Detección y reconocimiento de objetos usando imágenes RGB y nubes de puntos organizadas.

Jesus Cruz Navarro

Director de tesis: Dr. Jesús Savage Carmona

Posgrado en ciencia e ingeniería de la computación
Universidad Nacional Autónoma de México

15 de marzo de 2016

Contenido

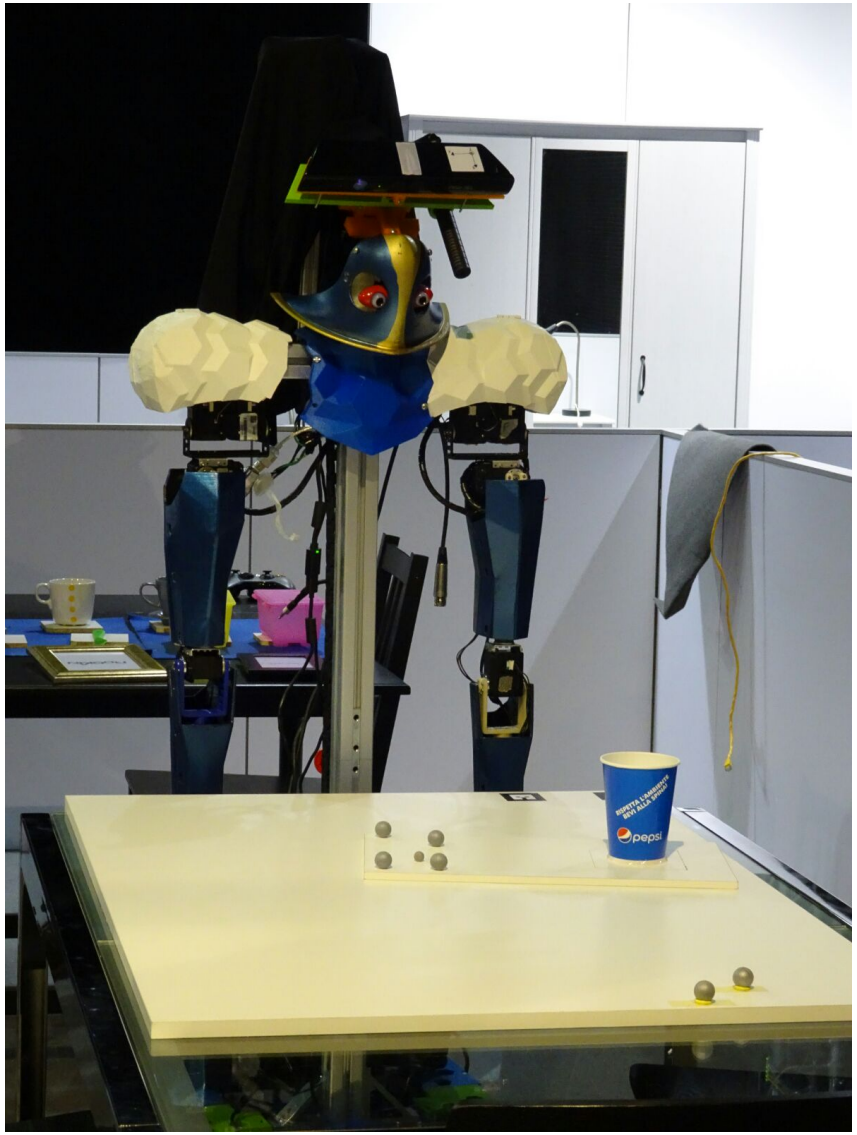
Introducción

Motivación

Justina

Robot Justina:

- Robot autónomo móvil de servicio.
- Funciones: detectar y reconocer objetos, entre otras.
- Reconocimiento basado en Puntos de Interés (SIFT).
- Sensor Kinect (Cámara RGB-D)
- Participación en competencias internacionales.



RoCKIn 2014

Functionality Benchmarck: Object Perception

RoCKIn 2014

Functionality Benchmarck: Object Perception

Problema:

Objetos con poca o nula textura !

Objetivo e hipótesis

Objetivo

Objetivo de la tesis:

Desarrollar un sistema que permita, de manera eficiente, **detectar** y **reconocer** objetos con poca o nula textura, que se encuentren **sobre planos horizontales** en una escena, utilizando la **imagen RGB** y la **nube de puntos organizada** que proporciona el sensor Kinect del robot Justina.

