Universitat Autònoma de Barcelona

Reconstrucció 3D: comparativa d'algoritmes de correspondència

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Gestió

realitzat per

Jaume Fàbrega Clavé

i dirigit per

Felipe Lumbreras

Escola d'Enginyeria

Sabadell, Setembre de 2012

El sotasignat, Felipe Lumbreras,

professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva direcció per

Jaume Fàbrega Clavé

I per a que consti firma la present.
Sabadell, Setembre de ***2012***

--
Signat: ***Felipe Lumbreras***

Resum

En aquest projecte s'analitzen dos algoritmes de correspondència entre imatges amb l'objectiu d'accelerar el procés de reconstrucció 3D mitjançant MVS. S'analitza tot el procés de reconstrucció i a partir d'un software existent es fa la comparació de l'algoritme SIFT i l'algoritme BRISK. A partir dels tests realitzats es conclou que el BRISK és més ràpid i millor per a una reconstrucció 3D.
Paraules clau: reconstrucció 3D, multi-vista estèreo, SIFT, BRISK

Resumen

En este proyecto se analizan dos algoritmos de correspondencia entre imágenes con el objetivo de acelerar el proceso de reconstrucción 3D mediante MVS. Se analiza todo el proceso de reconstrucción y a partir de un software existente se hace la comparación de el algoritmo SIFT y el algoritmo BRISK. A partir de los tests realizados se concluye que el BRISK es más rápido y mejor para una reconstrucción 3D.
Palabras clave: reconstrucción 3D, multi-vista estéreo, SIFT, BRISK

Abstract

In this project we analyze two matching algorithms between images in order to accelerate the process of 3D reconstruction by MVS. It analyzes the process of reconstruction and from an existing software we compare SIFT algorithm and BRISK algorithm. From the tests performed we conclude that BRISK algorithm is faster and better for a 3D reconstruction.
Keywords: 3D reconstruction, multi-view stereo, SIFT, BRISK

Agraïments

M'agradaria donar les gràcies al meu director de projecte Felipe Lumbreras per tot el seu suport, per fer que continués amb el projecte, per ajudar-me en tot el que ha pogut i més.

Gràcies Laia per recolzar-me sempre, per escoltar-me, per entendre'm.

Gràcies a Pemi pel seu temps, per ajudar-me amb les fotografies, i cedir-me tot el material fotogràfic necessari.

Donar les gràcies també a tots els amics, i als companys del centre excursionista pel suport i per fer-me desconnectar quan era necessari.

Gràcies a tots els que són a prop meu, i que d'alguna manera han fet que tot això fos possible.

Índex

1. Introducció

1.1. Introducció al problema

Des de l'inici dels temps, l'home ha volgut plasmar la realitat que l'envolta, i cada vegada més ha volgut que aquesta còpia fos més fidel a la realitat. Primerament la realitat es plasmava mitjançant pintures rupestres, escultures, pintures al oli, mosaics. D'entre totes aquestes maneres de representar la realitat la escultura era l'única que podia representar fidelment el volum de les coses. Tanmateix, el cost en temps i material de la realització d'una escultura era molt elevat, i també l'habilitat de qui feia l'escultura hi tenia molt a dir.

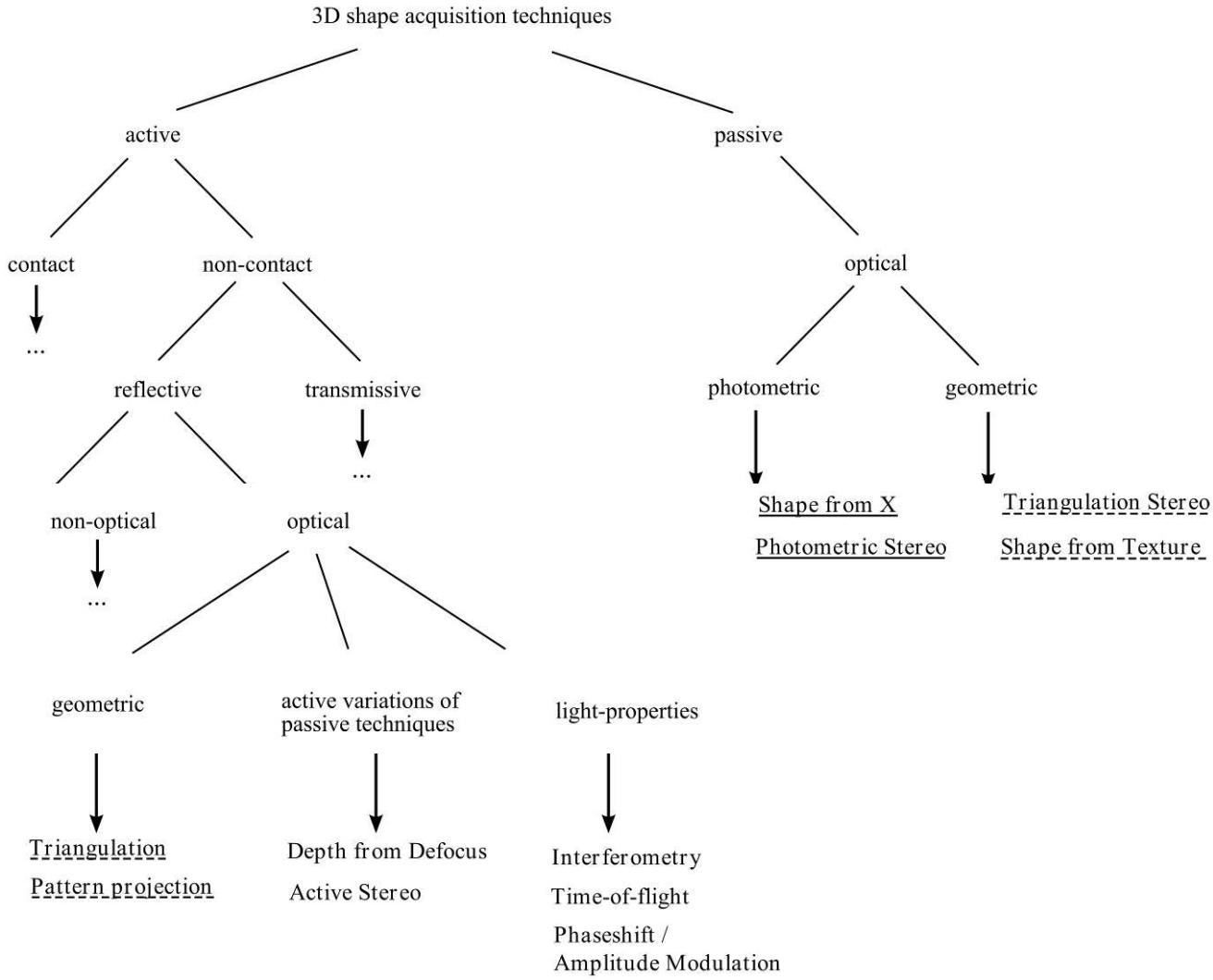
Des de l'aparició dels ordinadors, aquesta representació de la realitat es mostra a través d'una pantalla.

Mitjançant tècniques de representació del volum s'han modelitzat figures en tres dimensions. Aquesta feina és costosa, i com passava amb les escultures, la persona hi ha de tenir una traça especial.

Per això des de fa uns anys s'estan desenvolupant sistemes que puguin fer recreacions virtuals en tres dimensions d'espais i objectes reals, de manera que tenint una computadora i proporcionant la informació necessària qualsevol persona pugui obtenir una representació en 3D dels seus objectes o espais preferits.

1.2. Estat de l'art

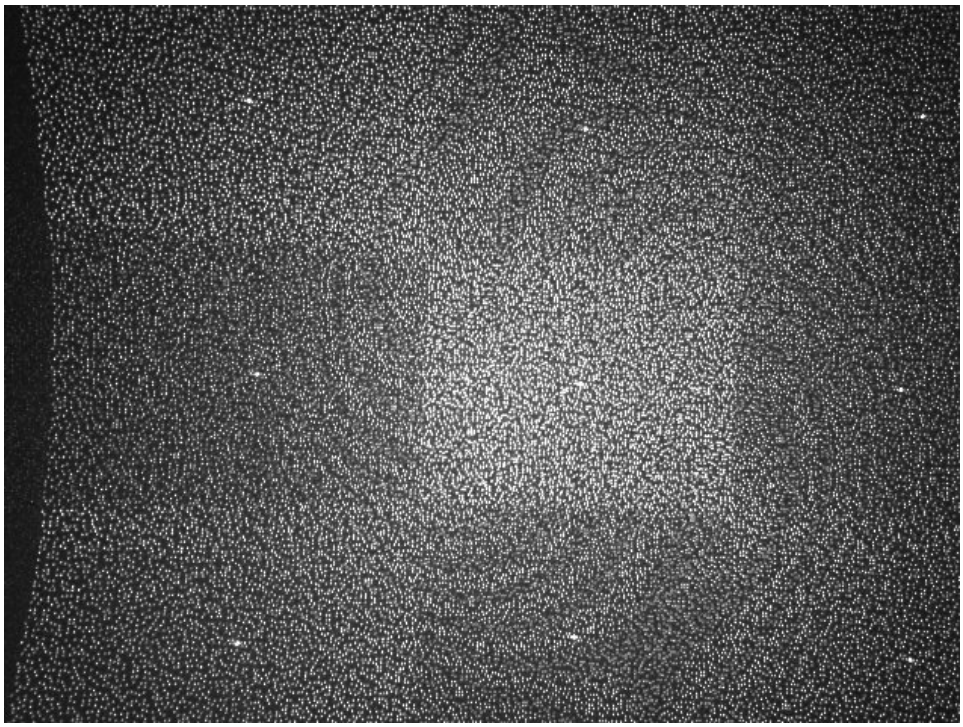
Per a solucionar aquest problema s'han plantejat diverses solucions, més o menys eficients, i més o menys costoses. D'entrada veurem una classificació dels sistemes que tenim per obtenir una forma en tres dimensions i una breu descripció de cadascun d'ells. Per començar tenim dos grans grups de tècniques, les passives i les actives.



