Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Título:

"Generación de Modelos 3D texturizados, a partir de datos parciales"

Tesis presentada por:
Nallely Sánchez Rodríguez.

Como requisito para obtener el título de la Licenciatura en:
Ingeniería en Ciencias de la Computación.

Asesor:
Dr. Rafael Lemuz López

Mayo 2008.

Resumen

El propósito de esta tesis es el desarrollo de un programa que facilite la generación de un modelo 3D único formado a partir de datos parciales, con apariencia real.

En muchas aplicaciones de visión artificial es necesario realizar comparaciones entre modelos 3D. En el caso particular del registrado, es necesario realizar una alineación entre modelos ***P*** y ***Q***. El objetivo es que los puntos representados en la escena del modelo ***P***, sean lo mas similar posible a los puntos representados en la escena del modelo ***Q***. Los modelos son obtenidos desde diferentes puntos de vista, por consiguiente tienen diferente apariencia; mismas que producen una serie de cambios imposibilitando la comparación directa de los mismos, ya que puntos equivalentes vistos desde distintas tomas no corresponden directamente. La técnica del registrado de modelos busca eliminar esas diferencias.

Por lo tanto el registrado consiste en encontrar una función de transformación geométrica que relacione dos modelos ***P*** y ***Q***; lo cual implica encontrar una serie de parámetros que aplicados al modelo ***P*** conviertan a este, en un modelo lo más parecido posible al modelo ***Q***.

El registrado se logra mediante los siguientes procesos: Primero se seleccionan las características que queremos comparar, en nuestro caso utilizamos *puntos*. Recordemos que los puntos seleccionados del modelo ***P*** deben ser lo más parecido posible a los puntos seleccionados del modelo ***Q***, para obtener una buena estimación inicial y facilitar el proceso de registro. Además se recomienda que los puntos sean tomados del contorno del modelo ya que de no ser así el algoritmo no realizará una buena alineación; como mínimo se deben de seleccionar 3 puntos, teniendo en cuenta que entre mayor sea la cantidad de puntos seleccionados mejor será la alineación de los datos parciales. Enseguida se procede con la extracción de características, que permitan comparar mediante una métrica de similitud los puntos de diferentes modelos. A continuación se realiza el proceso de búsqueda de correspondencia para lo cual utilizaremos un algoritmo de indexación, mismo que consiste en encontrar el vecino mas cercano. Mas adelante, procedemos con encontrar el conjunto de transformaciones elementales (*translación, rotación y escala*) las cuales determinan una función de transformación adecuada para el alineamiento de nuestros modelos 3D.

Agradecimientos

Muy en especial esta tesis esta dedicada a mis padres (Miguel Sánchez Acevedo. e Irene Rodríguez Sánchez.) y hermanos (Miguel, Arquímedes y Uriel) por que gracias a ellos he podido lograr este proyecto tan importante en mi vida profesional; gracias a sus consejos y a su ayuda total que fue un aliciente muy fuerte estoy aquí y me siento orgullosa de ello. Los amo mucho, y le doy gracias a Dios por haberme dado una familia como ustedes y por no haberme dejado ir antes de tiempo. ***Gracias Familia!!!***

Gracias a mi novio Jesús David Alcobas Rodríguez, por ser una persona maravillosa que me dio una ilusión, un apoyo, un motivo. Eres una persona que me impulsa a salir adelante, a lograr metas, a luchar y a decir que nada es imposible, y que todo lo puedo, gracias por comprender el tiempo que no te pude dar, TE AMO.

Agradezco a todas las personas que en un momento dado formaron parte de mi vida a lo largo de la carrera, ya que en mi mente y corazón se quedarán grabadas para siempre, gracias por regalarme esos momentos que para mi son y serán inolvidables. Ya que de todos esos momentos buenos y malos; se ha formado mi vida. He caído pero el orgullo que me queda es que me he sabido levantar. Gracias amiga (Teresa Pantoja), ya que tu fuiste una pieza clave en mi vida, me enseñaste a ser mas madura; fuiste suave y tierna, pero también fuerte a la vez. Contigo aprendí a quererme mucho mas, a valorarme y ser mas responsable. Gracias Carlos Rosas mi gran amigo y confidente de la universidad, gracias por haberme apoyado en todo momento, te quiero muchísimo; sabes?., es difícil encontrar un amigo como tu, pero yo ya lo encontré, gracias.

Contenido

Índice de Figuras

Estructura de la Tesis.

Capítulo I: En esta etapa se habla de la importancia y de las diversas aplicaciones que tiene la reconstruccion de modelos tridimensionales dentro de la ciencia y la tecnología, así como también se da a conocer el planteamiento del problema y el objetivo de esta tesis; proporcionando diversas técnicas para lograr la solución, dentro de ellas la estrategia utilizada, que es la del método del vecino mas cercano ya que es una de las técnicas mas importantes y populares; la cual facilita encontrar el vecino mas próximo de un elemento dado, y con esto podemos establecer correspondencias adecuadas entre los elementos de un modelo y otro. Damos a conocer los fundamentos del registrado; listamos todos los pasos del proceso, desde la selección de rasgos hasta la explicación de la determinación de la función de transformación estimando los parámetros de (rotación, traslación y escala) utilizando el método SVD, el cual permitirá la alineación de nuestros modelos. Así como también mencionamos la existencia de 2 métodos diferentes, que nos permiten encontrar los parámetros de transformación (*cuaternios y SVD*). Finalmente se mencionan las ventajas y desventajas de los módulos pertenecientes al algoritmo.

Capítulo II: En esta fase se habla de la importancia que ha tenido la búsqueda por similaridad debido a que la evolución de la tecnología nos ha llevado a contar con nuevos modelos de búsqueda para el tipo de datos no estructurados. También se da a conocer la implementación de los algoritmos de indexación que utilizan este tipo de búsqueda. Los algoritmos AESA e IAESA tienen el mismo objetivo ya que establecen la correspondencia entre los puntos de los modelos para alinearlos, la diferencia de los algoritmos radica en el criterio de la selección del próximo pivote. Así que, aquí se mencionan tanto las técnicas y los criterios que utiliza cada uno.

Capítulo III: En este capítulo se realiza la explicación del funcionamiento de los 3 algoritmos implementados AESA, IAESA y SVD, realizando las iteraciones de los mismos. Así como también se presentan resultados de varios experimentos realizados.

Finalmente se dan a conocer las conclusiones acerca de éste proyecto de tesis, así como las aportaciones y los trabajos futuros que éste puede hacer a la sociedad mexicana.

Las referencias bibliográficas en las cuales se respalda el presente trabajo se han marcado con corchetes y aparecen a lo largo del documento. Dichas referencias se localizan al final del documento.

CAPÍTULO I

I. Estado del Arte.

1.1 Introducción a la Reconstrucción de modelos 3D.

La construcción de modelos tridimensionales de objetos reales ha tenido un creciente interés. Los campos de aplicación de los modelos 3D son amplios, desde la inspección automática de piezas, diseño de productos industriales, medicina (En la cirugía Ortopédica, estimación de fragmentos cilíndricos para la reducción de fractura de hueso semiautomática, la cirugía asistida por computadora, diseño de prótesis.) hasta la creación de museos virtuales o modelos generados por computadora para películas, multimedia y comercio electrónico. Otra utilidad radica en poder comparar piezas o modelos y comprobar si se asemejan o si son defectuosos.

Estos modelos se construyen de varias formas, una de ellas es partiendo de una serie de modelos adquiridos desde diferentes puntos de vista (modelos parciales) los cuales al ser procesados por ciertos algoritmos dan como resultado un modelo único 3D.

El problema a tratar es el siguiente:
Dados dos modelos ***P*** y ***Q*** pertenecientes a la misma escena pero capturados desde diferentes puntos de vista, son utilizados para lograr el objetivo de: *"generar un modelo 3D único a partir de éstos modelos*".

Para realizar esta tarea es necesario establecer una correspondencia adecuada entre el conjunto de puntos del modelo ***P*** y el conjunto de puntos del modelo ***Q***, de tal manera que la solución a este problema consiste en encontrar ciertos parámetros de transformación (*rotación, traslación y escala*) que conjuntamente logren la alineación correcta de nuestros modelos generando un modelo único 3D.

La solución de nuestro objetivo se logra mediante los pasos sugeridos en la literatura: Primero seleccionamos un conjunto de rasgos (*puntos*) pertenecientes a ambos modelos. El proceso de selección lo hacemos tomando un punto del modelo ***P*** y un punto del modelo ***Q*** siendo éste lo más similar posible al punto del modelo ***P*** (*estimación inicial*) estas parejas de puntos se van seleccionando consecutivamente.

Existen varios tipos de características que permiten lograr una correspondencia, como lo son:

Líneas: En donde el proceso utiliza ángulos, posiciones, longitudes y puntos medios, para determinar la correspondencia.

Regiones: En donde el proceso utiliza información sobre las posiciones relativas y la orientación de las regiones.

Formas: En donde se utilizan los coeficientes de la transformada de Fourier; donde dos formas pueden ser comparadas por sus características de diseño geométrico, posiciones, orientaciones y escalas.

Puntos: Estos son los más utilizados por que sus coordenadas son directamente utilizadas para determinar los parámetros de una función de transformación en el registrado de modelos.

Una vez seleccionadas todas las parejas de puntos; procedemos con la extracción de características de ambos modelos, para más adelante mediante un algoritmo de indexación establecer las correspondencias de las mismas; para lograrlo se implementaron los algoritmos AESA e IAESA cuya diferencia radica en el criterio para seleccionar el próximo pivote, pero el objetivo es el mismo ya que ambos establecen correspondencias entre los modelos 3D, además de que utilizan el método del k vecino mas cercano, el cual es un método de clasificación no paramétrico que consiste en encontrar el vecino más próximo de un elemento. Mediante esta manera se encuentran los puntos de correspondencia. Estas características extraídas se utilizan para encontrar una función de transformación adecuada.

Para el proceso de búsqueda de relación existente entre nuestros modelos ***P*** y ***Q*** se establecen distancias entre los conjuntos de puntos correspondientes a los modelos 3D; éstos cálculos son muy importantes por que son utilizados como grado de relación, en donde la suma mas pequeña es considerada la verdadera posición [2a], en esta etapa se encuentra la cantidad de cambio que necesita el modelo ***P*** para alinearse con el modelo ***Q***. Para establecer estas correspondencias existen varios algoritmos basados en dos tipos de indexación algunos de ellos son: ***basados en pivotes*** (**BKT**, **VPT**, **VPF**, **FQT**, **FQA**, **AESA**, **IAESA**, etc.) y ***basados en particiones compactas*** (**BST**, **MTREE**, **SAT**, **GNAT**, **GHT**, etc.) [6a].

Finalmente se procede con la última etapa del registrado (algoritmo **ICP**) mediante el método SVD (Descomposición de Valores Singulares) que determina los parámetros de transformación rígida (*Traslación Rotación y Escala*). Estos parámetros permiten encontrar una función de transformación adecuada para la correspondencia de nuestros datos parciales, y así facilitar la alineación de nuestros modelos generando un modelo único 3D.

Por lo tanto el proceso empieza con una buena estimación inicial del modelo ***P***, hacia los puntos del modelo ***Q***.

Los métodos para determinar la correspondencia se discuten mas adelante.

Tenemos 3 términos que se utilizarán a lo largo de ésta tesis.

Modelo Q: Es el modelo que permanecerá sin cambios durante el proceso y será utilizado como referencia para las comparaciones posteriores

Modelo P: Es el modelo que se va a transformar hasta alinearlo con el modelo de la ***Q***.

Función de transformación: Es la función matemática que transforma la geometría del modelo ***P*** para asemejarla con la del modelo ***Q***.

1.2 Planteamiento del Problema.

El problema a ser resuelto es el siguiente:
"Dados **2** conjuntos de puntos **P** y **Q**, donde $p_i = (X_i, Y_i, Z_i)$ y $q_j = (X_j, Y_j, Z_j)$ tenemos que determinar todas las parejas (i, j) de los **2** conjuntos de puntos, donde p_i y q_j corresponden al mismo punto en la escena. La transformación obtenida es usada para alinear los puntos del conjunto **P** hacia los del conjunto **Q**."

El tipo de datos que estamos utilizando son puntos, así como sus propiedades espaciales, lo cual implica que el tipo de función seleccionada dependerá de la distancia euclidiana, esto es, de la exactitud de los puntos en correspondencia, densidad y organización. [2a]

Así como también se hace la estimación del método SVD que consiste en la descomposición de valores singulares de una matriz, y es utilizado en el registrado de modelos 3D, para encontrar los parámetros de transformación que permiten realizar la alineación de los puntos correspondientes entre los dos modelos.

En el caso tridimensional la detección de puntos llega a ser de la siguiente manera:

$$X = ax + by + cz + d,$$
$$Y = ex + fy + gz + h,$$
$$Z = ix + jy + kz + l.$$

Donde (*X, Y, Z*) son coordenadas de puntos en el modelo ***Q*** y (*x, y, z*) son las coordenadas de puntos en el modelo ***P*** [2a, pág. 65].

En el caso bidimensional, la correspondencia de puntos en los modelos puede relacionarse por afinidad o por transformación lineal, el cual se representa de la siguiente manera:

$$X = ax + by + c,$$
$$Y = dx + ey + f,$$

1.3 Objetivo.

El objetivo de éste proyecto consiste en *generar un modelo 3D único a partir de datos parciales, es decir, "alinear" 2 modelos en el espacio tridimensional.*

Originalmente las coordenadas de cada modelo difieren en su sistema de coordenadas. Para hacer coincidir sus sistemas de coordenadas se establecen correspondencias entre los conjuntos de puntos pertenecientes a dichos modelos. Para esto se lleva a cabo la implementación del algoritmo de indexación iAESA, así como también se debe de hallar una función de transformación que establezca la correspondencia de los modelos. Nuestra tarea es encontrar los parámetros de dicha función (rotación, traslación y cambio de escala) y llevar a cabo la alineación de nuestros modelos; ésta tarea se desarrolla implementando el método SVD. Finalmente se lleva a cabo el proceso de texturización del modelo único, esto para proporcionarle una apariencia real.

Hipótesis: suponemos que, es posible mejorar la apariencia de los modelos tridimensionales registrando las propiedades geométricas de los modelos parciales. En la figura 1 se muestra el diseño conceptual del sistema, es decir, las etapas de procesamiento que se desarrollaron para formar un modelo único tridimensional de apariencia real.

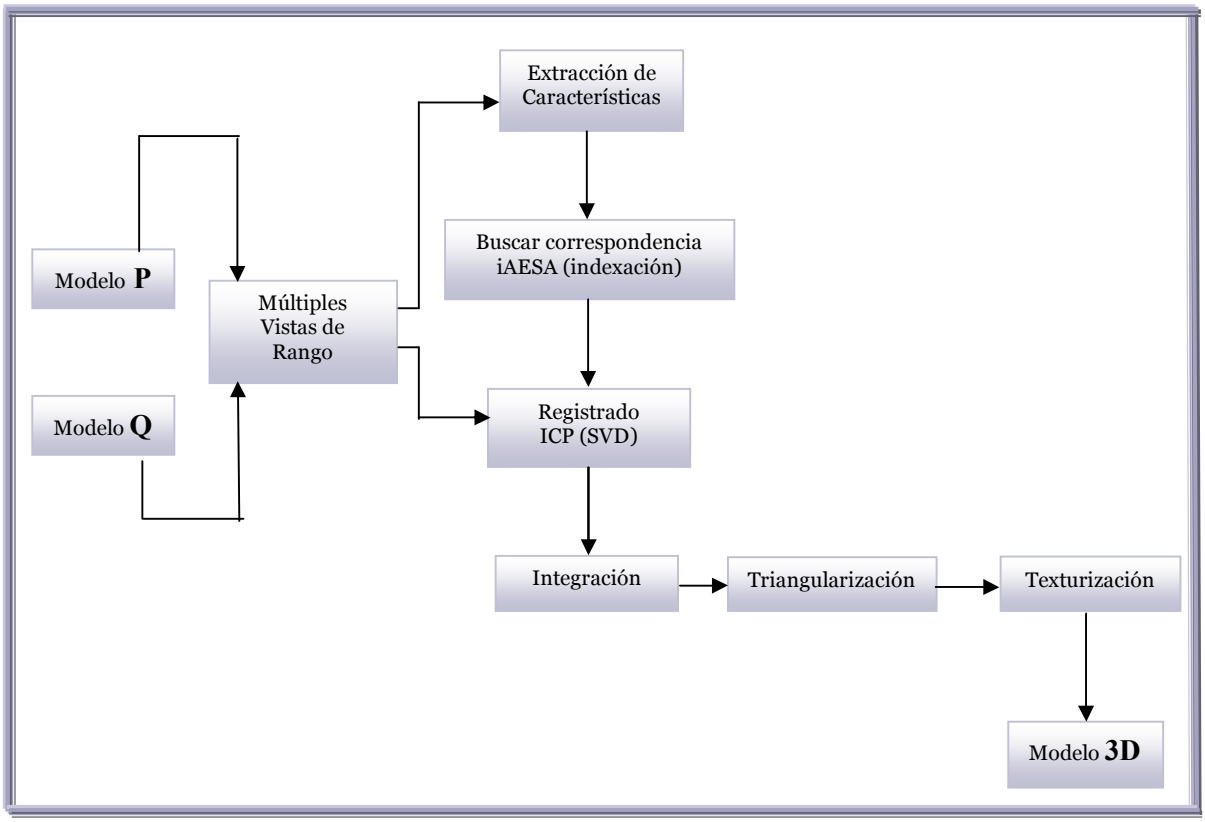


Fig. 1. Pasos para la obtención de un modelo 3D único formado a partir de modelos parciales en diferentes sistemas de coordenadas.

1.4 Notación Utilizada.

- ***Modelo Q:*** modelo de Referencia (Permanece sin cambios durante el proceso).
- ***Modelo P:*** modelo Objetivo (se le aplican las transformaciones para alinearla).
- ***Múltiples Vistas de Rango:*** Modelos adquiridos en diferentes posiciones en diferentes lapsos de tiempo.
- ***Extracción de Características:*** Se extraen las parejas de puntos pertenecientes a los modelos 3D, mismas que se utilizarán en las etapas de correspondencia basada en las características y registrado.
- ***Buscar Correspondencia IAESA:*** Se establece la correspondencia basada en las características (*puntos*).
- ***Registrado (SVD):*** En esta etapa se realiza la tarea de encontrar la transformación adecuada para tener todos los modelos bajo el mismo sistema de coordenadas.
- ***Integración:*** En esta etapa se realiza la integración de las partes que se traslapan de los modelos. Como resultado final se obtendrá un modelo global que contenga todas las características de las imágenes de rango y se tiene la opción de realizar la triangularización directa.
- ***Triangularización:*** Se realiza el modelado del objeto mediante un algoritmo de triangularización que genera una malla de triángulos.
- ***Texturización:*** Una vez alineado el modelo 3D, podemos pasar a la etapa final del modelado, que consiste en darle una apariencia realista a la escena, aplicándole la textura a la caras del modelo.

1.5 Aplicación en la reconstrucción de modelos 3D.

Mediante la reconstrucción tridimensional es posible crear modelos a escala de espacios físicos, objetos o productos, situaciones diversas y brindar así una forma de visualización más avanzada y más global.

Hoy en día surgen aplicaciones en las que se desea buscar objetos similares. A continuación se mencionan de forma detallada algunas aplicaciones en donde se emplea la reconstrucción 3D, para resolver diferentes tipos de problemas [3a]:

- ***Reconocimiento de objetos:*** Un sistema modelador 3D puede ser usado para crear vistas de un objeto. Esto puede ser entonces comparado con un conjunto de datos de modelos de objetos para identificarlo.

- ***Simuladores:*** Permite repetir una técnica tantas veces como sea necesario hasta su correcto y adecuado aprendizaje, él individuo se instruye dentro del entorno al que se enfrentará en la realidad y de esta forma se permiten corregir sus actividades rutinarias.

- ***Robots móviles:*** Se puede requerir que un robot se mueva por una escena y después trazar un mapa de su ambiente, que puede ser usado para la planificación del movimiento y así poder evitar colisiones (planificación de movimiento).

- ***Gráficos por computadora:*** En ciertas aplicaciones, podría utilizarse un sistema modelador 3D, para la creación de escenas artificiales y representación de objetos del mundo real.

1.6 Búsqueda del vecino más cercano

Uno de los métodos para establecer puntos de correspondencia es el enfoque del vecino más cercano. El cual consiste en encontrar el ***K*** vecino más cercano de cada elemento, en un conjunto de datos de *n* puntos o elementos pertenecientes a un espacio métrico.

La complejidad del problema es medida en cálculos de distancia, ya que de antemano se sabe que utilizando la fuerza bruta el problema es solucionado con n^2 cálculos de distancia (debido a que cada elemento calcularía la distancia hacia todo el conjunto de datos). La solución planteada para encontrar el ***K*** vecino más cercano consiste en hacer uso de un algoritmo de indexamiento para reducir los cálculos de distancia.

La tarea del registrado de modelos implica el proceso de encontrar una buena correspondencia entre el conjunto de puntos del modelo ***Q*** hacia el conjunto de puntos del modelo de ***P***, para realizar esta tarea es estimado el método del ***k*** vecino más cercano, ya que este método encuentra la interpolación entre los puntos de ambos modelos, calculando las distancias de un punto hacia los demás puntos del modelo, tomando a aquel elemento cuya distancia sea mínima. En la figura 2 se muestra el resultado de encontrar todos los ***K*** vecinos de un conjunto de datos.

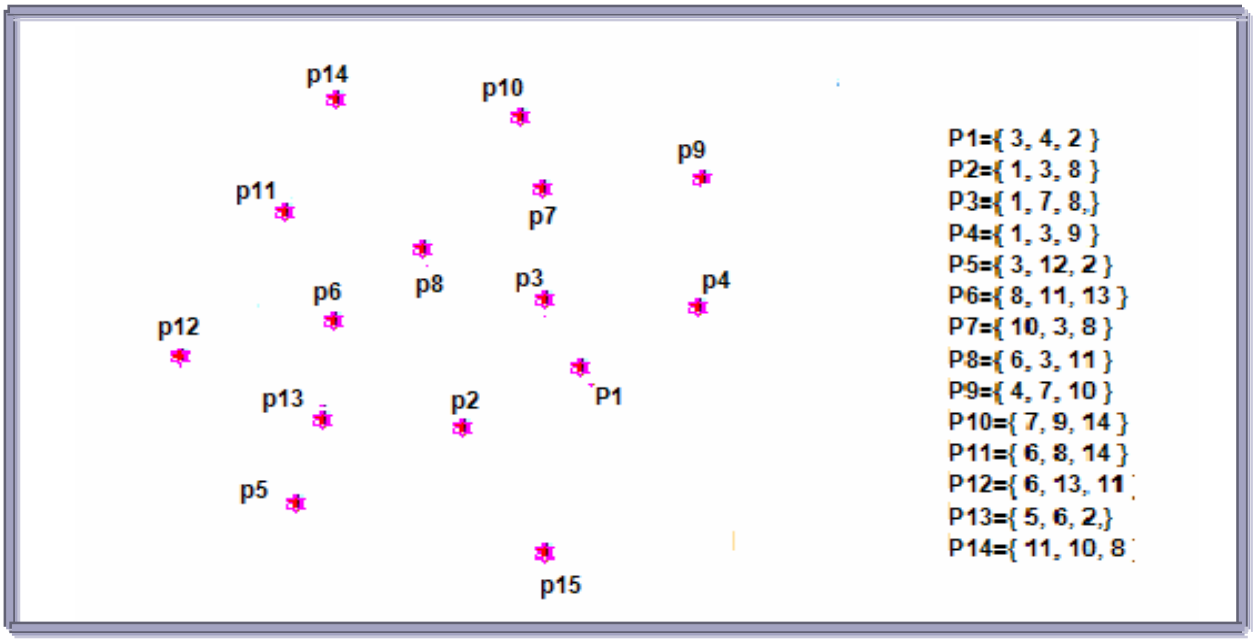


Fig. 2. Conjunto de datos en donde a cada elemento se le calculan sus k vecinos más cercanos.

El problema del ***k*** vecino más cercano es aplicado en diversas áreas de la tecnología computacional, como por ejemplo:

- ***Genética:*** Encontrar secuencias de ADN o proteínas similares en alguna base de datos genética.

- ***Reconocimiento de Imágenes:*** Encontrar imágenes similares.

- ***Compresión de video:*** Encontrar bloques en un modelo previo que son similares a bloques en una nuevo modelo.

- ***Minería de Datos:*** Encontrar series de tiempos aproximados o patrones no conocidos de una base de datos.

- ***Recuperación de información:*** Encontrar documentos relacionados a uno dado en una biblioteca digital.

- ***Bases de Datos Médica:*** Donde se almacenan imágenes 2d (rayos X) e imágenes 3d (tomografía computarizada). La identificación de casos anteriores con similares síntomas es muy útil en diagnósticos, enseñanza médica e investigación.

1.7 Técnicas para la Búsqueda de los K vecinos más cercanos.

La búsqueda del vecino más cercano es una técnica utilizada principalmente en el reconocimiento de formas. Sin embargo, esto no quiere decir que no pueda ser de utilidad en otras disciplinas. Por ejemplo se puede aplicar en hacer consultas a un conjunto de datos, a la corrección de errores tipográficos en los procesadores de textos, en bibliotecas, bases de datos, buscadores en Internet, etc. [5a]. En la figura 3 se muestra la clase en la que pertenece la mayoría de los vecinos más próximos a la consulta.

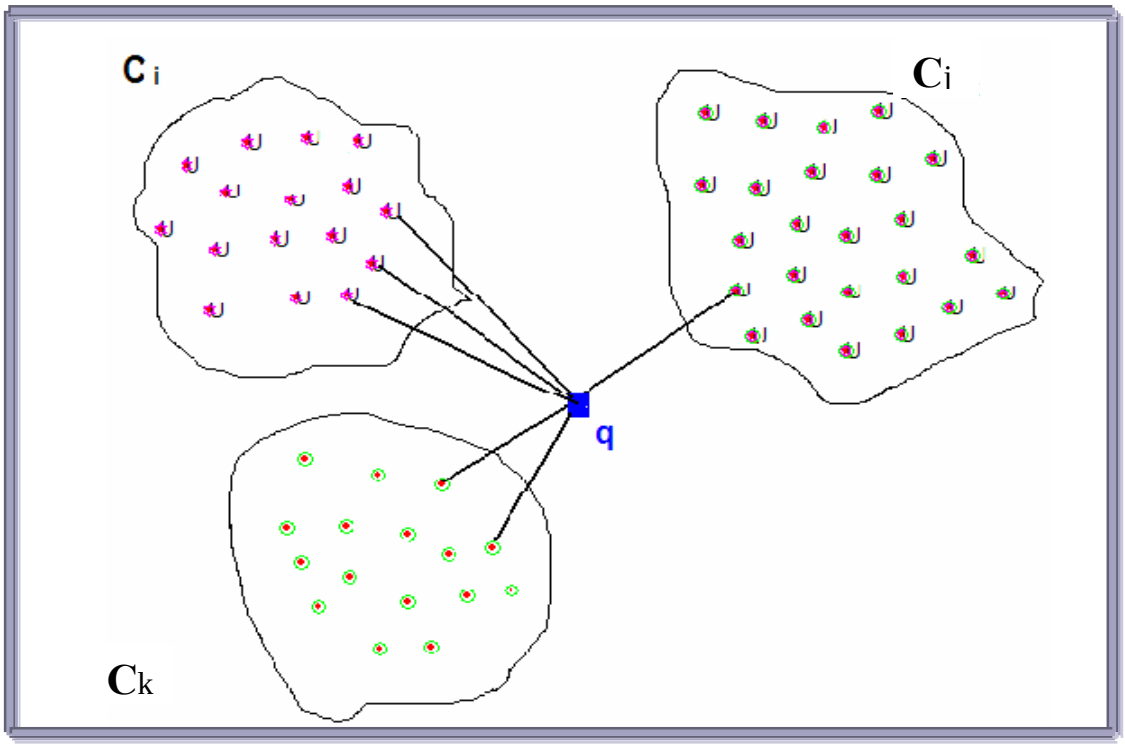


Fig. 3. Clasificador basado en la regla de los K vecinos más próximos. La muestra se clasifica en la clase en la que pertenece la mayoría de los K vecinos más próximos. En este ejemplo la clase mayoritaria es la C_i (3 vecinos)

Una simple elección entre el k vecino más cercano genera la predicción para cada caso.

- Se planteará el desarrollo del algoritmo, el cual consiste en indexar el conjunto de datos en tiempo lineal, después hacer $([q],r)_d$ consultas de rango con un radio seguro para conocer el ***k*** vecino más cercano.

Se puede definir el problema de la búsqueda del vecino más cercano de la siguiente manera:

Dado un conjunto de puntos $P = \{p_1, \ldots\ldots p_n\}$ en un espacio métrico ***X,*** con una función de distancia *d*, permitiendo algún pre-procesamiento en ***P*** de manera eficiente, se desea responder a dos tipos de solicitudes.

- ***Vecino más cercano:*** Localizar el punto *p* más cercano a $q \in X$.
- ***Rango:*** Dado un punto $q \in X$ y $r > 0$, regresar todos los puntos $p \in P$ que satisfagan $d(p,q) \leq r$.

Búsqueda exhaustiva.

Es el algoritmo más sencillo para la búsqueda del vecino más cercano, ya que calcula todas las distancias de un prototipo hacia los demás prototipos del "conjunto de datos" y asigna al conjunto de vecinos más cercanos, aquel cuya distancia sea mínima. En la práctica resulta poco aconsejable, ya que el coste en número de distancias calculadas es proporcional al tamaño del conjunto de prototipos.

En muchos de los problemas en los que se aplica la técnica de búsqueda de vecinos, éste método es prohibitivo debido a que el coste inherente al cálculo de la distancia puede ser elevado; en consecuencia se han ido desarrollando algoritmos más eficientes que evitan recorrer exhaustivamente todo el conjunto de datos.

La búsqueda del vecino mas cercano es una técnica relevante en áreas como: Bibliotecas, bases de datos, buscadores en Internet, etc.

Búsqueda de vecinos en espacios vectoriales (k-tree).

La mayoría de los métodos propuestos en la literatura pertenecen a este grupo, esto se debe a que para muchos problemas de reconocimiento de formas puede obtenerse una adecuada representación vectorial de los datos u objetos a reconocer. Estos métodos basan su funcionamiento en estructuras de datos que utilizan una representación vectorial de los puntos. El árbol de búsqueda *d*-dimensional, más conocido como k-*tree*, es la estructura de datos más utilizada en este tipo de algoritmos, desarrollada por Bentley en 1975 [6a]. Es un árbol binario de búsqueda, aquí se divide el dominio de búsqueda donde los puntos se encuentran ordenados en dos subdominios, y la comparación con el valor de partición nos dirá en cual de los 2 subdominios se encuentra cada punto.

EL k-*tree* se basa en la misma idea original de dividir el espacio de representación R^k en dos espacios disjuntos a partir de un hiperplano perpendicular al vector correspondiente a una de las coordenadas *k*.

El proceso de búsqueda sobre ésta estructura se lleva a cabo descendiendo por las ramas del árbol, eligiendo en cada nivel aquella subregión que contiene al punto. Cuando se llega a una hoja se examinan exhaustivamente todos los puntos que pertenecen a la misma y si la hiperesfera con centro en la muestra y radio dado por la distancia del vecino más cercano hasta el momento no corta ninguna región, el proceso de búsqueda ha terminado. De no ser así el proceso debe repetirse para éstas regiones. En la figura 4 se muestra gráficamente la función de este algoritmo. Los autores demuestran que para espacios vectoriales, el coste temporal del proceso de búsqueda es en promedio logarítmico con el número de puntos y tiene una dependencia exponencial de la dimensión.

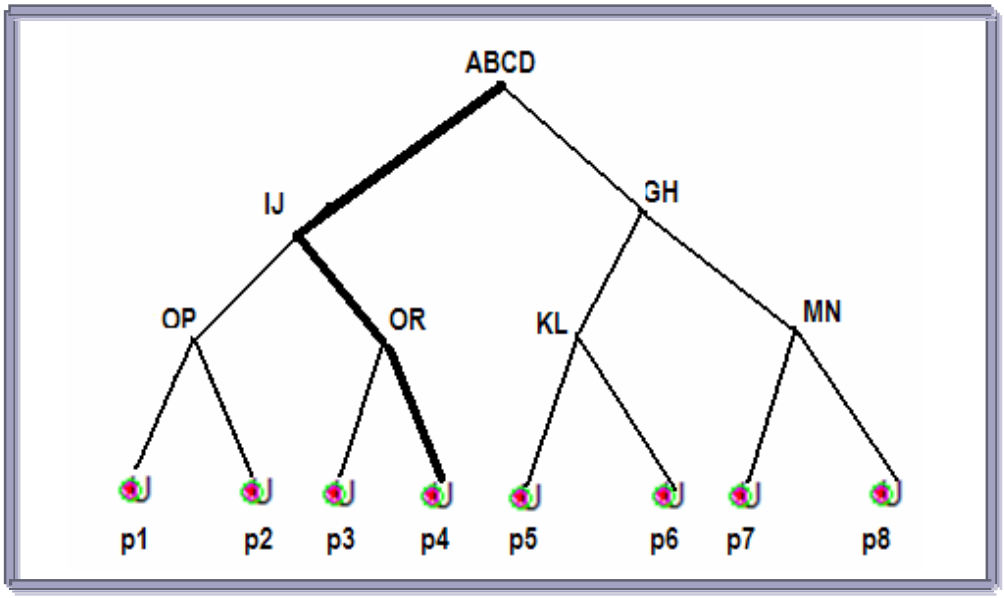


Fig. 4. K-tree (con k=2) Proceso y búsqueda del vecino más próximo.