Objetivo

Objetivo de la tesis:

Desarrollar un sistema que permita, de manera eficiente, **detectar** y **reconocer** objetos con poca o nula textura, que se encuentren **sobre planos horizontales** en una escena, utilizando la **imagen RGB** y la **nube de puntos organizada** que proporciona el sensor Kinect del robot Justina.

Detectar Decidir si existen o no (y cuantos) objetos sobre los plano horizontal en una escena.

Reconocer Decidir a que clase (de la base de entrenamiento) pertenecen los objetos detectados.

Hipótesis

Hipótesis de la tesis:

Se consiguen mejores resultados al identificar un objeto con poca o nula textura analizando sus dimensiones, la forma que proyecta este en el plano en el que se encuentra suspendido y su patrón de color, que utilizando técnicas basadas en la descripción de puntos característicos.

Hipótesis

Hipótesis de la tesis:

Se consiguen mejores resultados al identificar un objeto con poca o nula textura analizando sus dimensiones, la forma que proyecta este en el plano en el que se encuentra suspendido y su patrón de color, que utilizando técnicas basadas en la descripción de puntos característicos.

Para validar la hipótesis, se realizaron pruebas experimentales !

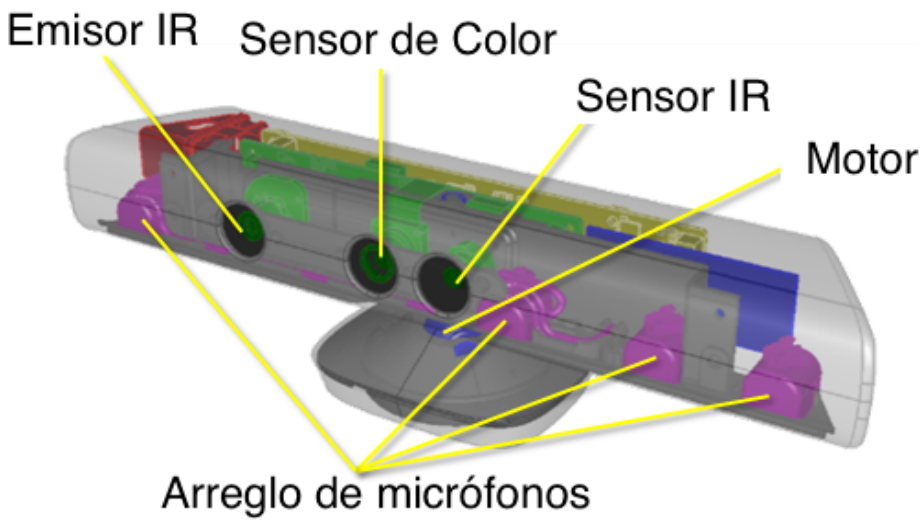
Antecedentes

El sensor Kinect

El sensor Kinect

Descripción:

- Cámara RGB-D
- Bajo costo
- Luz estructurada IR



Características:

- Ángulos de vision: 43° Vertical, 57° horizontal
- Resolución máx color: 1280 x 960
- Resolucion máx profundidad: 640 x 480
- Rango de profundidad: 0.8 - 4.0 m
- Frecuencia máx: 30 fps (640x480)

