracker podem visualitzar tots els punts 3D capturats amb el color real del punt. A la figura 4.23 es pot apreciar els punts 3D captats des de diferents vistes.

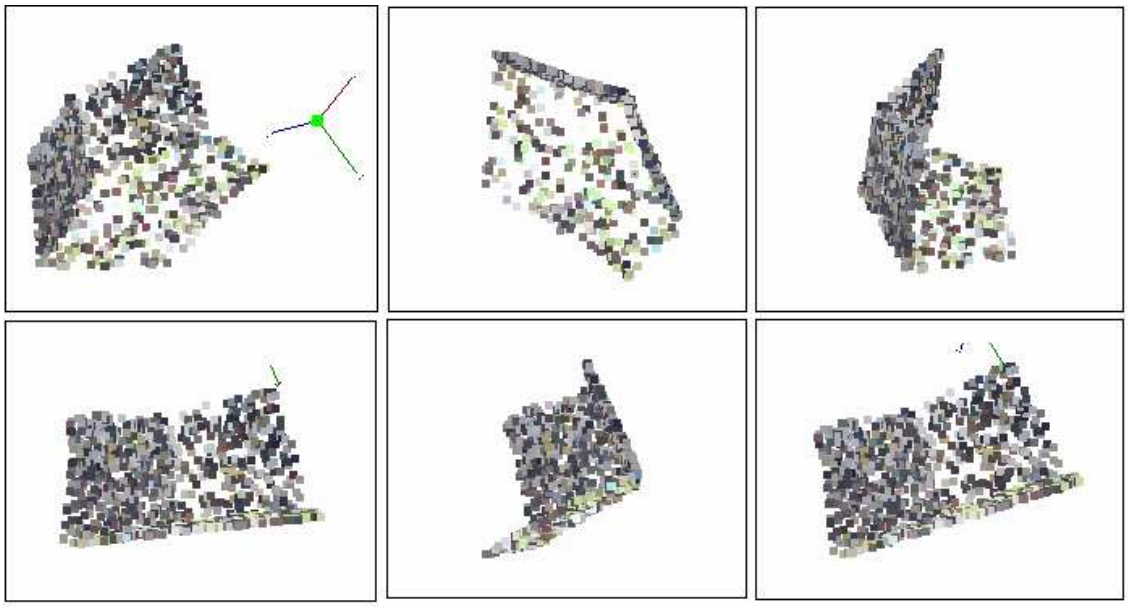


Figura 4.23 Representació 3D des de diferents punts de vista de l'estructura del setup de la figura 4.20.

Per tant, ens podem donar per contents del setup que hem muntat ja que hem trobat gran quantitat de punts i n'hem pogut extreure punts 3D per tot arreu de l'entorn.

### 4.3.3 Primer experiment: Objecte rígid + Objecte en moviment

El primer experiment que realitzarem per a provar els mètodes de segmentació que hem implementat es tractarà de dos objectes, els quals un estarà en moviment i l'altra romandrà immòbil. D'aquesta manera simulem un objecte deformable ja que l'objecte que estarà immòbil, els seus punts 3D que trobem gràcies al software que utilitzem pel Tracking els trobarà a la mateixa posició al llarg de la seqüència, en canvi l'objecte que movem, els seus punts seguiran una trajectòria amb la qual després podrem estimar l'error de posició dels punts 3D trobats aplicant el mètode RANSAC per trobar l'error de posició respecte el primer frame després d'aplicar la transformada inversa, tal com fèiem amb les dades sintètiques, això serà una aproximació a una deformació que en el següent experiment simularem una deformació movent objectes en diferents direccions.

Els objectes que utilitzem en aquest experiment es tracten de dues caixetes de cerilles que les seves dimensions són de 8x5x2.8 cm. En quan a l'entorn de treball, seguirem utilitzant el mateix.

A la figura 4.24 podem veure com es trobaran en la posició original les dues caixes de cerilles a utilitzar. Serà la caixa de l'esquerra de la imatge la que mourem estirant-la per un fil aproximadament uns 10cm cap a l'esquerra, l'altre en canvi, com s'ha dit, romandrà immòbil.

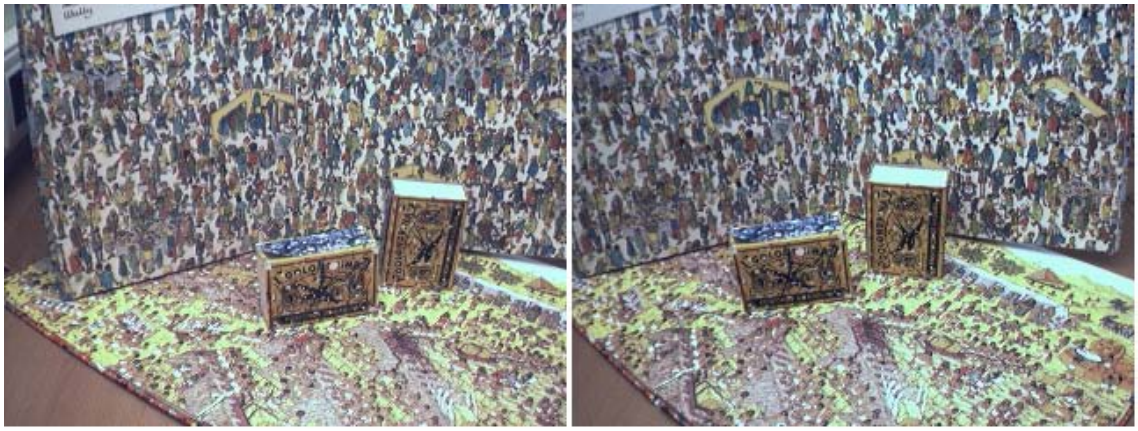


Figura 4.24 Posició inicial dels objectes des de la vista de les dues càmeres esquerra i dreta respectivament.

També com podem veure a la figura 4.24 el sol·lapament entre les dues càmeres és prou bo com per a què puguem fer un bon "tracking" i matching de correspondències, per tant hauríem de tenir bons resultats amb aquest setup.

A la figura 4.25 es mostra la posició final dels objectes, la caixa esquerra s'ha desplaçat aproximadament uns 8-10cm, més endavant abans de segmentar provarem de calcular-ne la distància euclidiana entre diferents punts que han estat desplaçats (deformats si ens ho mirem com una deformació d'un objecte que és el que es tracta).

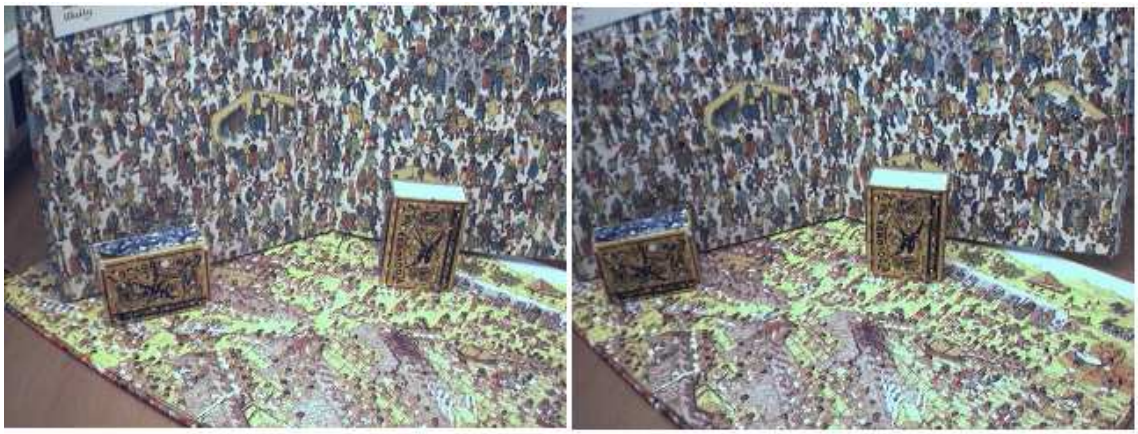


Figura 4.25. Posició final de la seqüència vist des de les dues càmeres, esquerra i dreta respectivament.

La captura de vídeo que s'ha realitzat amb el sistema estèreo entre la posició inicial i la posició final, en total hi ha 90 frames. Per a fer el tracking en total hem utilitzat 9 frames, saltant de 10 en 10 ja que el desplaçament s'ha fet a poc a poc per a poder-ne fer un bon tracking. Per tant, com que el frame rate de la captura de vídeo és de 15fps, el temps real d'aquesta seqüència seria de 90/15 = 6 segons. I els frames que s'han utilitzat per a fer el tracking són: 10, 20, 30, 40, 50, 60, 70, 80 i 90. Ho fem així ja que no cal fer-los servir tots perque amb aquests ja és representatiu. A la figura 4.26 hi podem observar un exemple del matching i tracking entre dos frames consecutius, a la part superior hi trobem el frame 80 i a la part inferior el frame 90.

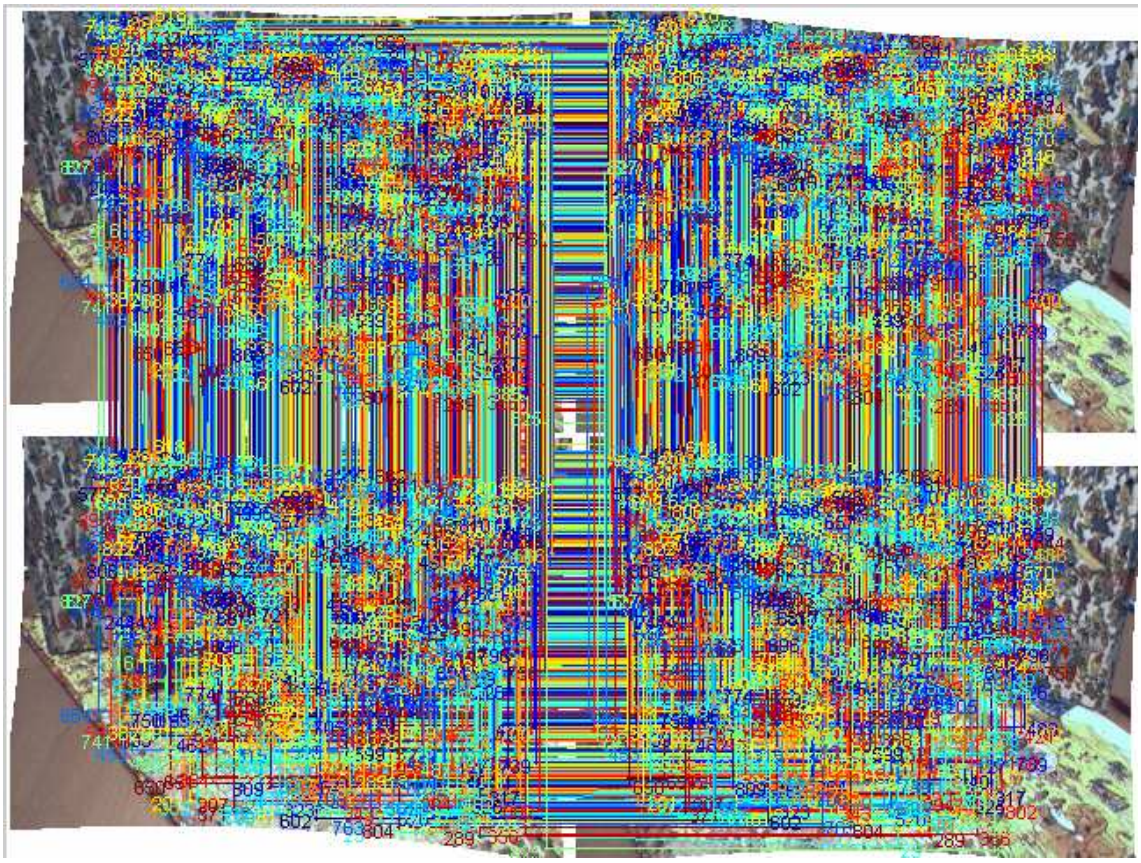


Figura 4.26. Quadruplet correspondències entre el frame 80 i el frame 90 i esquerra-dreta.