Els mètodes actius consisteixen en modificar propietats de l'entorn que interactuen amb l'objecte per obtenir informació de la seva forma, per exemple per contacte amb l'objecte, per reflexió de la llum, projecció de patrons, etc.

Pattern projection

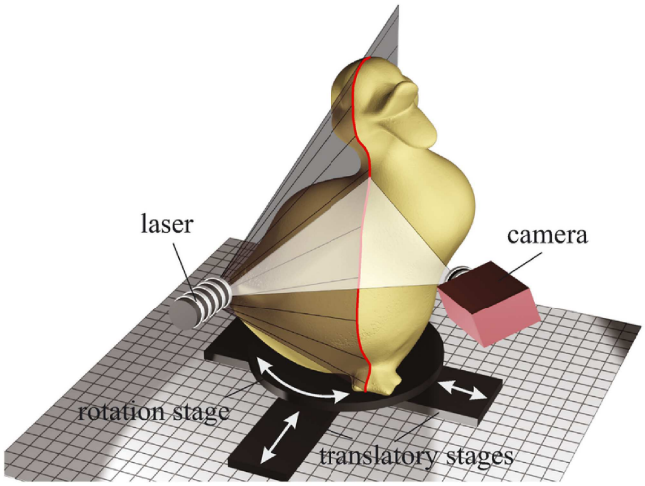
La projecció de patrons, utilitza un sistema d'il·luminació estructurada, de manera que projectant línies de llum, cercles, o qualsevol altre mena de patró sobre l'objecte i després observant la deformació que adquireix el patró sobre l'objecte. Aquest sistema ha arribat a l'abast de tothom amb el dispositiu creat per Microsoft, Kinect, que es tracta d'un emissor de llum infraroja que projecta un patró de punts sobre l'estança on es troba, i una càmera per captar aquest tipus de llum. I amb aquesta informació pot fer reconstruccions, i saber la distancia a la que es troben els diferents objectes.



Il·lustració 1: Patró de punts IR projectat per Kinect

Triangulació activa

El sistema de triangulació activa, està basat en una càmera i un raig làser, que formen un triangle amb el punt on impacta el làser. La posició de la càmera i la posició del raig làser són conegudes, i la seva orientació també, a partir d'aquí s'utilitzen aquestes dades per obtenir la profunditat amb l'ajuda de les relacions geomètriques



Il·lustració 2: Triangulació activa. Font de la imatge: Wikipedia, LaserPrincipale.png"

Time-of-flight

La tècnica de *Time-of-flight* està basada en la velocitat de la llum. S'utilitza un telèmetre làser de temps de vol, que el que fa és enviar un raig làser a l'objecte, i calcular el temps que triga en anar fins a l'objecte i tornar. Això només detecta la distància a un sol punt, per tant, per obtenir una reconstrucció hem d'escanejar diferents punts. Un telèmetre típic pot escanejar entre 10.000 i 100.000 punts per segon, i la seva precisió ve donada per la precisió amb la que podem mesurar el temps, ja que 3,3 pico-segons és aproximadament el que triga la llum en recórrer 1 mil·límetre.

Depth from defocus

El mètode de profunditat per desenfoc consisteix en obtenir una seqüència d'imatges amb diferent grau de desenfoc. Tots els punts que estiguin a la mateixa distància de la càmera tindran el mateix grau de d'enfoc. Per tant, analitzant aquest grau de desenfoc al llarg de tota la seqüència es pot obtenir el mapa de profunditats de tota l'escena.

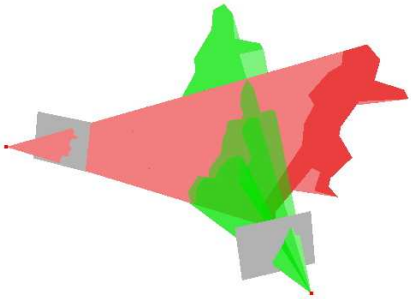
Triangulació passiva

El sistema de triangulació passiva és molt semblant al sistema actiu, però en aquest cas no hi ha raig làser, està basat en dues càmeres, que formen un triangle amb el punt on enfoquen. La posició i orientació de les càmeres són preestablertes i conegudes, s'analitzen les imatges obtingudes i s'identifiquen punts en comú a les dues imatges, i a partir d'aquí s'utilitzen aquestes dades per obtenir la profunditat amb l'ajuda de les relacions geomètriques.

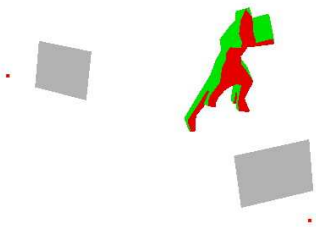
I lustració 3: Sistema de triangulació passiva on la posició i orientació de les càmeres son coneguts

Shape from silhouette

La tècnica de shape-from-silhouette, consisteix en obtenir N siluetes de l'objecte, per obtenir el con de la silueta que ve donat per les línies que uneixen cada punt de la silueta amb el punt de visió que seria el centre de la càmera, i després fer la intersecció de tots els cons obtinguts, i d'aquesta manera obtindríem el visual hull de l'objecte. Amb aquest mètode hi hauria parts de l'objecte que quedarien ocultes.



Il·lustració 4: Projeccions de les siluetes



Il·lustració 5: Intersecció dels cons de projecció

Shape from shading

Aquest mètode consisteix a avaluar la quantitat de llum o brillantor de la imatge de l'objecte. La imatge és processada sense la informació de color. A partir de la informació de la imatge es crea un mapa de vectors amb els que es farà la reconstrucció de l'objecte en 3D.

1.3. Objectius

El objectius d'aquest projecte no són el desenvolupament d'una eina de producció o una eina destinada al gran públic, més aviat els objectius d'aquest projecte estan relacionats amb la investigació i l'anàlisi de diferents mètodes de reconstrucció d'objectes en 3D.

L'objectiu principal d'aquest projecte és l'anàlisi d'un mètode de reconstrucció d'objectes en 3D, en concret la tècnica de MVS (multi-view stereo) ja que no requereix cap hardware addicional ni costós. Com que tot el procés es força feixuc, ens centrarem en l'àmbit de detecció de punts d'interès, descripció dels punts/zones d'interès, i cerca de correspondències. En concret consistirà en l'avaluació de dos algoritmes per a la detecció, extracció i correspondència de punts clau entre dues imatges. La idea és modificar una aplicació existent i de codi obert de reconstrucció 3D mitjançant MVS i optimitzar al màxim aquesta part del procés de reconstrucció, de manera que el seu cost en temps i recursos sigui mínim per, de cara al futur, poder obtenir una aplicació de reconstrucció en 3D per a dispositius mòbils que sigui eficient. Els algoritmes que s'analitzaran són el SIFT (Scale-Invariant Feature Transform), molt conegut i utilitzat, i el BRISK (Binary Robust Invariant Scalable Keypoints), algoritme aparegut recentment que optimitza molt els recursos.

1.4. Estructura de la memòria

Aquesta memòria esta estructurada de la següent manera:

- **Anàlisi de requisits:** En aquest capítol s'analitzaran els requisits per a dur a terme aquest projecte. També veurem si és viable la seva realització, i la planificació adoptada per dur-lo a terme.

- **Procediment:** En aquest capítol veurem tots els coneixements a tenir en compte per realitzar i entendre el projecte, en què consisteix la multi-vista estèreo, i què fan els algoritmes SIFT i BRISK.

- **Disseny:** En aquest capítol veurem com implementarem tot el conjunt d'algoritmes i procediments per a finalment obtenir una reconstrucció 3D. Això ens ajudarà a fer una bona implementació del projecte.

- **Resultats:** En aquest capítol veurem tot un conjunt de tests i proves de les que obtindrem unes dades per valorar l'efectivitat dels diferents algoritmes que s'implementaran al projecte.

- **Conclusions i treballs futurs:** Aquest serà l'últim capítol, i en ell hi exposarem les conclusions obtingudes en la realització d'aquest projecte, i també hi veurem les futures línies de treball que podrem seguir a partir dels resultats obtinguts i la feina realitzada.

2. Anàlisi de requisits

En aquest apartat es veuran els requisits necessaris per a la realització del projecte i per a la consecució dels objectius establerts.

2.1. Requisits funcionals

El requisit principal és proposar un mètode d'extracció de punts o regions d'interès, i cerca de correspondències entre imatges que millori el temps d'extracció i cerca dels mètodes actuals basats en l'algoritme SIFT de David Lowe[5]. El mètode ha de ser invariant a l'escalat, a la rotació, i independent de les condicions d'il·luminació.

2.2. Requisits no funcionals

El format del fitxers obtinguts en executar l'aplicació han de tenir el mateix format que en la implementació feta per David Lowe, de manera que els canvis en tot el procés siguin mínims.

2.3. Estudi de viabilitat

En aquest apartat s'estudiara la viabilitat de realitzar el projecte en diferents aspectes.

2.3.1. Viabilitat tècnica

Malgrat no tenir tots els coneixements tècnics, aquest projecte és viable tècnicament ja que els coneixements necessaris es poden obtenir mitjançant la xarxa, les biblioteques universitàries, i la informació que pugui aportar el professorat expert en aquest àmbit.

2.3.2. Viabilitat econòmica

Tenint en compte que aquest projecte és una part més de la meva formació, és viable econòmicament, ja que totes les eines que s'utilitzaran les pot proporcionar la universitat amb els seus repositoris de software o són gratuïtes.

Però si això no fos així, els cost del projecte hauria de contemplar les despeses de sistema operatiu Windows7, eines de desenvolupament Visual Studio 2010 o Eclipse, eines d'ofimàtica LibreOffice, llibreries de desenvolupament OpenCV.