Vecinos más cercanos con distancia Ponderada

El cálculo de los *k* vecinos más cercanos se puede ponderar con la contribución de cada vecino de acuerdo a la distancia entre él y el ejemplar a ser clasificado x_q, dando mayor peso a los vecinos más cercanos. Por ejemplo podemos ponderar el voto de cada vecino de acuerdo al cuadrado inverso de sus distancias.

$$\hat{f}(x_q) \leftarrow ard \max_{u \in V} \sum_{i=1}^{K} w_i \delta(u, f(x_i))$$

Donde:

$$wi \equiv \frac{1}{d(x_q, x_i)^2}$$

De esta manera se ve que no hay riesgo de permitir a todos los ejemplos de entrenamiento a contribuir a la clasificación de x_q, ya que al ser muy distantes no tendrían peso asociado. La desventaja de considerar todos los ejemplos es que sería muy lenta la respuesta. Se requiere siempre tener un método local en el que solo los vecinos más cercanos son considerados.

Esta mejora es muy efectiva en muchos problemas prácticos. Es robusto ante los ruidos de datos y suficientemente efectivo en conjuntos de datos grandes. Se puede ver que al tomar promedios ponderados de los k-vecinos más cercanos el algoritmo puede evitar impactos de ejemplos con ruido aislados.

Búsqueda de vecinos en espacios.

Se requiere la definición de una medida de disimilitud entre cada par de puntos en el espacio de representación, siendo esa medida una métrica. Se ordena un conjunto de prototipos y se almacena en una lista, la ordenación realizada es utilizada junto con la desigualdad triangular para evitar el cálculo de muchas distancias entre los prototipos y la muestra a clasificar.

Otro tipo de métodos que utilizan estructuras arborescentes para almacenar prototipos en donde el proceso consiste en una descomposición jerarquizada del conjunto de prototipos en subconjuntos disjuntos donde la búsqueda se basa en técnicas de ramificación y poda para reducir el número de prototipos (algoritmo propuesto por Fukunaga y Narendra [1a]).

El algoritmo AESA reduce significativamente el número de distancias calculadas respecto a otros métodos.

1.8 Introducción al Registrado de Modelos.

El registrado de modelos es un proceso para determinar elemento a elemento la correspondencia entre 2 modelos. También hace posible comparar información con respecto a un modelo de referencia y determinar diferencias que son causadas por cambios de escena.

En el proceso de correspondencia, se calculan las distancias entre los puntos del modelo ***P*** y los puntos del modelo ***Q***, las distancias juegan un papel muy importante ya que son utilizadas como grado de relación, es decir, se encuentra la cantidad de cambio que necesita el modelo ***P*** para alinearse con el modelo ***Q***, esto se determina utilizando la distancia euclideana entre los puntos de los modelos 3D, en donde la distancia más pequeña es considerada la verdadera posición. Por lo tanto el proceso empieza de la posición inicial en el modelo ***P***, hacia los puntos del modelo ***Q*** [2a].

La tarea del registrado de modelos implica establecer la relación de los modelos utilizando las características extraídas (*puntos*) con las que vamos a realizar cálculos matemáticos y determinar los parámetros de transformación. Para la realización de este proceso existen dos métodos: Cuaternios y SVD (Descomposición de Valores Singulares) nosotros utilizaremos el segundo, ya que es menos complicado y genera buenos resultados.

1.9 Rasgos de Correspondencia.

La correspondencia consiste en establecer una similitud entre los elementos del modelo ***P*** y los elementos homólogos del modelo ***Q***, la cual consiste en tomar un elemento perteneciente a la escena del modelo ***P*** que sea lo mas parecido posible al elemento representado en la escena del modelo ***Q***. Para ello se utiliza la métrica de la distancia, que es la que nos indica cual es el elemento homólogo de un elemento dado.

Uno de los métodos para encontrar la correspondencia es la correlación; donde se asume que el modelo izquierdo (***P***) y derecho (***Q***) son simplemente versiones desplazadas una de la otra, ambos modelos son superficies paralelas a la misma altura en condiciones idénticas de luz. El corrimiento adecuado es aquel que da cero al hacer la integral al cuadrado de la diferencia entre los dos modelos [3a].

Otra familia de aproximaciones está basada en encontrar bordes y después buscar puntos que correspondan. Los bordes se dice que son compatibles, si están lo suficientemente cerca en orientación y tienen la misma señal de contraste con la otra orilla.

Sin embargo los rasgos mas utilizados para el registrado de modelos son puntos ya que sus coordenadas pueden ser directamente utilizadas para determinar los parámetros de la función de transformación necesaria en el registrado de modelos. Los puntos

son determinados directamente o también son determinados de las intersecciones de líneas, regiones de centroides, o contornos de los modelos.

Asumimos que dos modelos están disponibles y que cada conjunto de puntos corresponde a un modelo.

El objetivo es determinar la correspondencia que existe entre los dos conjuntos de puntos. Debido al ruido y otros factores es posible que algunos puntos aparezcan en solo un conjunto de puntos. Tales puntos son llamados *outliers*. (*Es un valor que produce resultados inusuales, dato erróneo*).

Nosotros asumimos que las coordenadas de los puntos se han precalculado y son información disponible en [3a].

1.10 Proceso del Registrado de Modelos 3D.

Después de que los puntos correspondientes entre dos o más vistas ya han sido calculados, el siguiente paso es estimar la transformación rígida (rotación y/o traslación) es decir; encontrar los parámetros correctos para realizar una alineación adecuada, de los puntos del modelo ***P*** hacia los puntos del modelo ***Q***. Esto es el registrado de imágenes.

Debido a la enorme diversidad de aplicaciones del registrado de modelos, no se puede especificar un algoritmo que sirva para todas las situaciones, ya que éste no sólo debe tener en cuenta las mismas transformaciones geométricas presentes en el modelo, sino también el ruido presente de la misma [7b].

Proceso del registrado

1.-***Preprocesando:*** En esta fase se realiza la aplicación de distintas técnicas de procesado de modelos para resaltar las características que queremos comparar (puntos).

2.-***Selección de características:*** En esta fase se seleccionan las características que nos van a servir para realizar comparaciones de similitud entre los modelos 3D, como podrían ser puntos, líneas, regiones etc. Estas características se pueden extraer de forma manual o automática. Para un procesado más a fondo pueden ser representadas por sus puntos representativos (centros de gravedad, fin de línea, puntos característicos) llamados *puntos de control.*

3.-***Determinación del espacio de búsqueda:*** Determinar el conjunto (finito) de posibles transformaciones elementales (traslaciones, rotaciones, escalas).

4.- ***Extracción de la información:*** Extraer los datos 3D, a partir de la información 2D que se tiene

5.-***Determinación de la función de transformación:*** Integrar el conjunto de transformaciones elementales de tal manera que al aplicarlas conjuntamente, logren la alineación adecuada entre el modelo ***P*** y el modelo ***Q***; esto es; determinar una *función de transformación*.

6.-***Remuestreo:*** Aplicar la función de transformación al modelo ***P*** para hacerla corresponder con la estructura del modelo **Q**.

Existe una serie de problemas a la hora de hacer la implementación en cada uno de estos pasos, como se comentó anteriormente debemos ser cuidadosos a la hora de seleccionar las características para realizar el proceso de registro. Para obtener la función de mapeo, se debe tener en cuenta la información que tenemos a priori sobre la forma de adquirir el modelo y si no se dispone de dicha información, deberá escogerse una función lo más general y flexible posible. Finalmente escoger la técnica adecuada de remuestreo ya que depende de la precisión necesaria para la aplicación y de la complejidad computacional.

En las figuras 5, 6, 7, y 8 se observa un esquema del todo el proceso necesario para el registrado de modelos.

Veamos un sencillo ejemplo del proceso de registrado de modelos: Partimos de estos dos modelos, donde el segundo ha sufrido una modificación en su posición. Deseamos alinear estos modelos.

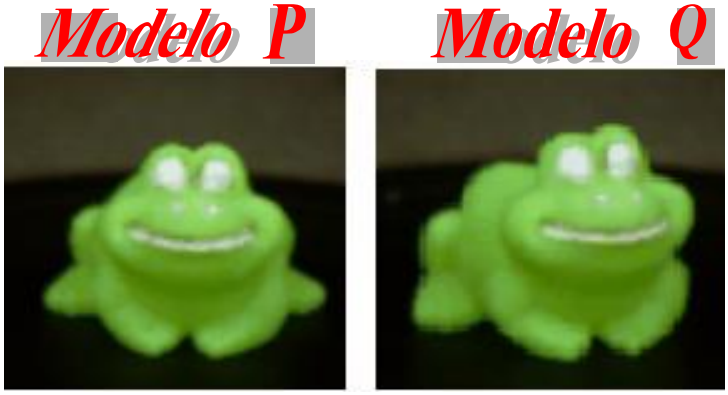


Fig. 5. Modelo P y Q Tridimensionales

Paso 1 y 2: Suponiendo que el preprocesado previo de detección de puntos se ha resuelto con anterioridad. En la figura 6 representamos gráficamente el proceso de selección de características *"puntos"*, marcadas con círculos rojos en ambos modelos.

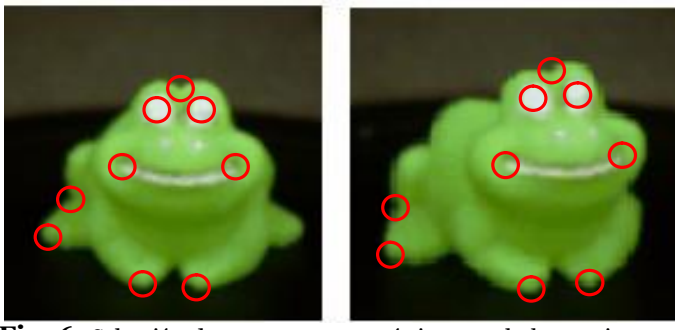


Fig. 6. Selección de rasgos o características con la herramienta cpselec que nos proporciona Matlab 7.0

Paso 3: Búsqueda de correspondencias. Resuelta por el método del vecino más cercano, mismo que facilita el conjunto de operaciones que harán corresponder a los puntos equivalentes en ambos modelos 3D. En éste caso, el objetivo es un mapeo punto por punto tal y como lo indica la imagen 7.

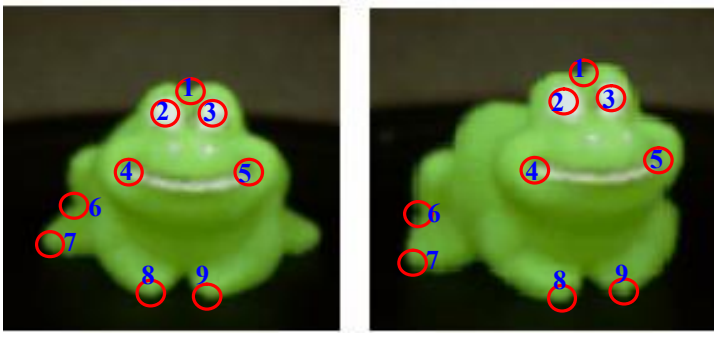


Fig. 7. Establecer correspondencia de las características (puntos del modelo)

Paso 4: Extraer información 3D, a partir de las coordenadas de textura 2D.

Paso 5: Determinar o calcular los parámetros de la función de transformación (rotación, traslación y escala) que permiten lograr la alineación de los modelos 3D. $f(t) = R + T + Sc$.

Con las transformaciones anteriores podremos determinar la función de transformación ya que es la que realiza globalmente la alineación que se está aplicando en este caso.

Paso 6: Utilizando la función Obtenida en el paso anterior podremos transformar el segundo modelo para poder alinearlo con el primero, obteniendo, en este sencillo ejemplo, un modelo idéntico al original, como se muestra en la figura 8. En un caso real, podríamos solapar los modelos y estudiar los cambios en: ojos, boca, etc.

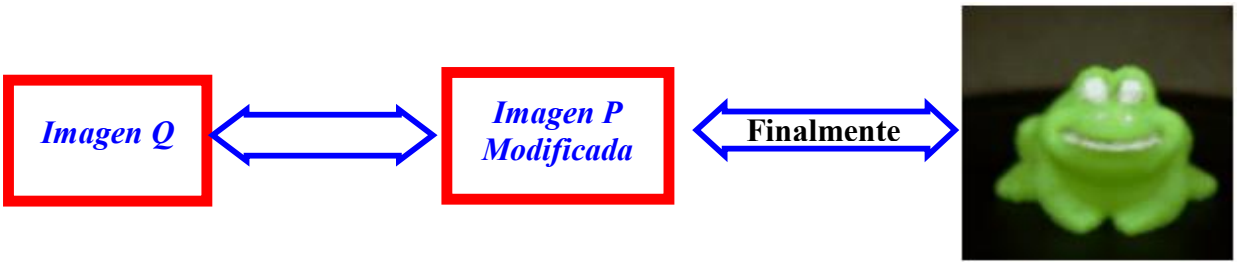


Fig. 8. Obtención de un modelo tridimensional único, formado a partir de datos parciales P y Q. En diferentes sistemas de referencia.

1.11 Fundamentos del Registrado.

En esta sección se considera el problema de registrar dos conjuntos de pares de puntos correspondientes a los modelos con las transformaciones rígidas. Esto puede ser definido como la identificación de la rotación ***R***, traslación ***T*** y escala ***S*** que lleven a los conjuntos ***P*** y ***Q*** a una alineación óptima, donde la definición de óptimo se basa generalmente en una cierta métrica de mínima distancia Euclidiana [3a].

Esto puede ser expresado algebraicamente, como encontrar (***R***, ***T***) tal que:

$$Q = RP + T \quad \text{..... 2.1}$$

La ecuación 2.1 expresa la idea de que un conjunto de datos es rotado y trasladado para que pueda ser registrado con otro.

Para el caso de dos conjuntos de puntos correspondientes $P = \{p_1, \ldots\ldots p_N\}$ y $Q = \{q_1, \ldots\ldots q_N\}$. Cada punto p_i corresponde al q_i. El objetivo es encontrar la matriz rotación ***R*** y vector de traslación ***T*** tal que:

$$q_i = Rp_i + T \quad \text{.....2.2}$$

Donde p_i y q_i son vectores que están conformados por 3 elementos, donde ***R*** es una matriz de rotación 3x3 y ***T*** es un vector de traslación de 3 elementos. A menos que los datos sean perfectos, será imposible encontrar la rotación y traslación tal que la ecuación 2.1 sea satisfecha en todos los puntos. En lugar de eso habrá un error ε en cada punto [8b].

$$\varepsilon = qQ_i - Rp_i - T \quad \text{.....2.3}$$

En general, el criterio de error del mínimo cuadrado, se utiliza para estimar los parámetros del registrado. Que es, minimizado por:

$$\sum^2 = \sum_{i=1}^{N} \left\| q_i - Rp_i - T \right\|^2 \quad \text{.....2.4}$$

Con respecto a ***R*** y ***T***

Varias técnicas se han propuesto para derivar una solución que reduce al mínimo la ecuación 2.4. Estas se describen a continuación.

1.11.1 Método SVD. (Descomposición de Valores Singulares).

El problema del registrado de modelos ha sido resuelto por varios métodos, uno de ellos es propuesto por uno de los primero trabajos de Horn, donde utiliza la técnica de cuaterniones, alrededor del mismo tiempo fue publicada una técnica equivalente por Arun et. y Horn basada en la descomposición de valores singulares [29d]; que permite resolver sistemas de ecuaciones, diagonalizar matrices singulares, resolver mínimos cuadrados, la aplicación más común es el cálculo rápido de la inversa de una matriz para resolver un sistema de ecuaciones $A^{-1} = VD^{-1}U^T$.

El primer paso es trasladar los centroides al origen, de igual manera que el cuaternión. Nosotros implementamos el método ***SVD*** ya que es más directo y se puede extender para trabajar en *n* dimensiones.

El método de la SVD es un algoritmo de factorización lineal conocido como descomposición de valores singulares; a partir del cual se genera una representación vectorial del espacio. Se utiliza para computar el registro óptimo de parámetros en correspondencia de puntos, este método es una solución para el conjunto de puntos 3D.

Es un vector o una matriz numérica que:

- Reduce los parámetros desconocidos de la traslación.
- Calcula la matriz de rotación desconocida ***SVD***.
- Calcula los parámetros de traslación.

Nota: Es importante para el análisis en los problemas que involucran matrices.

Definición:

Cualquier matriz ***A***, sea singular o no, puede ser descompuesta en un producto de tres matrices, de la forma:

$$A = U \sum V^T \text{, donde } \sum = diag(\sigma_1, ...\sigma_m) \in R^{nxm}$$

Donde ***U*** y ***V*** son matrices ortogonales, por lo que sus inversas son iguales a sus transpuestas y **∑** es una matriz diagonal en la cual se encuentran los valores singulares de la matriz **A**

Los valores singulares de una matriz $A \in R^{mxn}$ se definen como las raíces cuadradas no negativas de los valores propios de la matriz simétrica $A * A^T$, que es definida no negativa. Los valores singulares de ***A*** se ordenan en sentido decreciente [28d].
$\sigma_1(A) \geq \sigma_2(A) \geq .....\sigma_r(A)$; donde *r*=rango de la matriz *(min (m, n))*.

*A continuación les presentamos una representación ampliada de la **SVD**:*

$$A = \{u_i, u_2, ...u_r\} \begin{pmatrix} \sigma 1 & 0 & 0 & & 0 \\ 0 & \sigma_2 & 0 & & 0 \\ 0 & 0 & \cdot & \cdot\cdot & 0 \\ 0 & 0 & 0 & \cdot\cdot & \sigma 2 \end{pmatrix} * \begin{pmatrix} V_1^T \\ \vdots \\ V_2^T \\ V_r^T \end{pmatrix}$$

Esta descomposición también puede representarse como la suma de *r* matrices de rango 1, donde *r* es el rango de la matriz ***A***:

$$A = \sum_{i=1}^{r} \sigma_i * u_i * v_i$$

En la figura 9 se representan las 3 nuevas matrices (***U***) (***V***) (***E***) que dan origen a los vectores singulares de la representación matricial del espacio(***A***).

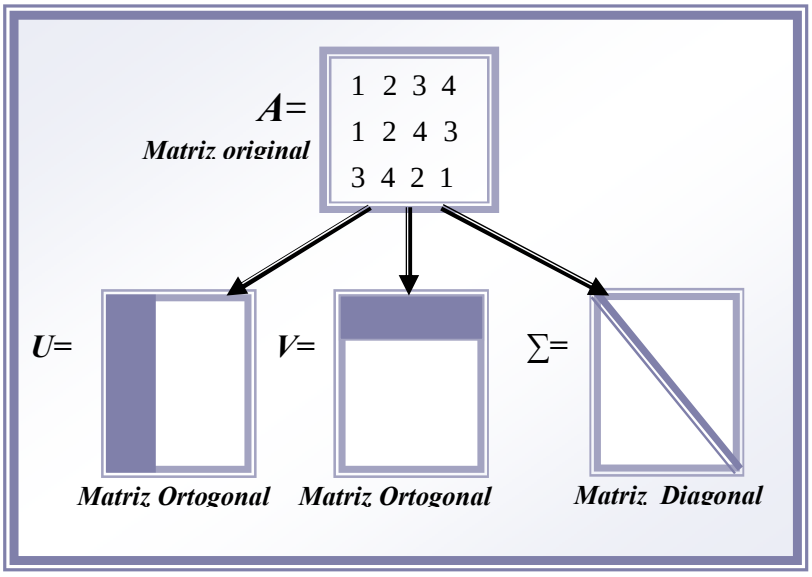


Fig. 9. Representación de la descomposición de la información de una matriz, y representarla como producto de 3 matrices.

Propiedades del método SVD.

- Define un espacio ortogonal en ***U*** cuyo número de vectores directores coincide con el rango de la matriz ***A***, y este espacio define la parte no singular de la matriz.
- Define un espacio ortogonal en ***V*** cuyo número de vectores directores coincide con la dimensión de la matriz ***A*** menos su rango y este espacio define la parte singular de la matriz.
- Los autovalores no nulos de $A^T * A$ y de $A * A^T$ son los cuadrados de los valores singulares no nulos de ***A***.
- Las columnas de ***U*** son los auto-vectores de $A * A^T$ y las columnas de ***V*** son los auto-vectores de $A^T * A$.
- Todo elemento de la diagonal de $S(\sigma_k)$ cumple que:

$$A * u_k = \sigma_k * v_k \text{, y que } A^T * v_k = \sigma_k * u_k$$

Siendo u_k una columna de ***U*** correspondiente a σ_k y v_k la columna de ***V*** correspondiente a σ_k [28d].

Parámetros de Transformación.

Los pasos para calcular los parámetros que permiten alinear los modelos se describen a continuación.
Dados dos modelo, ***P*** y ***Q,*** en donde el cálculo para encontrar la matriz de rotación ***R,*** se realiza a partir de la descomposición de valores singulares AB^T
($UU^T = VV^T = I$, $D=\text{diag}(d_i)$, $d_1 \geq d_2, \geq \ldots \geq d_m \geq 0$). Entonces el valor mínimo de $\|A - RB\|^2$ con respecto a R es:

$$\min_R \|P - RQ\|^2 = \|P\|^2 + \|Q\|^2 - 2tr(DS)$$

Donde

$$S = \begin{cases} I & if \det(PQ^T) \geq 0 \\ diag(1,1....1-1) & if \det(PQ^T) < 0 \end{cases}$$

Cuando el rango $(PQ^T) \geq m-1$, la matriz óptima de rotación R la cual logra el valor mínimo es:

$$R = USV^T$$

Donde S es elegido según:

$$S = \begin{cases} I & if\ \det(U)\det(V) = 1 \\ diag(1,1....1-1) & if\ \det(U)\det(V) = -1 \end{cases}$$

Así que, el valor mínimo de $\|P - RQ\|^2$ es logrado cuando $s_1 = s_2 = ...s_m = 1$ si el $\det(PQ^T) \geq 0$ y $s_1 = s_2 =,..., s_{m-1} = 1,\ s_m = -1$ si el $\det(PQ^T) < 0$, esto concluye la primera mitad del teorema.
Después determinamos la matriz de rotación ***R*** logrado el valor mínimo. Cuando el rango de $(PQ^T) = m$.

El valor mínimo se calcula como:

$$\varepsilon^2 = \sigma_Q^2 - \frac{tr(DS)^2}{\sigma_P^2}$$

Para realizar los cálculos de nuestro método contamos con éstas fórmulas:

$$\mu_P = \frac{1}{n}\sum_{i=1}^{n} P_i\ ,\ \ \mu_Q = \frac{1}{n}\sum_{i=1}^{n} Q_i\ ,$$

$$\sigma_P^2 = \frac{1}{n}\sum_{i=1}^{n} \left\| P_i - \mu_Q \right\|^2\ ,\ \ \sigma_Q^2 = \frac{1}{n}\sum_{i=1}^{n} \left\| Q_i - \mu_P \right\|^2\ ,$$

$$\sum\nolimits_{PQ} = \frac{1}{n}\sum_{i=1}^{n} (P_i - \mu_Q)(P_i - \mu_P)$$

Dejando una descomposición de valores singulares de la matriz de covarianza $\sum_{PQ}$ son UDV^T Donde D=diagonal y S es:

$$S = \begin{cases} I & if\ \det(\sum\nolimits_{PQ}) \geq 0 \\ diag(1,1.....1-1) & if\ \det(\sum\nolimits_{PQ}) < 0 \end{cases}$$

$\sum_{PQ}$ es la matriz de covarianza de P y Q, μ_P y μ_Q son vectores de P y ***Q***, y σ_P^2 y σ_Q^2 son las varianzas alrededor de la media de los vectores ***P*** y ***Q***, respectivamente [33d].

Cuando el rango de $\sum_{PQ} \geq m-1$ los parámetros de la transformación óptima son determinados únicamente así:

$$R = USV^T$$
$$t = \mu_Q - cR\mu_P$$
$$c = \frac{1}{\sigma_x^2 tr(DS)}$$

A continuación en la figura 10 se presentan 3 puntos pertenecientes al modelo ***P***, y en la figura 11 observamos los puntos pertenecientes al modelo ***Q***. En la figura 12 se representa gráficamente la alineación de los puntos pertenecientes a los modelos ***P*** y ***Q***. utilizando el método SVD.

Representación geométrica del método **SVD.**

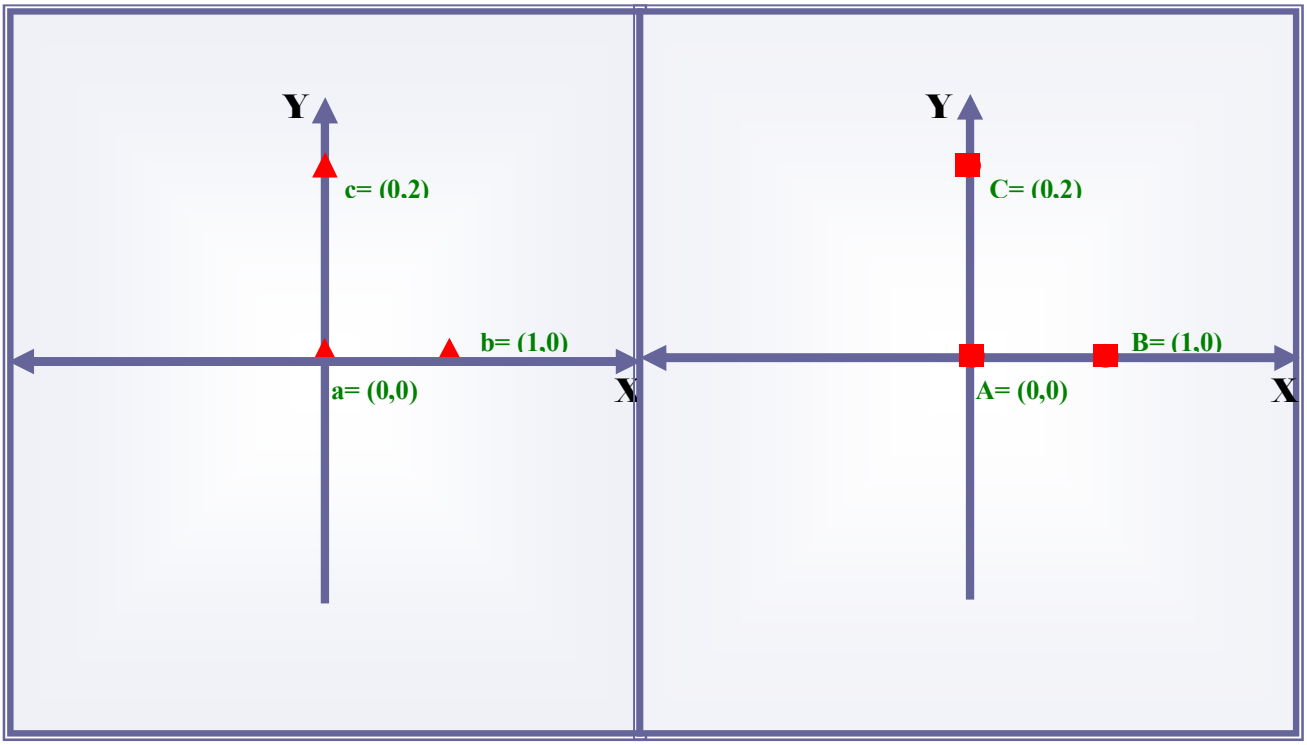


Fig. 10. Conjunto de puntos pertenecientes al modelo P.

Fig. 11. Conjunto de puntos pertenecientes al modelo Q

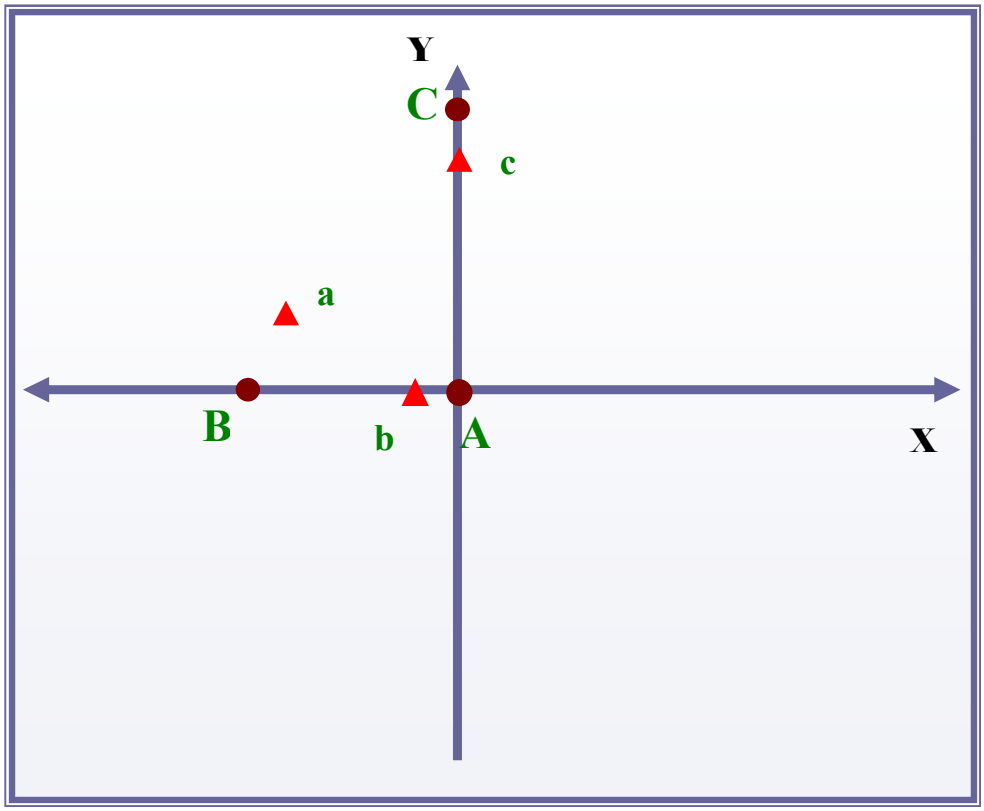


Fig. 12. Resultado de la alineación con la propiedad de la condición R utilizando el método SVD

Algoritmo SVD.
(Descomposición de Valores Singulares).

Método SVD (Descomposición de Valores Singulares)

1.-Calcular la matriz de covarianza Σ_{PQ} de forma similar al del método del cuaternión.

2.-Encontrar la descomposición de valor singular (SVD) de Σ_{PQ} tal que $\Sigma_{PQ} = \ddot{U} \wedge V^{T}$

3.-Calcular $\hat{R} = UV^{T}$

4.-Si el determinante $|\hat{R}| = -1$ entonces el conjunto $R = \hat{R}$

En caso contrario, si $|\hat{R}| = -1$ entonces el algoritmo ha fallado, y el método SVD no puede ser usado directamente.

Fig. 13. Pasos del Método SVD.

En el algoritmo descrito en la figura 13, es necesario comprobar el determinante del resultado; para estar seguros de la validez. Una determinante negativa, indica que la matriz de rotación calculada es una reflexión, que es generalmente indeseable.

Para corregir el problema Arun [3a] demostró que puede emplearse un simple procedimiento para asegurarse que existan solo las rotaciones apropiadas. Como se muestra a continuación en el siguiente algoritmo de la figura 14.

Algoritmo Asegurando rotaciones apropiadas.

Algoritmo: Asegurando rotaciones apropiadas

1.- Si $|\hat{R}| = -1$ entonces forma $V' = [v_1, v_2, -v_3]$ donde las v_i son las columnas de la SVD de la matriz V

2.- Calcular $\hat{R}' = U'V^{T}$

3.-Calcular $R = \hat{R}'$

Fig. 14. Pasos para asegurar rotaciones apropiadas

1.11.2 Quaternios.

Los cuaternios son la extensión de los números imaginarios llevados a una cuarta dimensión. Se les añade los elementos j y k.

La primera solución analítica para determinar la matriz de rotación requerida fue reportada por Horn en 1987 [8b], quién empleó un cuaternión para representar rotaciones. Los cuaterniones son una generalización de números complejos y pueden ser usados concisamente para representar rotaciones de 2D y 3D. Un cuaternión de rotación se puede representar como:

$$q_k = [q_o, q_1, q_2, q_3], \text{ donde } q_0 \geq 0 \text{ y } q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1.$$

Matemáticamente, un cuaternión puede ser visto como un número complejo con tres diferentes partes imaginarias:

$$q = q_0 + iq_x + jq_y + kq_z$$

Dados los ejes de rotación descritos por el vector unitario $n = [n_x, n_y, n_z]$ y el ángulo de rotación θ_n, el cuaternión puede ser calculado de la siguiente manera:

$$\begin{aligned} q_0 &= \cos(\theta_n / 2) \\ q_x &= \sin(\theta_n / 2) n_x \\ q_y &= \sin(\theta_n / 2) n_y \\ q_z &= \sin(\theta_n / 2) n_z \end{aligned}$$

Aquí se utilizan los cuaterniones para encontrar las rotaciones de los modelos.

El primer paso en el registrado, es trasladar el centroide del modelo ***P*** al centroide del modelo ***Q***, de tal manera ambos coincidan con las coordenadas de origen, esto se realiza de la siguiente manera:

1.- Calcular los centroides $\mu_p = \frac{1}{N}\sum_{i=1}^{N} p_i$ y $\mu_y = \frac{1}{N}\sum_{i=1}^{N} q_i$

2.- Construir el conjunto de puntos P' y Q' tal que $p_i' = p_i - \mu_p$ y $q_i' = q_i - \mu_q$

Después de estos pasos, el método del cuaternión es como se muestra en el siguiente algoritmo [3a]:

1.- Calcular la matriz de covarianza $\sum_{PQ} = \sum_{i=1}^{N} p_i' q_i'^T$ donde T indica matriz (o vector) de transposición.