Els punts 3D resultats de l'execució del STracker han sigut 497 punts. S'han perdut molts de punts degut a que les càmeres utilitzades no disposen d'una gran qualitat d'imatge degut al soroll que és considerable i a altres factors. A més a més s'han perdut punts del fons on al llarg de la seqüència la caixa de cerilles que s'ha anat desplaçant ha tapat els punts que quedaven al fons, i com que només ens interessen els punts que han aparegut al llarg de tota la seqüència llavors els hem descartat. També hem perdut punts en l'objecte en moviment ja sigui perquè el STracker els ha perdut o bé perquè el sol·lapament entre les dues càmeres per a aquell punt no era prou bo.

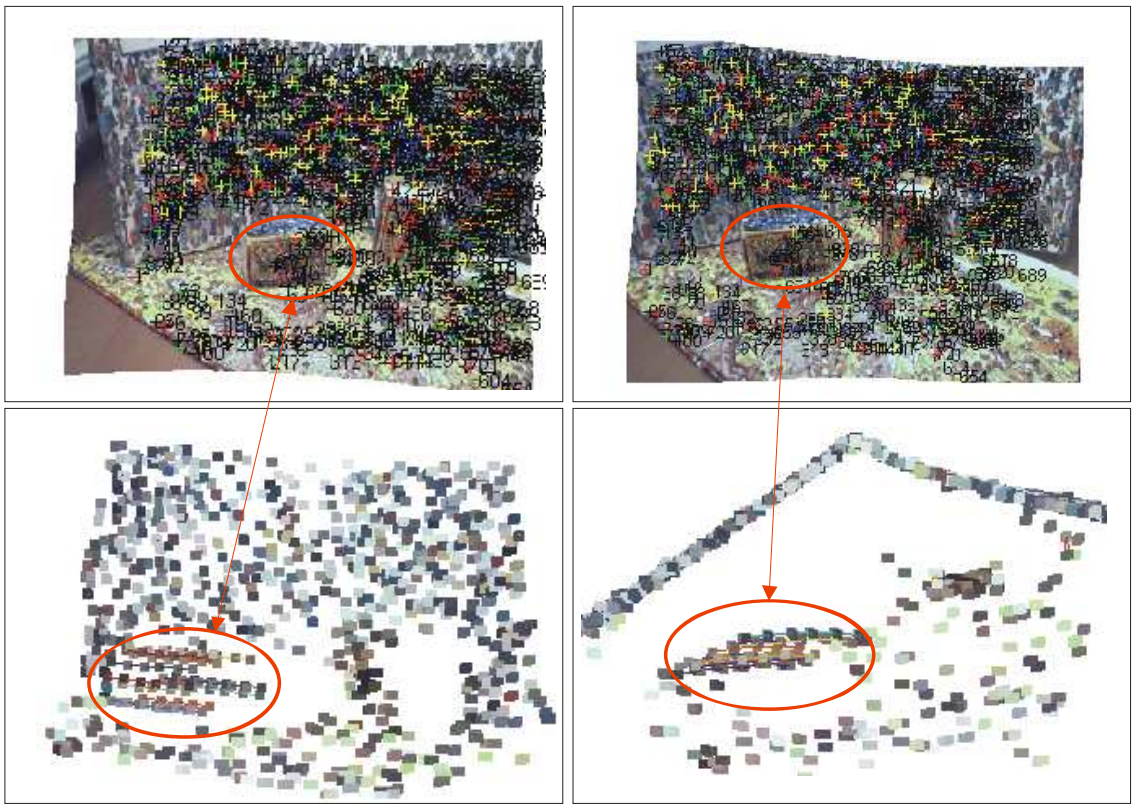


Figura 4.27. Després d'executar el STracker podem visualitzar els punts 3D que han aparegut al llarg de l'escena. A la part superior podem veure les dos imatges des de la posició de la càmera esquerra i dreta respectivament, a la part inferior apareixen tots els punts al llarg de tota la seqüència, encerclats surten els punts no rígids amb la trajectòria que han seguit.

Gràcies a una funció que he implementat per a tractar els punts 3D que ens ha retornat el STracker, he pogut filtrar per fondària els diferents punts i he fet un recompte dels punts de l'objecte rígid i l'objecte que s'ha desplaçat, i el resultat per a cada un és: 9 punts per l'objecte desplaçat i 33 punts per l'objecte rígid i 570 que formen el fons de l'escena, a la figura 4.27 podem veure el comportament dels punts 3D que hem pogut capturar. Així doncs tenim una proporció molt petita de punts no rígids comparats amb els punts rígids en aquesta escena. Tot i això veurem si podem segmentar bé o no.

**Mètode segmentació Otsu**

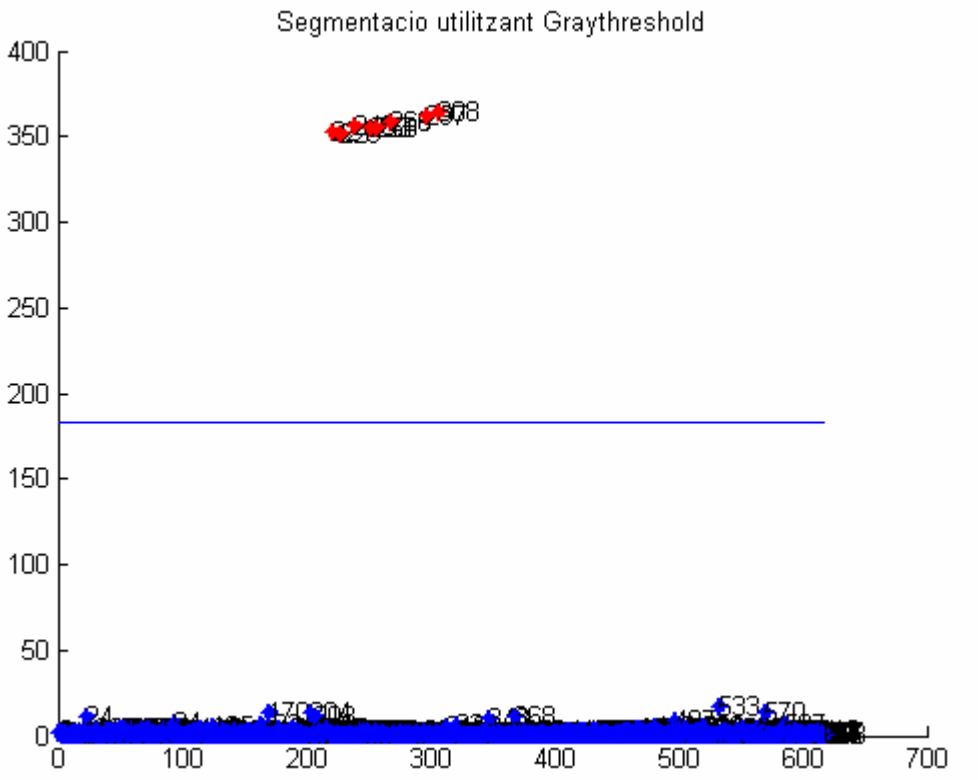



Figura 4.28. Gràfica resultant després d'aplicar el mètode de segmentació Otsu per a les dades. Llindar de segmentació = 183.61.

A la figura 4.28. Podem veure el resultat de segmentar els punts 3D, el resultat és òptim ja que no hem obtingut cap error de segmentació, tot hi haver-hi una proporció molt elevada de punts rígids respecte els punts no rígids, així doncs per aquesta prova 100% d'efectivitat.

**Mètode segmentació K-Means**

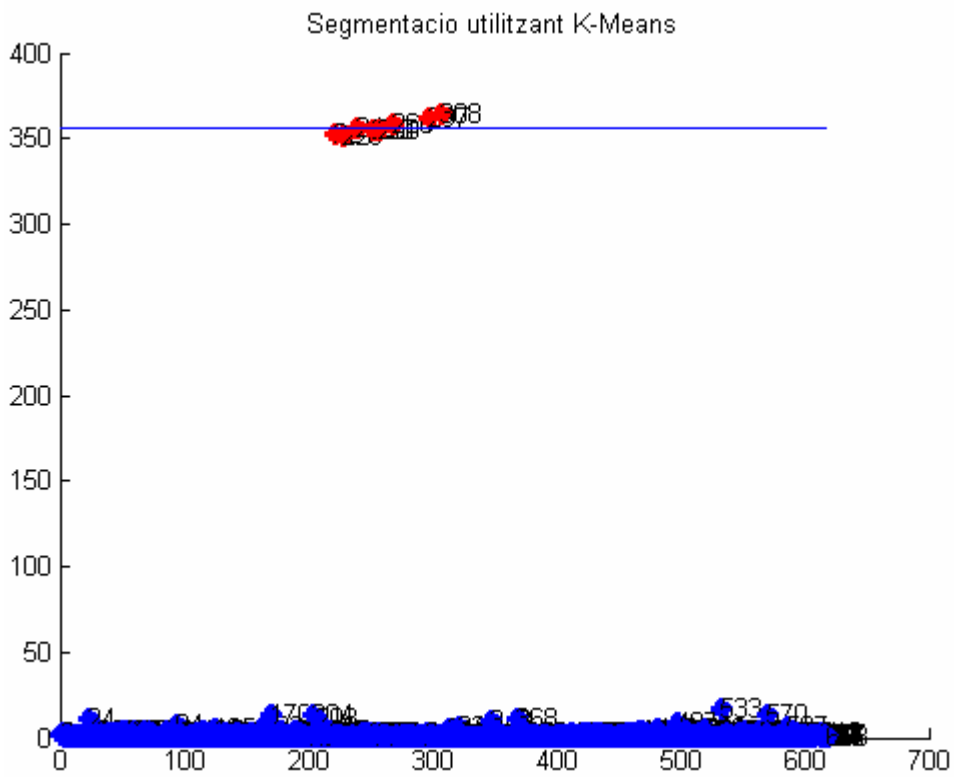



Figura 4.29. Gràfica resultant després d'aplicar el mètode de segmentació K-Means, els centroids són els següents: El centroid per el cluster 1 on hi tenim els punts no rígids és de 0.94 i el centroid del cluster 2 és de 355.95.