Per ser conseqüents, també haurem de comptabilitzar les hores de feina, que en aquest cas està establert amb un mínim de 300 hores. Com encara no sóc del tot enginyer, i vist la situació econòmica actual de crisi, he comptat un preu per hora de 15 euros, per tant el cost seria de 4500 euros.

Per tant tindríem dues opcions:

Opció A:

– Windows 7	309€
– Visual Studio 2010	Sobre uns 1000€
– LibreOffice	Gratuït
– Llibreries OpenCV	Gratuït
– PC	900€
– 300 hores de feina	4500€
Total Opció A	6709€

Opció B:

– Windows 7	309€
– Eclipse	Gratuït
– LibreOffice	Gratuït
– Llibreries OpenCV	Gratuït
– PC	900€
– 300 hores de feina	4500€
Total Opció B	5709€

Si tinguéssim en compte els costos triaríem la opció B per a realitzar el projecte, però com la universitat ens proporciona el software necessari i per comoditat amb l'eina Visual Studio 2010, el projecte es realitzarà amb l'opció A.

2.3.3. Viabilitat legal

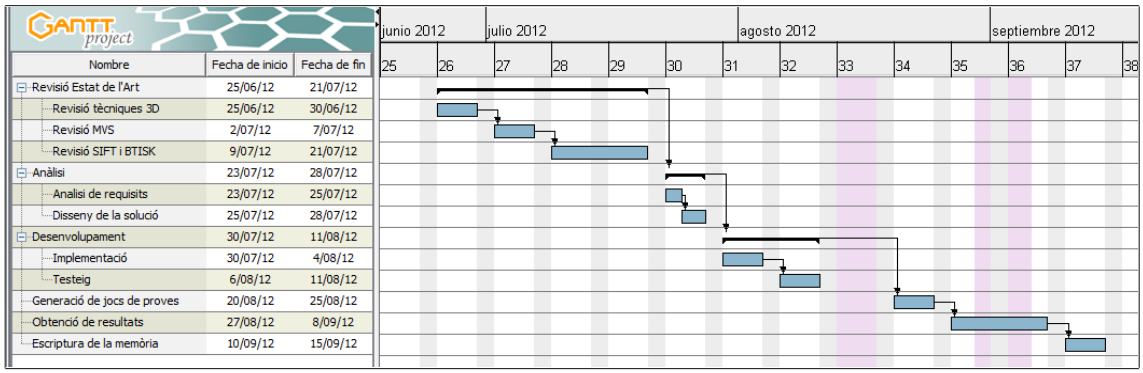
El projecte també és viable legalment ja que el tipus de projecte i aplicació no tracta cap mena de dada confidencial, totes les eines utilitzades tenen llicència vigent i obtinguda legalment, i les parts de codi font utilitzades al projecte són de codi obert.

2.4. Planificació

Tasques

- Revisar l'estat de l'art (4 setmanes)
  - Revisar les tècniques d'obtenció de 3D
  - Revisar en profunditat la tècnica MVS
  - Revisar els algoritmes SIFT i BRISK
- Anàlisi de requisits i disseny (1 setmana)
  - Anàlisi de requisits
  - Dissenyar una solució aprofitant el software existent
- Implementació i testeig (2 setmanes)
- Resultats (2 setmanes)
  - Generar jocs de proves
  - Obtenció de resultats
- Escriptura de la memòria (1 setmana)

La planificació s'ha fet tenint en compte jornades de sis hores diàries, i contemplant els períodes de vacances.



Nombre	Fecha de inicio	Fecha de fin
Revisió Estat de l'Art	25/06/12	21/07/12
Revisió tècniques 3D	25/06/12	30/06/12
Revisió MVS	2/07/12	7/07/12
Revisió SIFT i BTISK	9/07/12	21/07/12
Anàlisi	23/07/12	28/07/12
Analisi de requisits	23/07/12	25/07/12
Disseny de la solució	25/07/12	28/07/12
Desenvolupament	30/07/12	11/08/12
Implementació	30/07/12	4/08/12
Testeig	6/08/12	11/08/12
Generació de jocs de proves	20/08/12	25/08/12
Obtenció de resultats	27/08/12	8/09/12
Escriptura de la memòria	10/09/12	15/09/12

3. Procediment

3.1. Reconstrucció MVS (multi-view stereo)

Multi-view stereo és un mètode de reconstrucció 3D basat en la triangulació passiva. Aquest mètode a partir de parelles d'imatges crea un mapa de profunditats, i troba la posició i orientació de les càmeres [3].

3.2. Solucions basades en MVS

3.2.1. 123D Catch

És una aplicació propietària de la empresa Autodesk pensada per a fer reconstruccions i modelatges 3D a partir d'imatges. És bastant popular per la seva senzillesa i qualitat de les reconstruccions. No és gens exigent pel que fa a processament i memòria ja que tot el procés es fa en servidors gestionats per la mateixa empresa, l'usuari envia les fotografies, i rep els resultats. Si el resultats no han acabat de ser bons o li falten correspondències entre imatges, pots afegir-les manualment. També un cop obtinguts els resultats es poden editar, visualitzar, fer una gravació i exportar. Actualment han fet el llançament de l'aplicació per a dispositius iPad i en format web.

3.2.2. Insight 3D

Aquesta és una aplicació de codi obert, també destinada a fer reconstruccions i modelatges 3D a partir d'una sèrie d'imatges. És un software poc treballat i constantment deixa de funcionar. Funciona essencialment amb reconstruccions d'edificis, i amb jocs d'imatges relativament petits, l'aplicació es distribueix amb un exemple amb només quatre imatges.

3.2.3. Photosynth

Photosynth és una aplicació propietària desenvolupada per Microsoft. Aquesta aplicació

és capaç de, amb un conjunt d'imatges, crear un núvol de punts extraient característiques i cercant correspondències entre les imatges. L'algoritme utilitzat per a trobar aquestes característiques i correspondències ha estat desenvolupat per Microsoft Research, i es similar al SIFT que veurem més endavant. Els resultat es visualitzen via web, i et permet recórrer cadascuna de les imatges amb les perspectives necessàries per a que concordin amb l'escena, i permet també navegar pel núvol de punts.

3.2.4. PMVS

Patch-base Multi-View Stereo (PMVS) és un software de codi obert distribuït amb la llicència GPL. Aquest software a partir d'un conjunt d'imatges i paràmetres de la càmera pot fer la reconstrucció en 3D de l'estructura d'un objecte o edifici. Reconstrueix les estructures sòlides, és a dir si en una imatge apareix una persona i en una altra no, aquesta persona no es reconstruïda. L'aplicació ens dóna com a resultat un núvol de punts amb orientació, no tenim ni polígons ni malla, per això s'ha d'utilitzar una altre eina com ara el Meshlab.

3.3. Esquema MVS

Un reconstrucció mitjançant multi-view stereo ha de seguir els següents passos:

1. Cercar punts d'interès i correspondències entre ells a tota la seqüència d'imatges.
2. Inicialitzar la recuperació d'estructura i moviment
   2.1. Seleccionar dos imatges adequades per a la inicialització
   2.2. Relacionar aquestes dues vistes utilitzant geometria epipolar.
   2.3. Establir l'estructura inicial.
   2.4. Reconstruir l'estructura inicial.
3. Per cada vista addicional
   3.1. Deduir correspondències amb l'estructura i calcular la posició de la càmera utilitzant un algoritme robust.
   3.2. Refinar l'estructura existent.

3.3. Per a les vistes properes que ja s'han calculat

3.3.1. Relacionar aquesta vista amb la vista actual cercant correspondències entre punts característics i calculant la geometria epipolar.

3.3.2. Deduir noves correspondències a l'estructura basant-nos en les correspondències calculades, i afegir aquestes a la llista utilitzada al pas 3.1.

3.4. Refinar la posició a partir de totes les correspondències utilitzant un algoritme robust.

3.5. Inicialitzar una nova estructura de punts.

4. Refinar l'estructura

El pas 2.2. Relacionar les dues vistes mitjançant geometria epipolar, es pot desglossar en els següents passos:

1. Calcular un conjunt de correspondències potencials
2. Mentre el nombre de valors no atípics pel nombre de mostres sigui menor al 95%
   2.1. Seleccionar una mostra mínima (7 coincidències).
   2.2. Calcular solucions per F (matriu fonamental).
   2.3. Determinar valors no atípics.
3. Refina F basant-nos en tots els valors no atípics.
4. Cercar coincidències addicionals.
5. Refinar F basant-nos en totes les coincidències correctes.

3.4. Harris

Per començar comentaré una mica un dels primers algoritmes utilitzats en la cerca de correspondències entre imatges, el detector de Harris.

Harris es basa en la detecció de cantonades, aquests punts normalment no es veuen afectats per la rotació ni l'escalat, i per aquest motiu és un bon mètode per a trobar

correspondències entre imatges.
Una cantonada es caracteritza per ser una regió amb variacions d'intensitat en totes les seves direccions. D'aquesta manera Harris classifica les regions en tres tipus diferents, regió plana, on no hi ha canvis d'intensitat en cap direcció, regió marge, on nomes hi ha canvis en una direcció, i regió cantonada, on es poden apreciar grans canvis en totes direccions.

3.5. SIFT

3.5.1. Definició

El significat de SIFT és Scale Invariant Feature Transform, i això vol dir que fa una transformació de les característiques de la imatge per a que siguin invariants al escalat.
Es busca fer una extracció de característiques i aconseguir una invariància al escalat, a la rotació, i als canvis de la llum. D'aquesta manera podem comparar dues imatges del mateix objecte preses des de punts diferents per aconseguir extreure informació de profunditat.