2.- Formar la matriz $A = \sum_{PQ} - \sum_{PQ}^{T}$

3.- Construir el vector columna $\Delta = [A_{23} A_{31} A_{12}]^T$

4.- Formar la matriz Q de la siguiente forma:

$$\mathcal{Q} = \begin{pmatrix} tr(\Sigma_{PQ}) & \Delta^T \\ \Delta & \Sigma_{PQ} + \sum_{PQ}^{T} - tr(\Sigma_{PQ}) I_3 \end{pmatrix}$$

Donde $tr()$ es el operador de traza (trace) e $\boldsymbol{I_3}$ es la matriz identidad de 3x3

5.- Calcular el vector propio unitario $q_R = [q_0, q_1, q_2, q_3]$ de **Q** que corresponde al valor propio positivo más grande.

6.- La matriz de rotación ortonormal **R** se calcula de q_R según:

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q2_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 + q_2^2 - q_4^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0\, q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{pmatrix}$$

Después de calcular **R**, el vector de traslación **T** es calculado de la siguiente manera:
$T = \mu_P - R\mu Q$

Nota: el cuaternión y la matriz de rotación también se representan de la siguiente manera:
Cuaternión unitario: $q = [q_0, q_{x'}, q_{y'}, q_{z'}]$ que representa la rotación deseada de los conjuntos de puntos ***Q1*** y ***Q2***, es el vector propio que corresponde a los valores propios positivos más grandes [3a].

$$R = \begin{pmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_y q_x + q_0 q_z) & q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0\, q_x) & q_0^2 + q_z^2 - q_x^2 - q_y^2 \end{pmatrix}$$

La principal ventaja que tiene el método de la SVD sobre los cuaternios es que para determinar la matriz de rotación, no utiliza números complejos (imaginarios), ni funciones trigonométricas, es decir, el método SVD es directo de aplicar.

1.11.3 Transformaciones Geométricas.

Debido a que los modelos son definidos por las coordenadas de sus vértices, las transformaciones geométricas son procedimientos para calcular nuevas coordenadas para esos puntos. Es decir para poder obtener una alineación entre el modelo ***P*** y el modelo ***Q*.**

Las transformaciones básicas son: traslación, rotación y escalamiento, las cuales son descritas en esta sección. Estas transformaciones tienen dos tipos de representaciones: vectorial y matricial. Pueden representarse mediante ecuaciones que producen un cambio deseado para cada componente de un punto, es decir, una ecuación es utilizada para calcular el nuevo componente del punto *x*, otra para el punto *y* y otra para *z*. *[10b]*.

Traslación.

Si se quiere trasladar el origen del modelo se le aplica la siguiente ecuación:

$$x_j = x_i + x_0$$
$$y_j = y_i + y_0$$
$$z_j = z_i + z_0$$

En la fig. 15 se representa gráficamente la traslación de un modelo, con sus respectivas coordenadas homogéneas.

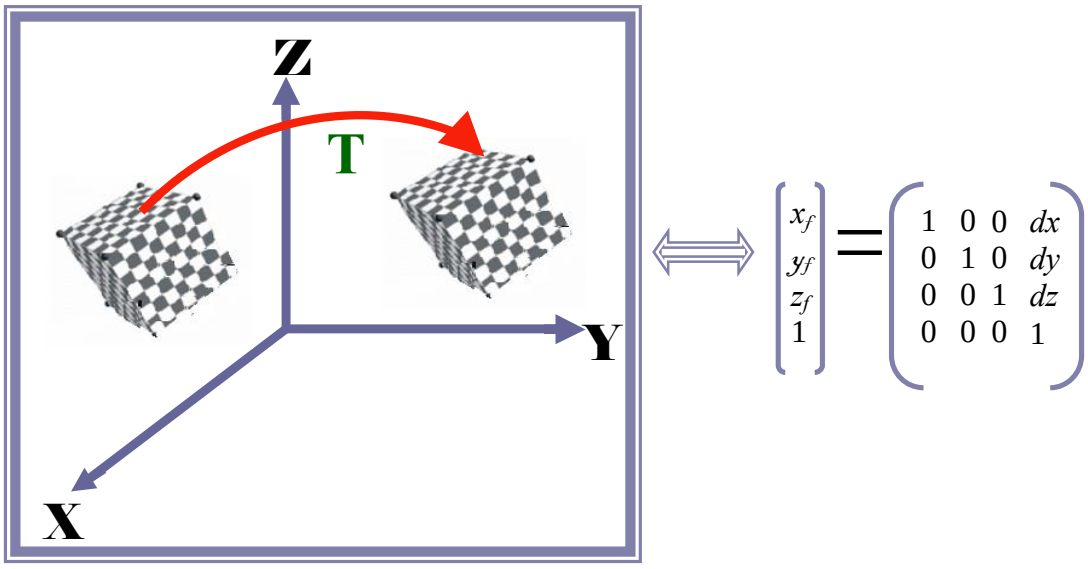


Fig. 15. En esta figura se representa gráficamente la manera de realizar la traslación a un modelo.

Rotación.

Esta transformación se realiza en una trayectoria circular se específica con un ángulo de rotación que determina la cantidad de rotación de cada vértice del objeto con respecto al origen de coordenadas. Esta transformación es la más compleja [10b]. Además se pueden definir funciones para realizar la rotación alrededor de cualquiera de los 3 ejes o alrededor de un eje arbitrario, pero teniendo en cuenta cierto criterio de signo para interpretar el sentido del giro. En la figura 16 lo representamos gráficamente.

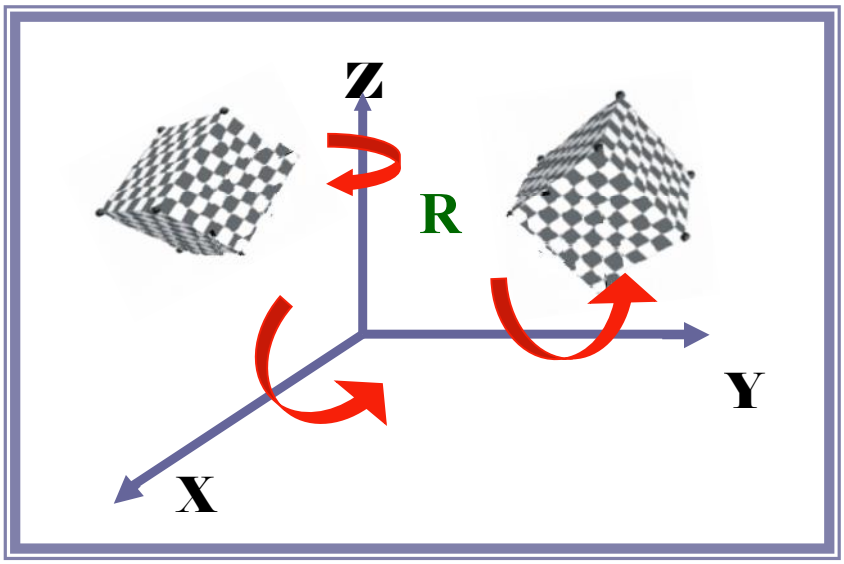


Fig. 16. En esta figura se ilustra gráficamente la manera de realizar la rotación a una figura tridimensional.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} \cos\theta_z & \sin\theta_z & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\sin\theta_z & \cos\theta_z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Mx (θ)=: ***Matriz de Rotación Alrededor del eje X*** My (θ)=: ***Matriz de Rotación Alrededor del eje Y*** Mz (θ)=: ***Matriz de Rotación Alrededor del eje Z***

Escalamiento.

Es una transformación geométrica que cambia el tamaño de los objetos. Esta acción es realizada multiplicando los puntos del cuerpo por los factores de escalamiento (S_x, S_y, S_z). Si S_x=S_y=S_z, el objeto es escalado manteniendo sus proporciones originales y se le llama escalamiento uniforme, cuando los factores de escalamiento no son iguales, el objeto es deformado con respecto a uno o varios de los ejes de coordenadas [10b]

Aunque el escalamiento puede ser representado con una matriz de 3x3, se utiliza el mecanismo de coordenadas homogéneas para ser compatible con la matriz de traslación, como se observa en la figura 17.

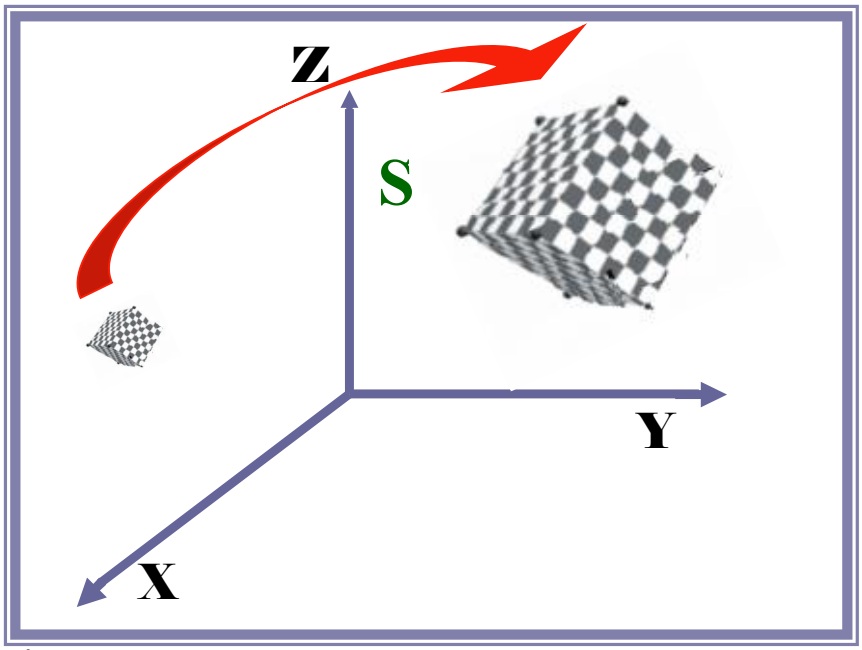


Fig. 17. En esta figura se ilustra gráficamente la manera de realizar el escalamiento a una figura tridimensional.

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} xS_x \\ yS_y \\ zS_z \\ 1 \end{pmatrix}$$

1.11.4 Algoritmo Iterative Closest Point (**ICP**).

Introducción.

El algoritmo ICP es extensamente utilizado para la alineación geométrica de modelos tridimensionales cuando se conoce una estimación inicial; es decir cuando inicialmente se tiene una estimación aproximada de los puntos de control o correspondencia de cada uno de los modelos **P** y ***Q***. La estimación inicial es requerida, por que el algoritmo ICP es utilizado para estimar una sintonización fina del movimiento y no para calcular estimaciones de movimiento grandes [3a]. El algoritmo ICP calcula el método para la estimación de la posición (rotación+traslación+escala) entre los datos de rango de una vista parcial y los del objeto completo al cual pertenece.

El algoritmo ICP fue propuesto por Besl y Mckay [11b] y también descrito por Chen y Medioni [20b]. De este algoritmo se mencionan dos propuestas: El primer grupo de técnicas tiene el inconveniente de que un modelo debe ser un subconjunto del siguiente y además es necesario conocer una buena aproximación inicial. En caso contrario, el alineamiento podría ser distinto para cada nube de puntos o el ir probando aproximaciones iniciales con objetos acercándonos a la solución correcta, hacer esto podría tardar mucho tiempo. El segundo grupo de técnicas requiere que se evalúen normales en los puntos de una superficie para hallar intersecciones con la otra superficie con la que se pretende alinear, aumentando la complejidad [13b].

El algoritmo se basa en encontrar el punto más cercano de una nube de puntos respecto a otra nube; este algoritmo siempre converge hacia un mínimo local de una distancia media cuadrática. Tomar una muestra del modelo con el que vamos a alinear, nos permite reducir el tiempo empleado y hacerlo robusto en presencia de *outliers (son puntos que tienen comportamiento diferente con respecto al resto de los datos, observaciones atípicas o erróneas)*. El objetivo es minimizar la distancia.

El procedimiento más costoso en términos de computación, es el establecimiento de la correspondencia del punto. Este algoritmo ***ICP*** original evita el problema de mínimo local usando muchos movimientos iniciales. El método es muy sencillo ya que a partir de un conjunto de datos de ***P*** y un subconjunto de datos de ***Q***, lo que hace es calcular para cada punto *p* el punto *q* más cercano a él. Una vez realizado este paso se evalúa la transformación ***T*** que minimiza la suma de los cuadrados de las distancias entre las parejas de puntos (*p, q*) y se aplica ***T*** sobre ***P*** y se repite el algoritmo hasta que la suma llegue a un mínimo [6a].

Una característica de esta técnica es su simplicidad, puesto que requiere solamente un procedimiento para determinar el punto más cercano en una entidad geométrica a un punto dado, y un procedimiento para calcular la transformación rígida entre dos conjuntos de puntos con correspondencia conocida [3a].

Descripción General del algoritmo.

Como se ha comentado a largo de esa tesis el método propuesto en este trabajo se basa en el cálculo de las componentes parciales sobre los datos de rango de unas vistas parciales, que denotaremos como ***X***. Éstas se definen a partir de los autovalores de la matriz de covarianza.

$$Q = X_c X_c^T \quad \text{.....1}$$

Donde Xc, son los datos desplazados respecto a su centroide. Xc es una matriz de $nx3$, por lo que los autovectores serán 3 y nos marcan las 3 direcciones principales de éstos. El primer autovector nos indica la dirección donde se encuentra la máxima varianza en los datos. Desde un punto de vista geométrico y suponiendo que los datos se encuentran equiespaciados, esto implica que la distancia entre los datos extremos proyectados sobre el primer autovector, es la máxima entre todas las posibles en Xc. El segundo marca una dirección normal a la primera, en la que existe en la varianza. El tercero forma con los otros dos un sistema ortonormal, hay que tener en cuenta que los autovalores son invariantes frente a las rotaciones y desplazamientos; y los autovectores frente a desplazamientos, correspondiéndose estos últimos con un sistema de referencia ligado a los propios datos. Esto hace que se puedan emplear los primeros para evaluar qué parte de los datos de rango se corresponden con la vista parcial y los segundos para calcular una primera aproximación a la transformación que deben sufrir los datos parciales para ser acoplados. Ésta solo reflejará la submatriz de rotación ya que los orígenes de los triedros coincidirán. La comparación de los autovalores da información de cuales son las posibles zonas en donde la vista parcial puede acoplarse, pero dicha información es de carácter global y como hemos dicho solo da información de la rotación. Será necesario la última fase de calculo que nos marque cual es la transformación definitiva. Y para ello emplearemos el algoritmo ***ICP***, que debe partir de una aproximación a la transformación inicial, que en nuestro caso corresponde en el acoplamiento de los autovectores. La zona donde se acoplará la vista y la transformación será medida mediante el error final del algoritmo.

Aplicación del Algoritmo.

El ***Iterative Closest Point*** (***ICP***) [11b] es un algoritmo iterativo que minimiza el error cuadrático medio.

$$e(k) = \frac{1}{n}\sum \left\| P - P' \right\|^2$$

Entre los n datos de rango de un objeto P', se denominan los datos de la escena y los datos del objeto, del conjunto de datos ***P***, que denominan los datos del modelo. En nuestro caso los datos de la escena son los datos de la vista parcial real Xc, normalizados y transformados mediante la matriz homogénea **T**, y los datos del modelo son aquellos datos del modelo completo Mn, que se encuentran más próximos a los datos de la escena. Éstos últimos variarían en cada paso de la iteración.

Al igual que en casos anteriores, el símbolo $\| \ \|$ representa la distancia euclideana. Determinados los datos y suponiendo que el (k) es mayor que el error final, se tiene que determinar la transformación que haga que el error e (k) sea mínimo en ese paso de iteración.
La solución para el vector desplazamiento, t es [12b]:

$$t = \frac{1}{n}\sum_{i=1}^{n} r_i - \frac{1}{n}\sum_{i=1}^{n} r_i'$$

Donde r_i y r_i' son las coordenadas de los datos del modelo y de la escena respectivamente.
Con respecto al cálculo de la matriz de rotación que minimiza el error, hemos empleado la aproximación de Horn [14b]. En ésta, se define **R** como los autovectores de la matriz **M** que se define como:

$$\mathbf{M} = \mathbf{P}\,(\mathbf{P}')^T$$

Esta es la aproximación que nos da una solución cerrada a dicho cálculo, lo que acelera de forma muy significativa el algoritmo ***ICP***. La matriz de transformación final será aquella en la que el error del ***ICP*** sea menor.

Taxonomía de las Variantes del ICP.

Han sido propuestas muchas variantes de éste algoritmo, afectando las fases de éste, la selección, la alineación de puntos, y la estrategia de minimización [13b].

En [13b] se enumeran muchas de éstas variantes y evalúan su efecto sobre la velocidad con la cual se alcanza la alineación correcta. Para mejorar la convergencia de los modelos con los rasgos mas pequeños, tales como: superficies inscritas; introducen una nueva variante basada en el muestreo [11b] uniforme del espacio. Concluyen proponiendo una combinación de las variantes del ***ICP*** optimizadas para la velocidad. Esta capacidad tiene uso potencial en la adquisición del modelo 3D en tiempo real.

Las comparaciones sugieren una combinación de las variantes del ***ICP*** que puedan alinear un par de modelos en 10 milisegundos, perceptiblemente más rápidamente que la mayoría de los sistemas de uso general del ***ICP***. La disponibilidad de un algoritmo en tiempo real del ***ICP*** puede permitir nuevos usos significativos y en seguir la exploración de modelos basados en 3D [13b].

El **ICP** se ha convertido en el método dominante para alinear modelos tridimensionales. Inicia con dos modelos y una aproximación inicial, generando pares de puntos correspondientes en los modelos y reduciendo a un mínimo error métrico.

Nosotros nos enfocamos solamente a alinear un solo par de modelos, y no tratar el problema de manera global [15b], [16b], [17b], [18b]. Desde la introducción del **ICP** por Chen y Medioni [19b] y [20b] se han introducido muchas variantes con el concepto básico del **ICP.**

Podemos clasificar éstas variantes como:

- **Seleccionar** puntos (rasgos) pertenecientes a ambos modelos.
- **Eligiendo** apropiadamente los pares correspondientes.
- **Rechazando** ciertos pares de puntos.
- **Asignando** un error métrico basado en los pares de puntos.
- **Minimizando** el error métrico.

Metodología de la Comparación.

La meta de [13b] fue comparar las características de convergencia de las variantes del **ICP**. Para limitar el alcance del problema y evitar una explosión combinatoria en el número de posibilidades. El algoritmo que se seleccionó como estándar fue propuesto en [21], sobre el cual incorporaron las características siguientes:

- Muestreo al azar de puntos en ambos modelos.
- Alineando cada punto seleccionado a la muestra más cercana del otro modelo que tiene una fuente normal dentro de los 45°.
- Elegir los pares de puntos.
- Rechazar los pares que tienen las distancias más grandes.
- El error métrico del punto al plano.
- La clásica iteración "seleccionar-alinear-minimizar" mas que alguna otra búsqueda para la alineación.

Comparaciones de las Variantes del ICP.

En [13b] fueron examinadas las variantes del **ICP**, para cada una de las etapas compararon su funcionamiento en las mismas escenas de prueba.

Selección de Puntos.

Se han propuesto las estrategias siguientes. Para el efecto de la selección de los pares de puntos sobre la convergencia del **ICP**.

- Utilizar siempre todos los puntos disponibles [19b].
- Realizar un submuestreo uniforme de los puntos disponibles.
- Muestra aleatoria (con una muestra diferente de puntos en cada iteración).
- Selección de puntos con el gradiente de alta intensidad, en las variantes que utilizan color o intensidad para ayudar en la alineación.
- Cada uno de los esquemas precedentes puede seleccionar puntos solamente en un modelo, o seleccionar puntos de ambos modelos.

Además de éstos se introduce una nueva estrategia de muestreo la cual consiste en elegir puntos tales que la distribución de normales entre puntos seleccionados sea lo mas grande posible. La motivación para esta estrategia es la observación, que para ciertos tipos de escenas los rasgos pequeños del modelo son vitales para determinar la alineación correcta. Una estrategia tal como la muestra aleatoria, seleccionará a menudo solamente algunas muestras en estas características, que lleva a una incapacidad para determinar ciertos componentes de la transformación correcta del cuerpo rígido [2a].

Alineación de Puntos.

La siguiente etapa del **ICP** es encontrar la correspondencia de puntos. Se han propuesto algoritmos para cada muestra de puntos seleccionados:

- Encontrar el punto más cercano al otro modelo [19b]. Éste proceso de cómputo puede acelerarse usando un árbol k-d o eligiendo el closest-point [21b].

- Encontrar la intersección del rayo que origina el punto del modelo de referencia en dirección a la superficie destino [20b]. Esto lo llamamos "normal shooting".

- Proyectar el punto del modelo ***P*** hacia el modelo de ***Q*** o destino del punto de vista de la cámara del rango del modelo destino [20b]. Esto se llama "calibración hacia atrás".

- Proyectar el punto fuente sobre el modelo destino, realizar entonces una búsqueda en el modelo de rango destino. La búsqueda podría ser una métrica basada en la distancia punto a punto [5], compatibilidad de intensidad o de color [20].

- Cualquiera de los métodos anteriores, solo se restringe a alinear puntos compatibles con el punto del modelo fuente o de referencia de acuerdo a una métrica. La métrica de compatibilidad basada sobre el color y el ángulo entre las normales, se ha explorado [21].

En particular las variantes que se comparan en [13b] son: closest-point, compatibilidad del closest–point, normal shooting, compatibilidad de puntos, proyección, y la búsqueda de proyección.

Se han enfocado a ver el error como una función del número de iteraciones, pero también es bueno verlo como una función de tiempo. Por que la etapa de alineación del algoritmo **ICP** es larga y se debe escoger el algoritmo que realice esta aplicación en un tiempo de ejecución corto. Por lo tanto, ven al mínimo error como una función de tiempo, para éstos algoritmos.

Algoritmo ICP.

El conjunto de puntos P y la segunda vista Q son seleccionados por el muestreo aleatorio, estos conjuntos van a ser procesados por el algoritmo **ICP** [11b].

Cada $p_i \in P$ es temporalmente emparejado al punto más cercano $q_j \in Q$, que satisface $\|p_i - q_j\| = \min\|p_i - r\|$, con $r \in Q$.

De este modo determinamos un conjunto de puntos $P = \{p_i\}$ $(1 \le i \le Ns)$ que denotamos como $P = C(P, Q)$

Los conjuntos P y Q son utilizados para estimar los parámetros del movimiento rígido T_{ICP} utilizando la representación de cuaternios. El conjunto P es sustituido por $Tr = RP + T$ y estos a su vez son iterados hasta que p_i y q_j convergen.

El algoritmo **ICP** (P, Tr, Q) se asume como lo indican los pasos de la figura 18:

Algoritmo: Iterative Closest Point (ICP)

1.- $k \leftarrow 1, P \leftarrow Tr(P), d_0 \leftarrow \infty$

2.- La correspondencia del punto establecida: $P \leftarrow C(P, Q)$

3.- El movimiento entre parejas puntos es estimado usando la representación de cuaternios $Tr \leftarrow Q(P, Q)$

4.- Aplicamos el movimiento estimado al conjunto de puntos $P \leftarrow Tr\,(P)$

5.- Evaluamos la distancia entre los conjuntos de puntos p y q: $d_k \leftarrow d(p_i, q_j)$

6.- Los pasos 2-5 son repetidos, incrementando $k \leftarrow k + 1$, hasta que d_{k-1} y $d_k < \tau\sigma$, donde $d_k = d(p_i, q_j)$ es la suma de los errores al cuadrado medios entre p_i y q_j:

$$d_k = \frac{1}{Np}\sum_{i=1}^{Np}\left\|p_i^I - p_i^{II}\right\|^2$$, donde τ es la tolerancia de convergencia

Fig. 18. Algoritmo ICP utilizado para el registrado de modelos.

1.11.5 Enfoques para la Construcción de Algoritmos de Indexación.

Un algoritmo de indexamiento es un procedimiento independiente para construir una estructura de datos (ó índice) es diseñado para ahorrar cálculos de distancia sobre muchas consultas en un conjunto de puntos, cuando se responde a una consulta de proximidad, esta estructura de datos puede ser costosa de construir, todos los algoritmos de indexamiento trabajan en base a descartar elementos usando la desigualdad del triangulo (única propiedad para ahorrar cálculos de distancia).

Finalmente el objetivo de este trabajo es implementar un algoritmo que sea capaz de encontrar el *k* vecino más cercano, utilizando un índice de complejidad lineal de la construcción, y con esto reducir el número de cálculos de distancia, que se compara con la solución al problema. Aplicando la fuerza bruta es resuelto utilizando n^2 cálculos de distancia.

Básicamente existen dos enfoques para la construcción de algoritmos de indexación en espacios métricos: por pivotes y por particiones compactas [5a].

Algoritmos de búsqueda por pivotes.

Los algoritmos basados en pivotes son herramientas efectivas para búsquedas de proximidad en espacios métricos; mismos que permiten negociar entre el espacio ocupado y el número de cálculos de distancia desarrollados en tiempos de consulta. Estos algoritmos durante la indexación, seleccionan *k* pivotes $\{pi, p2, p3, ...pk\}$, y le asignan a cada elemento del conjunto de datos el vector o firma:

$$\Phi(a) = d(a, p_1); d(a, p_2), ..., d(a, p_k)$$

Durante la búsqueda utilizan la desigualdad triangular junto con la firma de cada elemento para filtrar elementos del conjunto de datos sin medir su distancia a la *query*. Dada $(q,r)_d$ se computa la firma de la *query*: $\Phi(a) = d(a, p_1); d(a, p_2), ..., d(a, p_k)$ y luego se descartan todos aquellos elementos a, tales que para algún pivote p_i, se cumple que: $\left|d(q, p_i) - d(a, p_i)\right| > r$ es decir: Máx. $1 \le i \le k$ $\left|d(q, p_i) - d(a, p_i)\right| > r$.

Los elementos no descartados forman parte de una lista de candidatos, que posteriormente se comparan directamente con el *query*.

Dentro de los algoritmos basados en pivotes se encuentran mencionados en el diagrama de la figura 19.

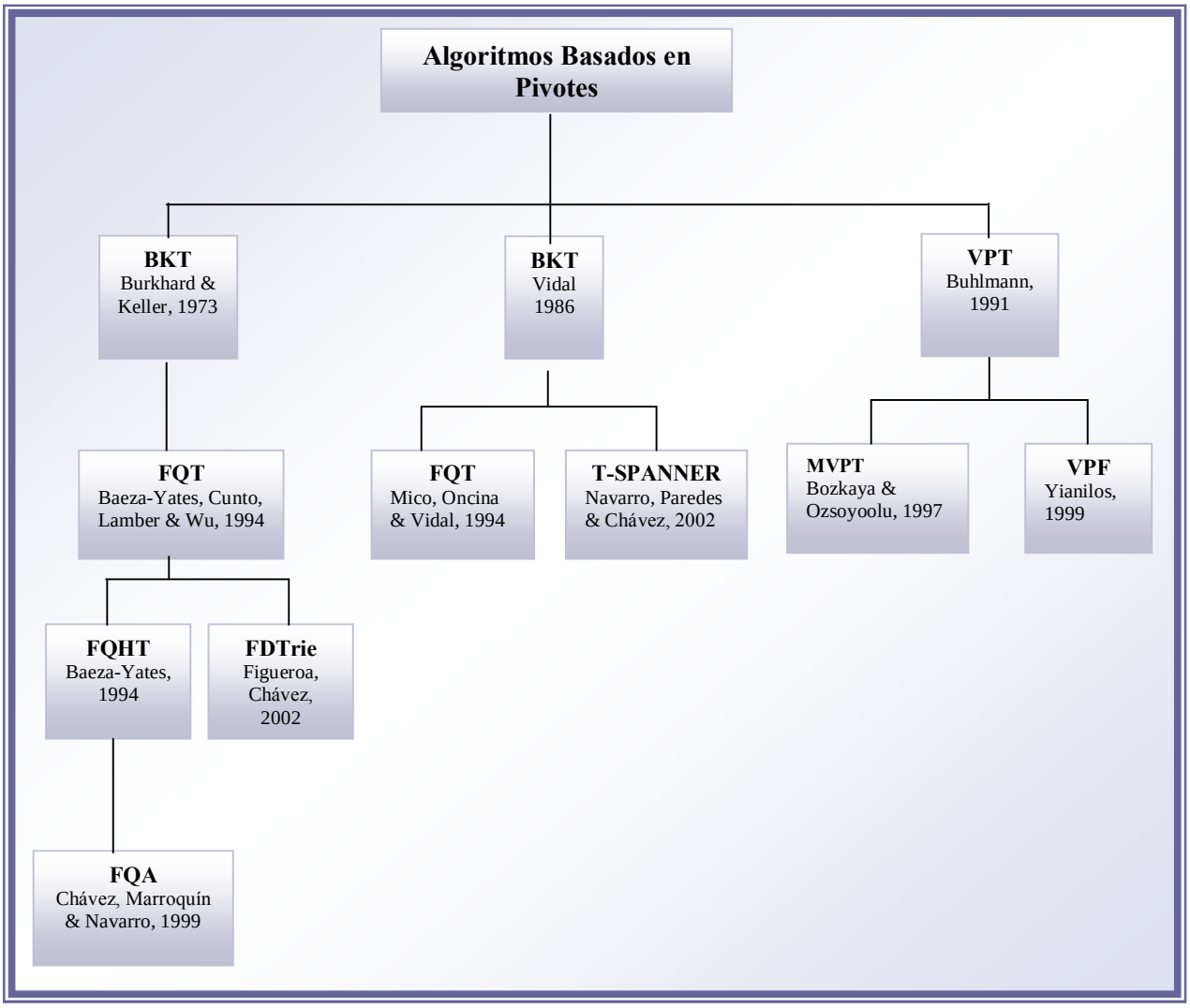


Fig. 19. Aquí se muestran todos los Algoritmos Basados en Pivotes. Que son efectivos para la búsqueda de aproximidad [5a]

Algoritmos de búsqueda por particiones compactas.

Estos algoritmos dividen el espacio en zonas tan compactas como sea posible, y almacenan un elemento (centro) representativo de la zona. Junto con el centro se almacena la información adicional que permitirá durante la búsqueda descartar aquellas zonas que no contengan elementos de interés. En la figura 20 se presenta la clasificación de los algoritmos basados en clusters o particiones compactas:

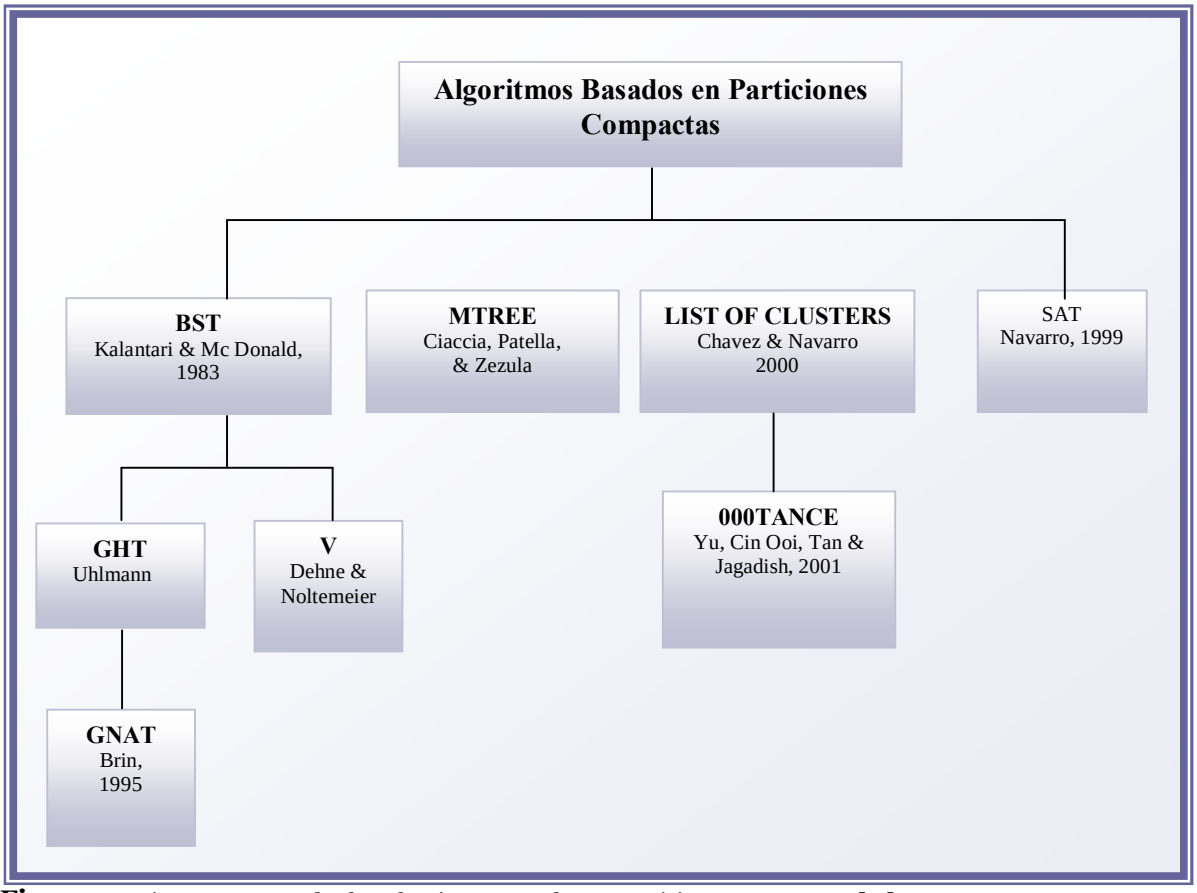


Fig. 20. Aquí se muestran todos los Algoritmos Basados en Particiones Compactas. [1a]

CAPÍTULO II

II. Análisis y Diseño.

2.1 Buscando proximidad en Espacios Métricos.

La búsqueda es un problema fundamental en ciencias computacionales y está presente virtualmente en cada aplicación de computadora.