Fent servir el mètode K-Means també ens ha donat excel·lents resultats per a aquest experiment, ja que fàcilment aquí podem distingir dos classes degut a la gran diferència d'errors que hi ha entre ambdues classes, a més que els errors estan agrupats quasi al mateix valor, si aquests errors fossin progressius i s'escampessin pel llarg de la gràfica llavors segurament sí que tindríem algun error de segmentació, però no és el cas.

**Mètode segmentació LDA**

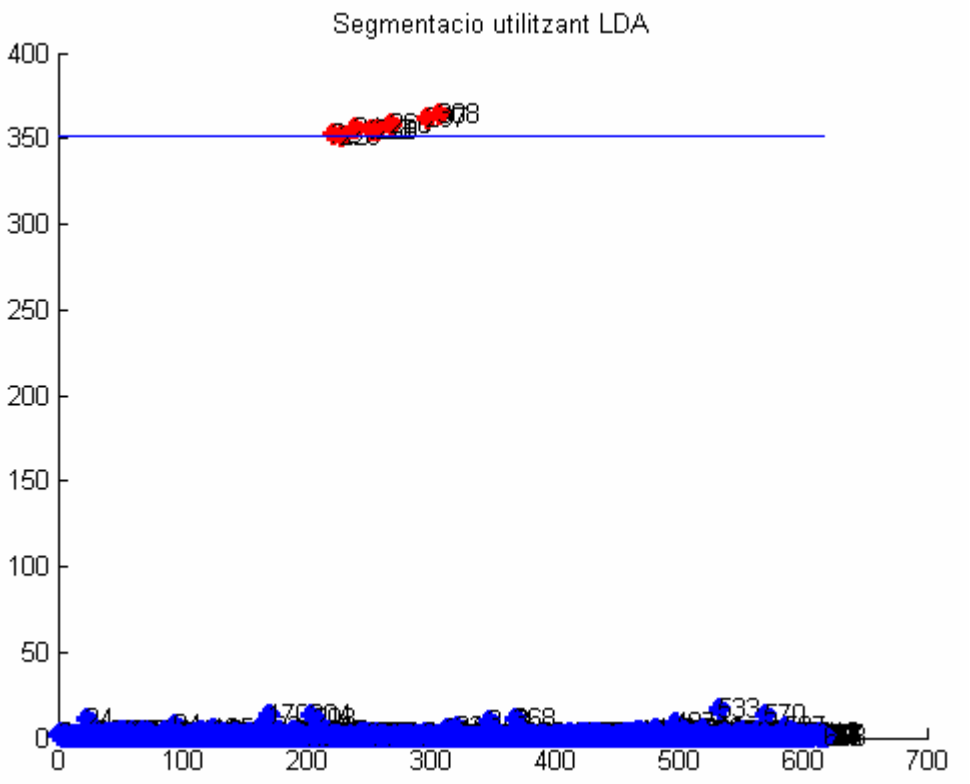



Figura 4.30. Gràfica resultant de l'experiment després d'aplicar el mètode de segmentació de LDA.

Aquest mètode que funciona en base a probabilitats i que les probabilitats assignades per a les dos classes és de 0.9 de probabilitat que un punt sigui rígid i un 0.1 que un punt sigui no rígid. Tot i que la proporció en aquest experiment és de 9/603 i la probabilitat seria molt més baixa ens ha donat excel·lents resultats sense donar-nos cap error de segmentació. Això també és degut a com en els altres mètodes, la gran diferència entre l'error dels punts rígids i l'error dels punts no rígids.

### 4.3.4 Segon experiment: Moviment d'una part de l'escena

El segon experiment serà el de moure una part de l'escenari, és a dir, mourem la meitat del llibre que es sosté dret i mantindrem totalment immòbil l'altre part del llibre, d'aquesta manera ens serà molt més fàcil obtenir gran quantitat de punts, tant per punts rígids com per punts no rígids. A més a més, la deformació que sofrirà no serà tant significativa com en el primer experiment, això significa que probablement la segmentació serà més difícil i imprecisa, i també que al tenir un petit desplaçament radial (des de punt de vista d'obrir i tancar un llibre, el radi respecte el llom és el mateix) els punts que es trobin més proper del mig del llibre patiran menys deformació respecte els punts que es troben més cap a l'extrem dret de la pàgina (veure figura 4.30).

Figura 4.31. En aquesta figura podem veure el primer frame, tant per la imatge esquerra com per la imatge dreta a la part superior de la figura, a la part inferior veiem el frame final per les seves respectives imatges.

Al tractar-se del fons que ocupa pràcticament tota l'àrea de sol·lapament entre les dues càmeres, ens és molt més fàcil trobar punts al llarg de tot l'objecte (el llibre obert). D'aquesta manera tenim una gran quantitat de punts tant per la part rígids com la part deformable, tot i que la primera lògicament en té més ja que també la base està rígida i alguns punts, com en el primer experiment, s'han perdut durant el tracking i per culpa del soroll.

La subseqüència que hem seleccionat per passar-li en el STracker té un total de 80 frames dels quals en total n'utilitzarem 5 ja que realitzarem un salt en la seqüència de 20 frames entre frame i frame, els frames de la seqüència seran els frames: 20, 40, 60 i 80.

A la figura 4.32 podem veure les correspondències dels punts 3D trobats sobre les imatges 2D que ha tractat el STracker.

Figura 4.32. Punts 3D resultants després de l'execució del STracker per a un total de 5 frames. A la part superior els punts 3D visualitzats sobre el primer frame per la càmera esquerra i dreta, i a la part inferior el moviment que han sofert els punts 3D al llarg de la seqüència.

**Mètode Otsu**

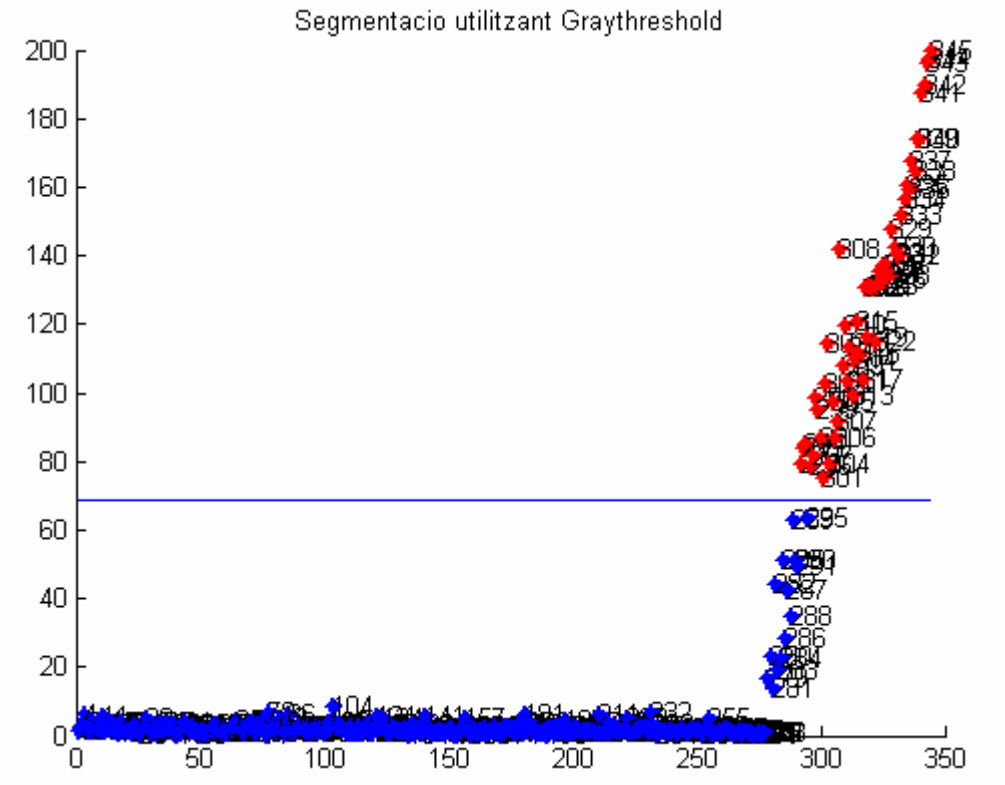



Figura 4.33. Segmentació utilitzant el mètode Otsu

Després d'aplicar el mètode de segmentació utilitzant el mètode Otsu, tal i com podem veure a la figura 4.33, aquí sí que obtenim errors de segmentació,

L'explicació a que l'error de posició és progressiu tal i com es pot veure a la gràfica és degut a que el llibre al tancar-se/obrir-se segueix una trajectòria circular, i lògicament els punts que es troben més proper del centre tindran menys desplaçament que els punts que es troben a l'extrem de la pàgina, si haguéssim pogut utilitzar una seqüència més llarga i obtenir-ne més informació segurament si que haguéssim pogut segmentar correctament, ja que agafem la suma dels errors de posició per un punt que hi ha entre frame i frame. A continuació veurem com es comporten els altres mètodes envers aquest experiment.

Si haguéssim utilitzat més frames, segurament la segmentació ens hagués millorat, degut a que el resultat del sumatori dels errors de posició s'hagués incrementat i per tant la segmentació hagués donat més bons resultats.