Aquestes característiques són punts clau anomenats keypoints, i els seus descriptors, que com bé diu el nom, el que fan és descriure l'entorn dels keypoints, informació important que s'extrau en part del gradient de la zona propera al keypoint. Els keypoints a part de la informació de posició (x, y) on es troben a la imatge, tenen també informació de direcció i escalat.

L'algoritme SIFT té la següent estructura, diferenciada en quatre fases, per a fer l'extracció de característiques:

- Detecció d'extrems d'espai-escala
- Localització de punts clau
- Assignació d'orientació

- Descripció de punts clau

3.5.2. Detecció d'extrems d'espai-escala

Aquesta primera fase de l'algoritme fa una primera selecció de keypoints de la imatge. Més endavant s'aniran aplicant restriccions a aquests keypoints i s'eliminaran aquells que no les compleixin.
Per aquesta primera selecció de keyponts el que es fa és aplicar un filtre que identifica punts característics i la seva escala, i això s'aplica a totes les escales possibles. Les localitzacions i escales que es repeteixin a diferents escales de la imatge seran els punts seleccionats.
Això fa que obtinguem un conjunt de punts invariants a l'escalat, un factor molt important a l'hora de detectar i reconèixer objectes.
Per aconseguir-ho s'utilitza una funció contínua anomenada scale-space: L(x, y, σ).

La funció L(x, y, σ) és produïda per una funció gausiana G(x, y, σ), i la imatge d'entrada I(x, y):

L(x, y, σ) = G(x, y, σ)*I(x, y),

on * és la convolució entre la imatge i la funció gausiana en x i en y,

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Per calcular tot l'espai de la funció L(x, y, σ) es construeix una piràmide amb la imatge convol·lucionada amb el filtre G(x, y, σ) aplicat amb diferent nivells del paràmetre σ.
Aquesta piràmide es defineix per dos factors, el factor octava, i el factor escala.
La octava representa totes les imatges generades a l'espai L que son del mateix tamany, però que varia el paràmetre σ amb el que s'han filtrat.
L'escala representa tot el conjunt d'imatges de l'espai L obtingudes amb el mateix nivell

de σ, i en diferents tamanys.

Per detectar keypoints estables no s'utilitza la funció anterior L, sinó una generada a partir de L, la funció utilitzada és Difference-of-gaussian, D(x, y, σ). Això no comporta un gran augment del cost computacional, aquesta funció simplement es calcula restant les imatges veïnes d'una mateixa octava.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma))*I(x, y)= L(x, y, k\sigma) - L(x, y, \sigma)$$

La funció de difference-of-gaussian es fa servir perquè s'aproxima molt a la funció Laplacian-of-gaussian ($\sigma^2\nabla^2G$). Els màxims i els mínims d'aquesta funció proporcionen característiques més estables comparat amb altres funcions utilitzades en aquest àmbit com el Hessià, gradient o la funció de cantonades de Harris.
Calcular el Laplacià d'una funció és d'elevat cost computacional, per aquest motiu s'aproxima amb la diferencia gaussiana D(x, y, σ).

A partir dels càlculs anteriors es calculen els màxims i mínims locals de l'espai D(x, y, σ). Cadascun dels píxels de cada imatge de la piràmide es compara amb els seus vuit veïns de la pròpia imatge, i els nou veïns tant de la imatge anterior com de la següent.

Aquest píxel serà seleccionat com a keypoint només si és major o menor que tots els seus 26 veïns (8+9+9). D'aquesta manera només es trobaran keypoints a les escales centrals de D(x, y, σ), ja que no hi ha imatges veïnes a la primera i última escala.

3.5.3. Localització de punts clau

En aquesta fase el que es pretén és emmagatzemar informació del keypoint, és a dir, l'escala i octava a les quals pertany, i la seva posició a la imatge. Aquesta fase també té com a objectiu reduir el nombre de keypoints segons dos criteris, keypoints amb baix contrast, i keypoints localitzats a les vores.

3.5.4. Assignació d'orientació

En aquesta fase de l'algoritme, cal calcular la orientació de cada keypoint. Això ens permetrà obtenir la invariància a la rotació, ja que utilitzarem aquesta orientació per a calcular els descriptors del keypoint..
Per obtenir l'orientació, cal avaluar el gradient de cadascun dels píxels d'una regió de 16x16 píxels que definim a sobre del keypoint que estem avaluant. El gradient és descrit pel seu mòdul i la seva inclinació, i es calcula mitjançant diferències entre píxels.

$$m(x,y)=\sqrt{((L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2)}$$

$$\theta(x,y)=\tan^{-1}\left(\frac{(L(x,y+1)-L(x,y-1))}{(L(x+1,y)-L(x-1,y))}\right)$$

Els càlculs es realitzaran a la imatge a la qual pertany el keypoint (octava i escala) que s'està analitzant.

La informació es desa en un histograma per a cada keypoint. Cada orientació θ es pondera amb el seu mòdul i una finestra circular gaussiana amb valor igual a 1.5 l'escala del keypoint. Aquestes ponderacions es fan per dos motius, donar més importància a les orientacions amb mòduls grans, i donar més importància als punts pròxims al keypoint.
En aquests instants la informació que tenim del keypoint es la seva posició, l'escala i octava on es troba, i la seva orientació. Tot seguit obtindrem els seus descriptors per a poder trobar aquest mateix punt en qualsevol altre imatge.

3.5.5. Generació de descriptors

A partir dels paràmetres calculats en els apartats anteriors podrem obtenir una descripció de l'entorn del punt que estem avaluant, de manera que sigui invariant a la rotació i a l'escalat. El descriptor ens permetrà cercar el mateix punt de l'escena en altres imatges.
Per a cada keypoint abans teníem una regió de 16x16, i ara dividirem aquesta en 16

regions de 4x4, de tal manera que obtindrem 16 histogrames. Cadascuna de les sub-regions es filtra amb una finestra gaussiana circular amb valor 0.5 vegades l'escala del keypoint. Aquests histogrames es defineixen amb 8 orientacions per a cobrir els 360º, i per tant el nombre d'elements del descriptor d'un keypoint és 128. Amb això ja tenim un descriptor robust als canvis d'il·luminació, però també ens cal que sigui robust als canvis de contrast, i als canvis de llum no lineals. Per a que sigui robust al contrast, només caldrà normalitzar cadascun dels 16 histogrames del descriptor.

3.5.6. Càlcul de correspondències

Existeixen diversos mètodes per a calcular la correspondència entre dos keypoints. Un d'ells és mitjançant la distància euclídia.

$$dif_i = \sqrt{(a_i - b_i)^2}$$

$$dif_{total} = \sum_{1}^{128} dif_i$$

Dif_i és la diferència euclídia entre els elements *i* dels descriptors de les dues imatges que s'estan comparant, i dif_{total} és la suma de totes les diferències dels 128 elements dels descriptors. Com menor sigui el valor de dif_{total} més probabilitat hi ha que ambdós descriptors de les dues imatges que s'analitzen facin referència al mateix punt de l'escena.

Aquestes operacions són costoses degut al gran nombre d'operacions, i per tant es busquen mecanismes d'optimització per reduir-ne les comparacions.

3.6. BRISK

3.6.1. Definició

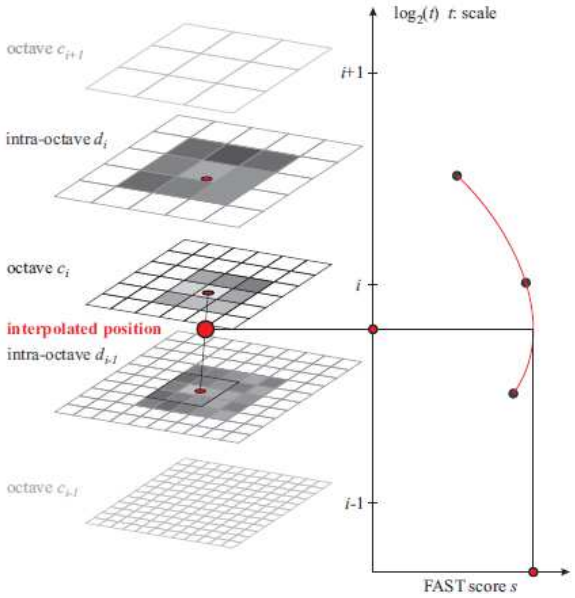
El mètode BRISK (Binary Robust Invariant Scalable Ketpoints) [4], rep aquest nom per

la seva robustesa a les variacions d'escalat, i al seu funcionament binari el qual li permet fer operacions més ràpidament.

Aquest mètode és computacionalment més ràpid que el vist anteriorment, el SIFT, i està basat principalment en la metodologia de l'algoritme AGAST, que és una extensió d'un altra mètode anomenat FAST [7][8], per a la detecció de punts, i es basa també en un altre mètode existent anomenat BRIEF [6] per a la descripció i el *matching* de punts.

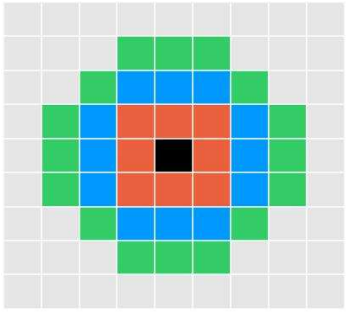
3.6.2. Detecció de punts clau al espai-escala

Per començar, construeix una piràmide de *n* octaves, i *n* intra-octaves, les octaves es calculen reduint progressivament la resolució de la imatge original a la meitat. Cada intra-octava, es troba entre dues octaves, on la primera intra-octava es calcula reduint la resolució de la imatge original dos terços (2/3), i la resta deriven d'aquesta reduint la resolució a la meitat successivament.