La operación de búsqueda ha sido tradicionalmente aplicada a "datos estructurados", como lo son las: bases de datos; que son construidos en entorno al concepto de búsqueda exacta y están divididas en registros; cada registro tiene una llave completa comparable. Las consultas a la base de datos regresan todos los registros cuyas llaves coinciden con la llave de búsqueda.

Sin embargo con la evolución de la información y las tecnologías de la comunicación, los bancos de información no estructurada han emergido. Tales como: reconstrucción de modelos, audio y video. He aquí la necesidad de contar con nuevos modelos de búsqueda en bancos de información no estructurados, como lo es la "*búsqueda por proximidad*" [11b].

El concepto de "búsqueda por similaridad ó proximidad" consiste en buscar a aquellos elementos del conjunto de datos que sean similares o cercanos a un elemento de consulta dado. La similaridad es modelada con una función de distancia que satisface la desigualdad del triángulo y el conjunto de objetos es llamado ***"espacio métrico".***

Normalmente la función distancia es definida por expertos en cada aplicación y en particular en el problema del registrado ya que nos dice qué similitud existe entre los puntos correspondientes de cada modelo y por consiguiente permite establecer la correspondencia entre los puntos pertenecientes a los modelos parciales. Puesto que la distancia es normalmente cara computacionalmente, la meta es realizar mínimas evaluaciones computacionales de distancia como posible sea; para satisfacer los cuestionamientos *query´s* . Se utilizan distancias precalculadas entre los elementos de la del conjunto de datos para acelerar los cuestionamientos. Para reducir el costo se puede construir un índice sobre los elementos de un modelo; éste es un dato estructurado que almacena información sobre algunas distancias entre los elementos del conjunto de datos. Esta información es utilizada para más adelante eliminar algunos elementos sin comprar+arlos directamente con el cuestionamiento.

En general la mayor información que un índice almacena, es el costo mas bajo del cuestionamiento logrado. En un conjunto de datos de *n* objetos, la mayor información que un índice puede almacenar es el: $\frac{n(n-1)}{2}$ distancias entre todas las parejas de elementos. Actualmente todo el desarrollo del espacio métrico indexado puede ser considerado como un cuestionamiento eficientemente mientras se va reduciendo la cantidad de información almacenada [27b].

Esta es la idea básica de los algoritmos como **AESA**, el cual almacena todas las distancias de los objetos de un conjunto de datos. En esta tesis mostramos la implementación de la mejora que se realizo en el algoritmo **AESA**; utilizando un método radicalmente diferente para seleccionar los elementos del conjunto de datos y compararlos con la interrogante *q*. Los experimentos hechos en [1a] muestran una mejoría de hasta un 75% en aplicaciones con conjuntos de datos.

Los algoritmos de búsqueda por medio de indexado son útiles para resolver el problema que se presenta en el registrado de modelos. Este problema consiste en: encontrar para cada elemento su ***K*** vecino más cercano, dado un modelo de *n* puntos o elementos en un espacio métrico. La complejidad del problema es medida en cálculos de distancia ya que de antemano se sabe que usando la fuerza bruta el problema es solucionado con n^2 cálculos de distancia.

La solución planteada para encontrar los ***K*** vecinos más cercanos consiste en hacer uso del algoritmo de indexamiento "**IAESA**"el cual se implemento en esta tesis y hace una mejora en el criterio de aproximación sobre el algoritmo AESA.

Este algoritmo funciona a base de permutaciones, estas se forman calculando la distancia entre el elemento (*query, o puntos pertenecientes al modelo*) y cada pivote perteneciente al conjunto de "permutantes" los cuales son seleccionados de manera aleatoria.

El cálculo de las distancias nos ayuda a formar las permutas proporcionando las posiciones en que se encuentran los puntos, de tal manera que a menor distancia mayor acercamiento existe entre los puntos, las permutaciones son ordenadas de manera creciente. Una vez formadas, se comparan de mayor a menor similaridad con la consulta, de esta manera encontramos el vecino mas cercano. Cada vez que IAESA elije un nuevo pivote se actualizan las permutaciones. El radio sirve para ver que elementos pertenecen a la hiperesfera de la consulta, es decir, cuales son los elementos más cercanos a la consulta, y obtener el vecino más cercano.

2.2 Notación y Conceptos básicos.

Denotaremos con ***X*** el universo de objetos válidos. Un subconjunto finito $U \subseteq X$, de cardinalidad *n*, que sería el conjunto sobre el que realizaremos las búsquedas. La función $d = X \times X \to R^+$ denotará una medida de distancia entre objetos de ***X***, esto significa que *a menor distancia más cercanos o similares son los objetos*.

El par (***X***, *d*) se denomina espacio métrico, específicamente en el problema del registrado tenemos que: dado un punto del modelo ***P***, se debe encontrar el punto más similar del modelo ***Q*** mediante una búsqueda por rango. Que denotaremos con: $(q', r)_d$ un elemento $q' \in X$, al que llamaremos *query* y un radio de tolerancia r; una búsqueda

por rango consiste en recuperar los objetos del conjunto de datos cuya distancia a la *query* no sea mayor que r, es decir: $(q',r)_d = \{u \in U : d(q';u) \le r\}$ mínima.

En particular, si solo es necesario obtener el vecino más cercano, entonces $(q',r)_d$, se pueden preprocesar los modelos por medio de un algoritmo de indexación con el objetivo de construir una estructura de datos índice, diseñada para ahorrar cálculos en el momento de resolver una búsqueda [27b].

La función distancia debe satisfacer, las siguientes propiedades, dados los siguientes modelos ***P*** y ***Q*** :

- **Reflexividad:** $\forall p,q \in E, d(p,q) = 0 \Leftrightarrow p = q$
- **Simetría:** $\forall p,q \in E, d(p,q) = d(q,p)$
- **Positividad Estricta:** $\forall p,q \in E, p \neq q, d(p,q) > 0$
- **Desigualdad Triangular:** $\forall p,qc \in E, d(p,q) + d(q,c) \ge d(p,c)$

Dados $U \subseteq X$ nuestro conjunto de datos de tamaño ***n,*** $q' \in X$ y $r \ge 0$.

La igualdad de las preguntas puede ser clasificada en 2 tipos básicos:

- ***Consultamos el Rango:*** $(q',r)_d = \{u \in U \mid \mathrm{d(u,q')} \le \mathrm{r}\}$

- ***Consultar el k-ésimo vecino más cercano:*** $kNN(q')_d = P$ tal que $\forall u \in P, u \in U - P$, $d(u,q') \le d(v,q')$ y $|P| = k$.

El acercamiento a éste tipo de interrogantes es para comparar totalmente el conjunto de datos contra las consultas. Ésta solución sin embargo, requiere *n* cálculos de distancia. Un índice es una estructura de datos en ***U***, esto resuelve preguntas de cualquier tipo intentando usar menos evaluaciones de *n* distancias.

La búsqueda siempre procede comparando *la query* contra un cierto número $u \in U$ descartando candidatos que usan esa distancia y la ayuda del índice, y así sucesivamente hasta que cada elemento es descartado o reportado.

El rendimiento de los algoritmos en los espacios métricos es afectado por la dimensión intrínseca (propia) de los datos. En altas dimensiones, no hay algoritmos que puedan evitar el análisis secuencial. ***AESA*** es también afectado por la dimensión a pesar de ser el mejor algoritmo de la búsqueda de aproximidad en espacios métricos

Un algoritmo de indexación se considera eficiente, cuando es capaz de realizar una búsqueda por similitud, haciendo una cantidad pequeña de cálculos de distancia, sublineal en la cantidad de elementos del conjunto de datos [1a].

2.3 Tipos de consultas de interés para los espacios métricos.

a) ***Consulta de rango o de proximidad:*** la cual consiste en recuperar todos los elementos los cuales están dentro de una distancia r de q. Esto es $\{u \in U | d(q,u) \le r\}$. Denotaremos esta consulta por $(q',r)_d$. En la figura 21 se representa este tipo de consulta.

b) ***Consulta del vecino más cercano:*** Este tipo de consulta recupera el elemento más cercano a $q' \in U$. Esto es: $\{u \in U | \forall u \in U, d(q',u) \le d(q',u)\ \}$ podemos dar una distancia máxima *r** tal que si el elemento más cercano esta a una distancia mayor que *r**; no queremos ningún reporte.

c) ***Consulta de los vecinos más cercanos:*** En la cual se recuperan los **K** elementos más cercanos a $q' \in U$. Esto es, recuperar un conjunto $P \subset U$ tal que $|P| = K$ y $\forall uP, u \in U - P, d(q',u) \le d(q',u)$.
El tipo de consulta (a) se soluciona de la siguiente manera. El conjunto $\{u \in U : d(q,u) \le r\}$ será llamado resultado de la consulta de proximidad.

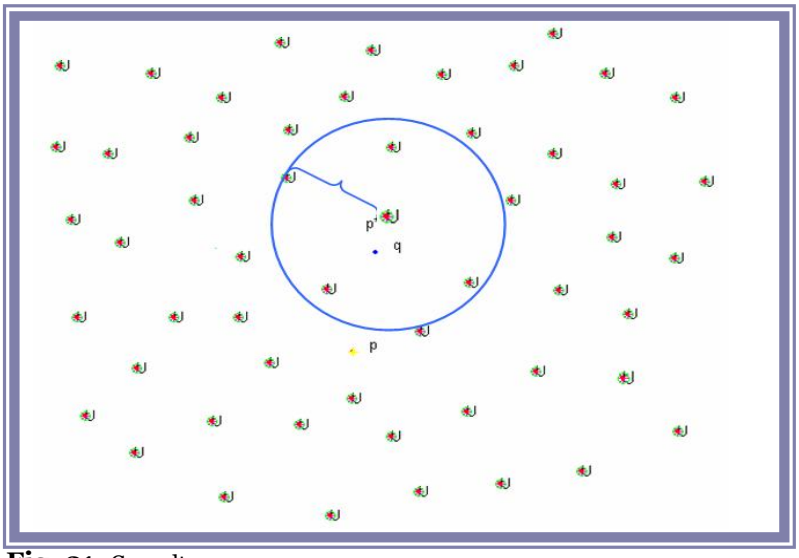


Fig. 21. Consulta por rango

Existen otros dos tipos de consultas que son solucionadas utilizando una variante de consulta de rango. Por ejemplo la consulta de tipo (b) la cual es normalmente es solucionada con una consulta de rango donde el radio es infinito y es reducido con los elementos más cercanos a la consulta a medida que son encontrados. Normalmente se trata de obtener los elementos más cercanos tan rápido como sea posible (el problema siempre es fácil para los radios pequeños). Otro tipo de consulta (c) es normalmente solucionado como una variante de tipo (b) donde los K elementos más cercanos son

guardados y la r activa es la distancia más larga de estos elementos a la consulta (Los elementos más lejanos no son de interés).

2.4 Buscando proximidad Utilizando AESA.

La "búsqueda de proximidad o similaridad", consiste en buscar en el modelo objetivo, aquellos elementos similares o cercanos a un elemento del modelo de referencia dado.

El Algoritmo de Búsqueda de Aproximación y Eliminación fue introducido por E. Vidal en 1986 [3a] mismo que por 20 años ha tenido la técnica requerida de indexación [3b].

AESA para calcular y almacenar distancias necesita una matriz como un índice registrando cada distancia $d(u,v) \quad \forall u,v \in U$, esto es $O(n^2)$ distancias. Durante el proceso de búsqueda, elige un elemento del conjunto de candidatos (*puntos del modelo*), llamado "*pivote*" y es comparado con la interrogante $query$, para podar o eliminar elementos del mismo conjunto de datos, es decir, cuanto mas cerca esté el *pivote* de la consulta mas eficaz es la poda; AESA utiliza esta matriz para descartar candidatos restantes (*elementos del modelo*) utilizando la desigualdad del triángulo. Nosotros implementamos una nueva técnica llamada iAESA propuesta en [1a] que presenta un esquema novedoso para elegir el próximo pivote.

En [27b] se menciona que en dimensiones muy grandes AESA y iAESA aun están bajo un análisis secuencial. Y que están explorando el uso de iAESA como un esquema probabilístico que pueda perder algunas respuestas relevantes, pero podría rápidamente encontrar la mayoría de ellos, en donde iAESA recupera 85% de las respuestas correctas examinando el apenas 10% del conjunto de datos. Eso es 80% menos de lo que se necesitaría para obtener el mismo resultado con probabilística de AESA.

Concretamente la estrategia de búsqueda es la principal aportación del algoritmo presentado en [1a]. En esta fase se selecciona un prototipo aproximadamente cercano a nuestra muestra que es la $query$, se calcula la distancia entre ambos y se actualiza el vecino más próximo, ésta distancia debe ser menor que la del más cercano hasta el momento. En la segunda fase de eliminación, usando la desigualdad triangular se eliminan para posteriores búsquedas todos los prototipos que no pueden estar más cercanos a la muestra que el actualmente se ha seleccionado como tal. El procedimiento termina cuando el conjunto de prototipos queda vacío.

En el caso de los conjuntos de datos grandes, donde $O(n^2)$ las distancias no pueden ser almacenadas, es todavía posible particionar el conjunto de datos con otra técnica y aplicar AESA sobre cada partición [3c].

Proceso de Búsqueda Utilizando AESA.

Tal como la mayoría de posicionamientos de los algoritmos de indexación AESA resuelve las interrogantes del vecino más cercano escogiendo un pivote $u \in U$ para comparar con la *query*, entonces filtrar a muchos candidatos de U como posible sea, y el proceso se repite hasta que todos los candidatos son comparados y descartados. AESA propone un método específico para seleccionar los pivotes [3a]. El próximo pivote para compararlos contra la *query* es elegido como el candidato u minimizando:

$$D(u) = \sum_{p \in |P|} \left| d(u,p) - d(p,q) \right| \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Donde **P** son los pivotes que son comparados con la *query*, (así las distancias $d(p,q)$ son conocidas, considerando las distancias $d(u,p)$ que son almacenadas en la matriz. La meta es minimizar $D(u)$ y encontrar un pivote lo más cerca posible de la *query*.

El algoritmo para encontrar el cuestionamiento más cercano se resume en 5 pasos:

- **Inicialización:** El conjunto de pivotes **P** y elementos filtrados **F** está vacío. Dados $D(u) \leftarrow 0$ for $u, D\max(u) \leftarrow 0$ y $r \leftarrow \infty$. Los pasos 2-5 son repetidos hasta $U = P \cup F$.

- **Aproximación:** Un nuevo pivote p es seleccionado de acuerdo a la ecuación (1). Eso es $p \leftarrow \arg\min_{u \in U-F-P} D(u)$.

- **Cálculos de distancia:** El elemento p es comparado con el cuestionamiento q calculando $d(p,q)$. El nuevo p será agregado al conjunto usado de pivotes **P**.

- **Actualización de NN:** Si $d(q,p) < r$, el vecino actual más cercano y r es actualizado. Cada objeto en $U - F - P$ es actualizado según el criterio de la aproximación, de acuerdo a la ecuación (1), eso es: $D(u) \leftarrow D(u) + |d(u,p) - d(p,q)|$ y $D\max(u) \leftarrow \max(D\max(u), |d(u,p) - d(p,q)|$.

- **Eliminando:** Estos $u \in U - F - P$ tal que $D\max(u) > r$ son descartados usando la desigualdad del triangulo. Los elementos filtrados en éste paso son agregados a F. El proceso continua al paso 2.

El proceso de la interrogante del vecino más cercano de AESA es presentado en la figura 17. El proceso de pregunta de rango $(q,e)_d$ puede ser implementado guardando r arreglando y reportando cada p esto es: $d(p,q) \le r$.

Estos algoritmos generalizan las interrogantes k-NN donde K>1, manteniendo un grupo con los elementos más cercanos p* encontrados hasta ahora, para que r sea la distancia actual $k-th$ del vecino más cercano.

Selección de pivotes utilizando AESA.

AESA es un algoritmo que pretende aproximar el radio de la consulta (q,r) mediante la eliminación sucesiva de candidatos (*elementos del modelo*). Para esto escoge un elemento $p \in U$ que se denomina *pivote*, calcula la distancia $d = d(p,q)$, y elimina a todos aquellos elementos u estén fuera del rango $[d-r, d+r]$. Luego toma un nuevo *pivote* dentro del conjunto de los que no son eliminados y repite el proceso hasta que obtiene el elemento más cercano a la *query* [27b]. De esta forma AESA puede ser visto como un método basado en pivotes donde $k \to n$, esto se representa en la fig. 22.

La selección del primer pivote es al azar, en cambio la selección del segundo pivote y de los siguientes sigue el criterio de escoger el pivote más promisorio dentro del conjunto de candidatos. La promisoriedad se define según la suma de las cotas inferiores de la distancia entre los elementos y la consulta (conocidas hasta el momento). Esto intenta elegir pivotes cercanos a la *query*. Este criterio se muestra a continuación en la fig. 22a.

$$sea \quad sumLB(u) = \sum_{i=0}^{k-1} \left| d(pi,q) - d(pi,u) \right|$$

Luego el pivote $pk \leftarrow \arg\min_{u \in Candidatos} \{sumLB(u)\}$

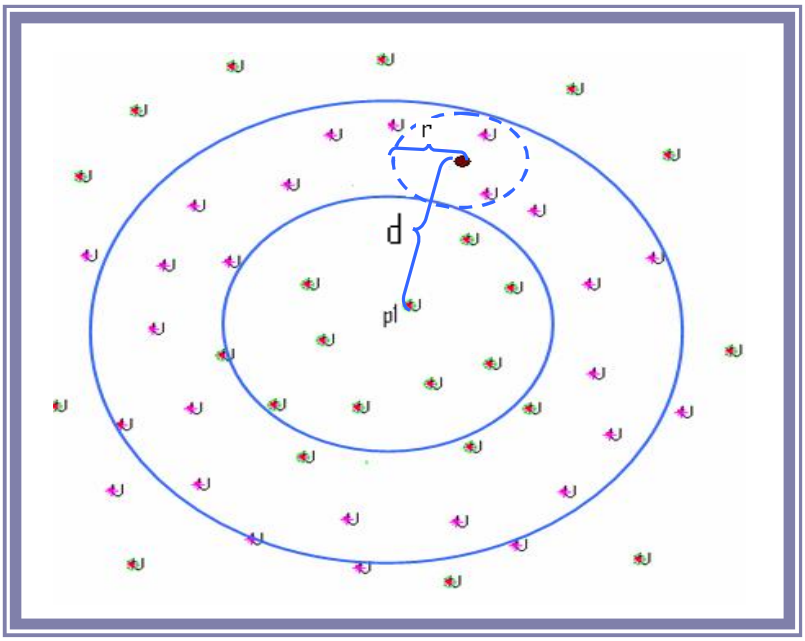


Fig. 22. Se muestra la eliminación de candidatos, para esto se escoge al pivote p1, se calcula la distancia y se establece el cascarón de exclusión de acuerdo al rango de distancias, con esto los elementos que pasan a la segunda iteración son los elementos que están dentro del cascarón.

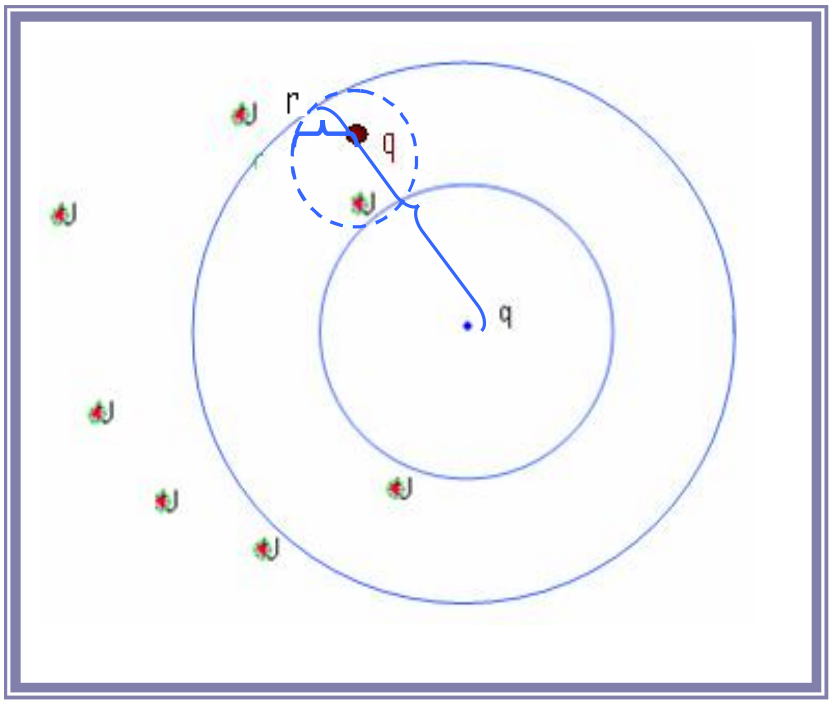


Fig. 22a. Se escoge un nuevo pivote p2, dentro de los objetos que no fueron excluidos y que optimiza la promisoriedad entre los candidatos no eliminados, se calcula la distancia y se establece el cascaron

Búsqueda por Aproximación y Eliminación AESA.

A continuación presentamos el esquema general que utiliza el algoritmo AESA para encontrar al vecino más próximo a la consulta.

La fase de aproximación trata de encontrar un candidato a ser el vecino mas próximo y la elección de este conlleva al cálculo de la distancia de este a la consulta, y como consecuencia la posible actualización del radio y del vecino mas cercano, el proceso de eliminación consiste en descartar aquellos individuos que no se encuentren dentro del radio de observación. Este procedimiento es resumido en 4 pasos:

- Entre los individuos del conjunto de entrenamiento, se selecciona un candidato a vecino más cercano (Aproximación).
- Se calcula la distancia del individuo en cuestión.
- Si la distancia es menor que la distancia del vecino más cercano hasta el momento, d_{nn} se actualiza el vecino más cercano y se eliminan del conjunto de entrenamiento aquellos individuos que no puedan estar adentro de una hiperesfera de radio d y con centro en la muestra, es decir, se eliminan aquellos individuos que no puedan estar más cerca de la muestra del vecino actual.
- Se repiten los pasos anteriores hasta que no queden individuos por seleccionar en el conjunto de entrenamiento, ya sea por que han sido previamente seleccionados o porque han sido eliminados. [19b]

Probabilidad de AESA.

Un problema serio de todos los algoritmos de espacios métricos, es que cuando la dimensión del espacio crece [3b], el conjunto de datos completo necesita ser revisado. En éste caso la probabilidad del algoritmo es una herramienta práctica. Cualquier algoritmo exacto puede convertirse en una probabilidad, permitiendo trabajar hasta el principio de un trabajo predefinido y midiendo cuantas respuestas relevantes encuentra.

Un conjunto de datos bien ordenado es el primero que obtiene más respuestas relevantes cruzando una pequeña fricción en el conjunto de datos.

Así el problema de encontrar una búsqueda de probabilidad, se traduce en encontrar un buen orden del conjunto de datos dado una pregunta. Bajo éste modelo, una versión probabilística de $k-NN$, $AESA$, $IAESA$ y $IAESA2$, consiste en revisar los objetos en algún fragmento del conjunto de datos e informando hasta encontrar el objeto k más cercano. Nosotros debemos sustituir la condición mientras por while

$|P| < percentaje\_of\_conjunto\_de\_datos$. Para el rango de preguntas nosotros simplemente informaríamos algún elemento relevante encontrado hasta que el análisis se detenga.

Pseudocódigo del Algoritmo AESA.

Primero se selecciona de manera aleatoria p_1, y el radio r de la consulta al vecino más cercano, donde r es inicialmente infinito. Con la distancia calculada se actualiza en vecino más cercano (NN) y el radio ($r \leftarrow d(p1,q)$), también se descartan algunos elementos que de acuerdo a la desigualdad triangular no son parte de la respuesta; el resto de los elementos se ordena por el criterio de proximidad, y se selecciona aquel con menor $D(u)$ y el proceso se repite hasta que todos los elementos hayan sido revisados o descartados.

Nótese que a medida de que los elementos se aproximan más a la consulta el radio se reduce, y por lo tanto se pueden filtrar más elementos. De aquí que sea de gran relevancia un buen criterio de proximidad para localizar rápidamente los elementos más cercanos a la *query*. En la figura 23 se presenta el pseudocódigo de AESA.

1.- Sea $P \leftarrow 0$ conjunto de pivotes //*Inicialización*
2.- Sea $F \leftarrow 0$ conjunto de elementos filtrados
3.- $r \leftarrow \infty$
4.- Para $u \in U$, $D(u) \leftarrow 0$, $D\max(u) \leftarrow 0$
5.- Mientras $U \neq P \cup F$ hacer
6.- $p \leftarrow \arg\min_{u \in U-P-F} D(u)$ //*Aproximación*
7.- $P \leftarrow P \cup \{p\}$
8.- $d \leftarrow d(q,p)$ //*Cálculo de distancia*
9.- si $d < r$ entonces
10.- $r \leftarrow d$ //*Actualización del vecino más cercano*
11.- $p^* \leftarrow p$
12.- Fin si
13.- Para $u \in U - P - F$ hacer
14.- $D\max(u) \leftarrow màx(D\max(u), |d - d_{u,p}|)$
15.- Si $D\max(u) > r$ entonces
16.- $F \leftarrow F \cup \{u\}$ //*Eliminación*
17.- Sino
18.- $D(u) \leftarrow D(u) + |d - d_{u,p}|$
19.- Fin si
20.-Fin Para

Fig. 23. Pseudocódigo del Algoritmo AESA

2.5 Políticas de selección de pivotes para índices Métricos.

Es evidente que el conjunto concreto de objetos seleccionados como pivotes influye en la eficiencia del método en la *"búsqueda por similitud"*.

El número de pivotes, su "ubicación" en el espacio métrico y la "ubicación" de cada uno de ellos con respecto a los demás pivotes determinan la capacidad del índice para descartar elementos sin compáralos directamente con la consulta.

La mayoría de los métodos de búsqueda basados en pivotes son de forma completamente aleatoria. Además no se conoce ninguna indicación para determinar el número óptimo de pivotes, un parámetro que depende del espacio métrico en concreto con que estamos trabajando. Estas cuestiones a las que quizá no se les ha dado la importancia que merecen hacen que la calidad de los resultados obtenidos dependa en concreto del espacio métrico y del azar.

En [26b] proponen tres técnicas para la selección de un buen grupo de pivotes. Dichas técnicas tratan de maximizar la media μ_D de la distribución de ***D***, donde:

$$D([p],[q]) = \max_{1 \le i \le k} \{|d(p, p_i) - d(q, p_i)|\}$$

De acuerdo a la referencia mencionada la técnica que muestra un mejor desempeño es la "*selección incremental*". Este método consiste en tomar una muestra de ***N*** elementos del conjunto de datos y seleccionar como primer pivote p_1 a aquel elemento que tenga el máximo valor para μ_D. El segundo pivote p_2 se elige de otra muestra de ***N*** elementos de forma tal que $\{p_1, p_2\}$ tengan el máximo valor para μ_D. Este proceso se repite hasta terminar de elegir los *k* pivotes necesitados [26b].

En dicho trabajo se muestra que un grupo de pivotes tiene dos características básicas:

- Los pivotes están alejados unos de otros, es decir, la distancia media entre pivotes es mayor que la distancia media entre elementos tomados al azar del espacio métrico.
- Los pivotes están alejados del resto de los elementos del espacio métrico.

Los elementos que tienen estas dos propiedades se denominan *outliers*. También se observa que, si bien, los pivotes tienen la propiedad de ser *outliers*, no todos los *outliers* son eficientes como pivotes.

A partir de los resultados experimentales se concluye que los conjuntos de *outliers* tienen un buen desempeño en espacios vectoriales uniformemente distribuidos, pero tienen un bajo performance en espacios métricos generales, peor aún; en una selección aleatoria.

Por otro lado, se presenta un enfoque alternativo que consiste en una selección dinámica del conjunto de pivotes y por consiguiente del índice sobre el que se resolverá la consulta. En lugar de seleccionar durante la construcción del índice un grupo de pivotes

que sea efectivo para todo el espacio métrico: se selecciona durante la búsqueda un grupo de pivotes que sea efectivo para la *query* q. Para ello, se construyen varios índices sobre el espacio con distintos grupos de pivotes (elegidos aleatoriamente); luego, durante una búsqueda $d(q,r)$ se selecciona aquel índice que sea más adecuado a q de acuerdo al conjunto de pivotes con el que fue construido.

A continuación presentamos algunas de las heurísticas existentes para la selección del índice adecuado:

- Selección por votos.
- pivote más cercano.
- pivote más lejano.
- menor masa total.
- pivote de menor masa.
- votación global.

A partir de la evaluación experimental de las mismas se muestra que las de mejor desempeño son las heurísticas pivote de menor masa y votación global. También se analizan las características que presentan los pivotes de los índices más competitivos, siendo la observación más importante que los mejores índices son aquellos cuyo grupo de pivotes tienen una *mayor varianza*.

2.6 Algoritmo Basado en Permutaciones.

Cada elemento del conjunto de datos forma su permutación (*conjunto)* en base a un conjunto de elementos, que en el caso de AESA los llamábamos *"pivotes"* ahora les llamaremos *"permutantes"* y el orden de la permutación es dado por distancia creciente del elemento hacia cada *"permutante"* en este criterio de aproximación simplemente aprovechamos la semejanza entre permutaciones para identificar aquellos elementos con mayor probabilidad de ser cercanos a la consulta ***q***.

La ventaja de esta técnica de permutaciones es que es posible obtener rápidamente un alto porcentaje de respuesta correcta, lo que da como resultado un algoritmo probabilístico.

La idea general de las permutaciones es:

Seleccionamos un conjunto pivotes $P = \{p_1,....p_k\} \subseteq U$, llamados "*permutantes*", en principio de manera aleatoria. Para cada $u \in U$, calculamos su $preorden \leq_u \in P$ y

formamos su permutación $\prod_u$. Los empates se deciden de forma arbitraria pero consistente.

Intuitivamente, si dos elementos u y v son cercanos entre sí, sus permutaciones serán similares. En particular, si dos elementos u, v son tales que $d(u,v)=0$, las permutaciones $\prod_u$ y $\prod_v$ serán iguales.

Basándose en esta observación se quiere revisar primero aquellas permutaciones con mayor semejanza de la consulta, esperando que las permutaciones sean un buen predictor de cercanía entre elementos. Por lo tanto, el objetivo de esta técnica es identificar a los elementos más cercanos entre sí usando la semejanza entre las permutaciones.

Comparación entre Permutaciones.

Al momento de la consulta se calcula $\prod_q$ para la consulta q utilizando $\leq_q$ sobre el mismo conjunto P. Luego para saber en qué orden revisar los elementos, ordenamos la permuta $\prod_u$, $u \in U$ de mayor a menor similaridad con respecto a $\prod_q$. De esta forma, esperamos que los elementos con permutaciones muy similares a $\prod_q$ estén espacialmente cerca de q y sean los más cercanos a la consulta.