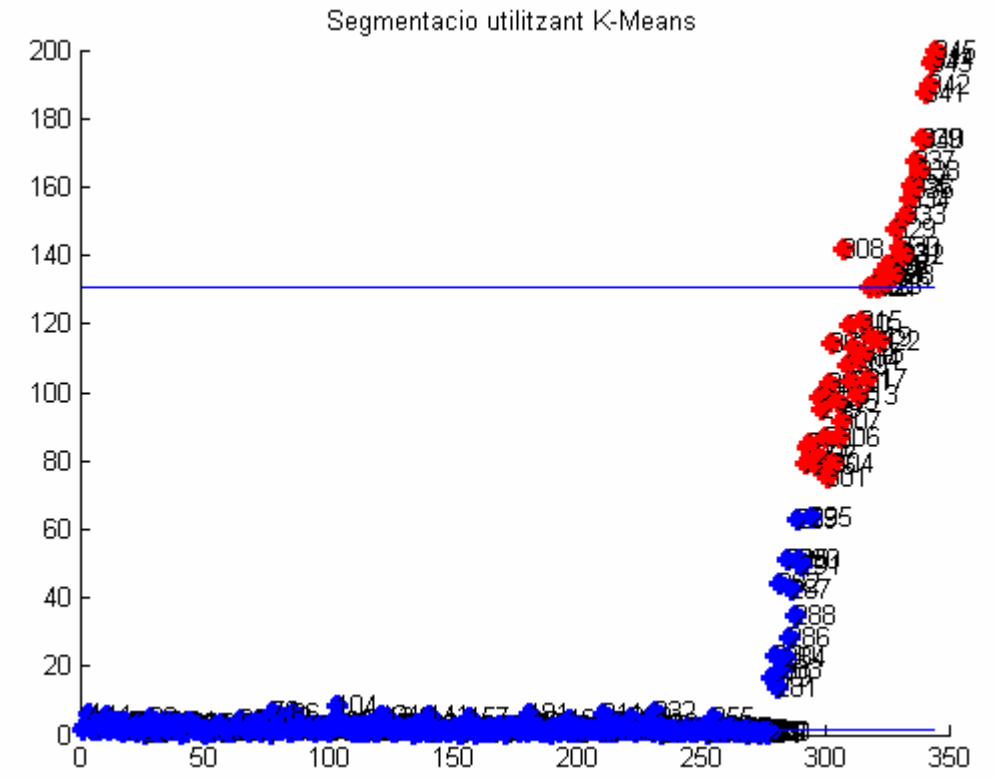



Figura 4.34. Segmentació utilitzant el mètode K-Means.

Aquest mètode al buscar el centre per a dos classes diferents, al ser progressiu els primers punts deformables que apareixen en color blau encara estan més proper del centroid de la classe dels punts rígids que no del centroid dels punts deformables. Si la deformació (error de posició) fos constant o semblant no ens trobaríem amb aquest problema, com en l'exemple anterior. Al funcionar així aquest mètode, és el que pitjors resultats ens ha donat dels 3 mètodes que hem implementat. Per tant podem considerar que no és gaire adient utilitzar aquest mètode on tinguem deformacions d'aquest tipus.

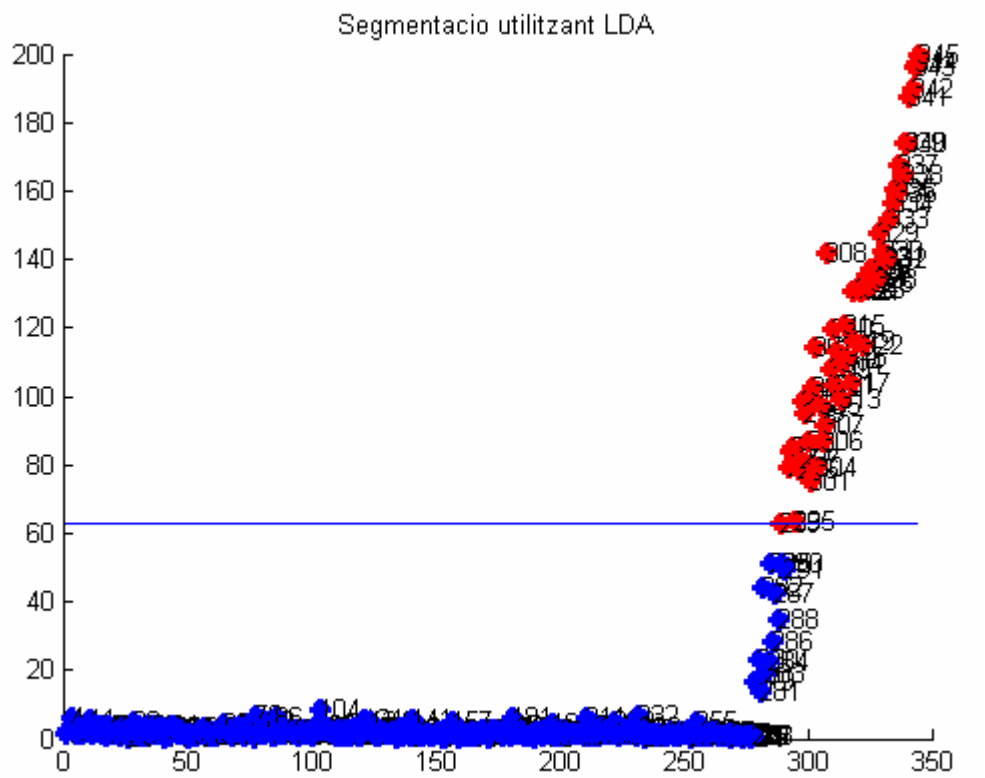



Figura 4.35. Segmentació utilitzant el mètode LDA.

Aquest sembla que és mètode amb menys error de segmentació, inclòs millor que el mètode Otsu on l'anterior els dos punts que estan tocant el llindar en aquest gràfic els considerava com a rígids.

### 4.3.5 Tercer experiment: Simulació d'una deformació movent dos objectes en direccions diferents

En aquest tercer experiment el que farem serà el següent: tindrem dos objectes els quals seran un tetrabrick de llet i una caixeta de cerilles els quals els dos objectes els mourem amb direccions diferents per tal de simular una deformació més real comparada amb el segon experiment on només movíem un sol objecte (una caixeta de cerilles) i l'altre es mantenia immòbil. L'entorn serà el mateix que en els casos anteriors. Aquí un objecte té més moviment que l'altre, per tant alhora de segmentar hauríem de poder veure que hi ha punts que tenen més error de posició que d'altres tot i que la segmentació els hauria de separar correctament de la resta de punts rígids (veure figura 4.36).

En aquest experiment agafarem un rang de 100 frames dels quals agafarem mostres cada 20 frames per a tenir així un total de 6 frames (20, 40, 60, 80, 100 i 120) que si a la captura de vídeo la realitzàvem a 15 frames per segon la seqüència de vídeo que tractem seria de 6'6 segons.

Figura 4.36. En aquest quadruplet podem veure a la part superior el primer frame inicial (frame 20) per a la càmera esquerra i dreta respectivament, i a la part inferior podem veure l'últim frame (frame 120) després per les seves respectives càmeres.

Cal recordar que a mesura que augmentàvem el nombre de frames a analitzar en el STracker anàvem perdent més punts i per tant no ens interessa fer una segmentació amb menys d'uns 15 punts deformables, d'aquesta manera podem provar els mètodes que hem desenvolupat a la part de les dades sintètiques de manera més semblant a aquella secció que no tenint tants pocs punts com tindríem utilitzant una seqüència amb més frames.

Després d'executar el STracker els punts que ens retorna que apareixen al llarg de tota la seqüència de 6 frames són un total de 412 punts. Si mostrem els punts trobats sobre el primer frame tant per la imatge esquerra i dreta podem comptar que tenim un total de 8 punts per a la caixeta de cerilles i per al tetrabrick un total de 41 punts i la resta que són la part rígida, o sigui la part de l'entorn que són 363 punts.

Figura 4.37. Aquí podem apreciar els punts 3D visualitzats sobre les imatges esquerra i dreta respectivament que corresponen al frame inicial de la seqüència que hem tractat a la part superior de la imatge, i a la part inferior els punts 3D al llarg de tota la seqüència on es pot veure el desplaçament que han tingut.

**Mètode Otsu**

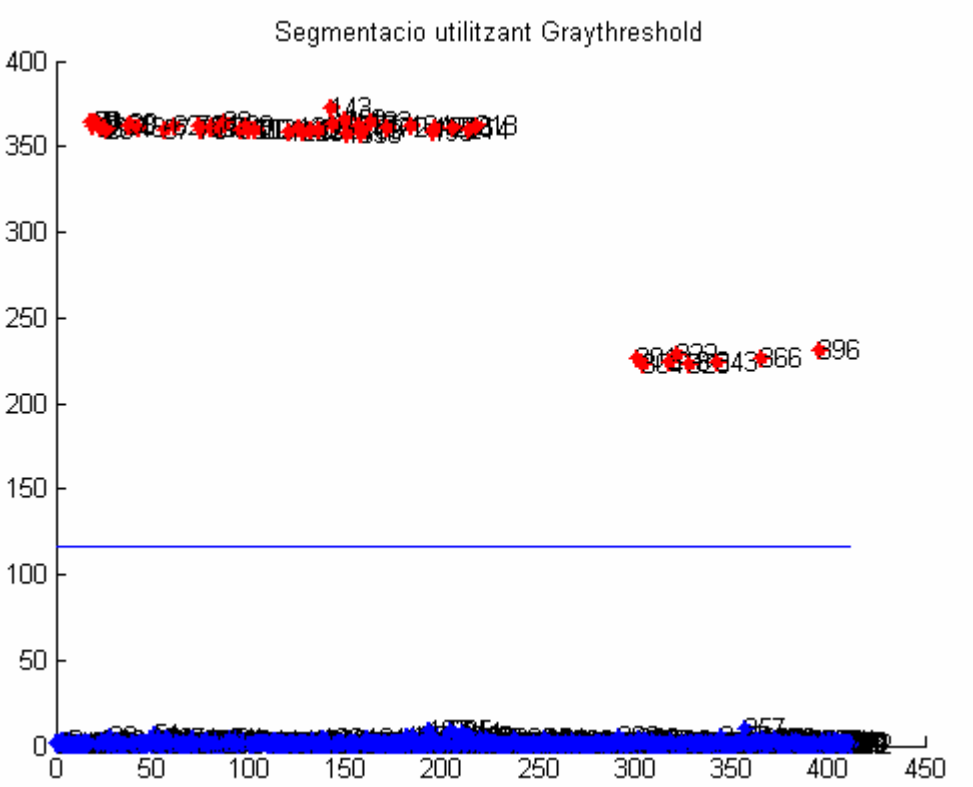



Figura 4.38. Resultat d'aplicar el mètode de segmentació Otsu, llindar = 116.16.