Seguidament, s'executa el FAST 9-16 a cada octava i intra-octava amb un llindar T per identificar possibles punts d'interès. L'algoritme FAST cerca cantonades d'una manera senzilla, comprovant que almenys 9 píxels consecutius, dels 16 píxels que formen un cercle de radi 3 al voltant del punt que s'analitza, siguin suficientment lluminosos o foscos que el píxel central. Aquests punts seran considerats possibles keypoints. A la

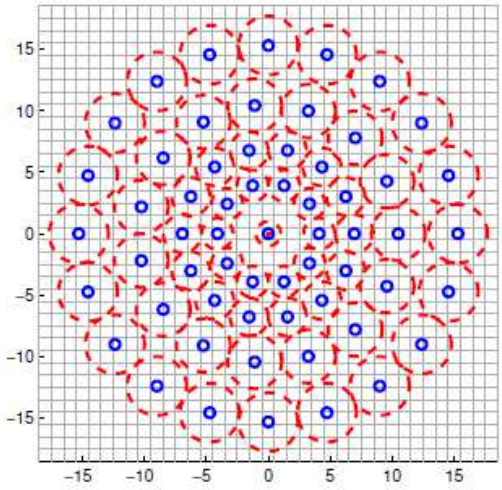
següent imatge es pot veure el patró 9-16 (verd), i el patró 5-8 (vermell) també utilitzat en aquest mètode.



Entre els punts seleccionats es suprimeixen els no màxims, és a dir aquells que no compleixen la condició de màxim respecte les valoracions dels seus 8 veïns.
Després d'això es fa un refinament de l'escala, per a obtenir-ne l'escala real en un àmbit continu, no discretitzat.

3.6.3. Generació de descriptors

Per a generar els descriptors primer de tot hem de trobar l'angle del punt que estem avaluant. Per això s'utilitza el següent patró. El patró està format per N punts, en aquest cas 60, i estan disposats en cercles concèntrics amb el keypoint.
Per començar, s'estima el gradient local per a cada parella de punts del patró, $g(P_i, P_j)$, finalment obtindrem N*(N-1)/2 gradients entre punts.



El conjunt de tots els gradients ve definit per:

$$A=\{(P_i,P_j)\in\mathbb{R}^2 x\,\mathbb{R}^2 | i<N\wedge j<i\wedge i,j\in\mathbb{N}\}$$

I els subconjunts de parelles de curta distància,

$$S=\{(P_i,P_j)\ \in\ A\ |\ \|P_j-P_i\|\ <\ \delta_{max}\}\ \subseteq\ A$$

i llarga distància

$$L=\{(P_i,P_j)\ \in\ A\ |\ \|P_j-P_i\|\ >\ \delta_{min}\}\ \subseteq\ A$$

on δ_{max} i δ_{min} són els llindars màxim i mínim per les distàncies.

Obtenim una estimació de la direcció del patró.

$$g=\begin{pmatrix}g_x\\g_y\end{pmatrix}=\frac{1}{|L|}\cdot\sum_{(P_i,P_j)\in L}g(P_i,P_j)$$

Finalment obtenim que l'angle de rotació és:

$$\alpha=arctan2(g_y,g_x)$$

Un cop calculat l'angle, generar el descriptor és més simple. Es compara la intensitat de les parelles de punts de curta distància amb el patró rotat, i es genera una cadena de bits, on cada bit és:

$$b=\begin{cases}1, & I(P_j^\alpha,\sigma_j)>I(P_i^\alpha,\sigma_i)\\ 0, & sinó\end{cases}\ \forall(P_i^\alpha,P_j^\alpha)\ \in\ S$$

Cada descriptor té una longitud total de 512 bits.

3.6.4. Càlcul de correspondències

Per a calcular la correspondència entre dos descriptors obtinguts amb BRISK, és ben simple, el nombre de bits diferents entre els descriptors és la mesura de la seva semblança. Per tant només cal fer la operació XOR entre els dos descriptors i contar el nombre de bits a 1, aquest tipus d'operacions són molt fàcils de computar, i gràcies a això obtenim un càlcul molt ràpid.

3.7. Comparativa entre SIFT i BRISK

La diferencia més notable entre el SIFT i el BRISK són els seus descriptors. El BRISK es caracteritza per utilitzar un tipus de descriptor binari, aquest descriptor està compost per 512 elements com hem vist abans, això significa que el cost en memòria d'un descriptor BRISK és de 64Bytes. D'altra banda, el descriptor del SIFT està format per 128 elements, on cada element pot prendre un valor entre 0 i 255, per tant cada element ocupa 1Byte, i això fa que el descriptor ocupi 128Bytes.

Així doncs, es pot veure que els descriptors BRISK ocupen la meitat que els descriptors SIFT en memòria, però aquest fet té encara més importància si el dispositiu on s'executa té la memòria limitada, i quan es té en compte que en una imatge es poden extreure més de 1000 punts d'interès, i els seus descriptors corresponents, i que s'avaluen per exemple unes 60 o 80 imatges.

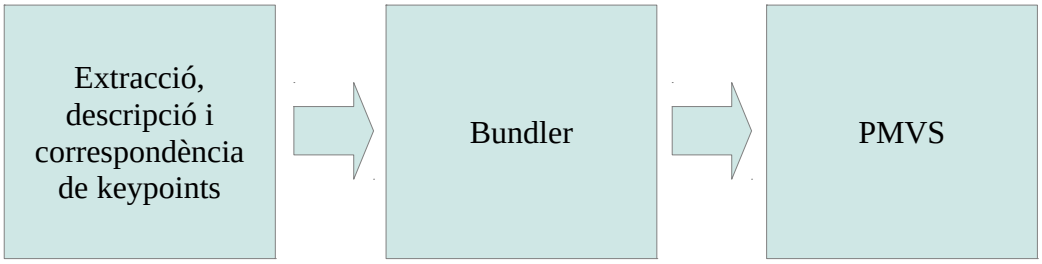
L'altra diferencia important entre els dos algoritmes, i que no es veu a simple vista, es basa en el mètode utilitzat per als diferents algoritmes per extreure els punts d'interès, on el SIFT fa operacions complicades, i el BRISK en canvi es basa en una cerca en arbre i en comparacions simples. Això i el descriptor binari del BRISK fa que aquest sigui molt més ràpid.

4. Disseny

En aquest apartat es veurà com s'estructura l'aplicació per a obtenir finalment una reconstrucció tridimensional, i es veurà de quina manera es dissenyarà i implementarà la primera part de tot el procés de reconstrucció.

Com s'ha vist a l'apartat de l'esquema MVS, la primera part consisteix en trobar punts d'interès a les imatges i buscar correspondències entre elles, aquesta primera part serà la que implementarem.

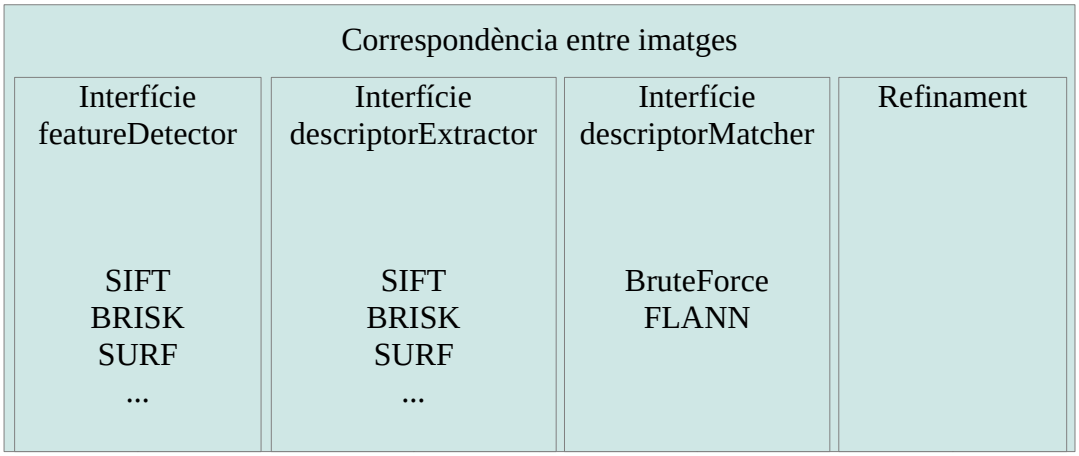
Després d'aquest procés, s'executa la resta amb la que s'obté alguns núvols de punts i la informació de les càmeres, d'aquesta part se n'encarrega una aplicació que es diu Bundler. Finalment, la reconstrucció s'acaba amb l'execució de l'aplicació PMVS, que agafa la informació obtinguda pel Bundler, i cerca més punts amb la finalitat d'obtindre un únic núvol de punts molt més dens i ric visualment.



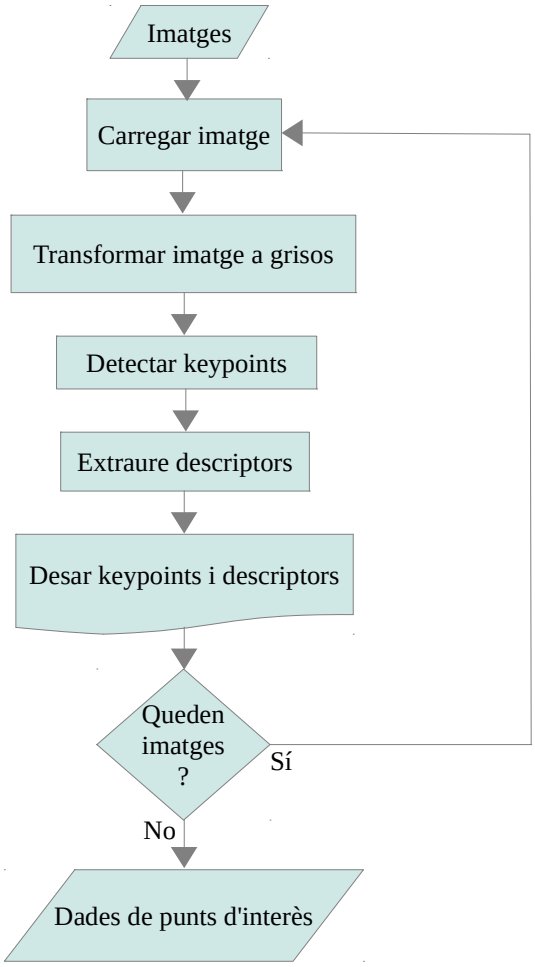
Com hem vist anteriorment, els procés d'extracció de punts d'interès i cerca de correspondències es pot dividir clarament en tres blocs: detecció de punts, extracció de descriptors, i cerca de correspondències. En la nostra implementació afegirem un refinament per eliminar correspondències que siguin potencialment errònies.