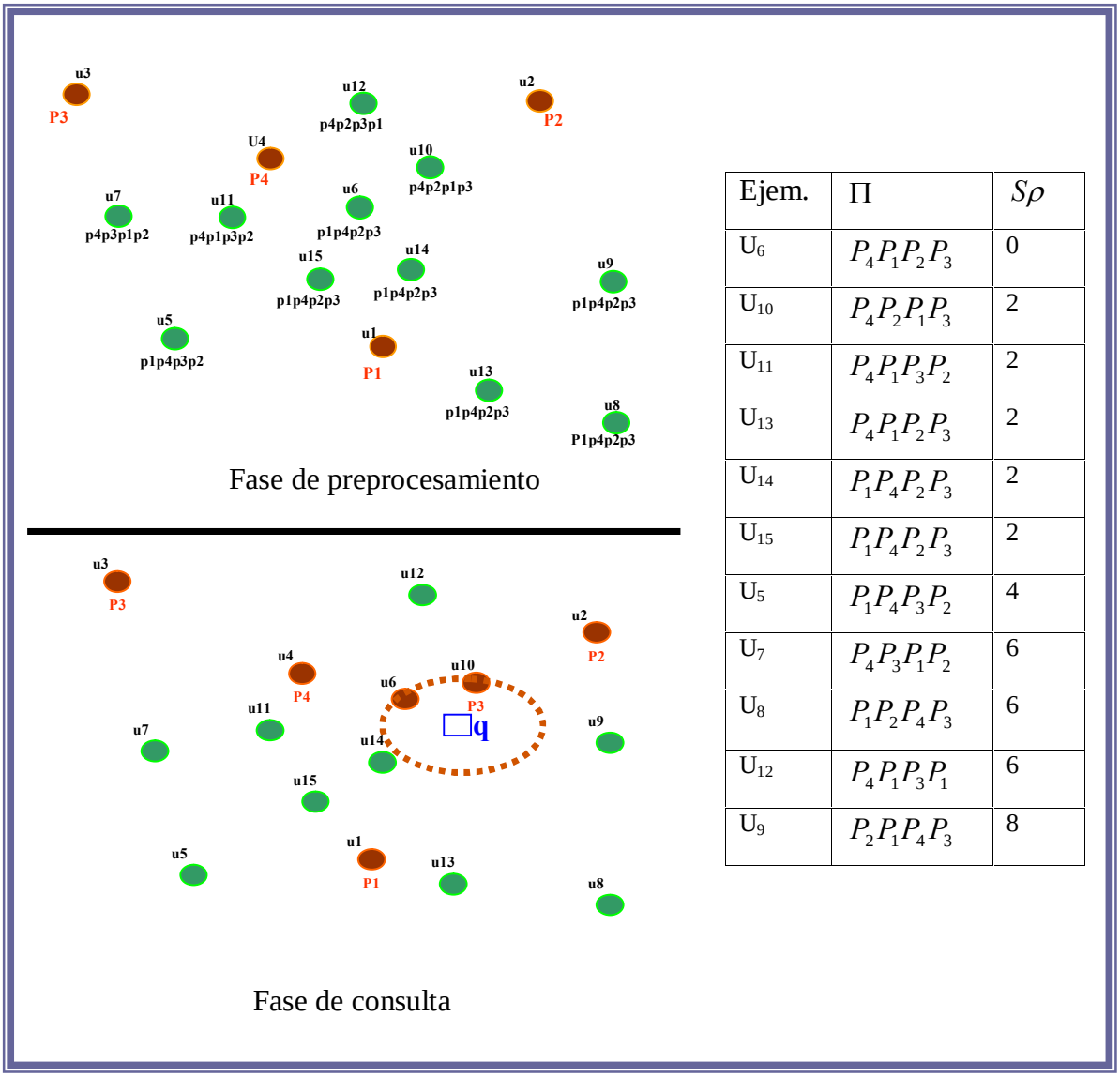
Como medida de similaridad entre permutaciones tenemos a: Spearman Rho, la cual denotaremos por $S\rho(\prod_q, \prod_u)$. Donde $S\rho$ es la suma de los cuadrados de las diferencias en las posiciones relativas de cada elemento en las dos permutaciones. Para cada $pi \in P$ calculamos su posición en $\prod_u$ y $\prod_p$, esto es $\prod_u^{-1}(pi)$ y $\prod_q^{-1}(pi)$, respectivamente. Formalmente [1a].

$$s\rho(\prod_q, \prod_u) = \sum_{i=1}^{k} \left| \prod_u^{-1}(p_i) - \prod_q^{-1}(p_i) \right|^2$$

A continuación daremos un ejemplo de cómo se calcula $S\rho(\coprod_q, \prod_u)$. Sea $\prod_q = p_1, p_2, p_3, p_4, p_5, p_6$ la permutación de la consulta, y sea $\prod_u = p_3, p_6, p_2, p_1, p_5, p_4$ la permutación de un elemento u. El elemento p_3 dentro de la permutación $\prod_u$ está desfasado dos posiciones con respecto a su posición en $\prod_q$. Las diferencias entre las permutaciones para cada elemento son: |1 – 3|, |2 – 6|, |3 – 2|, |4 –1|, |5–5|, |6–4|, y la suma de los cuadrados de todas las diferencias es $S\rho(\prod_q, \prod_u)$ =34.

En la figura 24 se muestra un ejemplo de las dos fases del algoritmo basado en permutaciones; la fase de pre-procesamiento (arriba) y la de consulta (abajo). En la fase

de pre-procesamiento se calculan las permutaciones para todos los elementos del conjunto de datos. En la fase de consulta, primero se calcula la permutación para q. Los elementos son ordenados respecto a la similaridad entre permutaciones (usando $s\rho$) y finalmente comparados secuencialmente en ese orden (primero $u6$, luego $u10$, etc.).



Ejem.	Π	$S\rho$
U_6	$P_4P_1P_2P_3$	0
U_{10}	$P_4P_2P_1P_3$	2
U_{11}	$P_4P_1P_3P_2$	2
U_{13}	$P_4P_1P_2P_3$	2
U_{14}	$P_1P_4P_2P_3$	2
U_{15}	$P_1P_4P_2P_3$	2
U_5	$P_1P_4P_3P_2$	4
U_7	$P_4P_3P_1P_2$	6
U_8	$P_1P_2P_4P_3$	6
U_{12}	$P_4P_1P_3P_1$	6
U_9	$P_2P_1P_4P_3$	8

Fig. 24. Ejemplo de las dos fases de un algoritmo basado en permutaciones. Las permutaciones fueron ordenadas por valor de $S\rho$ respecto a la permutación de la consulta. [1a]

La medida que usaremos es la de Spearman Footrule, que es la distancia L1 entre dos permutaciones. Formalmente:

$$F(\prod q, \prod u) = \sum_{i=1}^{k} \left| \prod_u^{-1}(pi) - \prod_q^{-1}(pi) \right|$$

Selección de Permutantes.

Mostraremos las diferentes heurísticas propuestas en [1a] para seleccionar permutantes ya que hasta ahora han sido seleccionados de manera aleatoria. Estas heurísticas consisten en minimizar o maximizar el valor de Spearman Rho, respecto a los permutantes que han sido seleccionados.

En el caso de seleccionar como permutantes los que minimicen el valor; significa que éstos estarán muy cerca entre ellos y con esto, las permutaciones serán más sensibles a pequeños cambios de posición en el espacio. De lo contrario si seleccionamos aquellos que maximizan el valor, significa de los permutantes serán muy distintos y podría ayudar a que pequeños cambios en una permutación indiquen mayor distancia entre los elementos.

La dimensión de los datos también afecta el desempeño de los permutantes, pues en dimensiones altas se minimiza la diferencia entre usar un permutante u otro. En particular, en éstos experimentos se aprecia que usando pocos permutantes es más visible el efecto de cómo fueron seleccionados, aunque sea pequeña la contribución.

Perspectiva desde el punto de vista de Permutaciones

Dados (q, r) con $radio > 0$, clasificaremos U de acuerdo a cada permutación $\prod_u$ diferenciada de $\prod_q$, entonces nosotros recorreremos el conjunto de datos en ese orden, esperando encontrar rápidamente los elementos más cercanos a la *query.* Con éste fin nosotros utilizamos la métrica de Spearman Footrule como una medida de similitud entre permutaciones, denotada por $F(\prod_q, \prod_u)$. Sumamos las diferencias de las posiciones relativas de cada elemento en la permutación. Esto es, $pi \in P$ calculamos su posición en $\amalg_u$ y $\amalg_q$. Se interpreta $\amalg_u(p_i)$ como la posición del elemento p_i en $\amalg_u$. Entonces resumimos los valores absolutos en diferencias $\amalg_u(p_i) - \amalg_q(p_i)$. Formalmente

$$F(\amalg_q, \amalg_u) = \sum_{1 \le i \le k} \left| \amalg_u(p_i) - \amalg_q(p_i) \right|$$

De esta manera se recorren los elementos $u \in U$ del más pequeño al más grande $F(\amalg_q, \amalg_u)$. Un ejemplo $F(\amalg_q, \amalg_u)$. Dado $\amalg_q = p_1, p_2, p_3, p_4, p_5, p_6$ sea la permutación de la pregunta, y $\amalg_u = p_3, p_6, p_2, p_1, p_5, p_4$ sea la permutación del elemento de $u \in U$. En particular el elemento p_3 en la permutación $\amalg_u$ es encontrado dos posiciones fuera con respecto a su posición en $\amalg_q$. Las diferencias entre las permutaciones son:

$$|3-1|+|6-2|+|3-2|+|4-1|+|5-5|+|6-4|=12$$

2.7 iAESA.

Como se había mencionado en secciones anteriores nosotros llevamos a cabo la implementación de los algoritmos propuestos en [1a] en donde uno de ellos se propone un método diferente para seleccionar el próximo pivote; éste consiste en utilizar las permutaciones, en donde seleccionaremos como siguiente pivote el elemento cuya permutación es la más similar a la permutación de la consulta.

IAESA es una variante del algoritmo AESA. En donde la única diferencia es que el criterio de aproximación.

Proceso de Búsqueda de iAESA.

El nuevo algoritmo de búsqueda basado en AESA consiste en reemplazar el criterio de aproximación, por similaridad entre permutaciones (Métrica de Footrule). Esto es en lugar de que minimice $D(u)$, se usará que minimice la diferencia de las permutaciones F ($\amalg_q, \amalg_u$). Las permutaciones se forman con los pivotes utilizados en cada iteración, por lo tanto estas se actualizan cada vez que se selecciona un pivote.

El algoritmo es prácticamente el mismo que AESA: se inicializa a cero el valor del Footrule de cada elemento; cada vez que se escoge un nuevo pivote p se actualiza la permutación de todos los elementos y de la consulta; se recalcula el valor del Footrule; y el resto del proceso es el mismo.

Se utilizó la métrica de Footrule y no Spearman Rho por que este último tiene ligeramente mayor tiempo de procesamiento [27b].

Pasos del Algoritmo iAESA.

Se inicia seleccionando el pivote p_1, se mide la distancia entre $d(p_1, q)$, al igual que en AESA. La distancia $d(p_1, q)$ es utilizada para reducir el radio al NN y para filtrar los elementos que de acuerdo a la desigualdad del triangulo no pueden ser parte de la respuesta. Con el resto del conjunto de datos se empieza a formar la permutación (*inicialmente*($\{p_i\}$)

Nótese que el segundo permutante puede ser cualquiera pues todo el conjunto de datos tiene la misma permutación. Se selecciona otro elemento en este caso p_2 (paso 1) se mide la distancia con la consulta q y se reduce el radio; y así sucesivamente con el resto del conjunto de datos.

Filtrar los elementos posibles, es decir los que no se encuentren dentro del radio de la *query*. Con dos permutantes ya es posible seleccionar las permutaciones con mayor semejanza a la consulta.

Algoritmo iAESA.

1.- Sea $P \leftarrow 0$ conjunto de pivotes *//Inicialización*
2.- Sea $F \leftarrow 0$ conjunto de elementos filtrados
3.- $r \leftarrow \infty$, $\prod_q \leftarrow <>$
4.- Para $u \in U$, $F(u) \leftarrow 0$, $\prod_u \leftarrow <>$, $D\max(u) \leftarrow 0$
5.- Mientras $U \neq P \cup F$ hacer
6.- $p \leftarrow \arg\min_{u \in U-P-F} F(u)$ *//Aproximación*
7.- $P \leftarrow P \cup \{p\}$
8.-insertar p en $\prod q$
9.- $d \leftarrow d(q, p)$ *//Cálculo de distancia*
10.- si $d < r$ entonces
11.- $r \leftarrow d$ *//Actualización del vecino más cercano*
12.- $p^* \leftarrow p$
13.- Fin si
14.- Para $u \in U - P - F$ hacer
15.- $D\max(u) \leftarrow màx(D\max(u), |d - d_{u,p}|)$
16.- Si $D\max(u) > r$ entonces
16.- $F \leftarrow F \cup \{u\}$ *//Eliminación*
17.- Sino
18.- $D(u) \leftarrow D(u) + |d - d_{u,p}|$
19.- insertar p en $\prod_u$
20.- $F(u) \leftarrow F(\prod_q, \prod_u)$
21.- Fin si
22.-Fin Para
23.-Fin Mientras
24.-Regresa p^* *//Regresa el vecino más cercano*

Fig. 25. Pseudocódigo del Algoritmo iAESA

CAPÍTULO III

III. Implementación y Pruebas.

3.1 Introducción.

Para desarrollar el proceso de reconstrucción en nuestros modelos 3D, empleamos los algoritmos (AESA e IESA) mencionados anteriormente a detalle en el capítulo II y el método (SVD) mencionado en el capítulo I. Las datos de prueba (imágenes 2D y modelos 3D) fueron tomadas de la base de datos [34d].

Una vez que se tiene un par de modelos 3D (datos parciales) se procede a seleccionar los rasgos de correspondencia como pueden ser: bordes, esquinas, puntos, etc.; en nuestro caso utilizamos "*puntos*".

Primero seleccionamos las características en ambos modelos, esto es, seleccionamos un punto del modelo ***P*** y uno del modelo ***Q***, procurando que el punto del modelo ***Q*** sea lo mas parecido posible al del modelo ***P***, esto lo realizamos con la herramienta *"cpselect"* que nos ofrece matlab 7.0. misma que almacena las parejas de puntos seleccionadas en matrices para poder extraerlas y manipularlas con el fin de calcular la estimación de las correspondencias, esta estimación la logramos con la implementación del algoritmo IAESA perteneciente a la familia AESA. Recordemos que este paso es muy importante ya que un mal resultado en la etapa de correspondencias hará que se obtengan resultados no válidos en las siguientes etapas. Una vez establecidas las correspondencias entre los modelos, es decir, qué característica del primer modelo corresponde a la del segundo; se procede con la última etapa del registrado, donde estimamos el método SVD para encontrar los parámetros de transformación que permiten alinear los modelos y con esto lograr el objetivo de generar un modelo único 3D.

3.2 Implementación de los Programas.

Esta tesis fue desarrollada en una PC de escritorio que cuenta con: el Sistema operativo Microsoft Windows XP, Procesador Intel Pentium IV a una velocidad de 1.80 Ghz., Memoria Ram de 512 y disco duro de 60 GB., haciendo uso del la plataforma Matlab versión 7.0.

Mediante estas herramientas se desarrolló un conjunto de programas los cuales realizan una tarea específica como se puede ver en la figura 26 donde representamos el diagrama de flujo de los pasos desarrollados para generar un modelo único 3D, a partir de datos parciales (modelo ***P*** y modelo ***Q***).

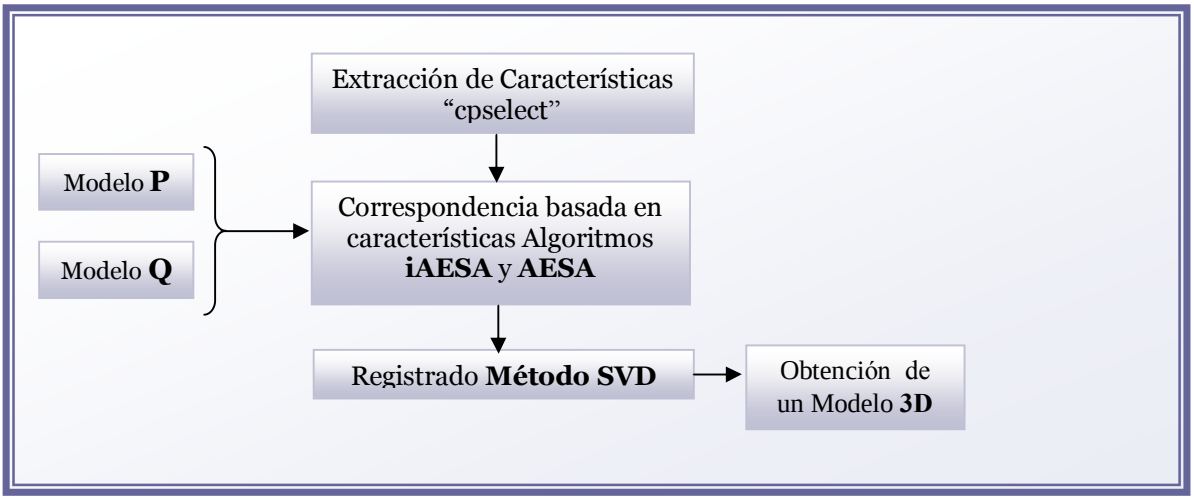


Fig. 26. Algoritmo para la alineación de los modelos Tridimensionales

Para efectuar el registrado de modelos 3D, se lleva a cabo la implementación del algoritmo SVD que se explicó a detalle en el capítulo I, éste algoritmo es uno de los más utilizados para desarrollar el proceso de registrado de modelos o superficies 3D, una desventaja es que requiere un conjunto de puntos los cuales, deben de estar semi-alineados (estimación inicial) y mediante un proceso iterativo es posible mejorar la alineación de estos, el algoritmo proporciona buenos resultados siempre y cuando los datos de entrada sean correctos y exista una buena aproximación inicial, ya que en dado caso que la correspondencia sea inadecuada el algoritmo no logrará encontrar los parámetros correctos para conseguir una transformación rígida que deje a ambos modelos alineados bajo un mismo sistema de coordenadas. El ICP no realiza estimaciones grandes, sino más bien finas

La extracción de características, los programas para establecer la correspondencia, el método SVD, el proceso de triangularización, la texturización de los modelos fueron desarrollados en matlab ver.7.0.

Pasos de la Implementación:

1.- Selección de características: Para realizar el módulo de extracción de características (*puntos*) del par de modelos ***P*** y ***Q*** se utilizó la herramienta *"cpselect"* que facilita la plataforma matlab 7.0. Misma que nos permite seleccionar pares de puntos correspondientes a nuestros modelos, es decir, seleccionar un rasgo del modelo **P** y uno del modelo **Q** sucesivamente.

2.- Extracción de características: Para realizar el módulo de la extracción de características, la herramienta "*cpselect*" proporciona estructuras donde al terminar de seleccionar las parejas de puntos quedan almacenados.

3.- Establecimiento de correspondencias: Para obtener el modulo de la estimación de correspondencias entre las características, se lleva a cabo la implementación de los algoritmos:

AESA: (Algoritmo de Búsqueda por aproximación y Eliminación) e *IAESA:* (Algoritmo basado en Permutaciones).
Ambos algoritmos utilizan una métrica de distancia y el método del vecino más cercano. La diferencia entre ellos es el enfoque de solución para determinar qué punto del modelo ***P*** corresponde con que punto del modelo ***Q***, es decir; el criterio para elegir el próximo pivote cambia en el algoritmo iAESA ya que este se basa en el concepto de permutaciones, es decir, cada elemento tiene un conjunto de puntos, correspondiente en el cual se encuentran todos los puntos que son cercanos a el, en orden ascendente conforme a la distancia con el elemento. Con esta técnica es posible obtener rápidamente un alto porcentaje de respuesta. Ya que es un algoritmo probabilístico.

4.-Alineación: Para establecer la alineación de los modelos, se utiliza el método **SVD** (Descomposición de Valores Singulares) que determina los parámetros de rotación, traslación y cambio escalar mediante el método propuesto por horn [29d], para alinear los modelos generando un modelo tridimensional único.

3.3 Algoritmo AESA.

A continuación en la figura 27 presentamos los modelos 3D que utilizaremos para desarrollar el proceso de registrado.

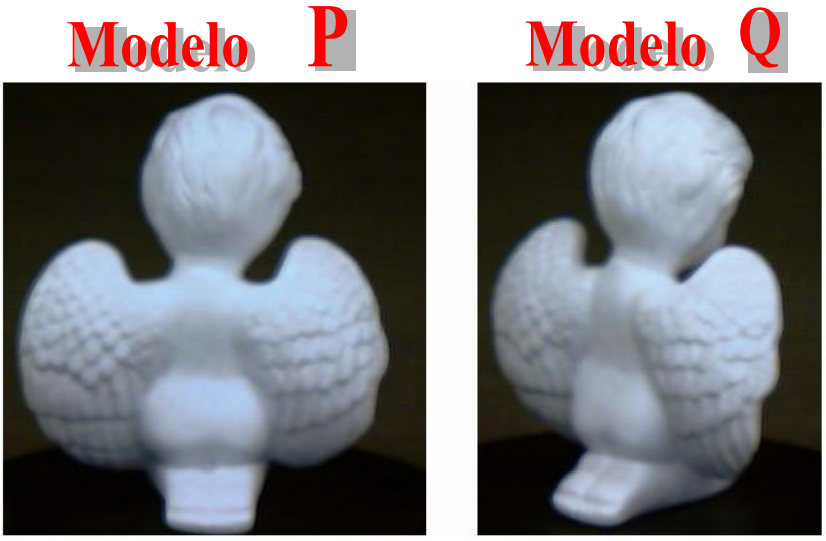


Fig. 27. Representación de los modelos **P** y **Q** 3D (datos parciales).

1.- *Selección de características:* En esta etapa utilizamos la herramienta *"cpselect"* que nos proporciona la plataforma de matlab 7.0, misma que permite visualizar ambos modelos al mismo tiempo para poder seleccionar parejas de puntos, tomados del modelo **P** y del modelo **Q,** siendo estos puntos lo más parecidos posibles. Esto se representa en la figura 28.

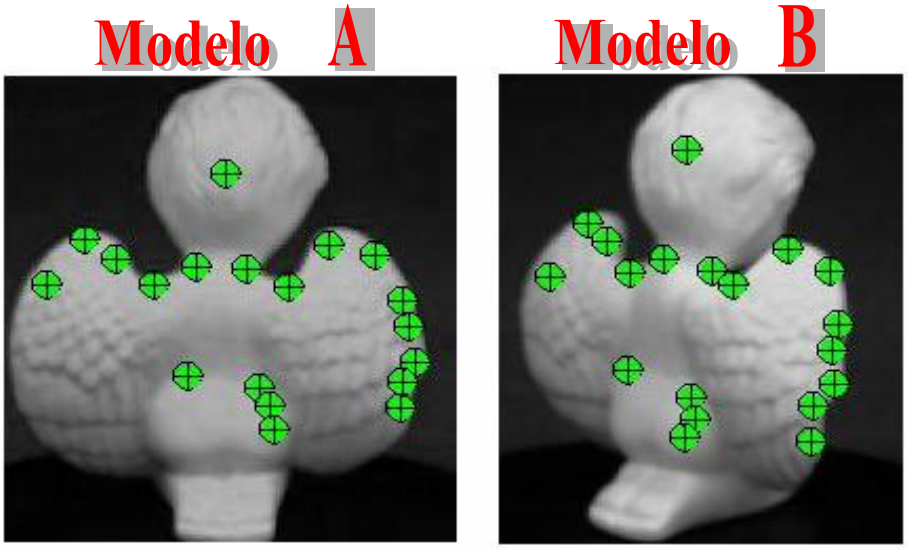


Fig. 28. Selección de rasgos o características del modelo **P** parecidos o semejantes al los puntos del modelo **Q**

1.- *Extracción de características:* En esta etapa la herramienta *"cpselect"* también nos facilita estructuras "matrices o vectores" para almacenar los puntos seleccionados. Y de esta manera poder manipularlas con facilidad en la siguiente fase, donde se realizan los cálculos de correspondencia.

2.- *Establecimiento de correspondencias:* Una vez extraídos los puntos de cada modelo se procede con la implementación del algortimo AESA / IAESA ambos algoritmos determinan las correspondencias de los modelos ***P*** y ***Q***.

Primero presentaremos el desarrollo del algoritmo AESA, y mas adelante el del algoritmo IAESA.

A continuación presentamos las características extraídas (*puntos*) que utilizaremos para desarrollar el proceso de correspondencia de los modelos. Recordemos que AESA es un algoritmo basado en pivotes y que además trabaja por búsqueda de Aproximación y Eliminación.

Enseguida presentamos las variables que utilizamos para implementar esta aplicación:

U: *Conjunto de puntos del modelo "P" (13 elementos),* ***q:*** "*query*" elemento perteneciente al modelo "Q", ***F****=[ ] Conjunto de elementos eliminados,* ***Pi***=[ ] *Conjunto de pivotes o puntos de control,* ***r***=∞ *radio,* ***p**** *es el elemento más cercano a la query ,* ***subU*** *conjunto de elementos que no están en* ***F*** *ni en* ***Pi****,* ***U****!=PUF Condición de paro,* que me dice: si el conjunto **U** es igual a la unión del conjunto de pivotes ***Pi*** y el conjunto de elementos eliminados ***F***. El algoritmo termina cuando ya no hay más puntos del modelo por analizar.

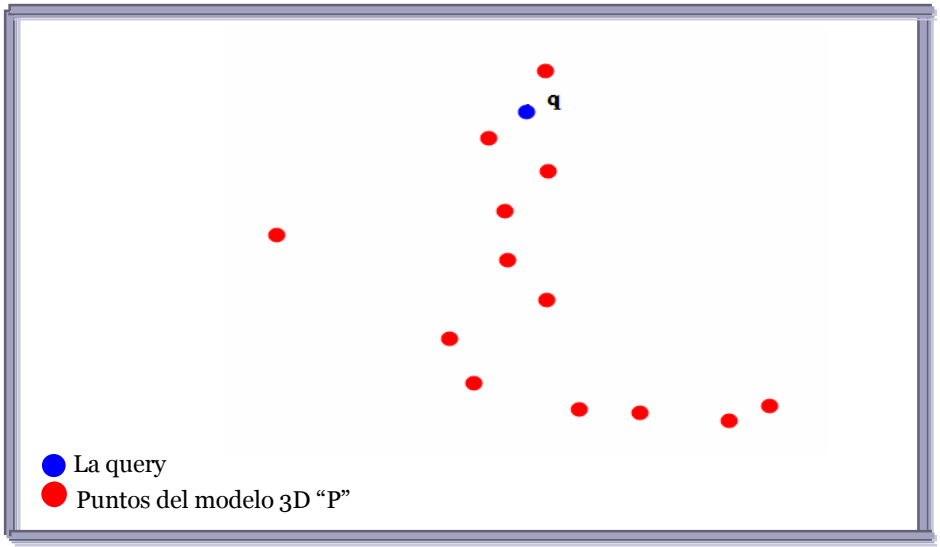


Fig. 29. Representación de los puntos del modelo "P" y la consulta. Para AESA

En la figura 29 se presentan los puntos del modelo ***P*** en color rojo y el punto "*query*" del modelo ***Q*** en color azul, se requiere establecer correspondencia, entre el punto del modelo ***Q*** "*query*", con el punto mas cercano a él del modelo ***P***, que corresponda correctamente.

AESA antes de empezar cada iteración pregunta si la condición de paro ya se cumplió. Si esta condición se cumple el algoritmo finaliza, de lo contrario AESA sigue seleccionando elementos para compararlos con la consulta.

Primera Iteración del Algoritmo AESA

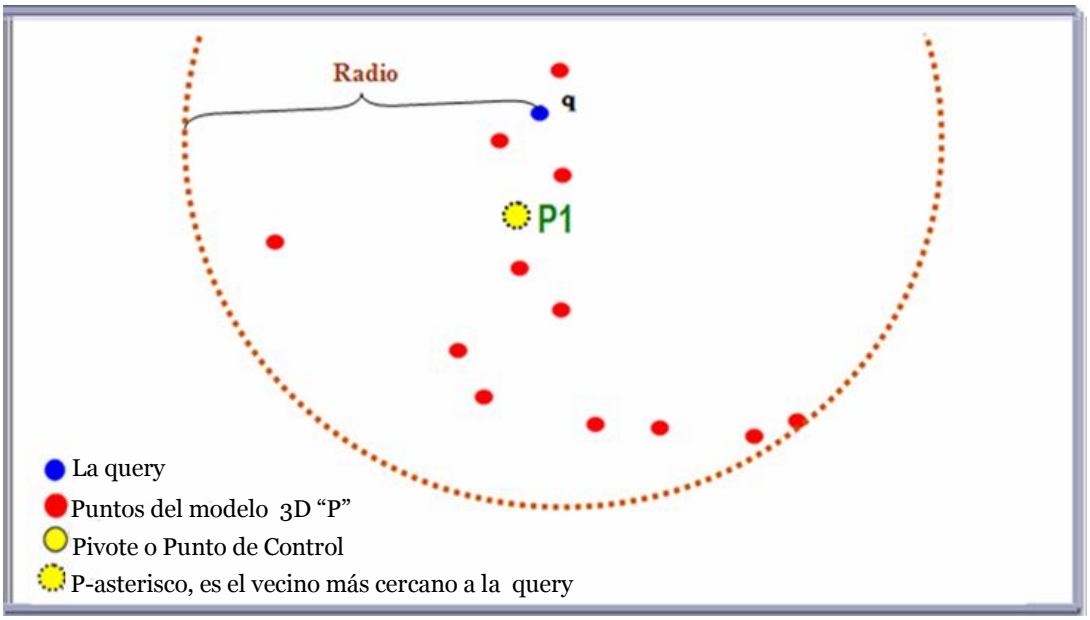


Fig. 30. Primera Iteración de IAESA donde el r = infinito.

En la figura 30 se observa que el algoritmo AESA elige el primer *pivote* p_1 de color amarillo (mapa punteado), inicialmente el radio toma el valor de infinito, esto quiere decir, que todos los puntos del modelo ***P*** son considerados para calcular la distancia de cada punto con la "*query*" ya que todos los elementos se encuentran dentro del radio de la "*query*", mismo que se irá actualizando en cada iteración.

Otra condición que valida AESA es que los pivotes seleccionados deben de estar dentro del radio de la consulta, de lo contrario ese elemento deberá ser descartado.

A continuación en la figura 31 se muestra gráficamente como se van numerando los puntos del modelo ***P*** en el orden en que fueron seleccionados, para calcular su distancia con el "*query*", es decir; todos los elementos $u \in U$. Las distancias se van calculando y almacenando en un vector llamado $D(u)$.En la primera iteración AESA calcula la distancia de p_1 con la interrogante "*query*", este valor es comparado con el valor inicial del radio y como este valor es menor, el radio actualiza su valor tomando el valor de la distancia que hay entre p_1 y la "*query*". De esta manera se actualiza el radio y por lo tanto aquellos elementos que se encuentren fuera del nuevo radio serán descartados. AESA le asigna el pivote p_1 a $p*$ que es quien tiene el elemento más cercano a el "*query*".

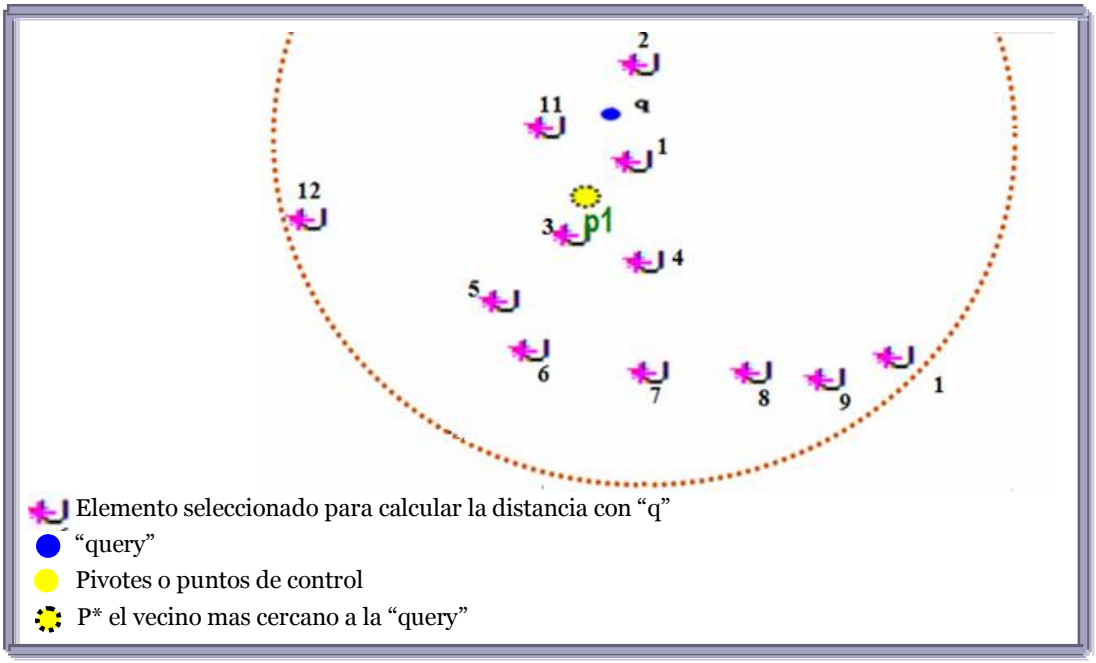


Fig. 31. En esta grafica se aprecia como se van seleccionando los elementos del modelo que se encuentran dentro de la hiperesfera que forma el radio de la query.

Segunda Iteración del Algoritmo AESA

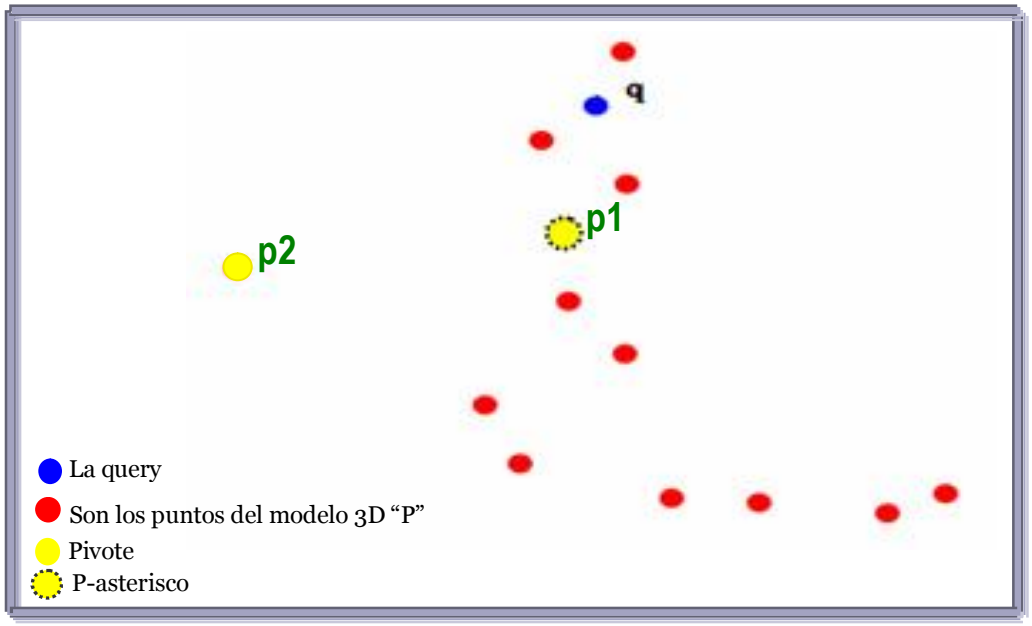


Fig. 32. Se observa que el algoritmo AESA elige un segundo pivote

En la figura 32 representamos que en la segunda iteración, AESA toma un segundo pivote p_2, calcula la distancia de p_2 con la "*query*", compara este valor con el valor del radio r y como este valor es mayor que el valor del radio, no se actualiza ni el radio ni $p*$. Durante las iteraciones se va actualizando la matriz de distancias $D(u)$, esto se realiza para poder comparar elementos mas adelante e ir descartándolos y almacenarlos en el conjunto F; por ahora el conjunto F sigue vació; AESA sigue tomando elementos del conjunto $\boldsymbol{U}$ para calcular su distancia, de igual manera se enumeran los elementos en el orden en que los tomó AESA. En la figura 33 se representa gráficamente el orden en que fueron seleccionados los elementos para calcular su distancia con la consulta.