Com podem veure a la figura 4.38 el resultat de la segmentació aplicant el mètode Otsu ha segmentat correctament al 100% exempt d'errors. Al haver-hi tanta deformació i diferència entre els punts rígids i no rígids la segmentació serà probablement sempre perfecte. El llindar de segmentació que ens ha retornat aquest mètode és de 116.16.

**Mètode K-Means**

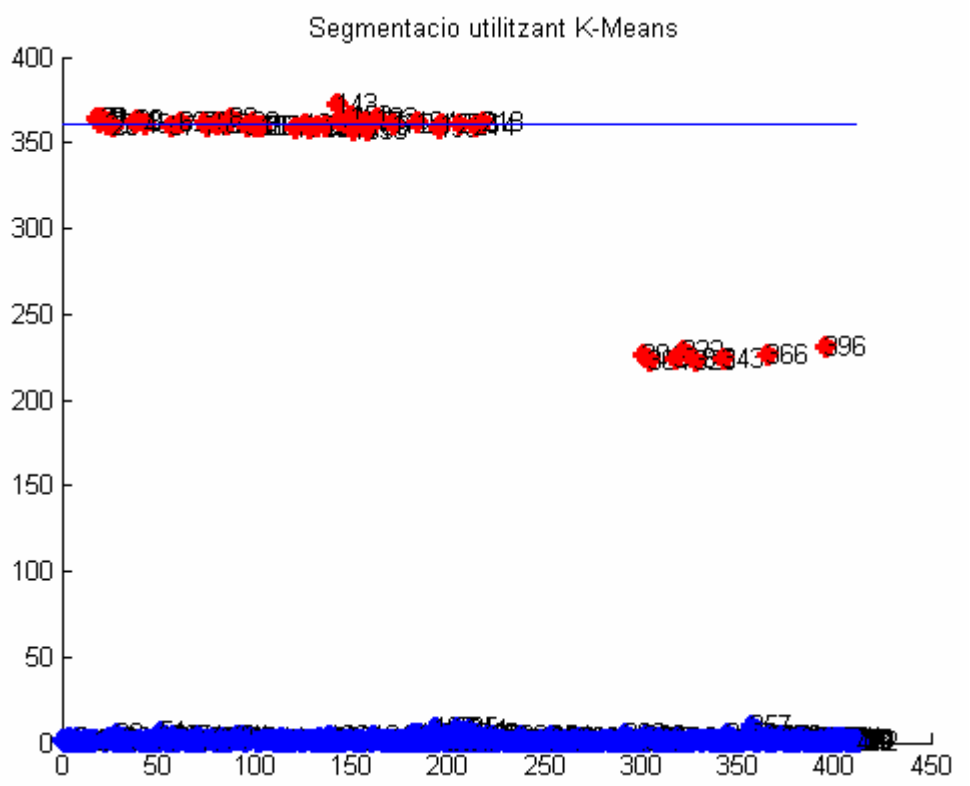



Figura 4.39. Resultat d'aplicar el mètode de segmentació K-means amb dos classes, valors dels centroids: centroid 1 = 1.08, centroid 2 = 360.32.

Aplicant la segmentació utilitzant el mètode K-Means (figura 4.39) podem veure que també hem pogut segmentar correctament el 100% dels punts tal com el mètode anterior. El centroid de la classe dels rígids és de 1.08, tot i que no es pot apreciar a la imatge per la gran quantitat de punts que hi ha propers a 0 i el centroid de la classe dels no rígids és de 360.32.

**Mètode LDA.**

Aquest mètode com que juga amb probabilitats i com que sabem que tenim pocs punts no rígids, més o menys estimem que la relació seria 1 no rígid per cada 10 rígids, dons hi assignem les següents probabilitats: P1 = 0.9 i P2 = 0.1 on P1 seria la probabilitat de la classe dels punts rígids i P2 seria la probabilitat de que un punt sigui de la classe dels punts no rígids.

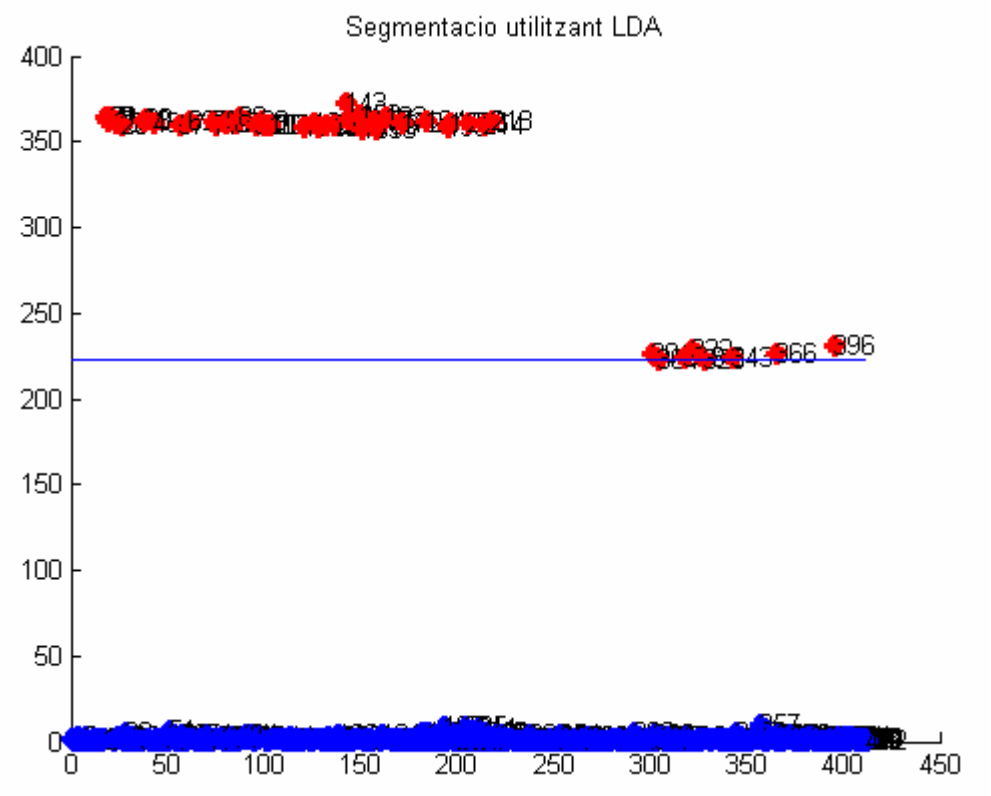



Figura 4.40. Resultat d'aplicar el mètode de segmentació LDA amb probabilitats P1=0.9 i P2=0.1. Llindar = 223.00.

Com podem apreciar a la gràfica la segmentació s'ha realitzat perfectament sense cap error de segmentació. A les dades sintètiques teníem molts errors de segmentació amb aquest mètode degut a que hi teníem molt poca deformació comparada amb la deformació (moviment) que hem sotmès als objectes. La gràfica ens demostra que aquest mètode quan existeix molta deformació, o sigui, molta diferència entre les dos classes i s'hi assigna la probabilitat aproximada per a les corresponents classes ens pot donar resultats 100% fiables com és aquest cas. Tot i que cal recordar que s'ha d'estimar les probabilitats, i no és fàcil ja que a priori no coneixem el percentatge de punts per a cada classe.

### 4.3.6 Quart experiment: Simulació d'una deformació movent 3 objectes en direccions diferents

En aquest últim experiment, provarem què passa quan movem tres objectes en direccions diferents. Un objecte anirà en direcció a la càmera i els altres dos aniran en direcció cap a la dreta des de la perspectiva de les càmeres, tot i que hi haurà un angle diferent en quan a direcció entre aquests dos.

Algun objecte tindrà més moviment que algun altre, igual que en l'experiment anterior. Això ho podrem veure a les gràfiques quan veiem punts que es troben agrupats en un valor d'error que aquest error és l'error de posició que han tingut els objectes respecte el primer frame. A més a més, en aquest experiment no els hi donarem en general tant de moviment com en l'anterior, i trobarem més punts no rígids que en el cas anterior degut a l'aparició d'un altre objecte relativament voluminós per a l'escena (veure figura 4.41).

En aquesta seqüència treballarem sobre un total de 100 frames dels quals els tractarem saltant de 20 en 20 frames. O sigui, els frames que utilitzarem en aquesta seqüència seran els frames: 0, 20, 40, 60, 80 i 100.

Figura 4.41. Quadruplet on veiem a la part superior el primer frame (frame 0) per a la càmera esquerra i la càmera dreta respectivament i a la part inferior podem veure l'últim frame (frame 100) per les seves respectives càmeres.

Figura 4.42. Frame inicial rectificat de la càmera esquerra i la càmera dreta amb els punts 3D que ens ha retornat el STracker. Són els punts que ha pogut seguir al llarg de la seqüència de 6 frames, tal com es mostra a la part inferior d'aquesta figura.

Després d'executar la seqüència tal com es pot veure a la figura 4.42, hi tenim un total de 338 punts dels quals 61 punts són no rígids i 277 són rígids. Dels 61 que no són rígids tenim 6 són de la caixeta de cerilles, 21 són de la caixeta que està tombada horitzontalment i la resta, 34, són del tetrabrick de llet. Aquest són els punts que apareixen al llarg de tota la seqüència de 6 frames que ha tractat el STracker. Així doncs la proporció entre punts rígids i no rígids és lleugerament superior a 1/5, 11/50 exactament.

L'objecte que ha sofert menys moviment és el tetrabrick de llet on la suma dels seus errors de posició per punt frame a frame es troba a prop dels 110mm. Llavors els altres dos objectes, la caixa de cerilles i l'altre caixeta de cartró que està tombada horitzontalment han tingut un desplaçament molt semblant : la suma dels errors de posició per la caixeta per tots els seus punts es situen al voltant dels 225mm i l'altre objecte al voltant dels 210-218mm. A continuació a les gràfiques quan realitzem la segmentació ho podrem comprovar.