Tota aquesta informació l'haurem de desar per a utilitzar-la en fases futures del procés de reconstrucció.

Cadascun dels blocs de cerca de correspondències entre imatges s'implementarà mitjançant interfícies, de manera que tot el procés sigui versàtil, i poder afegir noves implementacions i algoritmes que vagin sorgint.

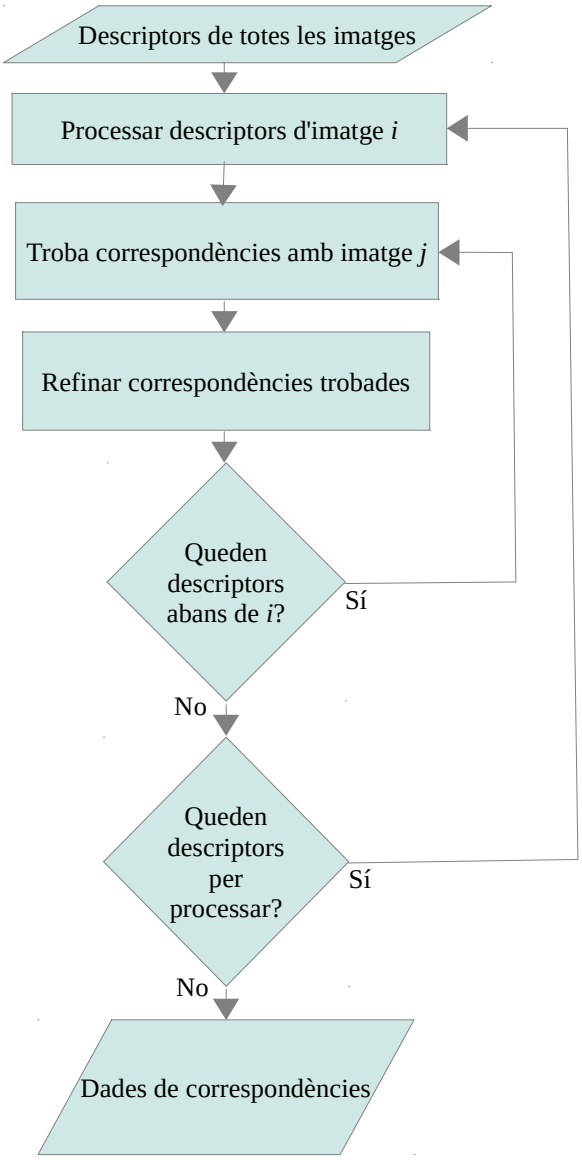


Per veure més detalladament quina serà l'estructura de l'aplicació, la dividirem en dos diagrames de flux, el primer ens mostrarà el procés de detecció de punts i extracció de descriptors, que generarà part de la informació necessària per executar els següents processos. El segon diagrama de flux s'ocuparà de detallar el procés de trobar correspondències entre les imatges, refinar-les, i finalment desar tota la informació de correspondències.



Per a cada imatge a processar, es convertida a grisos, i se li extrau la informació de punts d'interès i descriptors i es desa en un fitxer de text que porta el nom de la imatge i la extensió de fitxer *.key*. Per tant hi haurà tants fitxers *.key* com imatges processades.

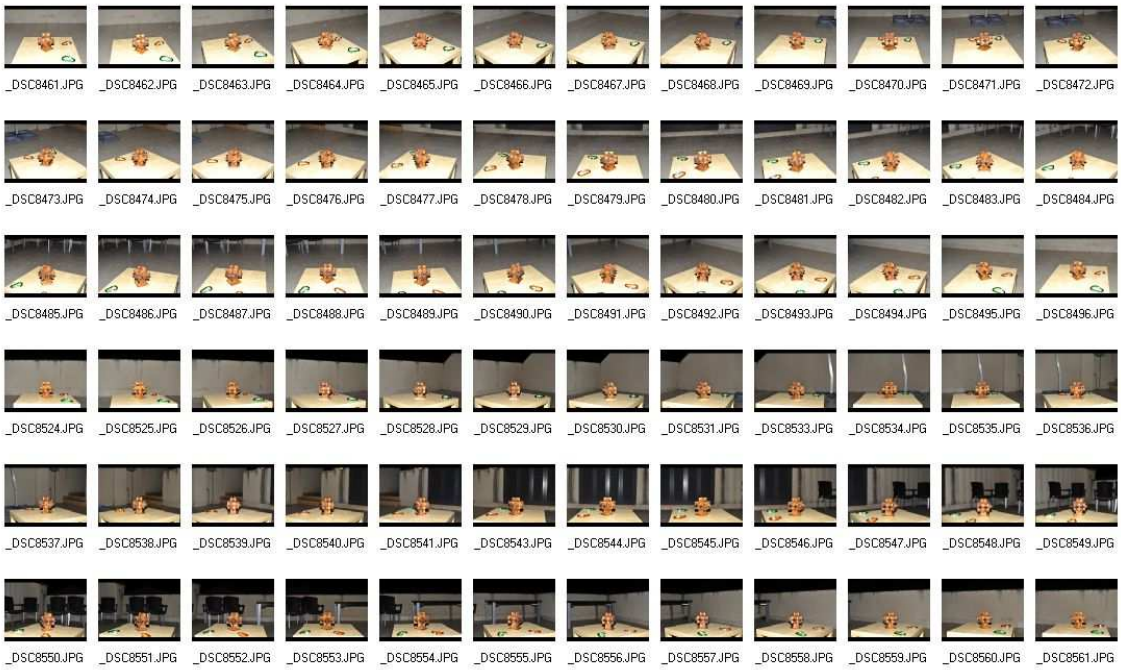
El següent diagrama mostra quina és la manera com es faran les comparacions dels descriptors obtinguts, i això donarà com a resultat un sol arxiu on apareixeran totes les correspondències que hagin superat el refinament.



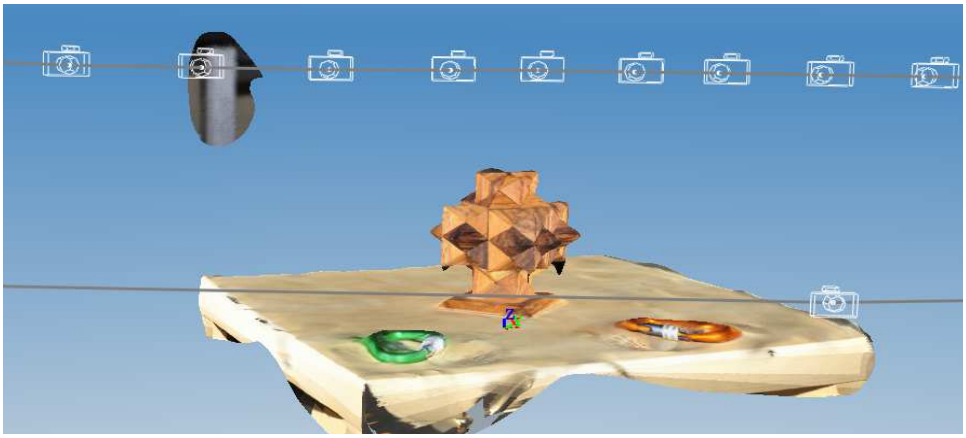
En aquest diagrama es mostra com els descriptors de cada una de les imatges és comparat amb els descriptors de totes les imatges que s'han processat anteriorment. Finalment el que s'obté és que cada descriptor és comparat amb els descriptors de totes les altres imatges.

5. Resultats

Abans d'obtenir qualsevol resultat amb el meu desenvolupament, es va decidir veure quins resultats donaven les aplicacions existents, i la primera solució provada va ser 123D Catch, el joc d'imatges va ser el següent.



Amb tot el joc d'imatges (72 imatges), el programa no va donar resultats, i es va tornar a executar amb la meitat de les imatges, 36 imatges van ser preses a la mateixa alçada que l'objecte, i unes altres 36 imatges van ser preses des d'una posició més elevada.

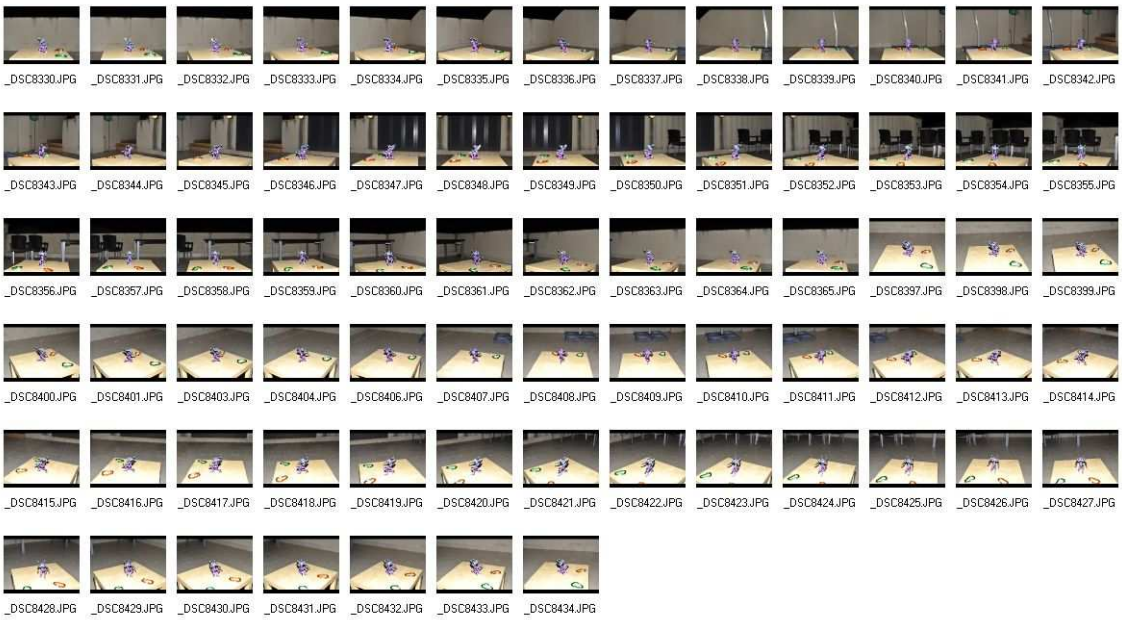


Als annexes es pot veure un vídeo on es reprodueix el moviment que va seguir la càmera per prendre les fotografies.