El módulo de visión

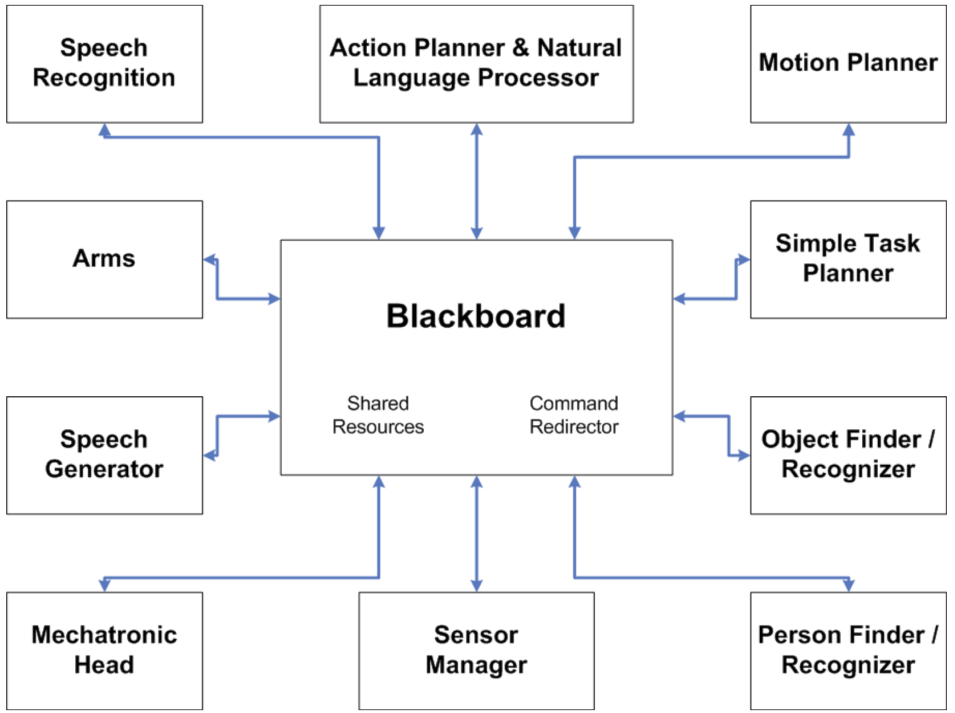
El módulo de vision

El sistema:

- Framework: VirBot
- Arquitectura: Blackboard / ROS

El modulo de visión:

- Comando-Respuesta
- Administra las cámaras
- Escrito en C++
- OpenNI
- OpenCV

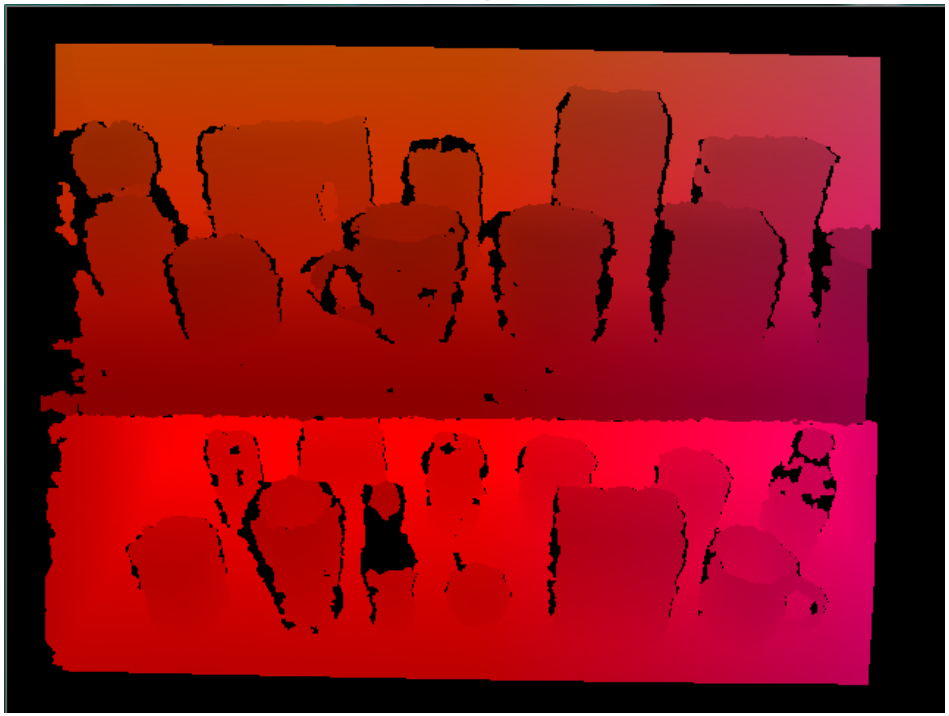


Estructuras de datos

OpenCV + OpenNI

Imágen RGB

Nube de puntos organizada



Detección de objetos

Transformación de la nube de puntos

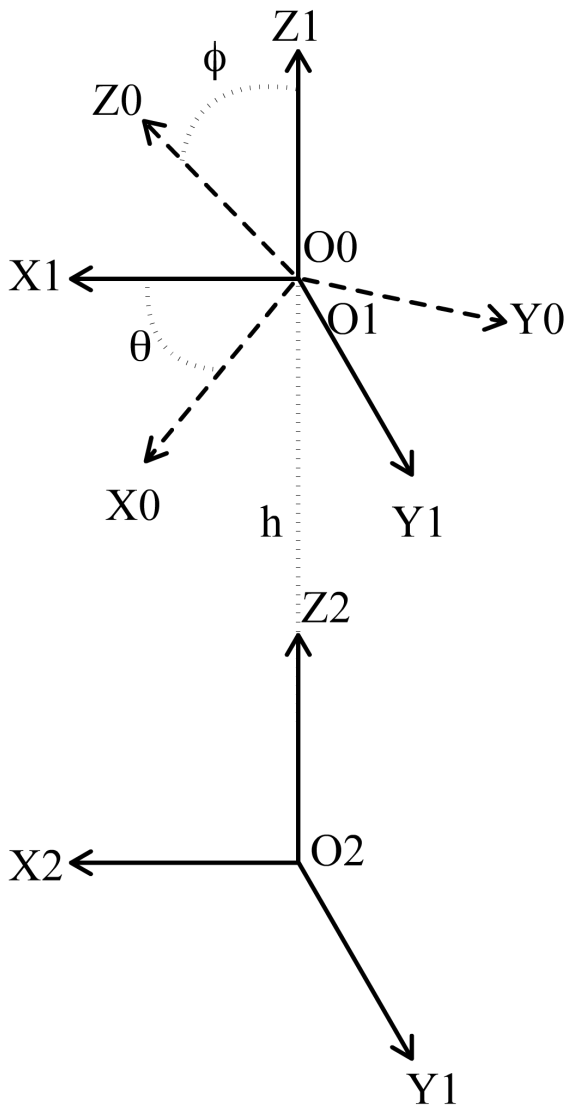
Transformación de la nube de puntos

Kinect montado sobre la cabeza (2DOF)

Transformación del sistema de referencia del Kinect (no estático) al sistema de referencia del robot (en la base)

Objetivo:

- Rápida identificación de los planos horizontales.
- Simplificación del reconocimiento.



Transformación homogénea

$$p_{2i} = \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix} p_{0i}$$

- θ - ángulo pan (guiñada) de la cabeza.
- ϕ - ángulo tilt (cabeceo) de la cabeza.
- h - la altura del Kinect.

Extracción de los planos horizontales

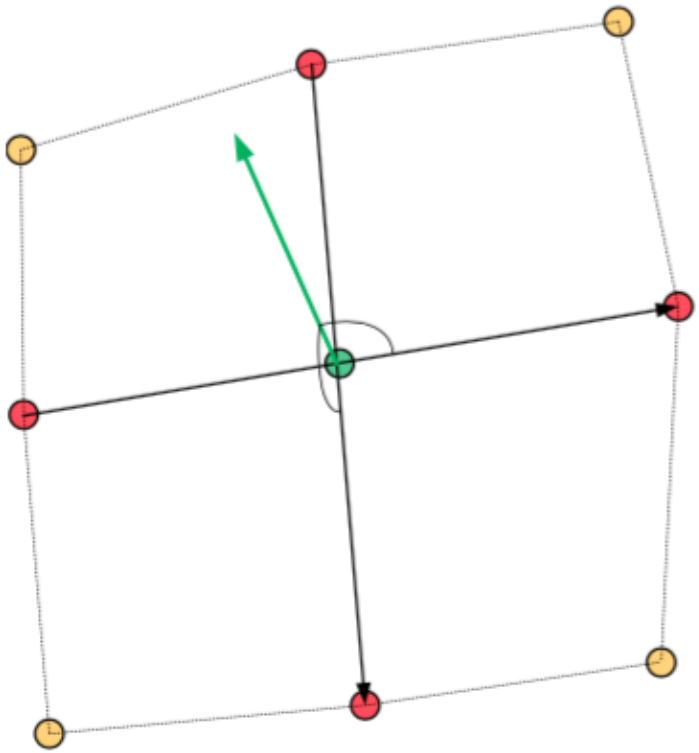
Filtro de bloque

- El sensor Kinect tiene ruido considerable.
- Normales no uniformes.
- Filtro de Bloque de tamaño $k \times k$
- Promediar puntos vecinos
- "Aplanar las superficie"

$$K = \frac{1}{k^2} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \quad \forall k > 0, k \in \text{impares}$$