**Segmentació utilitzant el mètode Otsu**

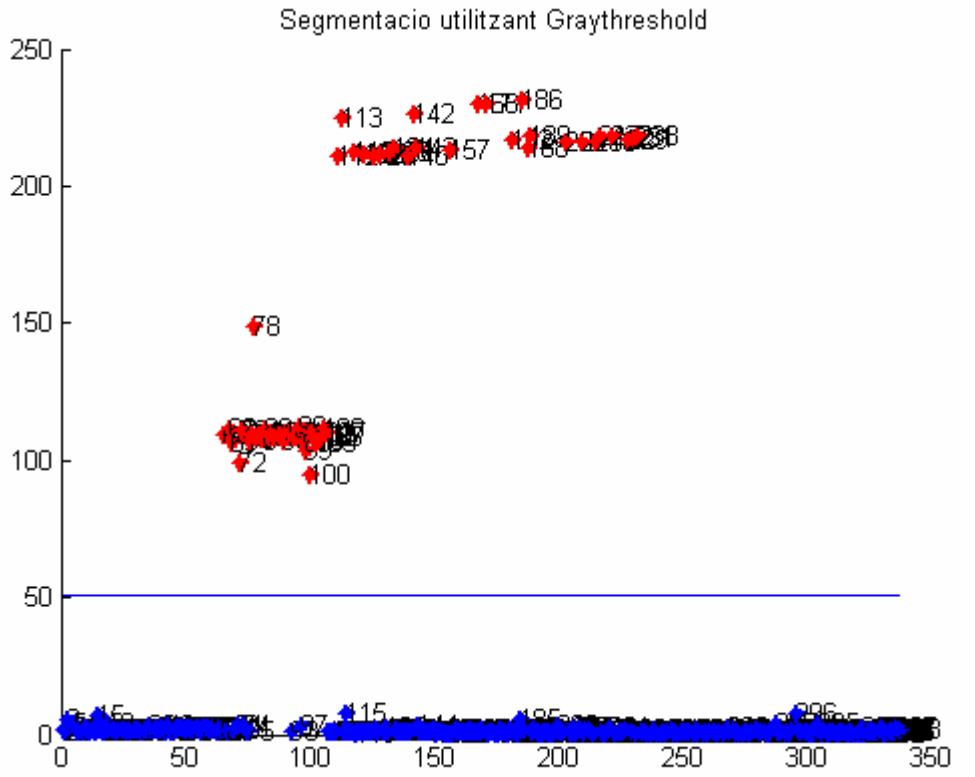



Figura 4.43. Segmentació dels punts 3D utilitzant el mètode Otsu. Llindar = 50.74.

En aquest altre cas la segmentació també ens ha donat bons resultats tot i no haver-hi tanta deformació com en el cas anterior i al haver-hi un objecte que ha tingut un desplaçament en què l'error de posició es troba justament entre els punts dels objectes que han sofert més moviment i els punts rígids. Tot i això amb el mètode Otsu hem segmentat al 100% correctament utilitzant un llindar de 50.74.

**Segmentació utilitzant el mètode K-Means**

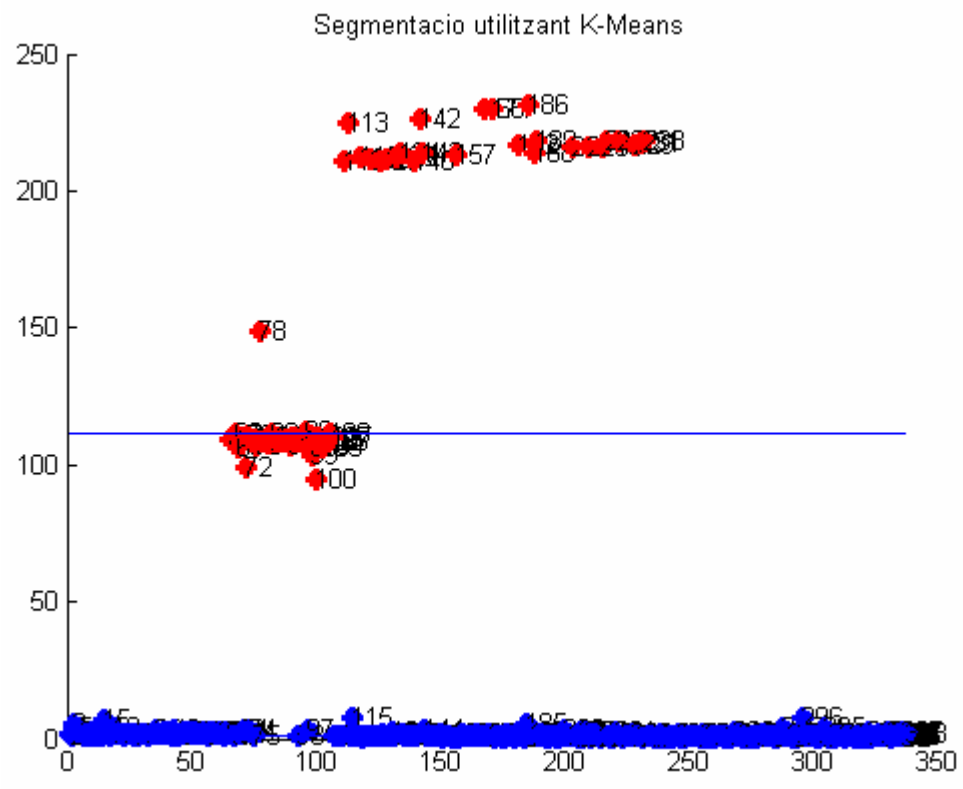



Figura 4.44. Segmentació utilitzant el mètode K-Means. Valors dels centroids: centroid 1 = 1.15. Centroid 2 = 111.28.

Un altre cop el procés de segmentació utilitzant aquest mètode no ens ha donat cap error de segmentació. El centroid del cluster 1 es troba a 1.15, la línia no es pot apreciar degut a la quantitat de punts que es troben propers a 0. i el centroid per al cluster 2 (el grup dels punts no rígids) es troba a 111.28 molt a prop del grup de punts no rígids del tetrabrick que han sofert poc moviment envers els punts trobats pel STracker dels altres dos objectes.

**Segmentació utilitzant el mètode LDA**

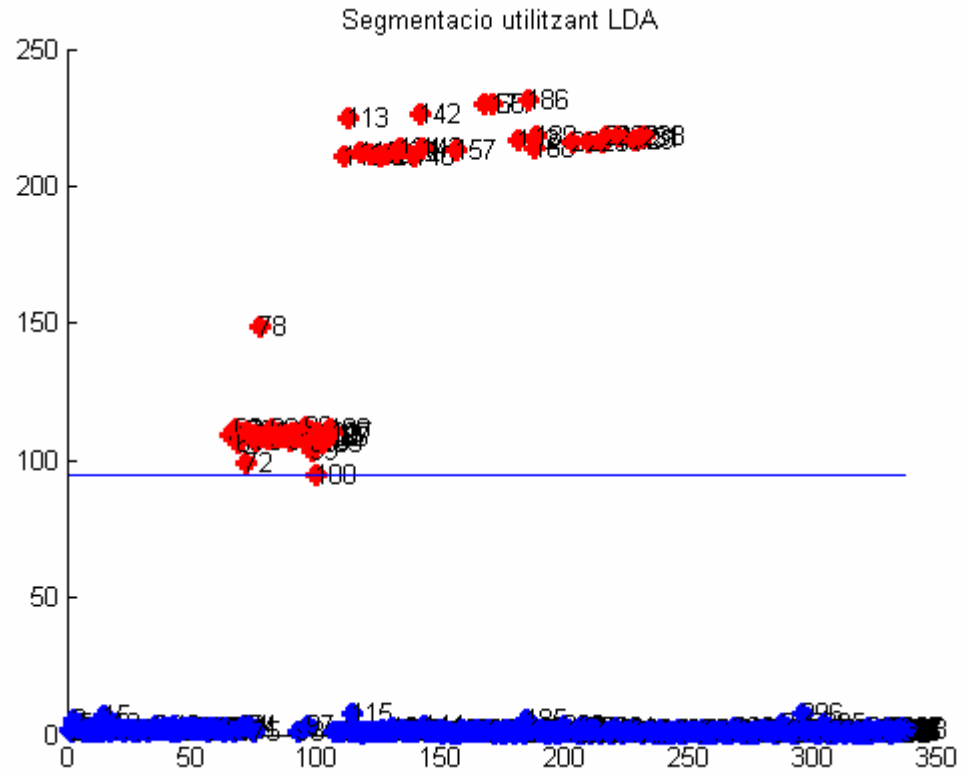



Figura 4.45. Segmentació utilitzant el mètode LDA. Llindar = 94.55.

Finalment l'últim mètode de segmentació que implementem, el LDA, també ens ha segmentat correctament tots els punts d'aquest experiment utilitzant un llindar de 94.55. Sembla doncs que tampoc influeix que hi hagi grans diferències entre els errors de posició per als diferents 3 objectes alhora de segmentar utilitzant aquest mètode. Tot i que la proporció de punts no rígids/rígids és d'aproximadament 1/5 aquí hem utilitzat les mateixes probabilitats que en els casos anteriors: 0.9 de probabilitat que un punt sigui un punt rígid i un 0.1 de probabilitat que un punt sigui no rígid, tot i això ho ha segmentat perfectament.

# Capítol 5

# Conclusions i ampliacions futures

## 5.1 Conclusions

Al llarg del desenvolupament d'aquest projecte hem anat assolint tots els objectius que ens vàrem proposar al principi del projecte i de la documentació, tal i com estan llistats a l'inici d'aquesta documentació al capítol de la introducció.

A partir d'aquest projecte ens hem introduït al món de la reconstrucció 3D fent unes proves reals de reconstrucció a partir d'un sistema estèreo. Després de desenvolupar el sistema estèreo, hem après a realitzar un acurat procés de calibració de les càmeres, primer calibrant individualment les càmeres per a després realitzar el procés de calibració estèreo per a l'extracció dels paràmetres extrínsecs. Amb una bona calibració llavors hem pogut fer unes captures reals de vídeo per a generar-ne la reconstrucció 3D a partir d'una seqüència d'imatges on hem realitzat un seguiment de diferents punts que es trobaven en moviment per a després posar a prova els mètodes que hem desenvolupat per a realitzar una segmentació de punts rígids i no rígids de l'estructura 3D al llarg d'una seqüència.

Els mètodes de segmentació que hem analitzat, dissenyat i implementat per a segmentar punts rígids i no rígids, s'han avaluat primer sobre un conjunt gran de dades sintètiques, amb les que teníem control total d'aquestes, on podíem jugar amb tots els paràmetres possibles d'aquestes dades, tals com serien: la deformació que patirien els punts 3D no rígids, afegint-hi soroll gaussià, augmentant o disminuint tots els punts, tant rígids com no rígids.