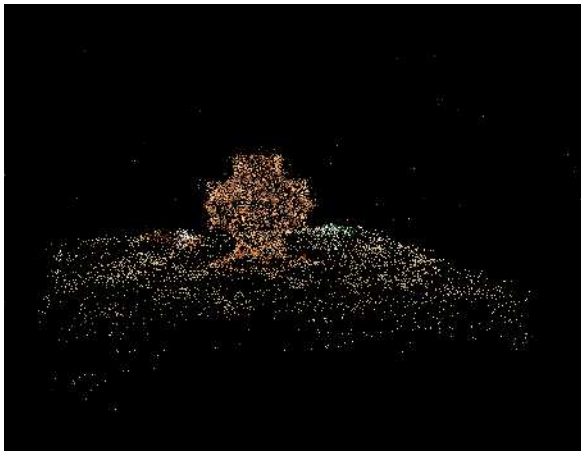
La segona solució a prova va ser Insight3D, aquesta sembla més complicada, amb més opcions de parametrització. S'han realitzat molts intents amb diferents paràmetres, i sempre hi ha hagut algun error, no s'ha pogut veure cap reconstrucció amb aquesta aplicació.

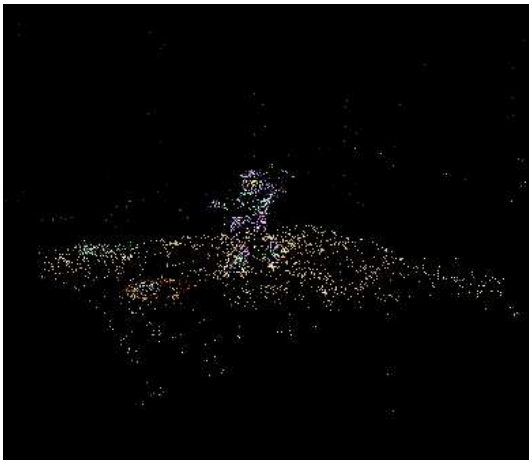
Després es va testejar el Photosynth de Microsoft amb el mateix joc d'imatges, en aquesta aplicació la visualització només ens permet veure el núvol de punts generats des de la perspectiva d'alguna de les càmeres, o veure les fotografies amb les transformacions corresponents perquè encaixin, i tinguin la posició i orientació adequades.

Aquí es va provar també amb una altre joc d'imatges, i tal com havia passat amb 123D Catch, les imatges preses a diferents nivells no van acabar de funcionar, però només les imatges preses a la mateixa alçada que l'objecte, si que va donar bons resultats.



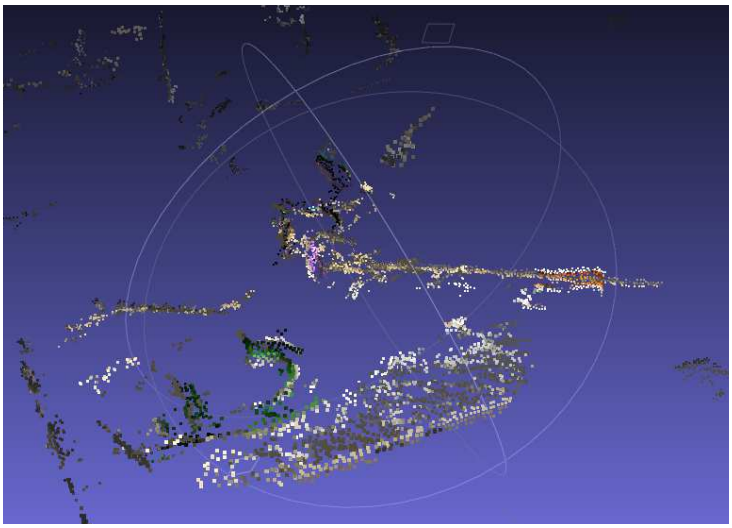
Aquí es mostren els resultats dels dos objectes en forma de núvol de punts.





A partir d'aquí veurem els resultats obtinguts amb el Bundler i PMVS amb les dades que s'han obtingut amb la nostra implementació del SIFT i el BRISK, també veurem diferències entre temps i detecció de punts d'interès.
Aquí a part dels jocs d'imatges que hem creat nosaltres, també hem fet servir altres jocs d'imatges que s'han utilitzat també en altres treballs [9].

En un principi es van fer tests amb els jocs de proves utilitzats anteriorment, i com anteriorment van sorgir problemes degut a les imatges preses a diferents alçades, de manera que en alguna reconstrucció a part de ser pobre, apareixia la forma de dues taules sobreposades.



De manera que es va començar a testejar amb jocs de prova que ja havien estat utilitzats per a fer aquest tipus de reconstruccions.

El primer joc de proves amb el que es van obtenir bons resultats després d'ajustar paràmetres va ser el joc de proves anomenat “fountain-P11”.

Amb aquest joc d'imatges s'han obtingut resultats tant amb el BRISK com amb el SIFT, per tant és un bon punt per a comparar-los.

El primer que es pot veure i salta a la vista, és el temps d'execució dels diferents algoritmes.

	Temps d'execució en segons
BRISK	389.6s
SIFT	2058.8s

Es pot veure clarament que el BRISK ha anat molt més ràpid, en concret ha anat 5 cops més ràpid. I els resultats finals són els següents.

I lustració 6: Resultats final amb BRISK

Il·lustració 7: Resultats finals amb SIFT

Si observem les dades:

Imatge	Keypoins	Temps (s)	Temps per a cada keypoint (s)
0	11365	13,12	0,001154
1	13392	14,29	0,001067
2	13078	14,10	0,001078
3	12955	14,18	0,001095
4	12718	14,09	0,001108
5	12335	13,88	0,001125
6	12093	13,70	0,001133
7	11135	13,03	0,001170
8	11476	13,21	0,001151
9	12114	13,64	0,001126
10	12483	13,78	0,001104
			0,001119

Taula 1: Temps de detecció de keypoints i extracció de descriptors per a SIFT2.4

Imatge	Keypoints	Temps (s)	Temps per a cada keypoint (s)
0	5821	1,94	0,000333
1	6653	2,07	0,000311
2	6432	2,03	0,000316
3	6824	2,14	0,000314
4	6867	2,14	0,000312
5	6189	1,95	0,000315
6	7377	2,29	0,000310
7	6506	2,03	0,000313
8	8325	2,59	0,000312
9	11248	3,39	0,000302
10	13334	3,98	0,000298
			0,000312

Taula 2: Temps de detecció de keypoints i extracció de descriptors per a BRISK20

Com es pot veure, el llindar utilitzat per al SIFT fa que aquest trobi molts més punts d'interès, però tot i així no és el fet de trobar més keypoints el que fa que sigui més lent, sinó el temps empleat en detectar el keypoint i extraure el seu descriptor també és més elevat.

Si comparem només els temps de detecció.

Imatge	Keypoins	Temps (s)	Temps per a cada keypoint (s)
0	11365	4,81	0,000423
1	13392	5,20	0,000388
2	13078	5,20	0,000398
3	12955	5,30	0,000409
4	12718	5,33	0,000419
5	12335	5,31	0,000430
6	12093	5,27	0,000436
7	11135	5,06	0,000454
8	11476	5,10	0,000444
9	12114	5,21	0,000430
10	12483	5,15	0,000413
			0,000422

Taula 3: Temps de detecció de keypoints per a SIFT2.4

Imatge	Keypoints	Temps (s)	Temps per a cada keypoint (s)
0	5821	0,28	0,000049
1	6653	0,24	0,000037
2	6432	0,24	0,000037
3	6824	0,25	0,000037
4	6867	0,24	0,000036
5	6189	0,23	0,000038
6	7377	0,26	0,000035
7	6506	0,24	0,000037
8	8325	0,28	0,000034
9	11248	0,32	0,000029
10	13334	0,36	0,000027
			0,000036

Taula 4: Temps de detecció de keypoints per a BRISK20

El temps de detecció de cada keypoint és de 0,4 mil·lèsimes de segon per al SIFT, però en canvi és de 0,03 mil·lèsimes de segon per al BRISK, aquí la diferència de temps és molt més notable.