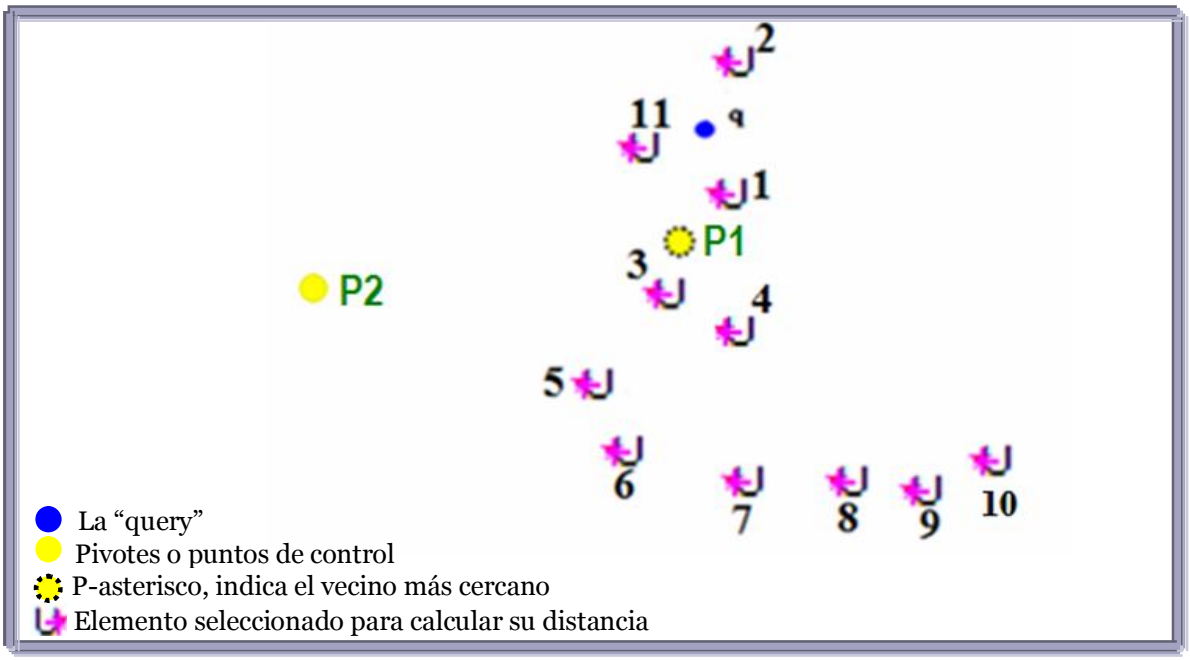


Fig. 33. Se observa el orden en que el algoritmo AESA va eligiendo cada elemento

Tercera Iteración del Algoritmo AESA

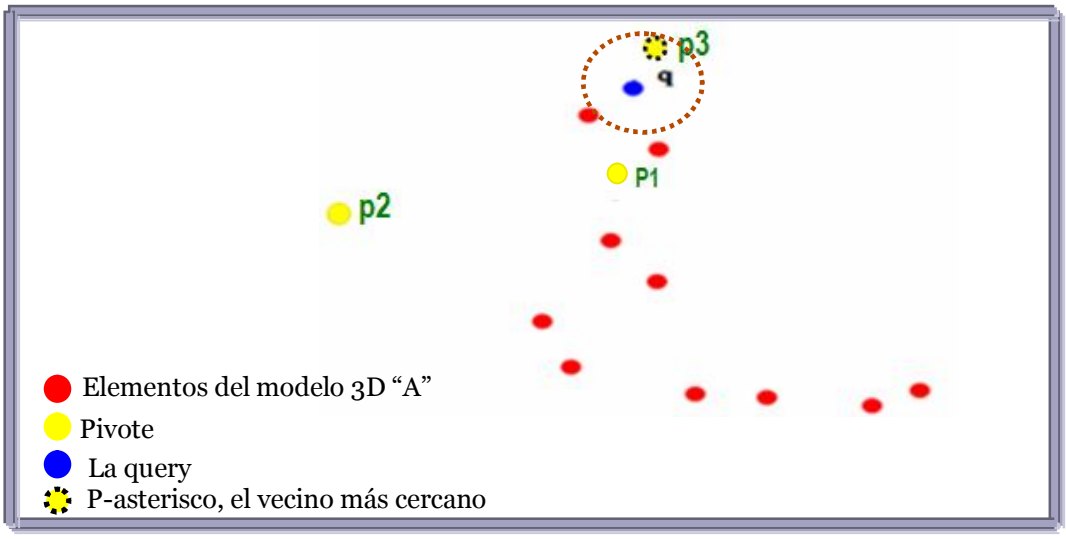


Fig. 34. AESA elige un tercer pivote y el radio es reducido

En la figura 34 se muestra que en la tercera iteración, AESA elige un tercer pivote p_3, calcula su distancia con la *query* y compara la distancia calculada con el valor del radio y como este valor es menor que el valor del radio, éste se vuelve a actualizar junto con $p*$, ahora p_3 es el vecino mas cercano a la *query*. A continuación AESA va seleccionando los elementos del modelo ***P*** para seguir calculando distancias y si estos puntos no se encuentran dentro del nuevo radio, AESA los va descartando y almacenando en el conjunto F. A continuación en la figura 35 se van numerando en el orden en que fueron seleccionados y eliminados. En este ejemplo se eliminaron 10 elementos, mismos que fueron almacenados en el conjunto F. Finalmente se cumple la condición de paro donde $U = P \cup F$, quedando como vecino más cercano el valor de $p* = p_3$. La implementación de este algoritmo se encuentra en el apéndice A.

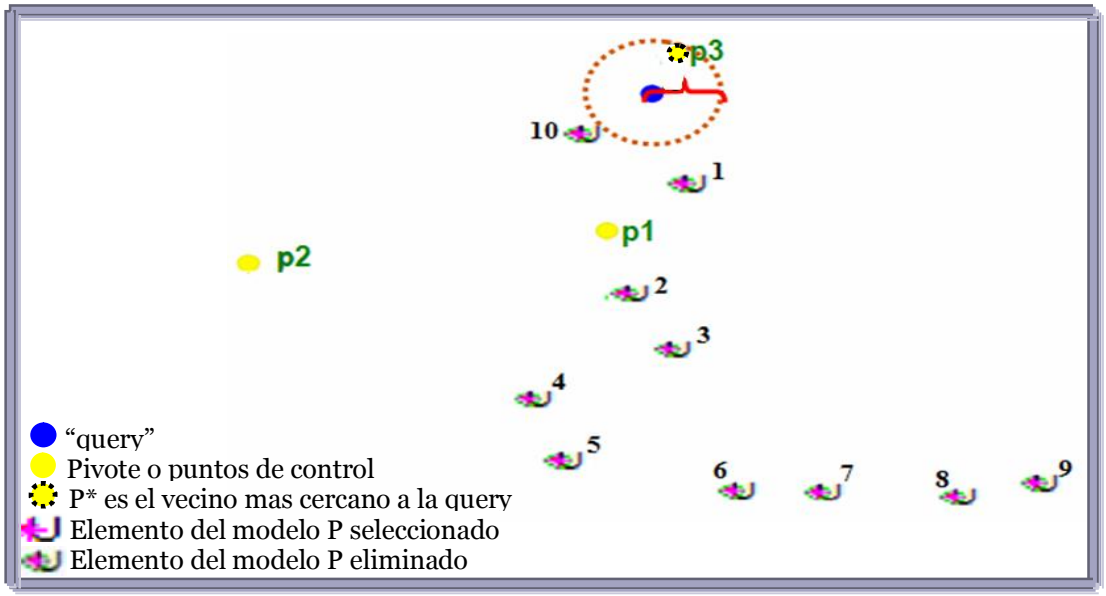


Fig. 35. Elementos eliminados por el algoritmo AESA

3.4 Algoritmo iAESA.

Para realizar una comparación entre el algoritmo AESA y el iAESA se han tomado los mismos modelos. Recordemos que este algoritmo esta basado en permutaciones, lo cual implica que cada elemento del modelo ***P***, forma su permuta (*conjunto de pivotes*) en base a los "*permutantes*". El orden de cada permuta esta dada de manera creciente referente a la distancia calculada entre cada pivote y el punto o elemento de ***U***, este es el criterio que maneja iAESA; única diferencia que existe con el algoritmo AESA para elegir el próximo pivote.

A continuación presentamos las características de mis modelos (conjunto *de puntos*) para desarrollar el proceso de correspondencia. Las variables que utilizamos realmente son similares a las utilizadas en el AESA, solo que ahora el conjunto de pivotes es llamado ***Permutantes*** y en lugar de utilizar $D(u)$ que minimiza la distancia, utilizaremos $F(u)$ que minimiza la diferencia de las posiciones, es decir, que minimiza la diferencia de los valores absolutos de la permuta Q ($\prod_q$) y la permuta de U ($\prod_u$), mismas que serán formadas con los pivotes seleccionados en cada iteración, por lo tanto estas se irán actualizando. La implementación de este algoritmo se puede ver en el apéndice B

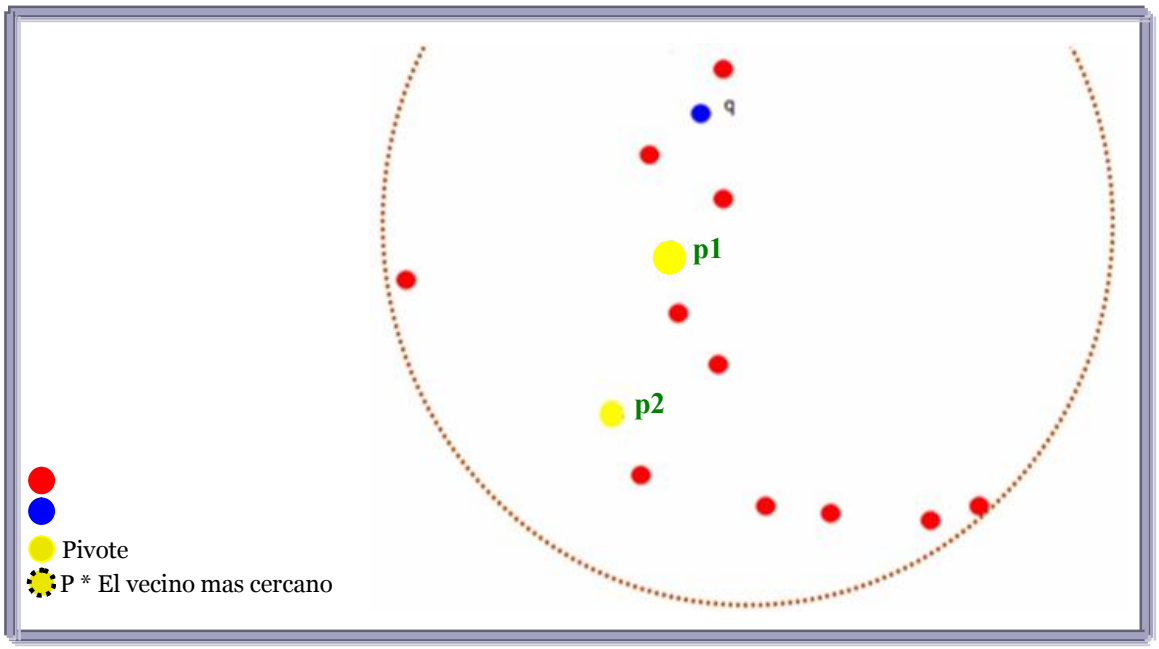


Fig. 36. Inicialmente el algoritmo IAESA toma 2 pivotes p1 y p2 (amarillos)

En la figura 36 presentamos los puntos del modelo ***P*** en color rojo y del *query* perteneciente al modelo ***Q*** en color azul. Se requiere establecer una correspondencia adecuada entre la "*query*" y el punto mas cercano a ella.

En la figura 37 mostramos que IAESA inicialmente tiene dos pivotes elegidos aleatoriamente, lo cual implica que el conjunto de "*Permutantes*" contiene dos pivotes. El radio toma el valor de infinito, lo cual quiere decir que todos los puntos del modelo ***P*** serán tomados en cuenta para ser seleccionados y comparados con la *query* . Posteriormente el radio se irá actualizando con las iteraciones.

IAESA antes de empezar a seleccionar elementos y calcular distancias con la "*query*" estima la condición de paro, misma que consiste en comparar si son iguales los elementos del conjunto ***U*** con los elementos del conjunto formado por la unión de ***Pi*** "permutantes" y ***F*** "elementos eliminados." Si se cumple la condición el algoritmo finaliza de lo contrario entra al ciclo para seguir seleccionando elementos y calcular su distancia con la consulta.

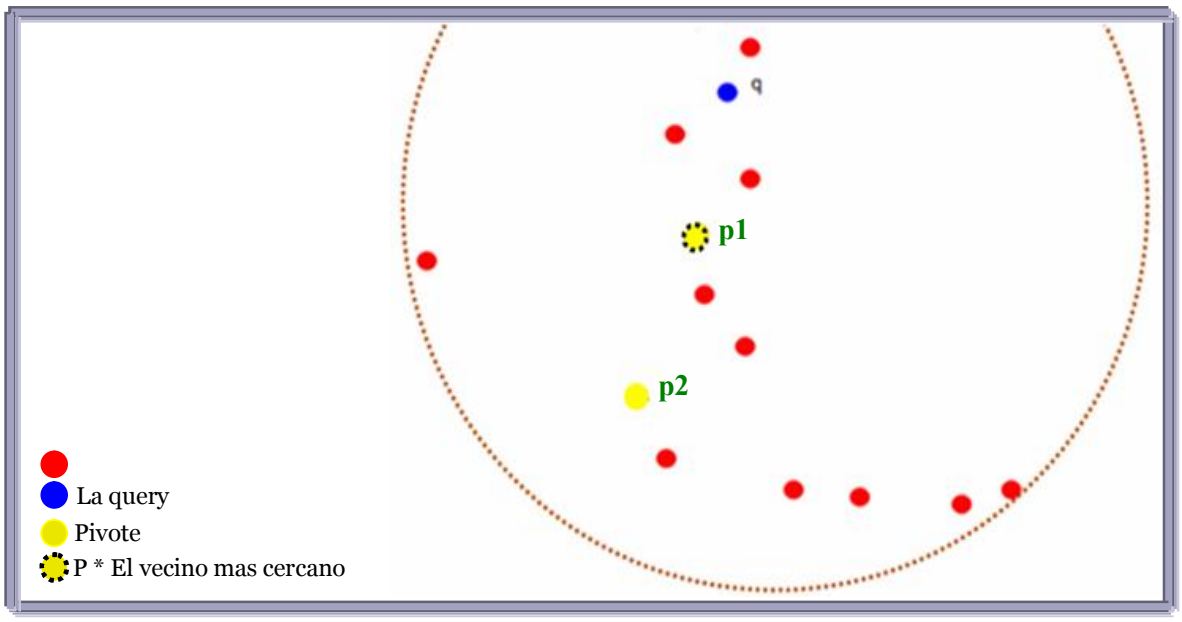


Fig. 37. En la 1era. Iteración IAESA toma 2 pivotes para formar las permutas tanto de los elementos como de la consulta

Enseguida se calculan las distancias de cada uno de los permutantes $\{p_1, p_2\}$ con la *query* e inmediatamente las ordena de mayor a menor similaridad con respecto a la consulta para formar su permuta $\amalg_q$, se toma el pivote que haya obtenido la distancia más pequeña, esto es, el mas cercano a la consulta, en este caso es p_1 ; a continuación

este valor es comparado con el valor del radio y como este valor calculado es menor, el radio es actualizado y a p* le es asignado p_1, de tal manera que por el momento p_1 es el vecino mas cercano a la consulta.

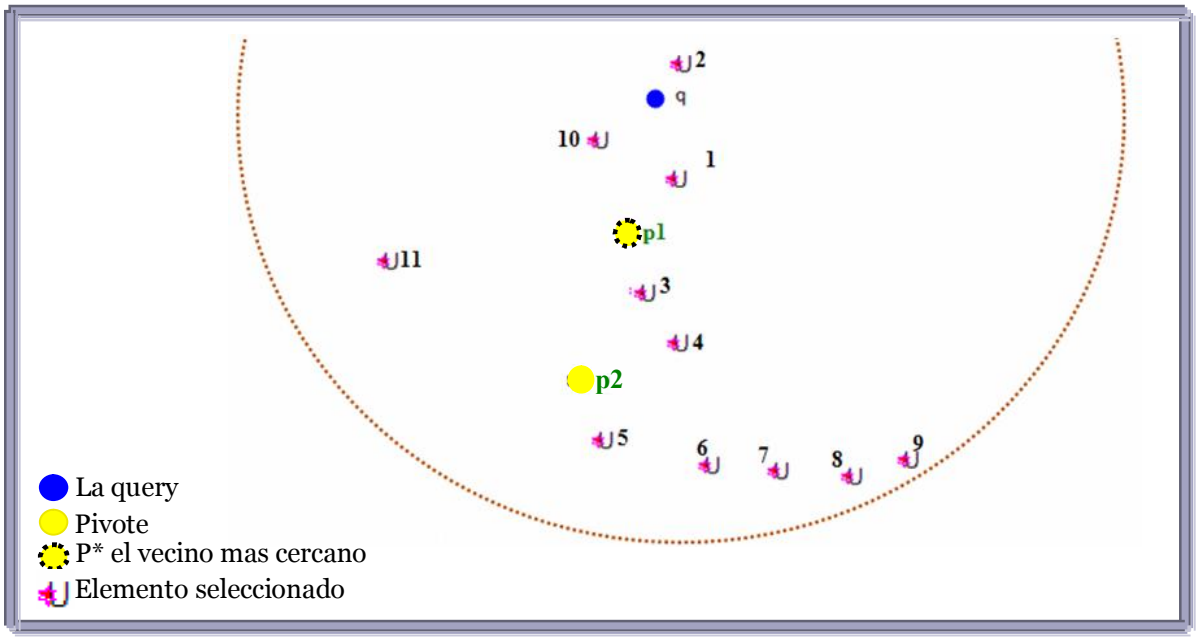


Fig. 38. Mostramos el orden en que fueron tomados los elementos del modelo para formar sus permutas durante la iteración del algoritmo iAESA

En la figura 38 presentamos el orden en que fueron seleccionados los elementos para calcular su distancia con los "*permutantes*". Y así ir formando las permutas tanto de la "*query*" como de los elementos.

IAESA también valida otra condición que consiste en comprobar que el elemento seleccionado se encuentre dentro del radio de la consulta. Si la condición se cumple calcula la distancia de ese elemento con cada uno de los permutantes e inmediatamente se inserta en las permutas de acuerdo a la posición que le corresponda, de lo contrario IAESA lo descarta, es decir, ese elemento queda almacenado en el conjunto de elementos filtrados F.

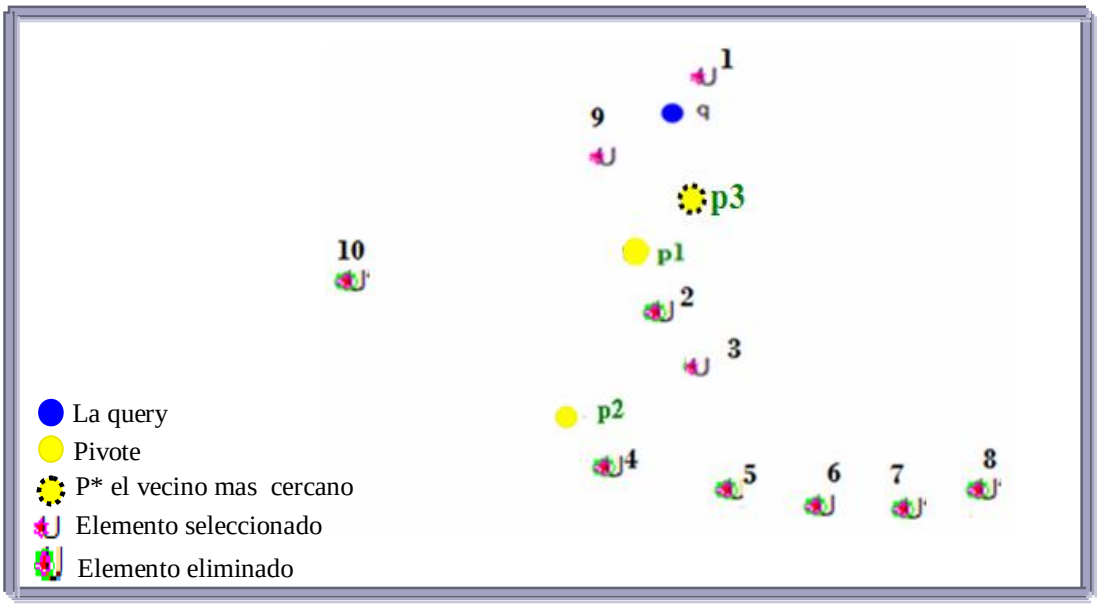


Fig. 39. IAESA elige un tercer pivote y actualiza el radio y la p*

En la figura 39 se aprecia que en la primera iteración iAESA elige un tercer pivote, actualiza la permuta de la *query* Π_q de tal manera que los elementos de la permuta queden ordenados de mayor a menor similaridad con respecto a la consulta. IAESA calcula la distancia del nuevo permutante p_3 con la *query* en este caso se observa claramente como el valor de esa distancia calculada es menor que el valor del el radio r, debido a esto, el radio actualiza su valor junto con $p*$ de tal manera que en esta ocasión p_3 es el vecino mas cercano.

Todos los elementos que están de color verde representados en la figura 34, nos indican que fueron eliminados debido a que no se situaron dentro del nuevo radio y por lo tanto son almacenados en el conjunto F.

Debido a un problema de perspectiva no se aprecia que el elemento 3 esta situado dentro del radio. Para comprobar esto les presentamos en la figura 40 una toma de estos mismos puntos rotados; en donde claramente se visualiza que efectivamente ese punto si se encuentra dentro del radio de la *query*, solo recordemos que estamos en un espacio tridimensional.

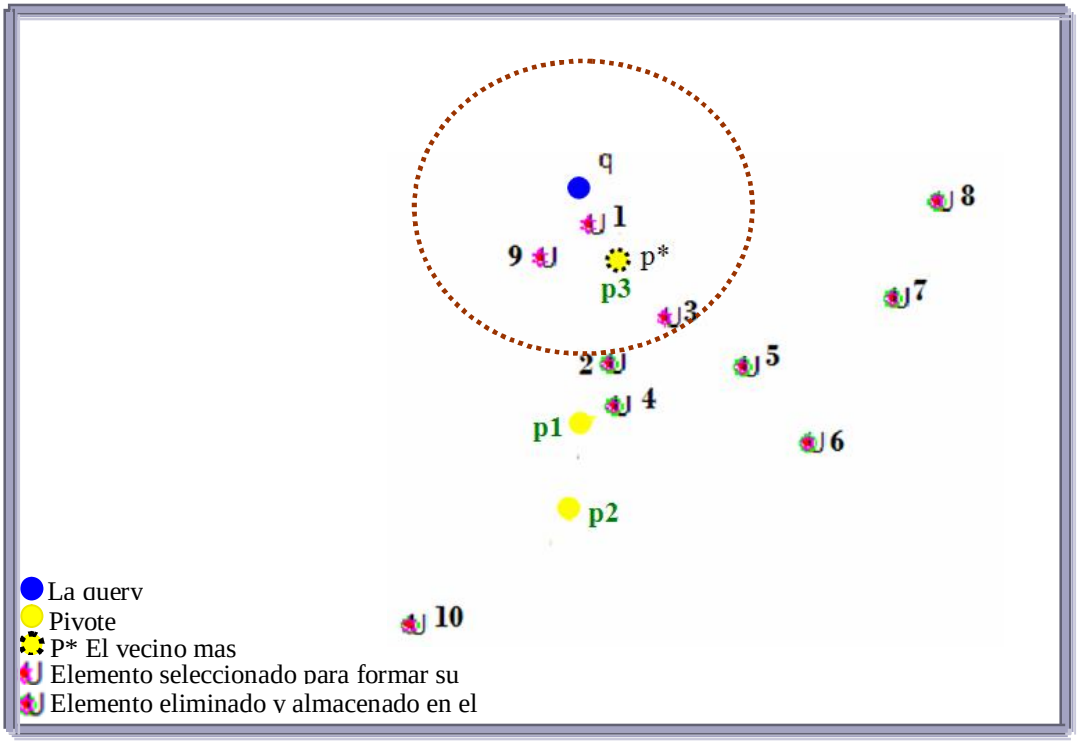


Fig. 40. Mostramos una diferente Perspectiva de los puntos del modelo 3D, para poder visualizar que efectivamente los puntos u1,u9 y u3 se encuentran dentro del radio de la Q

Segunda Iteración del Algoritmo iAESA

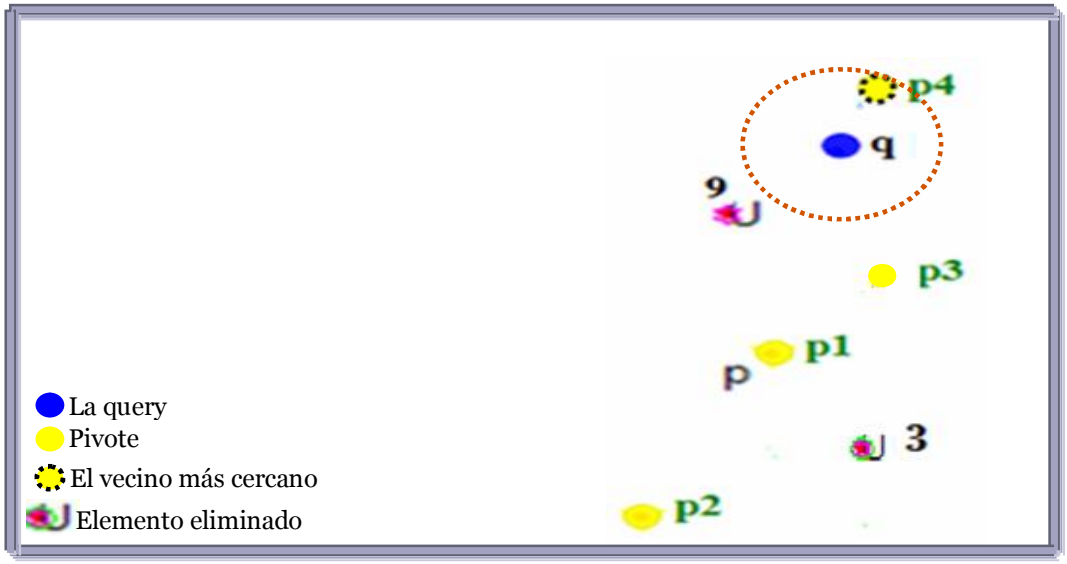


Fig. 41 IAESA elige un cuarto pivote, se vuelve a actualizar el radio y la p*

Como podemos ver en la figura 41, en la segunda iteración IAESA toma un cuarto pivote, calcula la distancia del nuevo permutante p_4 con la *query*, este permutante se inserta en la permuta de la *query* respecto a la posición que le corresponde. Debido el que valor de la distancia calculada fue menor que el valor del radio, este una vez más se actualiza junto con el $p*$, ahora el vecino mas cercano es el pivote p_4, IAESA empieza a seleccionar a los elementos restantes del modelo ***P***, para actualizar sus permutas con el nuevo permutante y compararlas con la permuta de la *query*. El elemento U3 es eliminado, ya que este elemento esta fuera del radio.

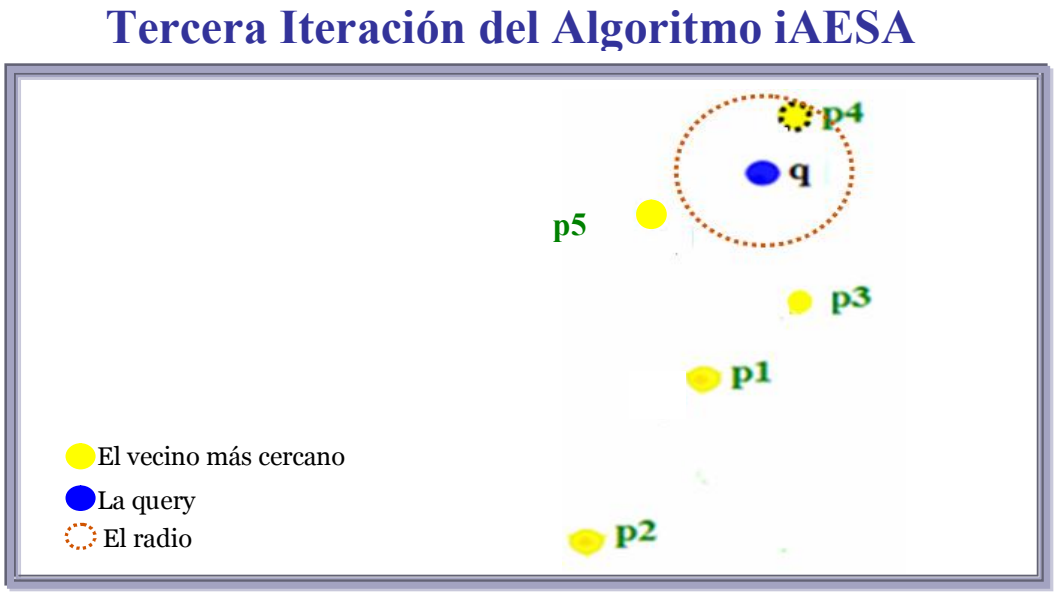


Fig. 42 IAESA elige un quinto pivote, en donde la distancia es mayor. Por lo tanto el radio no se actualiza

En la figura 42 representamos la tercera iteración del algoritmo IAESA donde es elegido un quinto pivote el cual es seleccionado para calcular su distancia con la *query*; la distancia calculada es mayor que el radio por lo tanto el radio no se actualiza, de tal manera que p_5 es un pivote que no pertenece a la hiperesfera de la *query*. En esta iteración ya no hay más elementos que seleccionar. Y como en todas las iteraciones se pregunta la condición de paro, es decir, si los elementos de U son iguales a los elementos de ***Pi*** "permutantes" unión ***F*** "elementos eliminados" el algoritmo finaliza. En este ejemplo tenemos que se cumple la condición de paro, por lo tanto IAESA finaliza, quedando como vecino mas cercano el pivote p_4, mismo que obtuvo la distancia mas pequeña de todos los elementos hacia la *query*, en la iteración numero 2.

3.2 Método SVD.

Una vez que ya tenemos el conjunto de correspondencias establecidas pertenecientes a nuestros modelos ***P*** y ***Q***, pasamos a la siguiente etapa de registrado. En la que se desarrolla el proceso de alineación de nuestros modelos 3D. Para realizar esta tarea se utiliza el método de Descomposición de Valores Singulares (SVD), ya que este método permite obtener los parámetros de transformación (Rotación, Traslación y Escala) necesarios para llevar a cabo la alineación de nuestros modelos. En la figura 43 se muestran nuestros modelos ***P*** y ***Q*** (*datos parciales*) utilizados para desarrollar esta etapa.