Els primers resultats obtinguts han sigut durant l'avaluació dels mètodes de segmentació sobre les dades sintètiques. Els mètodes que hem avaluat han sigut

el mètode d'Otsu, el mètode K-Means (utilitzant dos classes) i el mètode LDA (Linear Discriminant Analysis). Amb aquestes dades sintètiques cap mètode ens ha donat un 100% de segmentació correcta amb tot aquest conjunt de dades, en cada un dels mètodes hem obtingut errors de segmentació, tot i que s'ha de dir que algun mètode ha donat més errors de segmentació que els altres. Això a mesura que variàvem els paràmetres de les dades sintètiques (variant nombre de punts rígids o no rígids, soroll i deformació) aconseguíem diferents resultats de segmentació amb diferents errors de segmentació. El mètode que ens ha donat més bons resultats en aquest apartat és el mètode d'Otsu on hem obtingut molts pocs errors de segmentació en comparació amb els altres dos mètodes, llavors el segueix el mètode K-Means en què també ens ha donat forces bons resultats i finalment el mètode LDA que ens ha donat els pitjors resultats.

Els factors que han influït més alhora de generar-nos errors de segmentació ha sigut sobretot el del nivell de deformació. Quan generàvem poca deformació per als punts no rígids la segmentació es complicava molt, ja que les dos classes (rígids i no rígids), el seus valors d'error de posició que els diferenciàvem mútuament era molt menor que si hi tinguéssim molta més deformació, tal i com ens ha passat en els experiments reals. Llavors un altre factor que també ens ha influït molt alhora de generar-nos errors de segmentació és la proporció entre punts rígids i no rígids, en algun mètode ens ha influït més que un altre degut al tractament de dades que realitzen cada un dels mètodes. Per exemple, el mètode K-Means ens ha donat més errors de segmentació quan teníem 10 punts rígids i 50 punts no rígids que el mètode LDA on pràcticament no hem obtingut errors de segmentació. D'aquesta manera podem deduir que aquests mètodes de segmentació es poden aplicar en diferents circumstàncies i obtenir més bons resultats amb un mètode que l'altre, i al revés amb diferents circumstàncies, tal com aquest exemple que acabem de comentar.

Llavors, després d'observar els resultats en aquestes dades sintètiques, amb les captures reals que hem fet després de realitzar una bona calibració del sistema i utilitzant el STracker per a fer un seguiment del conjunt de punts 3D localitzats en una escena al llarg d'una seqüència hem convertit el conjunt de dades que ens ha retornat el STracker a on conjunt de dades on les deixàvem llestes per a aplicar els tres mètodes de segmentació que havíem posat a prova amb les dades sintètiques.

Els resultats que hem obtingut amb les dades reals han estat satisfactoris, ja que en tots els experiments que hem realitzat, excepte un on teníem una deformació progressiva punt a punt, doncs no hem obtingut cap error de segmentació. El motiu principal d'aquest fet és que en les proves reals, els objectes els quals hem donat moviment (simulant deformacions) els hi hem donat molta més deformació que en les dades sintètiques, on aquesta deformació era molt més petita. I en el cas excepcional que hem comentat, la deformació on era progressiva (experiment 2 del capítol de dades reals), el mètode que ens ha donat més bons resultats és el mètode LDA, on hem obtingut menys errors de segmentació que els altres dos mètodes, tot i ser el mètode que pitjors resultats ens havia donat quan hem avaluat els mètodes sobre dades sintètiques.
Amb aquests resultats que hem obtingut al llarg del projecte, tant en dades sintètiques com en dades reals podem arribar a les conclusions que cada mètode pot ser utilitzat depenent dels factors que hem comentat obtenint així menys errors de segmentació utilitzant un mètode en comptes d'un altre. Tot i que en general, podem concloure que Otsu és una bona opció.

## 5.2 Ampliacions futures

Al llarg d'aquest projecte hem anat assolint cada un dels objectius que ens hauríem proposat al principi del projecte, però tot i això encara ens podríem proposar altres objectius per a seguir tractant i ampliant aquest tema, com podrien ser:

- **Avaluació dels mètodes de segmentació amb més experiments reals:** Podríem provar els tres mètodes de segmentació que hem estudiat amb més experiments reals on hi intervinguessin més factors, com una seqüència més llarga, menys deformació pels punts no rígids, etc...

- **Avaluació d'experiments reals amb càmeres d'alta resolució:** En aquest projecte hem treballat amb dues càmeres FireWire on la seva resolució és de 640x480 píxels i tenen un soroll considerable. Si provéssim la Toolbox junt amb el STracker utilitzant unes càmeres d'alta resolució, el més segur és que obtinguéssim un total de punts 3D per a una seqüència molt més elevat que els que hem obtingut amb aquestes càmeres.

- **Ampliació del conjunt de mètodes de segmentació:** En aquest projecte només hem provat tres mètodes diferents per a segmentar punts rígids i no rígids. Per tant, segurament trobaríem altres mètodes els quals podríem adaptar al camp que hem tractat per a realitzar segmentacions entre punts rígids i no rígids.

- **Implementació dels mètodes en C o C++:** Durant aquest projecte hem treballat exclusivament amb Matlab, llavors si dissenyéssim i implementéssim tots els mètodes i funcions que hem desenvolupat per la nostre Toolbox les desenvolupéssim amb C o C++ guanyaríem molt en quant a eficiència i podríem treballar amb un conjunt de dades molt més elevat amb un temps inferior

- **Reconstrucció 3D del model deformable:** Amb els mètodes que hem desenvolupat i provat durant aquest projecte podríem fer la reconstrucció 3D del model deformable després de la segmentació de punts rígids i no rígids, utilitzant tècniques de "Non-rigid Structure from Motion".

# Bibliografia

[1] Jean-Yves Bouguet “Camera Calibration Toolbox for Matlab.” http://www.vision.caltech.edu/bouguetj/calib_doc/index.html. 2009

[2] “Middlebury Stereo Vision Page.” http://vision.middlebury.edu/stereo/. 2009

[3] Xavier Lladó, Alessio Del Blue, Lourdes Agapito"Recovering Euclidian Deformable Models from Stereo-motion", ICPR (International Conference on Pattern Recognition), 2008

[4] Ricard Campos Dausà "Mètodes de reconstrucció 3D d'objectes a partir del moviment en seqüències d'imatges". PFC, ETIS. Juny 2007

[5] Jordi Ferrer Plana "Optical Seafloor Mapping", Tesis Doctoral, Juny 2008

[6] Augusto Salazar, Luis Gonzalez Sanchez, Flavio Prieto "Sistema de adquisición de imágenes de rango con base en estèreo-activo", 2006

[7] Three-Dimensional Computer Vision: A Geometric Approach, O. Faugeras, MIT Press, 1996, pp. 33-68

[8] Multiple View Geometry in Computer Vision, R. Hartley and A. Zisserman, Cambridge University Press, 2000, pp. 138-183

[9] Iryna Gordon and David G. Lowe, "What and where: 3D object recognition with accurate pose," in Toward Category-Level Object Recognition, (Springer-Verlag, 2006), pp. 67-8

[10] Univesritat de Girona, "Apunts Visió per computador Tema 3 - Formació de la imatge", 2009

[11] Universitat de Girona, "Apunts Visió per Computador Tema 6 - Visió 3D", 2009

[12] Pablo Figueroa, Apunts Universidad de los Andes "Geometría de dos vistas", 2008

[13] Instituto de Ingeniería Eléctrica, Universidad de la República "Estimación de la geometría en visión por computador", Tema 4, 2008

[14] Robyn Owens. "Epipolar geometry". http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT10/node3.html, 2003

[15] Luís Miguel Jiménez, José María Sebastián, Grupo de Visión, UPM, "Estimación Robusta - RANSAC", 2008

[16] Apunts Universidad Nacional de Quilmes - "Segmentación por Umbralización, Mètodo Otsu", 2005

[17] "K-Mean Clustering Tutorials" http://people.revoledu.com/kardi/tutorial/kMean/index.html, 2009

[18] "Discriminant Analysis Tutorial" http://people.revoledu.com/kardi/tutorial/LDA/index.html, 2009

# Toolbox Matlab: Manual per a la utilització de funcions.

La Toolbox que hem desenvolupat al llarg d'aquest projecte conté tot un conjunt de funcions les quals ens han sigut de gran utilitat per a posar a prova els diferents mètodes, generar dades i per tractar-les. A continuació es mostra un esquema de les funcions agrupades per camp.

- **Funcions per a dades sintètiques.**
  - error_per_punt
  - error_ransac
  - segmenta_dades_sintètiques
  - guarda_dades

- **Funcions per a la segmentació**
  - segmentacio_otsu
  - segmentacio_kmeans
  - segmentacio_lda

- **Funcions per a tractar les dades reals.**
  - TestReal
  - filtra_fondaria
  - seleccionar_punts_aleatoriament
  - calcula_resultats_dades_reals
  - segmenta_dades_reals

- **Funcions per a visualitzar.**
  - plot_3D
  - visualitza_3D
  - mostra_punts

- **Altres funcions**
  - distancia_euclidiana
  - carrega_calibracio

- **Funcions externes**
  - rotation_matrix2euler

- rotation_euler2matrix
- ransac
- ransacRT
- stereo_triangulation

Després de llista totes les funcions que hem desenvolupat i utilitzat al llarg del projecte, ara entrem a detallar-les més.

# Funcions sobre dades sintètiques

### ➢ Funció error_per_punt

**Descripció:** Aquesta funció s'encarrega de fer el sumatori de l'error de posició després d'executar Ransac i fer la transformada inversa per cada punt. Retorna una matriu amb tantes columnes com punts hi hagi, i tantes files com frames tingui la seqüència.

**Invocació:** ret = error_per_punt(Mat)

**Input:**

Mat: matriu que conté l'error de posició per a frame punt a punt, hi ha tants punts com nombre de files i tants frames com columnes.

**Output:**

ret = retorna un vector després de fer tot el sumatori de l'error de tots els frames per a cada punt.

### ➢ Funció error_ransac

**Descripció:** Aquesta funció realitza la funció de calcular l'error de posició d'un punt. Aquest error el calcula fent la diferència de la posició del punt en el frame original respecte la posició en què es troba en frame que es tracta després haver fet la transformada inversa en rotació i translació.

**Invocació:** error_posicio = error_ransac(coord1, coord2, R, t)

**Input:**

coord1: coordenada 3D del frame inicial.

coord2: coordenada 3D del frame actual.

R: matriu de rotació retornada per RansacRT.

t: vector de translació retornada per RansacRT.

**Output:**

error_posició: error en mil·límetres de la diferència entre coord1 i el resultat de fer la transformada inversa amb R i t a coord2.