Execució de la comparació	Keypoints imatge 1	Keypoints imatge 2	Temps (s)	Temps per comparació (s)	Correspondències trobades
1	11365	13392	35,08	0,00000023	1522
2	11365	13078	34,48	0,00000023	514
3	13392	13078	40,50	0,00000023	2047
4	11365	12955	34,02	0,00000023	229
5	13392	12955	40,09	0,00000023	938
6	13078	12955	39,19	0,00000023	2130
7	11365	12718	33,38	0,00000023	149
8	13392	12718	39,38	0,00000023	484
9	13078	12718	38,69	0,00000023	938
10	12955	12718	38,32	0,00000023	2027
11	11365	12335	32,51	0,00000023	106
12	13392	12335	38,22	0,00000023	300
13	13078	12335	37,35	0,00000023	464
14	12955	12335	37,28	0,00000023	955
15	12718	12335	36,71	0,00000023	2159
16	11365	12093	31,95	0,00000023	78
17	13392	12093	37,57	0,00000023	113
18	13078	12093	36,67	0,00000023	181
19	12955	12093	36,34	0,00000023	342
20	12718	12093	35,95	0,00000023	890
21	12335	12093	34,43	0,00000023	2200
22	11365	11135	29,15	0,00000023	36
23	13392	11135	34,36	0,00000023	55
24	13078	11135	33,54	0,00000023	71
25	12955	11135	33,21	0,00000023	105
26	12718	11135	32,63	0,00000023	250
27	12335	11135	31,63	0,00000023	718
28	12093	11135	31,03	0,00000023	1854
29	11365	11476	30,09	0,00000023	18
30	13392	11476	35,47	0,00000023	21
31	13078	11476	33,88	0,00000023	20
32	12955	11476	33,50	0,00000023	28
33	12718	11476	32,93	0,00000023	69
34	12335	11476	32,09	0,00000023	150
35	12093	11476	31,15	0,00000022	465
36	11135	11476	29,31	0,00000023	1141
37	11365	12114	31,77	0,00000023	17
38	13392	12114	37,22	0,00000023	16
39	13078	12114	36,62	0,00000023	25
40	12955	12114	36,58	0,00000023	31
41	12718	12114	35,66	0,00000023	45
42	12335	12114	34,54	0,00000023	71
43	12093	12114	33,80	0,00000023	121
45	11135	12114	30,76	0,00000023	358
46	11476	12114	31,50	0,00000023	1513
47	11365	12483	32,08	0,00000023	21
48	13392	12483	37,48	0,00000022	14
49	13078	12483	36,48	0,00000022	17
50	12955	12483	36,38	0,00000022	24
51	12718	12483	35,59	0,00000022	28
52	12335	12483	34,46	0,00000022	37
53	12093	12483	33,66	0,00000022	52
54	11135	12483	31,04	0,00000022	57
55	11476	12483	31,98	0,00000022	216
56	12114	12483	33,87	0,00000022	1111
				0,00000023	27541

Taula 5: Temps de cerca de correspondències per a SIFT

Execució de la comparació	Keypoints imatge 1	Keypoints imatge 2	Temps (s)	Temps per comparació (s)	Correspondències trobades
1	5821	6653	4,20	0,00000011	1075
2	5821	6432	3,99	0,00000011	493
3	6653	6432	4,57	0,00000011	1481
4	5821	6824	4,22	0,00000011	263
5	6653	6824	4,81	0,00000011	708
6	6432	6824	4,66	0,00000011	1478
7	5821	6867	4,25	0,00000011	111
8	6653	6867	4,93	0,00000011	330
9	6432	6867	4,68	0,00000011	735
10	6824	6867	4,97	0,00000011	1401
11	5821	6189	3,90	0,00000011	78
12	6653	6189	4,38	0,00000011	179
13	6432	6189	4,23	0,00000011	418
14	6824	6189	4,49	0,00000011	784
15	6867	6189	4,51	0,00000011	1482
16	5821	7377	4,58	0,00000011	78
17	6653	7377	5,23	0,00000011	147
18	6432	7377	5,14	0,00000011	258
19	6824	7377	5,37	0,00000011	466
20	6867	7377	5,43	0,00000011	849
21	6189	7377	4,91	0,00000011	1583
22	5821	6506	4,08	0,00000011	98
23	6653	6506	4,67	0,00000011	152
24	6432	6506	4,60	0,00000011	222
25	6824	6506	4,78	0,00000011	344
26	6867	6506	4,84	0,00000011	445
27	6189	6506	4,33	0,00000011	783
28	7377	6506	5,29	0,00000011	1453
29	5821	8325	5,20	0,00000011	56
30	6653	8325	6,05	0,00000011	95
31	6432	8325	5,76	0,00000011	126
32	6824	8325	6,08	0,00000011	142
33	6867	8325	6,22	0,00000011	181
34	6189	8325	5,49	0,00000011	305
35	7377	8325	6,61	0,00000011	674
36	6506	8325	5,85	0,00000011	1160
37	5821	11248	7,05	0,00000011	62
38	6653	11248	8,19	0,00000011	103
39	6432	11248	7,74	0,00000011	113
40	6824	11248	8,22	0,00000011	179
41	6867	11248	8,42	0,00000011	122
42	6189	11248	7,45	0,00000011	165
43	7377	11248	8,88	0,00000011	267
45	6506	11248	7,91	0,00000011	508
46	8325	11248	10,24	0,00000011	2060
47	5821	13334	8,37	0,00000011	91
48	6653	13334	9,71	0,00000011	111
49	6432	13334	9,22	0,00000011	99
50	6824	13334	9,73	0,00000011	169
51	6867	13334	10,00	0,00000011	93
52	6189	13334	8,82	0,00000011	110
53	7377	13334	10,49	0,00000011	208
54	6506	13334	9,32	0,00000011	296
55	8325	13334	12,11	0,00000011	684
56	11248	13334	16,39	0,00000011	2126
				0,00000011	28199

Taula 6: Temps de cerca de correspondències per a BRISK

Per acabar de comparar els temps, veurem les diferències en la cerca de correspondències.
Tal com es pot veure a les taules, els temps per comparació entre dos descriptors per al SIFT és de 229 nanosegons, mentre que els temps per al BRISK són de 107 nanosegons.

Amb totes aquestes dades queda patent que el BRISK és més ràpid que el SIFT, tot i així, s'han fet més tests i molts no han reconstruït res, de manera que s'ha intentat esbrinar els motius d'aquests casos.

Aquí trobem que el principal problema és que les imatges tenen molta distància entre sí, i això fa que moltes de les correspondències siguin errònies, a mes la gran quantitat d'elements iguals fa incrementar el nombre d'errors.
Aquests factors són presents en altres jocs d'imatges que s'han intentat reconstruir, com per exemple el primer joc d'imatges que sí ha funcionat amb 123d Catch, on el principal problema és que gran part de la imatge és fons, i per tant canvia constantment, i les cantonades de la taula, on troba correspondències d'una cantonada amb una altra cantonada diferent quan hem fet un quart de volta a aquesta, i així amb moltes altres imatges.

6. Conclusions i treballs futurs

6.1. Conclusions

Per a la realització d'aquest projecte s'ha fet un repàs de l'estat de l'art de les tècniques existents d'obtenció de 3D, s'ha revisat en profunditat el funcionament d'un algoritme fortament implantat en aquest camp com és el SIFT, i s'ha revisat també de manera exhaustiva el funcionament d'un altre algoritme que millora l'anterior, i més recent com és el BRISK.

A partir d'aquí s'ha desenvolupat un software que utilitza ambdós algoritmes i s'ha procedit a realitzar diversos tests per a comprovar que es podia obtenir una reconstrucció a partir dels resultats obtinguts. Amb els algoritmes funcionant s'han comparat utilitzant uns paràmetres per a obtenir uns resultats similars pel que fa al nombre de punts d'interès i correspondències, i s'ha avaluat la rapidesa de cadascun. Finalment, amb totes les dades obtingudes de nombre de punts d'interès, correspondències, i temps de cadascuna de les parts dels algoritmes, podem dir que el BRISK és efectivament més ràpid i precís que el SIFT.

6.2. Treballs futurs

Com s'ha pogut veure en la comparativa el BRISK és més eficient en termes de memòria, i per tant una línia lògica a seguir seria fer una migració de tots aquests aplicatius per a que funcionin en dispositius mòbils amb poca capacitat de memòria i processament. Això seria un gran avantatge de cara a l'arribada de perifèrics cada vegada més orientats al 3D com ara les impressores de 3D.

També està previst de cara el futur, canviar el tipus de dades d'entrada de l'aplicació que actualment és mitjançant un conjunt d'imatges i un fitxer de text que les enumera, per un arxiu de vídeo, de manera que es milloraria el fet de no tenir prou imatges per a la reconstrucció o no tenir bones imatges com hem vist als resultats. A més seria molt més còmode d'obtenir les imatges i d'executar l'aplicació

7. Recursos

7.1. Referències

1. S. Herbort i C. Wöhler (2011) An Introduction to Image-based 3D Surface Reconstruction and a
Survey of Photometric Stereo Methods

2. Jean-S Franco i E. Boyer (2009) Efficient polyhedral modeling from silhouttes

3. M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, R. Koch (2004) Visual Modeling with a hand-held camera.

4. S. Leutenegger, M. Chili, R.Y. Siegwart (2011) BRISK: Binary Robust Invariant Scalable Keypoints

5. David G. Lowe (2004) Distinctive Images from Scale-Invariant Keypoints

6. M. Calonder, V. Lepetit, C. Strecha, P. Fua BRIEF: Binary Robust Independent Elementary Features

7. E. Rosten, T. Drummond (2006) Machine learning for high-speed corner detection

8. E. Rosten, T. Drummond (2005) Fusing points and lines for high performance tracking

9.C. Strecha, W. von Hansen, L. Van Gool, P. Fua, U. Thoennessen (2008)On Benchmarking Camera Calibration and Multi-View Stereo for High
Resolution Imagery

7.2. Webgrafia

http://www.edwardrosten.com/work/fast.html

http://www6.in.tum.de/Main/ResearchAgast

http://www.cs.ubc.ca/~lowe/keypoints/

http://www.asl.ethz.ch/people/lestefan/personal/BRISK

http://grail.cs.washington.edu/software/pmvs/

http://phototour.cs.washington.edu/bundler/

_____

Firmat: Jaume Fàbrega Clavé

Sabadell, Setembre de 2012