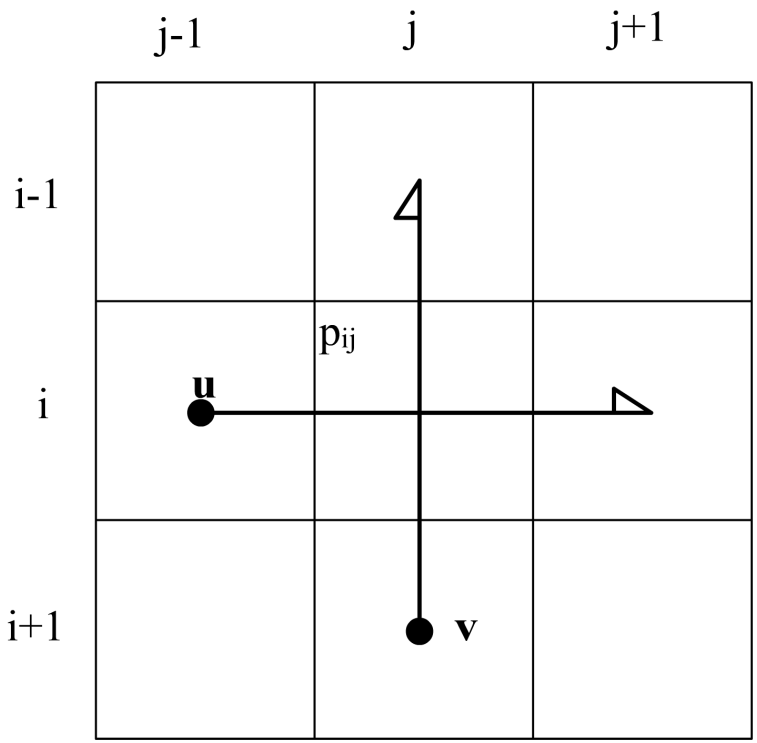
Normales Locales

- Planos son regiones con curvaturas similares.
- Curvatura en un punto p_i se puede determinar como la normal del plano tangente que mejor se adecue a los k vecinos más cercanos de p_i en un radio r.



Cálculo de normales locales

- Cálculo rápido de normales utilizado por [**?**].
- Pixeles vecinos en lugar de k vecinos en radio r.
- Rapidez vs exactitud



Para p_{ij} :

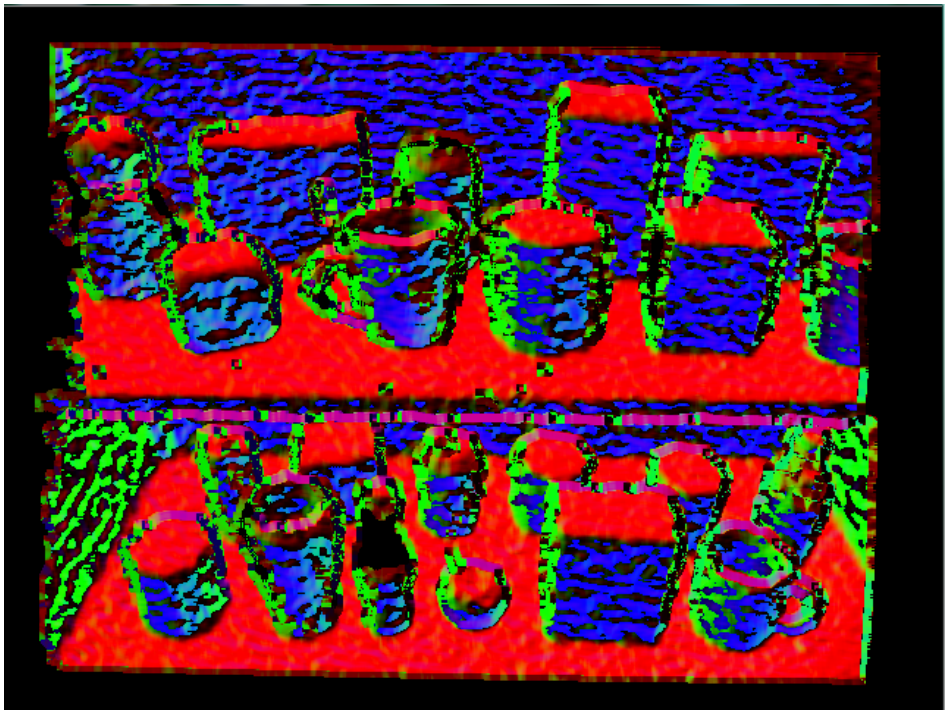
$$u = p_{i,j+1} - p_{i,j-1}$$
$$v = p_{i-1,j} - p_{i+1,j}$$

$$n_{ij} = u \times v$$

Ejemplo: Normales locales

Imágen RGB:

Normales locales:



Identificación de planos usando RANSAC

RANSAC: *RANdom SAmple Consensus*

Metodo iterativo para adecuar un modelo a un conjunto de datos.
Robusto a ruido (outliers).

Pasos:

1. Aleatoriamente escoger el mínimo número de datos para obtener un modelo.
2. Verificar los datos de todo el conjunto que se adecuan al modelo.
3. Si el número de datos es mayor a un umbral (no. de inliers). Terminar y regresar modelo.
4. Si no, regresar al paso 1, hasta cierto número de iteraciones.