### ➢ Funció segmenta_dades_sintetiques

**Descripció:** Aquesta funció el que realitza és crear-ne tot un conjunt gran de dades sintètiques aleatòries, i guarda totes les dades generades en diferents fitxers els qual el Matlab carregarà en el *workspace*. Les dades sintètiques es separen per el total de punts rígids, punts no rígids i per nivell de deformació. Crida a la funció crea_dades passant els paràmetres corresponents per a generar aquests fitxers.

**Invocació:** [l_otsu, l_kmeans, l_lda] = segmentadades_sintetiques(errors)

**Input:**

errors: Vector amb el sumatori dels errors de posició per a cada punt.

**Output:**

l_otsu: llindar retornat del mètode Otsu.

l_kmeans: vector de tamany dos, amb els valors dels dos clusters de cada classe retornat per el mètode K-Means.

l_lda: llindar retornat del mètode LDA.

### ➢ Funció guarda_dades

**Descripció:** La funció guarda dades s'encarrega de guardar els resultats de la segmentació de les dades sintètiques prèviament calculats amb la funció segmenta_dades_sintetiques. Separa en tres fitxers els quals són per a cada un dels mètodes de segmentació (Otsu, K-Means i LDA). Utilitza les dades del *workspace*.

**Invocació:** guarda_dades()

**Output:**

Crea els fitxer segmentació_otsu.txt, segmentació_kmeans.txt, segmentació_lda.txt

## Funcions per a la segmentació

### ➢ Funció segmentacio_otsu

**Descripció:** Executa el mètode Otsu explicat en aquest PFC. L'entrada d'aquesta funció és un vector el qual és el sumatori de tots els errors de

posició per a cada un dels punts. Donats aquests valors realitza la segmentació utilitzant el mètode Otsu.

**Invocació:** llindar = segmentacio_otsu(sum_errors)

**Input:**

sum_errors: Vector amb el sumatori dels errors de posició per a cada punt

**Output:**

llindar: llindar de segmentació.

### ➢ Funció segmentacio_kmeans

Descripció: Executa el mètode K-Means utilitzant una inicialització aleatòria dels clústers utilitzant dos classes (punts rígids i punts no rígids). L'entrada d'aquesta funció és un vector el quals cada un dels seus valors, com en la funció anterior, és el sumatori de l'error de posició per a cada un dels punts. Amb aquests valors segmenta utilitzant el mètode K-Means

**Invocació:** llindar = segmentacio_kmeans(sum_errors)

**Input:**

sum_errors: Vector amb el sumatori dels errors de posició per a cada punt

**Output:**

llindar: llindar de segmentació.

### ➢ Funció segmentacio_lda

**Descripció:** Executa el mètode LDA utilitzant una probabilitat del 0.5 per a les dos classes. El paràmetre d'entrada és el mateix que en els dos mètodes anteriors.

**Invocació:** llindar = segmentacio_lda(sum_errors)

**Input:**

sum_errors: Vector amb el sumatori dels errors de posició per a cada punt

**Output:**

llindar: llindar de segmentació.

# Funcions per a tractar les dades reals

## ➢ Funció TestReal

**Descripció:** Donada una seqüència de X frames, realitza tot el procés d'executar tot el STracker i retornar els punts 3D els quals s'han trobat al llarg de tota la seqüència.

**Invocació:** TestReal()

**Input:**

Treballa amb les dades que estan guardades en fitxers amb extensió Matlab on hi ha guardats els paràmetres de calibració, i amb les dades carregades al workspace sobre la seqüència i altres constants, variables i opcions que es troben dintre aquesta funció.

**Output:**

Guarda al workspace tots els punts 3D que han aparegut al llarg de la seqüència, també guarda al workspace els punts 2D corresponents a cada imatge de cada frame per a la càmera esquerra i càmera dreta.

## ➢ Funció filtra_fondaria

**Descripció:** Passant per paràmetre una fondària mínima i una fondària màxima en mil·límetres, eliminem tots els punts 3D que es troben fora de l'interval entre la mínima i la màxima que ens ha retornat el STracker. Si passem l'últim paràmetre, frame, podrem visualitzar els mateixos punts en 2D sobre el frame corresponent tant per la imatge esquerra com per la imatge dreta.

**Invocació:** llista_punts = filtra_fondaria(punts3D, fond_min, fond_max, frame)

**Input:**

punts3D : llista de punts 3D

fond_min : fondària mínima.

fond_max : fondària màxima.

frame : (opcional), frame per a visualitzar els punts en 2D sobre el frame indicat.

**Output:**

llista_punts : punts que es troben entre fond_min i fond_max

### ➢ Funció seleccionar_punts _aleatoriament

**Descripció:** Donats tots els punts 3D que ens ha retornat el STracker, seleccionem el nombre de punts que es passa per paràmetre per seleccionar-los aleatòriament utilitzant la funció *randint* de Matlab.

**Invocació:** llista_punts= seleccionar_punts_aleatoriament(llista, num)

**Input:**

llista: llista d'índexs de punts.

num: número de punts a seleccionar aleatòriament.

**Output:**

llista_punts: llista de punts aleatoris retornats.

### ➢ Funció calcula_resultats_dades_reals

**Descripció:** Aquesta funció una vegada executada la funció TestReal el qual ens ha retornat un conjunt de punts 3D que han aparegut al llarg de la seqüència real, junt amb la posició de cada punt per a cada frame de la seqüència, ens retorna un vector amb el sumatori els errors de posició per a cada un dels punts 3D al llarg de la seqüència després de fer la transformada inversa que ens ha retornat Ransac.

**Invocació:** errors = calcula_resultats_dades_reals(mat)

**Input:**

mat: matriu amb 3xn files les quals són les coordenades dels punts 3D per a cada columna que són els frames.

**Output:**

errors: sumatori dels errors de posició

### ➢ Funció segmenta_dades_reals

**Descripció:** Passant el sumatori dels errors de posició per a una seqüència real, s'encarrega de cridar tots els mètodes de segmentació que hem treballat al llarg d'aquest PFC (Otsu, K-Means i LDA).

**Invocació:** [l_otsu, l_kmeans, l_lda] = segmenta_dades_reals(errors)

**Input:**

errors: Vector amb el sumatori dels errors de posició per a cada punt.

**Output:**

l_otsu: llindar retornat del mètode Otsu.

l_kmeans: vector de tamany dos, amb els valors dels dos clusters de cada classe retornat per el mètode K-Means.
l_lda: llindar retornat del mètode LDA.

# Funcions per a visualitzar

## ➢ Funció plot_3D

**Descripció:** Donada una matriu 3xn punt, mostra un plot amb cada un dels punts en tres dimensions.
**Invocació:** plot_3D(p)
**Input:**
p: punts 3D.
**Output:**
Obre un plot 3D.

## ➢ Funció visualitza_sequencia_3D

**Descripció:** Donada una matriu (3xframes)x nombre de punts 3D, aquesta funció ens mostra tots els punts al llarg de la seqüència sobre un plot.
**Invocació:** visualitza_sequencia_3D(mat)
**Input:**
mat: matriu de 3xN files (N són els punts) i m columnes (frames).
Output :
Obre un plot amb tota la seqüència 3D per a tots els punts.

## ➢ Funció mostra_punts

**Descripció:** Passant per paràmetre un vector amb el número de punts que es volen visualitzar es mostren els punts en 2D equivalents en la imatge esquerra i la imatge dreta, indicant també sobre quin frame es volen obrir els dos plots.
**Invocació:** mostra_punts(punts3D, frame)
**Input:**
punts3D: matriu coordenades dels punts 3D (3xn punts).
frame: frame a visualitzar per als dos plots

## Altres funcions

### ➢ Funció distancia_euclidiana

**Descripció:** Donats dos punts 3D retorna la distància euclidiana entre ells.

**Invocació:** dist = distancia_euclidiana(p1,p2)

**Input:**

p1: coordenades punt 1.

p2: coordenades punt 2.

**Output:**

dist: distància euclidiana entre els dos punts

## Funcions externes

### ➢ Funció rotation_matrix2euler

**Descripció:** Retorna els angles eulerians d'una matriu de rotació.

**Invocació:** R = rotation_matrix2euler(mat_R)

**Input:**

mat_R: matriu de rotació 3x3.

**Output:**

R: vector 3x1 amb els angles eulerians.

### ➢ Funció rotation_euler2matrix

**Descripció:** Retorna una matriu de rotació d'un vector d'angles eulerians.

**Invocació:** mat_R = rotation_matrix2euler(R)

**Input :**

R : vector 3x1 amb els angles eulerians.

**Output :**

mat_R : matriu de rotació 3x3.

### ➢ Funció ransacRT

**Descripció:** Funció que passada una tolerància per a trobar inliers executa el mètode Ransac donada una matriu 3xn  punts. Calcula el model de la transformada, tant en rotació com en translació, entre dos frames utilitzant

un mínim de tres inliers per a trobar el model sempre i quan es trobi amb un llindar. Retorna una matriu de rotació i el vector de translació junt amb els inliers trobats.

**Invocació:** [R t inliers] = ransacRT(p1, p_orig, llindar)

**Input:**

p1: matriu 3xn punts amb tots els punts del frame actual

p_orign: matriu 3xn punts amb les coordenades originals de la seqüència

llindar: llindar per a trobar inliers

**Output:**

R: matriu de rotació 3x3

t: vector de translació 3x1

Inliers: llista d'inliers utilitzats per a trobar el model de rotació i translació

## ➢ Funció stereo_triangulation

**Descripció:** Entrant per paràmetres tots els paràmetres de calibració del sistema estèreo, ens retorna els punts 3D estimats fent els càlculs i triangulacions corresponents per a trobar la coordenada 3D de tota la llista de punts que també es passen par paràmetre.

**Invocació:**

[XL,XR]=stereo_triangulation(xL,xR,om,T,fc_left,cc_left,kc_left,alpha_c_left,fc_right,cc_right,kc_right,alpha_c_right)

**Input:**

xL i xR: Matriu 2xN amb les coordenades dels mateixos punts en 2D per a cada càmera.

om: Matriu de rotacio entre ambdues càmera (la càmera dreta respecte la càmera esquerra),

T: vector de translació entre les dues càmeres.

fc_left,cc_left,kc_left,alpha_c_left: paràmetres intrínsecs càmera esquerra.

fc_right,cc_right,kc_right,alpha_c_right: paràmetres intrínsecs càmera esquerra.

**Output:**

XL: Llista de punts 3D tenint com a origen del sistema de coordenades la càmera esquerra

XR: Llista de punts 3D tenint com a origen del sistema de coordenades la càmera esquerra