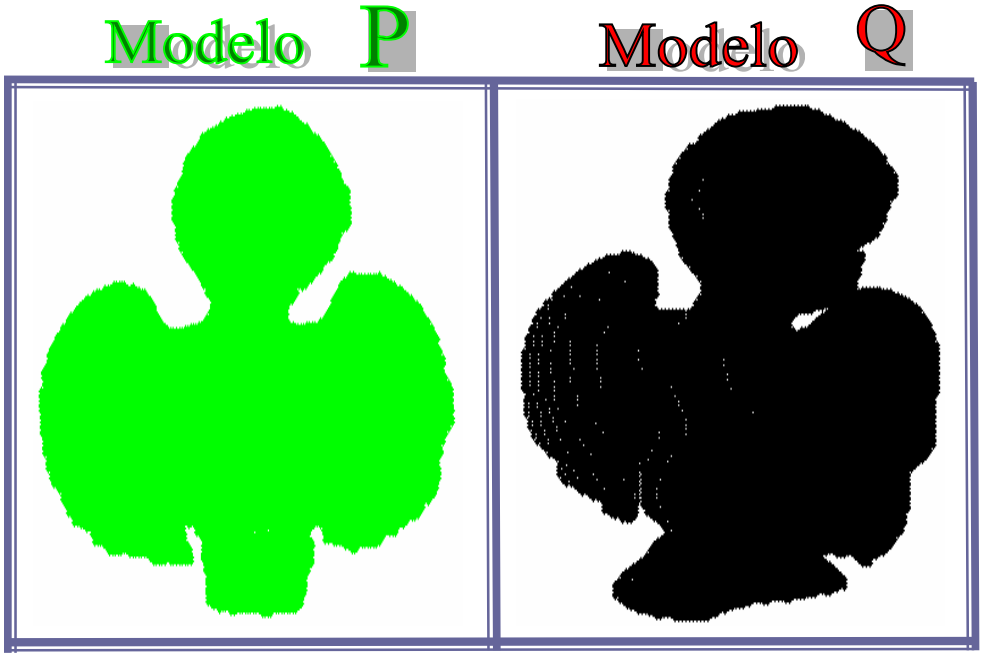


Fig. 43 a) Modelo "P" a este modelo se le van a aplicar las transformaciones (rotación, traslación y escala) para alinearlo con el modelo "Q"

B) Modelo "Q" este es el modelo que se toma de referencia, es decir no se le aplica ninguna transformación (rotación, traslación y escala)

El centroide de nuestro conjunto de puntos del modelo ***P*** $mx3$ es diferente al del modelo ***Q*** $mx3$, esto implica que nuestros modelos se encuentran situados en diferente origen. Para desarrollar el proceso de alineación el primer paso es lograr que ambos modelos se encuentren en el mismo centro. Esto se logra calculando la media del modelo ***P*** y del modelo ***Q***; dicho calculo nos da un vector de $3x1$ para cada modelo correspondiente. El siguiente paso es calcular la transpuesta de cada conjunto de puntos, para poder realizar la diferencia del vector media con cada vértice del conjunto de puntos correspondiente a los modelos. Y con éstos cálculos hemos logrado mover a nuestros modelos a un mismo origen, como se muestra en la figura 44.

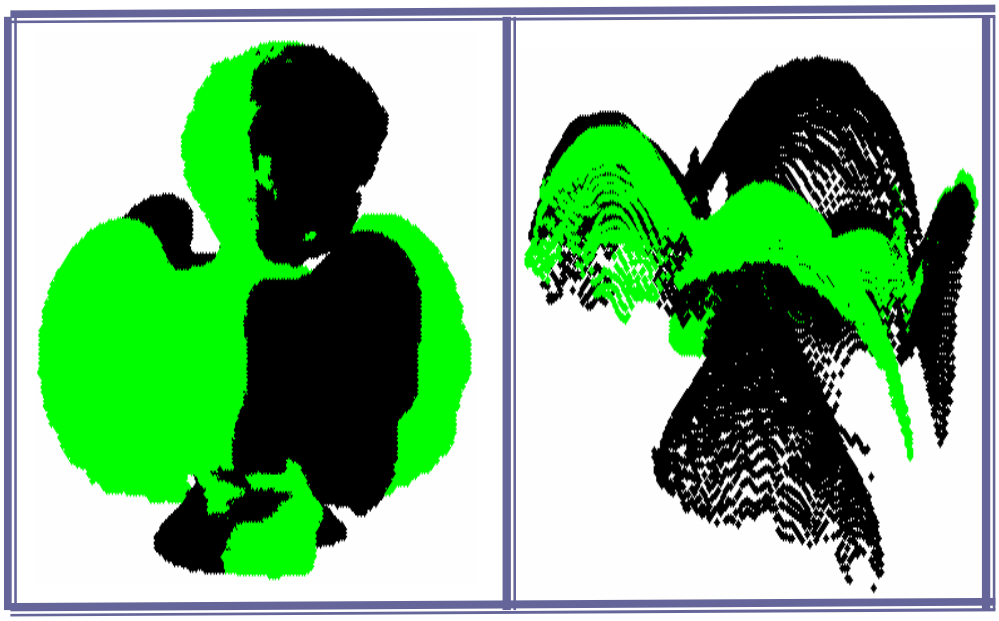


Fig. 44 En la figura de la derecha te presentamos el modelo P (verde) y Q (negro) situados en el mismo origen, y en la figura de la izquierda los podemos apreciar en diferente ángulo

Una vez situados los modelos en el mismo origen, procedemos a encontrar la matriz de rotación que se realiza a partir de la descomposición de valores singulares de $P*Q^T$, es decir; con la multiplicación del conjunto de puntos del modelo ***P*** con la transpuesta del conjunto de puntos del modelo ***Q***, esto es: $X = P*Q^T$ en donde a la nueva matriz X se le aplica la factorización lineal que consiste en descomponerla como un producto de 3 matrices: $U*V*S$. Donde U y V son matrices ortogonales y S es la matriz identidad. Por lo tanto la multiplicación de esos valores con S transpuesta, proporciona la matriz de rotación $R = U*S*V^T$.

A continuación la matriz de rotación es multiplicada con la matriz que tiene el conjunto de puntos del modelo ***P***, este conjunto de puntos es rotado sobre el nuevo centroide de la figura y con esto logramos que el modelo ***P*** tenga la misma orientación que tiene el modelo ***Q***; Como se muestra en la figura 40.

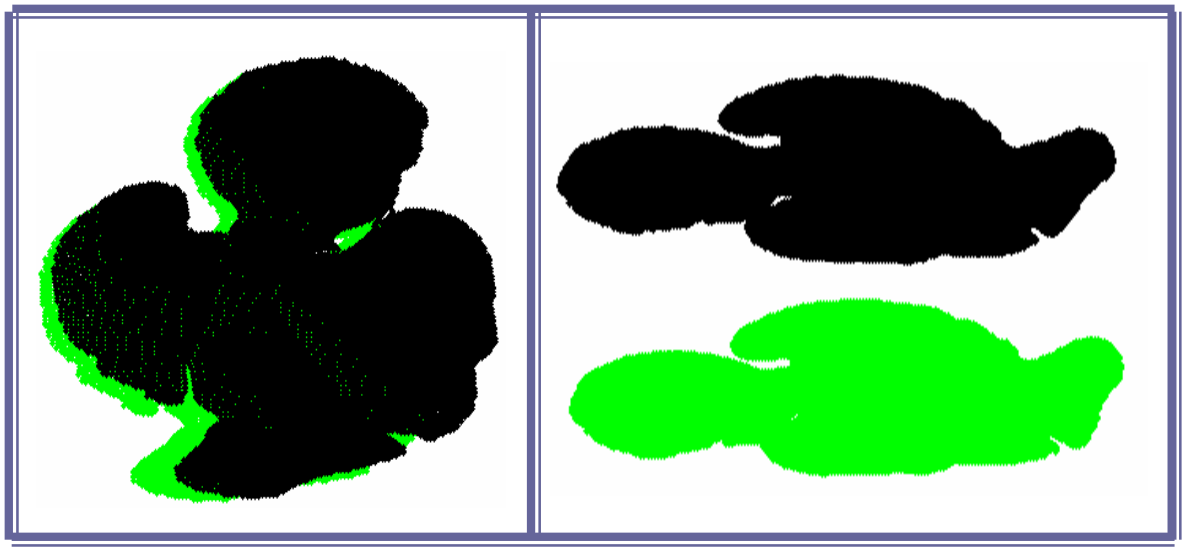


Fig. 45 Conjunto de puntos del modelo P es rotado, para tener la misma orientación que el modelo Q, y lograr la alineación de los modelos. Se tomaron como mínimo 3 parejas de puntos correspondientes a cada modelo, en esta figura se observa que ya se aplico la rotación y la traslación En esta figura se observa claramente que falta aplicar el parámetro de traslación, para lograr la alineación de los modelos P y Q

Como podemos ver en la figura 45 los modelos no están completamente alineados, ya que se encuentran ligeramente alejados. Para solucionar este problema a la matriz que tiene el producto del conjunto de puntos P con la matriz de rotación, se le suma a cada vértice la media del modelo Q. como se muestra en la figura 46.

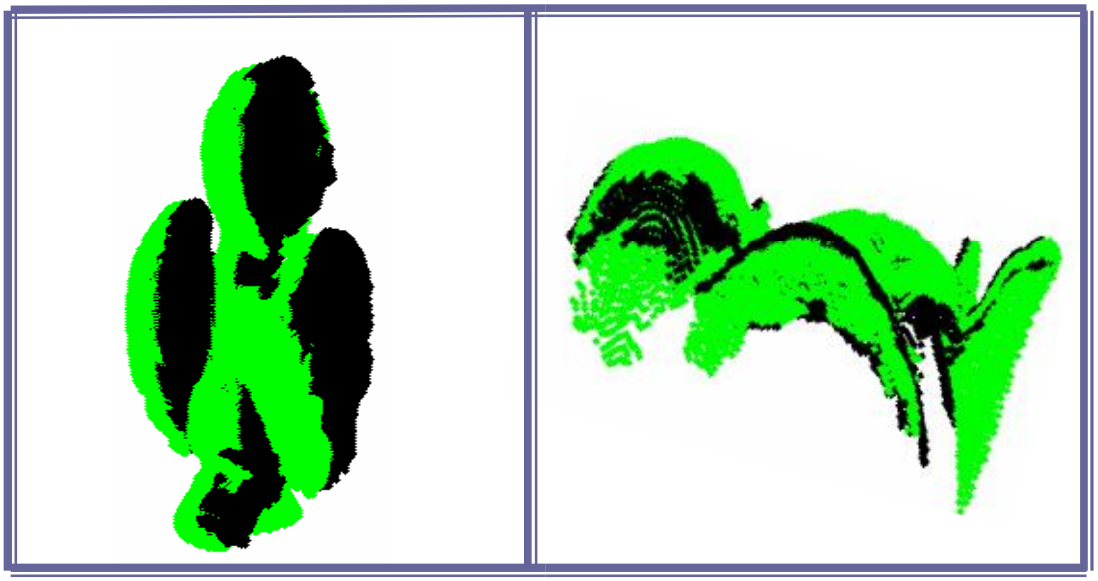


Fig. 46 En la figura de la izquierda presentamos al Modelo P (verde) que se encuentra alineado con el modelo Q (negro). En estas 2 figuras ilustramos nuestro objetivo; tener nuestro modelo 3D único, formado a partir de datos parciales P y Q.

En la figura 46 podemos apreciar que el modelo ***P*** ya se encuentra alineado con el modelo ***Q***. Es decir el objetivo de generar un modelo 3D único formado a partir de datos parciales (***P*** y ***Q***) se ha logrado. A continuación en la figura 47 se muestran tomas desde diferentes puntos de vista, con el fin de observar que los modelos se encuentran alineados, es decir que el modelo ***P*** se encuentre alineado con el modelo ***Q***.

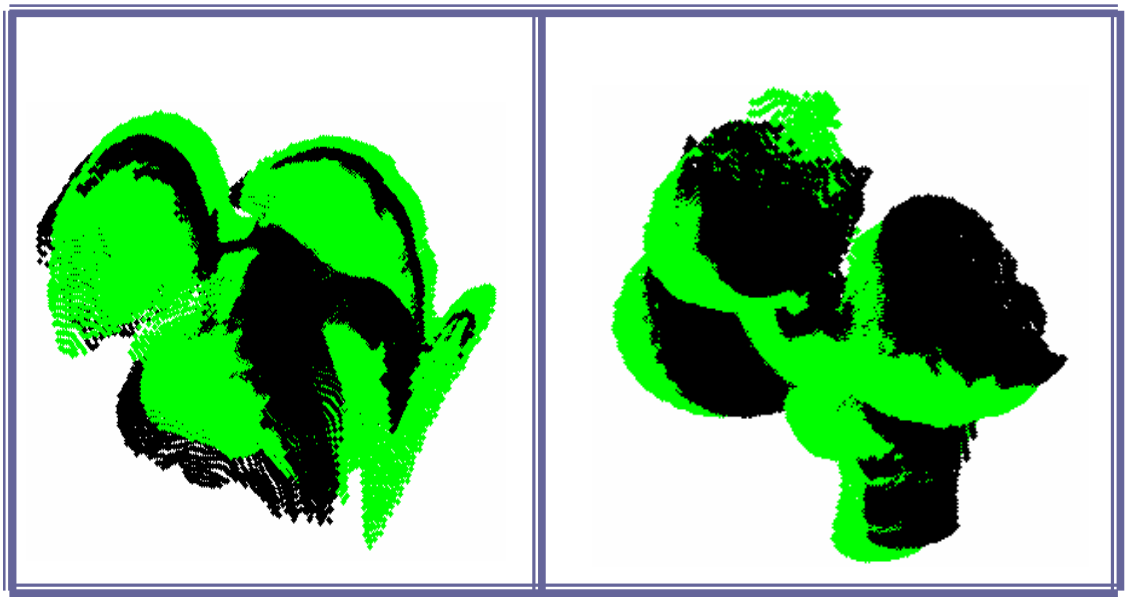


Fig. 47 Presentamos el Modelo único 3D tomado desde diferentes puntos de vista.

3.6 Presentación de Pruebas Obtenidas.

"Pruebas realizadas con cubos"

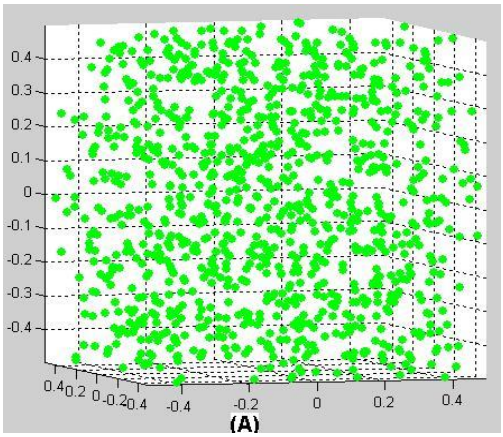


Imagen 1. Conjunto de puntos del primer modelo "P". Con centroide diferente al modelo "Q"

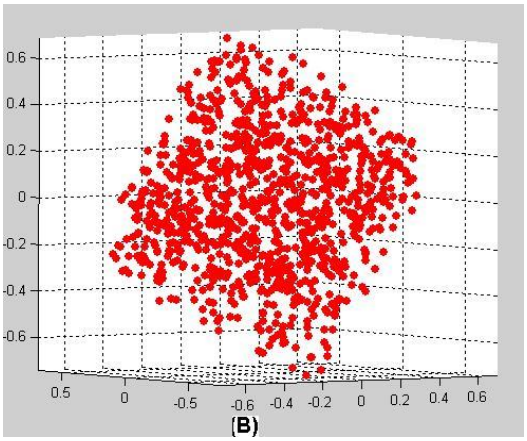


Imagen 2. Conjunto de puntos del segundo modelo "Q". Con centroide diferente al modelo "P"

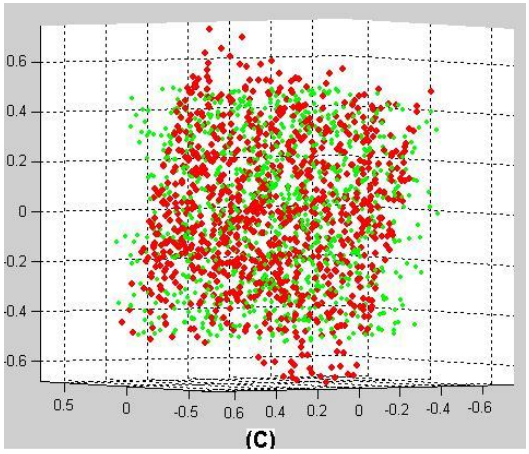


Imagen 3. Conjuntos de puntos pertenecientes a los modelos P y Q colocados en el mismo centroide.

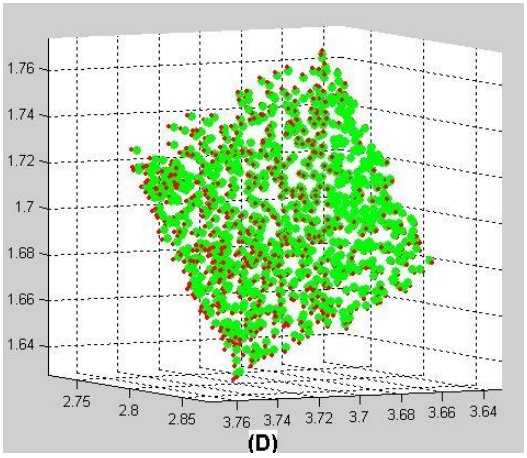


Imagen 4. Alineación del modelo P con Q formando un modelo 3D único. Aplicando los parámetros de transformación (método SVD) al modelo P.

Fig. 48 Representación del método del registrado para la alineación de los modelos parciales P y Q, formando un modelo 3D único utilizando CUBOS 3D

"Pruebas realizadas con Ángeles"

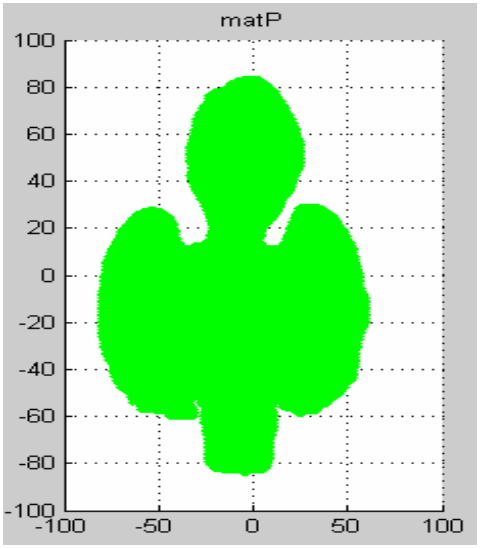


Imagen 1. Conjunto de puntos del primer modelo "P". Con centroide diferente al modelo "Q"

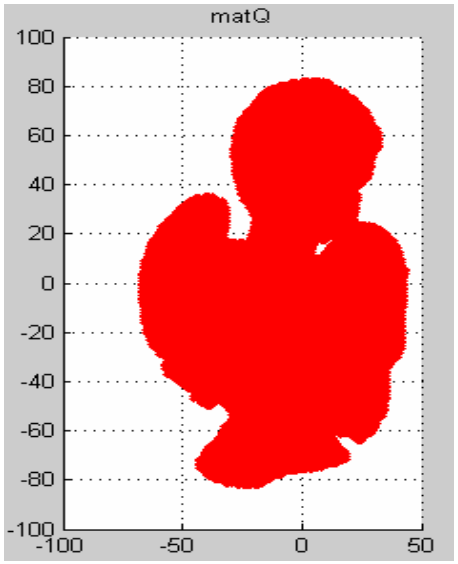


Imagen 2. Conjunto de puntos del segundo modelo "Q". Con centroide diferente al modelo "P"

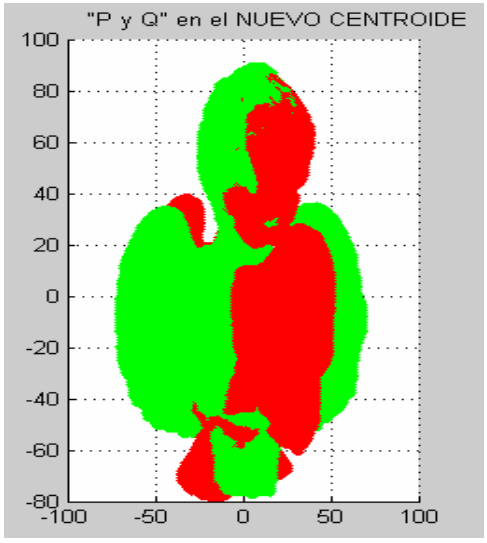


Imagen 3. Conjuntos de puntos pertenecientes a los modelos P y Q colocados en el mismo centroide.

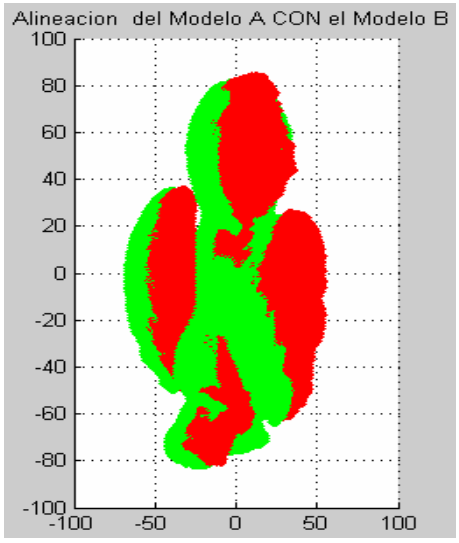


Imagen 4. Alineación del modelo P con Q formando un modelo 3D único. Aplicando los parámetros de transformación (método SVD) al modelo P.

Fig. 49 Representación del método del registrado para la alineación de los modelos parciales P y Q, formando un modelo 3D único utilizando Ángeles.

"Pruebas realizadas con EL OSO POOH"

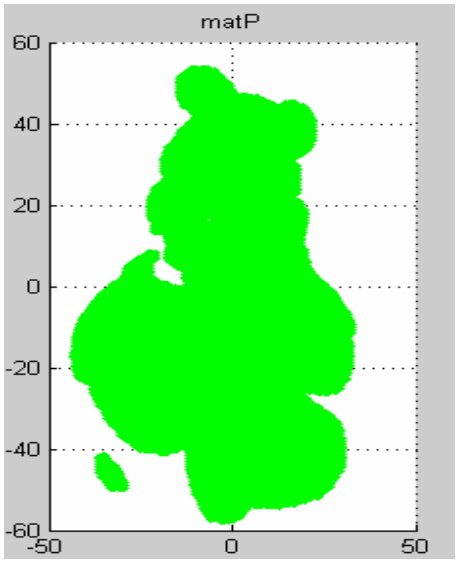


Imagen 1. Conjunto de puntos del primer modelo "P". Con centroide diferente al modelo "Q"

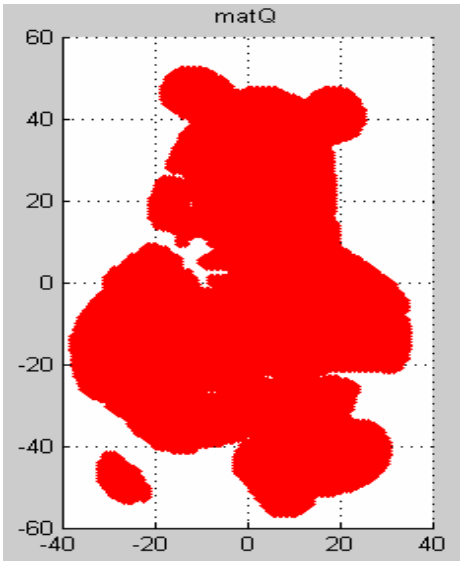


Imagen 2. Conjunto de puntos del segundo modelo "Q". Con centroide diferente al modelo "P"

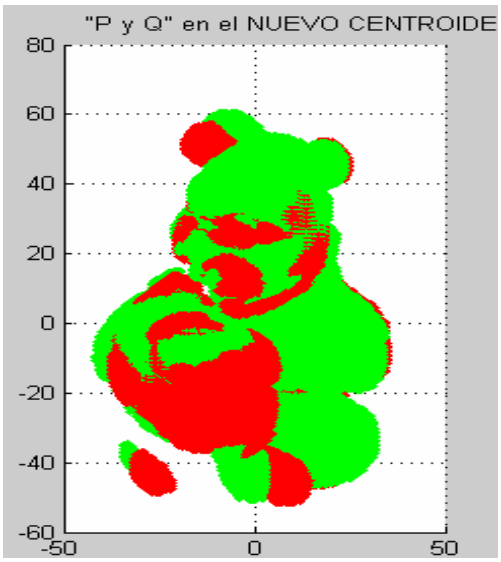


Imagen 3. Conjuntos de puntos pertenecientes a los modelos P y Q colocados en el mismo centroide.

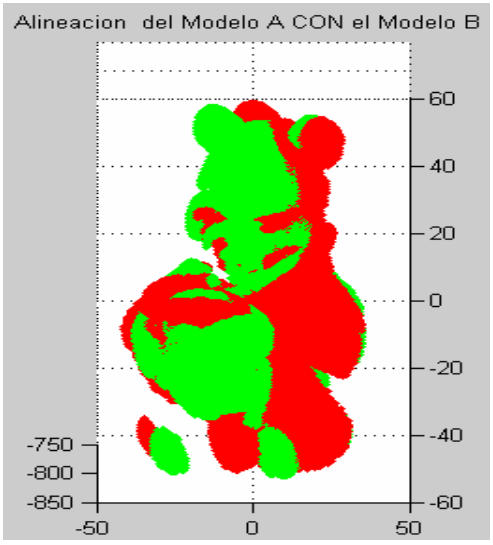


Imagen 4. Alineación del modelo P con Q formando un modelo 3D único. Aplicando los parámetros de transformación (método SVD) al modelo P.

Fig. 50 Representación del método del registrado para la alineación de los modelos parciales P y Q, formando un modelo 3D único utilizando el Oso Pooh.

"Pruebas realizadas con Ranas"

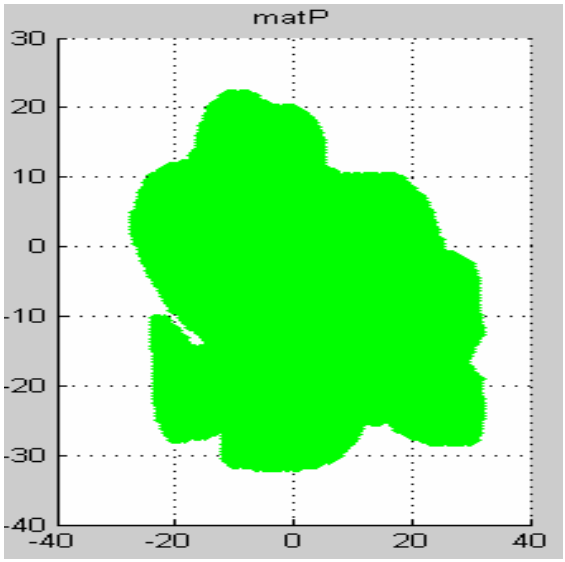


Imagen 1. Conjunto de puntos del primer modelo "P". Con centroide diferente al modelo "Q"

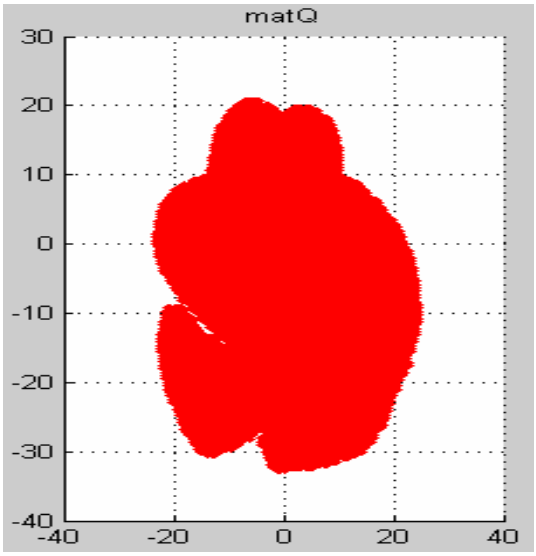


Imagen 2. Conjunto de puntos del segundo modelo "Q". Con centroide diferente al modelo "P"

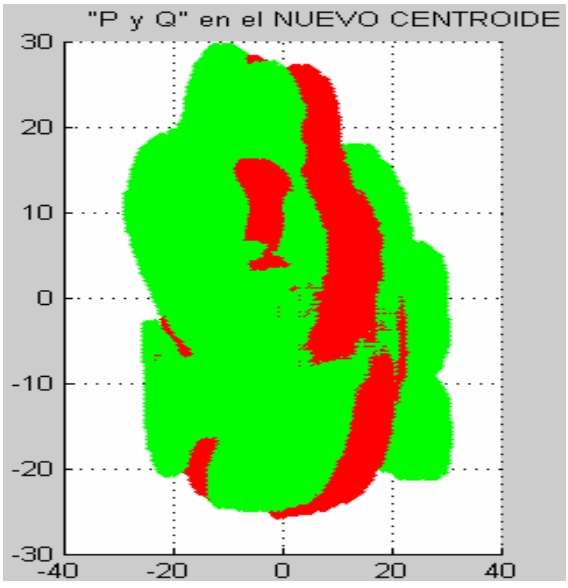


Imagen 3. Conjuntos de puntos pertenecientes a los modelos P y Q colocados en el mismo centroide.

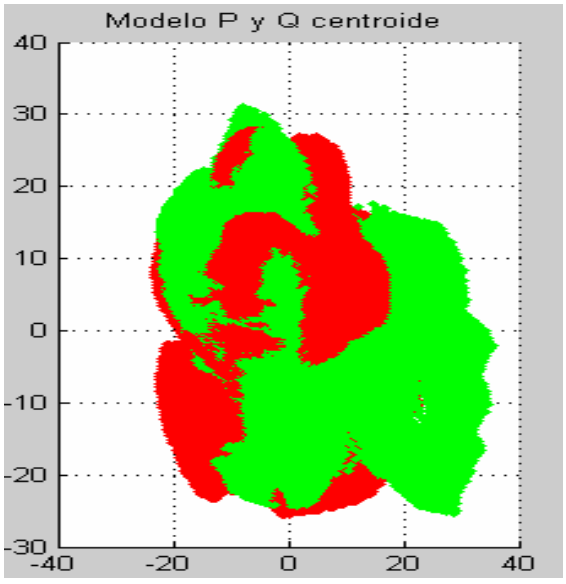


Imagen 4. Alineación del modelo P con Q formando un modelo 3D único. Aplicando los parámetros de transformación (método SVD) al modelo P.

Fig. 51 Representación del método del registrado para la alineación de los modelos parciales P y Q, formando un modelo 3D único utilizando las Ranas.

| "Pruebas realizadas con BUDDHA"

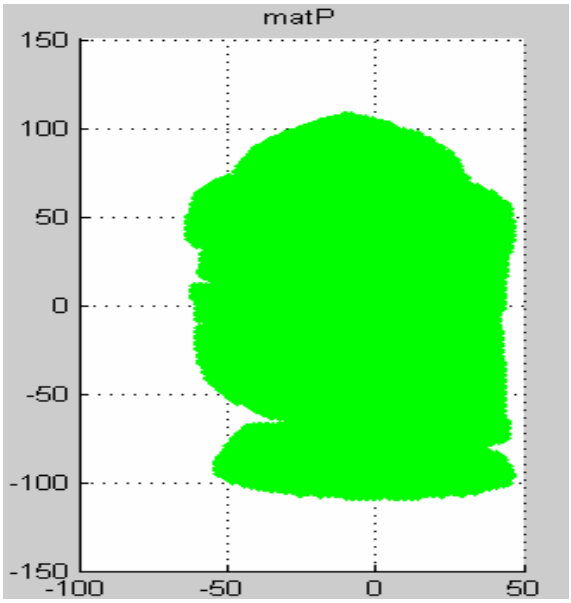


Imagen 1. Conjunto de puntos del primer modelo "P". Con centroide diferente al modelo "Q"

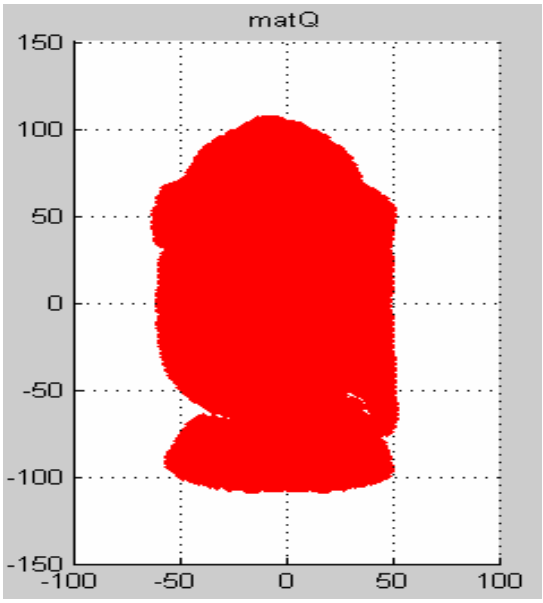


Imagen 2. Conjunto de puntos del segundo modelo "Q". Con centroide diferente al modelo "P"

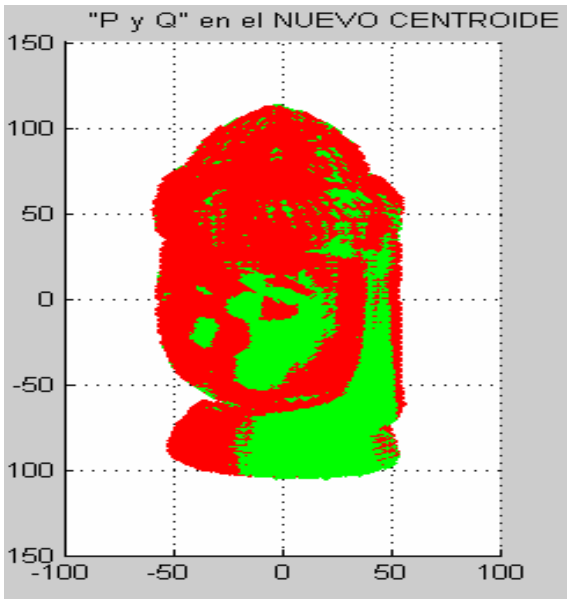


Imagen 3. Conjuntos de puntos pertenecientes a los modelos P y Q colocados en el mismo centroide.

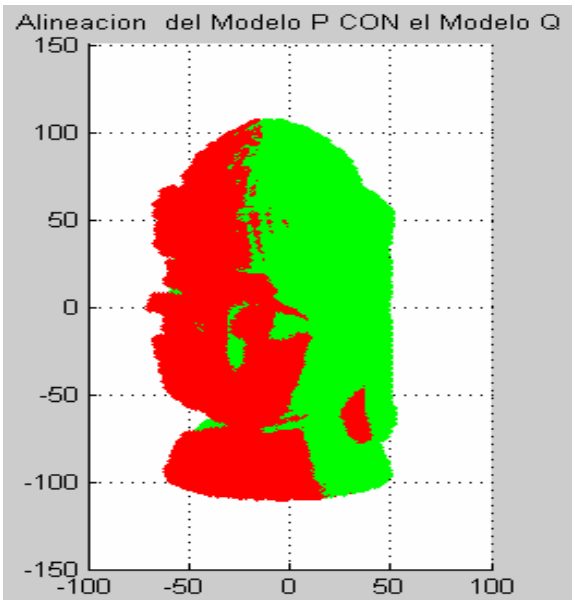


Imagen 4. Alineación del modelo P con Q formando un modelo 3D único. Aplicando los parámetros de transformación (método SVD) al modelo P.

Fig. 52 Representación del método del registrado para la alineación de los modelos parciales P y Q, formando un modelo 3D único utilizando Buddhas

"Pruebas realizadas con Pato"

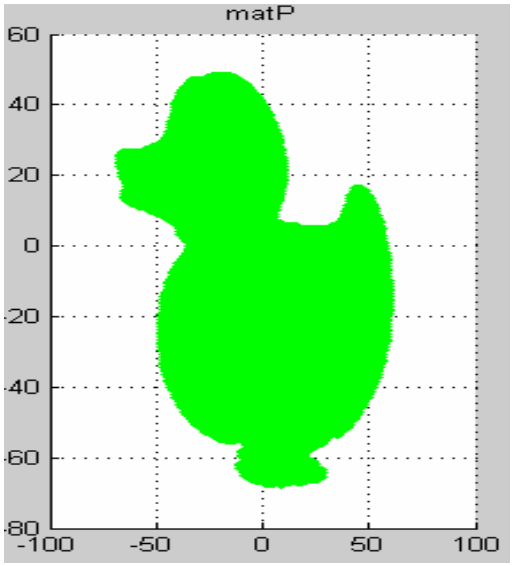


Imagen 1. Conjunto de puntos del primer modelo "P". Con centroide diferente al modelo "Q"

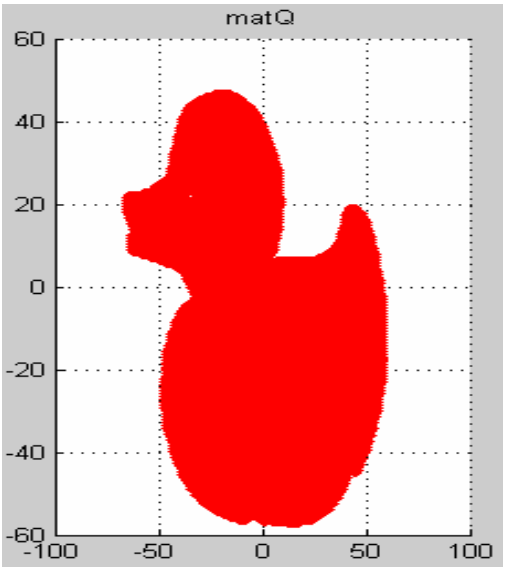


Imagen 2. Conjunto de puntos del segundo modelo "Q". Con centroide diferente al modelo "P"

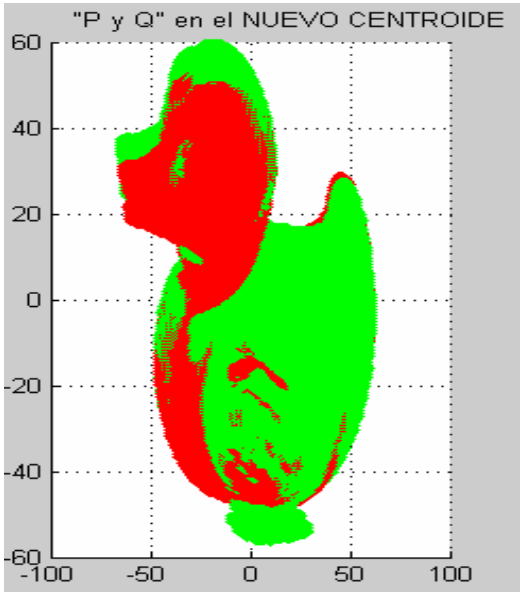


Imagen 3. Conjuntos de puntos pertenecientes a los modelos P y Q colocados en el mismo centroide.

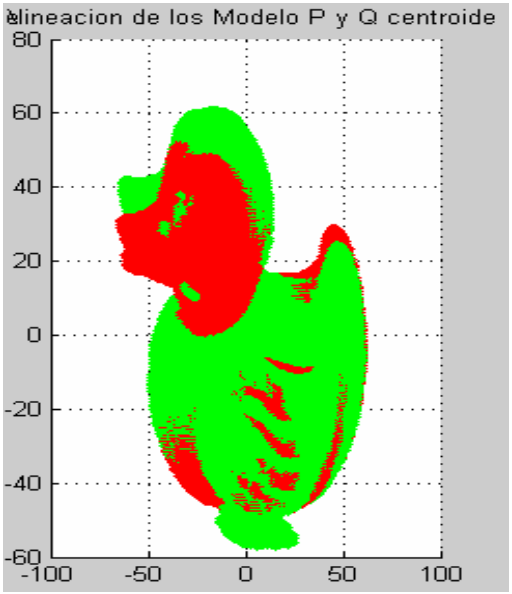


Imagen 4. Alineación del modelo P con Q formando un modelo 3D único. Aplicando los parámetros de transformación (método SVD) al modelo P.

Fig. 53 Representación del método del registrado para la alineación de los modelos parciales P y Q, formando un modelo 3D único utilizando Patos

"Ángeles texturizados"

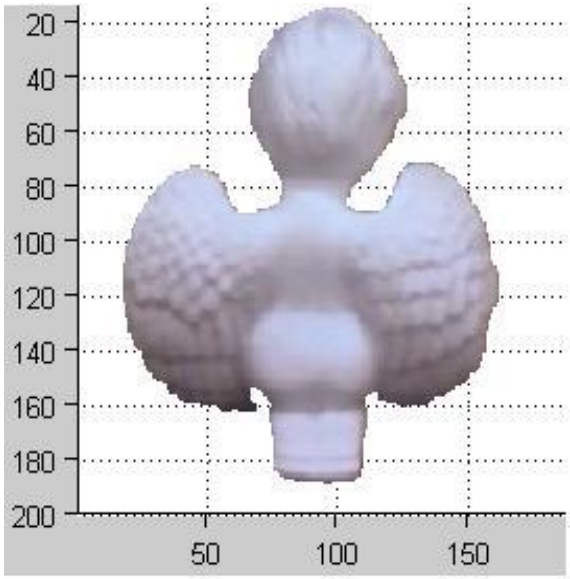


Imagen 1: Conjunto de puntos del modelo "P" texturizado.

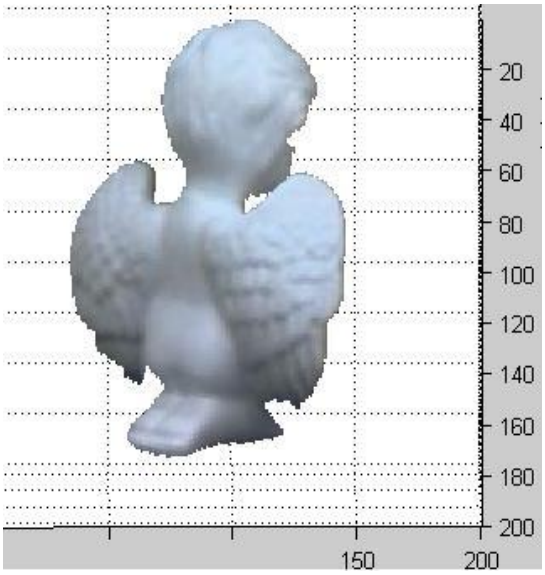


Imagen 2: Conjunto de puntos del modelo "Q" texturizado.

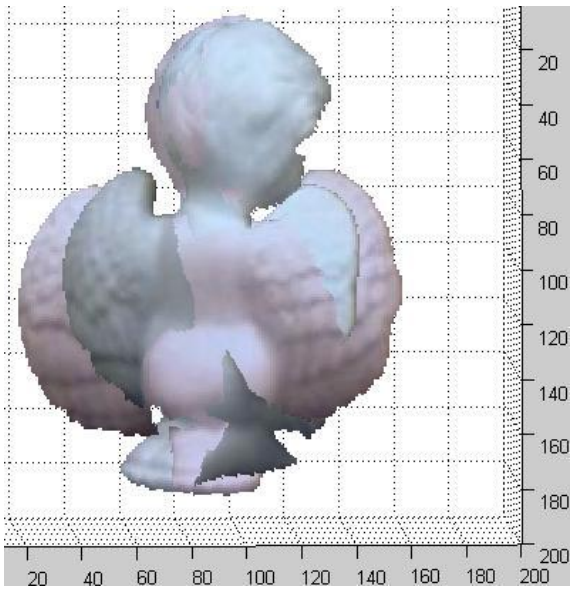


Imagen 3: Conjunto de puntos de los modelos "P" y "Q" texturizados, en el mismo centroide.

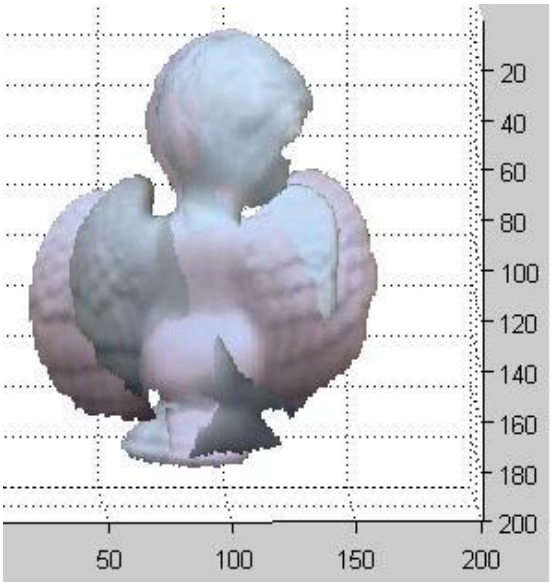


Imagen 4: Alineación del modelo 3D único Texturizado.

Fig. 54 Texturización del modelo 3D único para darle una apariencia Real. "Ángeles"

"Ositos Pooh texturizados"

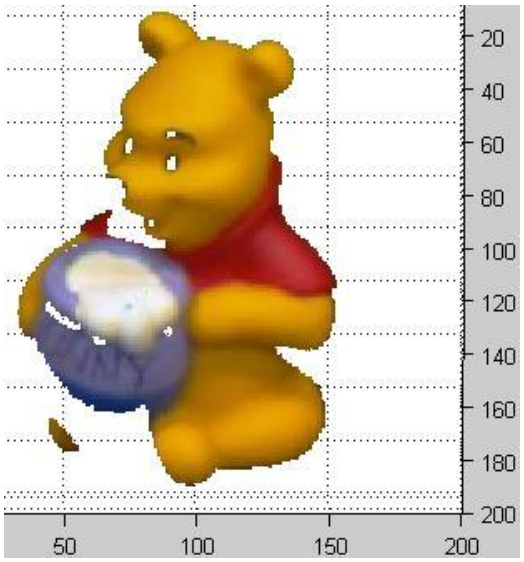


Imagen 1: Conjunto de puntos del modelo "P" texturizado.

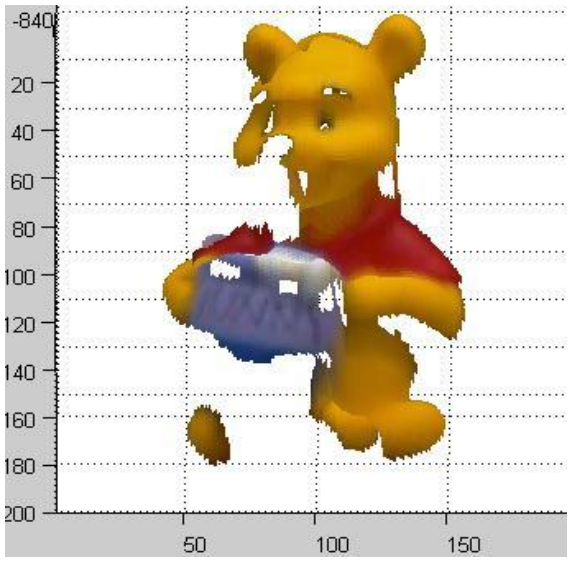


Imagen 2: Conjunto de puntos del modelo "Q" texturizado.

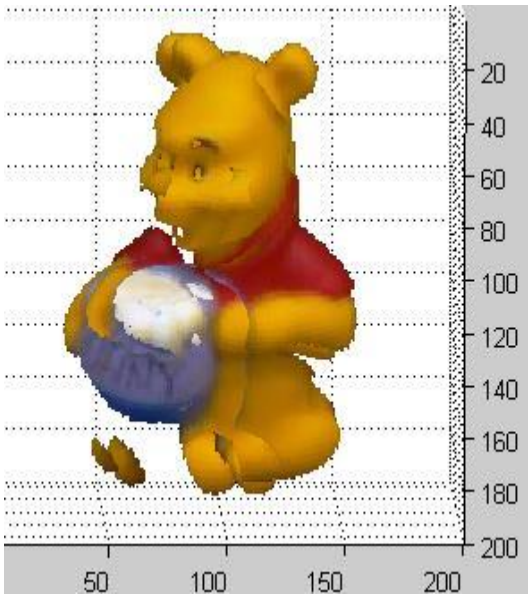


Imagen 3: Conjunto de puntos de los modelos "P" y "Q" texturizados, en el mismo centroide.

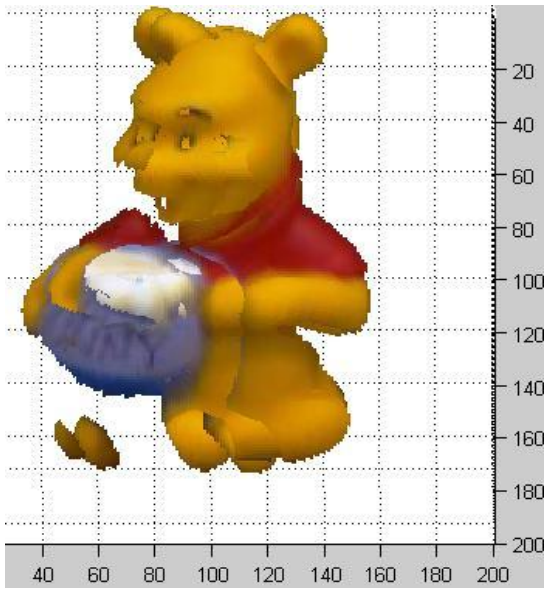


Imagen 4: Alineación del modelo 3D único Texturizado.

Fig. 55 Texturización del modelo 3D único con fin de darle una apariencia Real. "Oso pooh"

"Ranas Texturizadas"

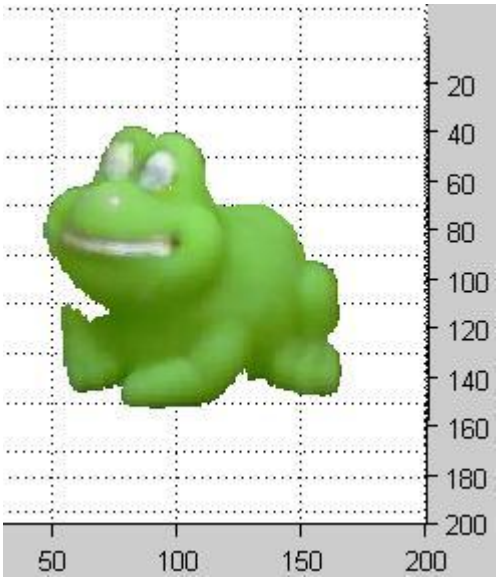


Imagen 1: Conjunto de puntos del modelo "P" texturizado.

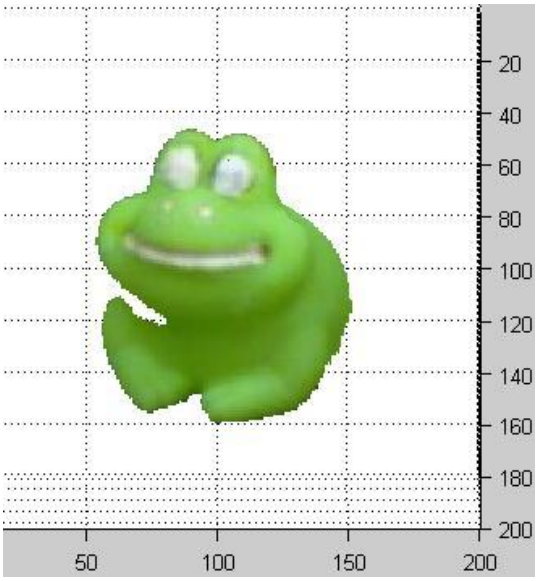


Imagen 2: Conjunto de puntos del modelo "Q" texturizado.

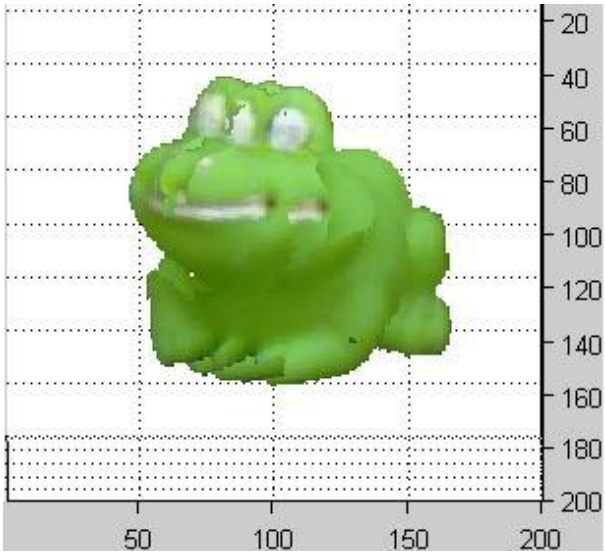


Imagen 3: Conjunto de puntos de los modelos "P" y "Q" texturizados, en el mismo centroide

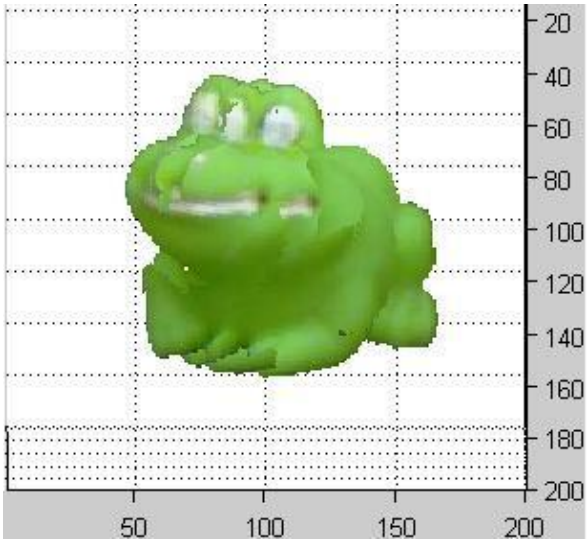


Imagen 4: Alineación del modelo 3D único Texturizado.

Fig. 56 Texturización del modelo 3D único con fin de darle una apariencia Real. "Ranas"

"Buddhas Texturizados"

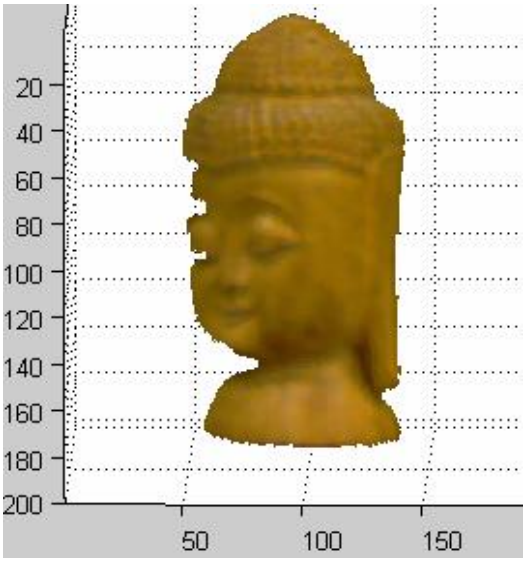


Imagen 1: Conjunto de puntos del modelo "P" texturizado.

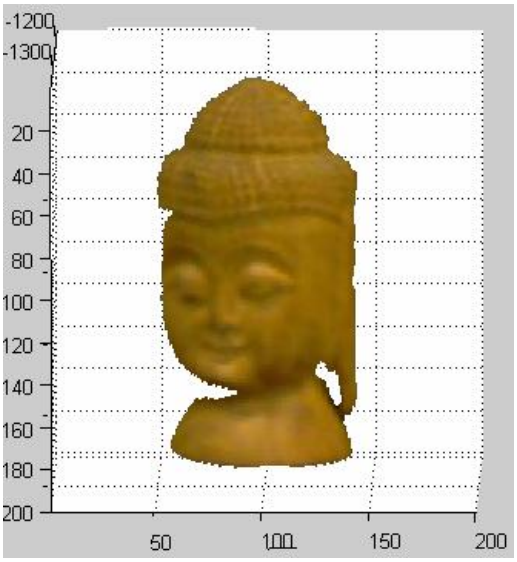


Imagen 2: Conjunto de puntos del modelo "Q" texturizado.

Imagen 3: Conjunto de puntos de los modelos "P" y "Q" texturizados, en el mismo centroide

Imagen 4: Alineación del modelo 3D único Texturizado.

Fig. 57 Texturización del modelo 3D único con fin de darle una apariencia Real. "Buddhas"

"Patos Texturizados"

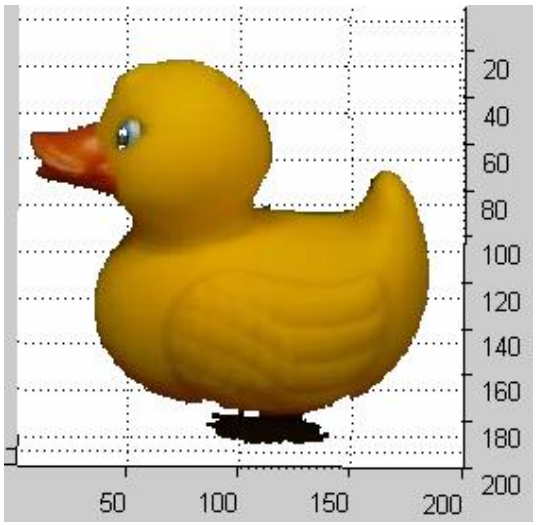


Imagen 1: Conjunto de puntos del modelo "P" texturizado.

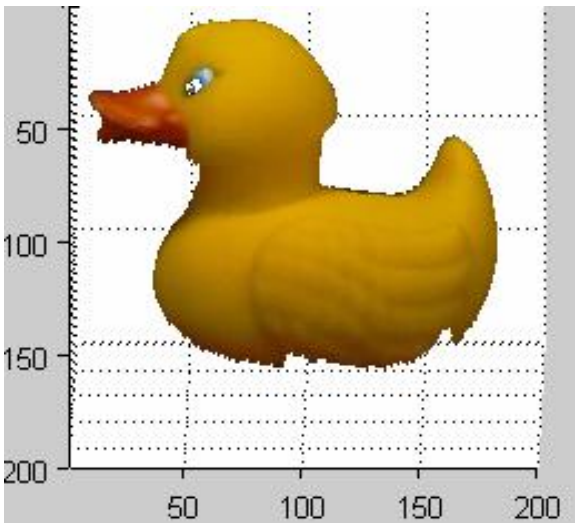


Imagen 2: Conjunto de puntos del modelo "Q" texturizado.

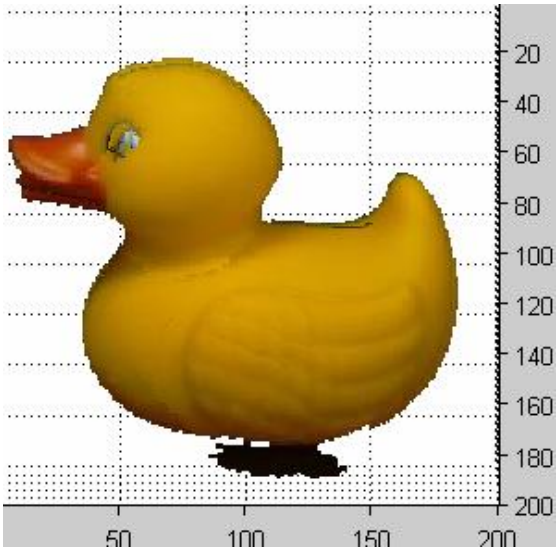


Imagen 3: Conjunto de puntos de los modelos "P" y "Q" texturizados, en el mismo centroide

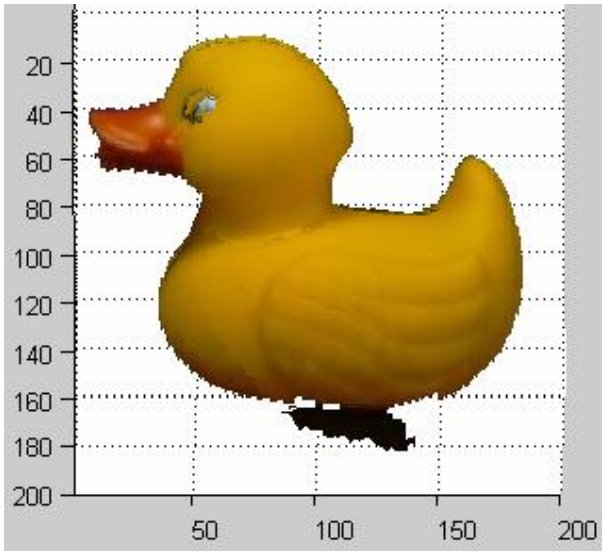


Imagen 4: Alineación del modelo 3D único Texturizado.

Fig. 58 Texturización del modelo 3D único con fin de darle una apariencia Real. "Patos"

Conclusiones y Trabajos Futuros.

Conclusiones.

- En este proyecto se programaron algoritmos que permiten formar un modelo único tridimensional de apariencia realista a partir de *datos parciales*, para lograr esta tarea se programaron los módulos que integran el algoritmo **ICP** (*Iterative Closest Point*); en particular se implementaron 3 algoritmos: AESA, IAESA y SVD.

- Los algoritmos (AESA e IAESA) son algoritmos de indexación, que permiten realizar "*la búsqueda del vecino más cercano de un elemento dado*", solo que IAESA tiene la ventaja de ser mas rápido, ya que en [27b] mencionan que 7IAESA recupera 85% de las respuestas correctas examinando apenas el 10% del conjunto de datos, esto es, 80% menos de lo que necesitaría AESA para obtener el mismo resultado, lo que significa que en menos iteraciones logra encontrar el vecino mas cercano. Esto lo corroboramos en los experimentos presentados anteriormente.

- El tercer algoritmo implementado es el método SVD (*Descomposición de Valores Singulares*) que estima los parámetros de la función de transformación (*rotación, traslación y escala*) misma que facilita la alineación de un modelo con otro.

- Finalmente se realiza el proceso de texturización de los modelos con fin de darle al modelo final una apariencia real.

A pesar de que muchas de las ideas presentadas fueron extraídas de diferentes artículos de otros investigadores, las aportaciones más importantes son:
La implementación y pruebas de las ideas planteadas en los artículos. y La aplicación de IAESA en el problema de registrado de modelos 3D ya que anteriormente solo se ha utilizado en problemas de conjuntos de datos.

La implementación de los algoritmos se presenta en los apéndices A, B y C.

2 Trabajo a Futuro.

- Utilizar diferentes métricas de similaridad que toleren cambios de iluminación ruido, cambios de vista al momento de capturar vistas parciales de los modelos.

- Explorar alguno de los métodos de selección de pivotes presentados en la sección 3.

- Explorar la selección de puntos de forma automática para forzar que dichos puntos seleccionados pertenezcan al contorno del modelo 3D.

- Integrar los módulos para completar el proceso iterativo del algoritmo ICP.

Bibliografía

[1a] Indexación Efectiva de Espacios Métricos usando Permutaciones por Karina Mariela Figueroa Mora Universidad de Chile Facultad de Ciencias Físicas y Matemáticas Depto. De Ciencias de la Computación

[2a] 2D and 3D Image Registration for medical, Remote Sensing, and Industrial Applications

[3a] Modelado Dinámico de Objetos en 3D por Sergio Juárez Velásquez supervisada por: Miguel Arias Estrada INAOE

[4a] E. Chávez Algoritmos para detección de cúmulos en nubes de datos 4ª. Jornadas iberoamericanas de informática 1997.

[5a] "Algoritmos de búsqueda de vecinos más próximos en espacios métricos" Tesis Doctoral de María Luisa Micó Andrés, Departamento de Sistemas informáticos y Computación. Universidad Politécnica de Valencia, 1 de Marzo de 1996.

[6a] El método k-nn (K nearest neighbors Fix and Hodges, 1951 http://es.wikipedia.org/wiki/Categor%C3%ADa:Algoritmos

[8b] J.G. Mcgreger, "Data-Drive Registration of Laser Range Images"

[9b] Ning Li, "Retrieving Camera Parameters from Real Video Images", Master Thesis, The University of British Columbia, August 1998.

[10b] "Interprete y navegador de mundos virtuales". Tesis Licenciatura. Ingeniería en Sistemas Computacionales. Departamento de Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas Puebla. Enero 2001 Cardona Amozuurrutia, J. F.

[11b] Paul. J. Besl and N.D. McKay. A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2):239–256, February 1992

[12b] David A. Forsyth and Jean Ponce. Computer

[13b] Efficient Variants of the ICP Algorithm Szymon Rusinkiewicz Marc Levoy Stanford University

[14b] Berthold K.P. Horn. Closed form solutions of absolute orientation using orthonormal matrices. Journal of the Optical Society A, 5(7):1127–1135, July 1988.

[15b] Bergevin, R., Soucy, M., Gagnon, H., and Laurendeau, D. "Towards a General Multi-View Registration Technique," Trans. PAMI, Vol. 18, No. 5, 1996.

[16b] Masuda, T., Sakaue, K., and Yokoya, N. "Registration and Integration of Multiple Range Images for 3-D Model Construction," Proc. CVPR, 1996.

[17b] Pulli, K. Surface Reconstruction and Display from Range and Color Data, Ph. D. Dissertation, University of Washington, 1997.

[18b] Pulli, K. "Multiview Registration for Large Data Sets," Proc. 3DIM, 1999.

[19b] Besl, P. and McKay, N. "A Method for Registration of 3-D Shapes," Trans. PAMI, Vol. 14, No. 2, 1992.

[20b] Chen, Y. and Medioni, G. "Object Modeling by Registration of Multiple Range Images," Proc. IEEE Conf. on Robotics and Automation, 1991

[21b] Fast and Accurate Shape-Based Registration, Ph. D. Dissertation, Carnegie Mellon University, 1996

[22b] Arun, K., Huang, T., and Blostein, S. "Least-Squares Fitting of Two 3-D Point Sets," Trans. PAMI, Vol. 9, No. 5, 1987.

[23b] Johnson, A. and Kang, S. "Registration and Integration of Textured 3-D Data," Proc. 3DIM, 1997.We now look at the x and z components of the uncertainty in the, 1996.

[24b] Masuda, T., Sakaue, K., and Yokoya, N. "Registration and Integration of Multiple Range Images for 3-D Model Construction," Proc. CVPR, 1996

[25b] Blais, G. and Levine, M. "Registering Multiview Range Data to Create 3D Computer Objects," Trans. PAMI, Vol. 17, No. 8, 1995.

[26b] "Políticas de Selección de Pivotes para Índices Métricos" Norma Edith Herrera, (Departamento de Informática Universidad Nacional de San Luís) Anabella C. De Battista, Andrés J. Pascal (Departamento de Sistemas de información Universidad Tecnológica Nacional Regional Concepción del Uruguay)

[27b] "On the Least Cost For Proximity Searching in Metric Spaces". En WEA'06 (5th) Workshop Experimental Algorithms), Menorca, España. Mayo 2006. Páginas 270-290, LNCS4007, Springer. Karina Figueroa, Edgar Chávez, Gonzalo Navarro y Rodrigo Paredes.

[28d] SVD Singular Value Descomposition. Universidad Técnica Federico Santa María. Depto. de Matemáticas Análisis Numérico.

[29d] Ning Li, "Retrieving Camera Parameters from Real Video Images" Master Thesis, The University of British Columbia, August 1998.

[30d] The mathematical Informatics Section, Information Scienc Division, Electrotechnical Laboratory, 1-14 Umezono, Tasukuda-shi, Ibaraki 305 Japan.

[31d] B: K: P Horn "Closed-form solution of absolute orientation using of ortonormal matrices" J. Opt. Soc. Amer. A, vol 5 no. 7, pp 1127-1135, 1987.

[32d] T.S Huang, S.D. Blostein and e. A margerum," Least-squares estimation of motion parameters from 3D point correspondences" in Proc. IEEE Conf. Computer Visión and Pattern Recognition, Miami Beach, FL,1986, pp24-26

[33d] Estimación in Medical Tecnology Elvin Eren 15 junio 2004

[34d] Signal Análisis and Machina Perception Laboratory Department of Electrical Engineering http://sampl.ece.ohio-state.edu/data/3DDB/RID/minolta/