RANSAC para planos

Pasos:

1. Aleatoriamente escoger el mínimo número de datos (3) de las normales locales para obtener un modelo del plano: $\pi : Ax + By + Cz + D = 0$
   - Planos grandes, puntos lejanos $d > |p_1 - p_2|, |p_1 - p_3|, |p_2 - p_3|$
   - Planos horizontales, componente n_z de la normal del plano sea mayor umbral
2. Verificar los datos de todo el conjunto que se adecuan al modelo.
   - Verificación: distancia punto-plano menor a cierto umbral $d_\pi > \frac{|Ap_{ix}+Bp_{iy}+Cp_{iz}+D|}{\sqrt{A^2+B^2+C^2}}$
3. Si el número de datos es mayor a un umbral (no. de inliers). (Refinar), (guardar modelo), (buscar más planos), Terminar y regresar modelos
   - Refinar los modelos usando PCA (x 2) [**?**].
   - Un punto del plano será el promedio de los puntos.
   - Normal del plano será eigenvector cuyo eigenvalor sea el menor.
   - Eliminar puntos del plano encontrado y continuar la búsqueda.
4. Si no, regresar al paso 1, hasta cierto número de iteraciones.

Ejemplo: Planos encontrados

Planos encotrados:

Planos segmentados de la imagen:

Segmentación de objetos

Objetos sobre los planos

- Obtener puntos por arriba de los planos.
- Se crea máscara binaria de los puntos a una distancia $d_{min} < d < d_{max}$

Objetos sobre un plano:

Máscara de objetos:

Segmentación

Separar un objeto de otro.

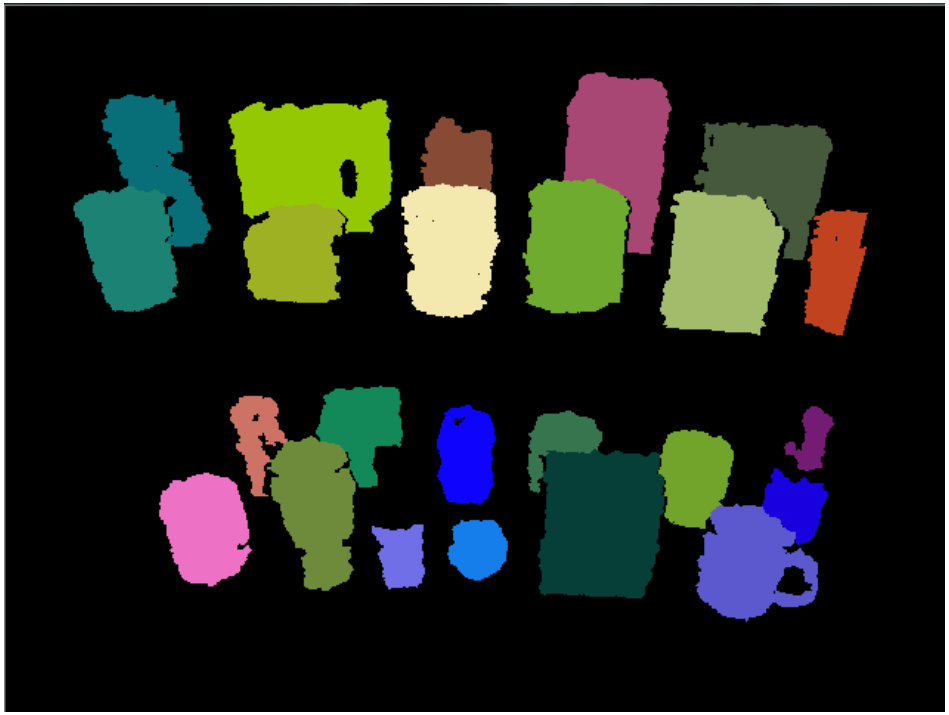
- Aduecuación de algoritmo *Connected-component labeling* para nubes de puntos organizadas.
- Conectividad 8.

Connected-component labeling:

1. Iterar sobre todos los puntos de la mascara binaría.
2. Si se encuentra uno encendido que no este etiquetado, etiquetar y meter a una cola.
3. Mientras que la cola no este vacia, obtener un punto p. Examinar sus vecinos, los que esten encendidos, no etiquetados y **que se encuentran a una distancia euclidian** $d_{umbral} > d$ **del punto** p, etiquetar (con la misma etiqueta que p) y meter a la cola.
4. Incrementar la etiqueta e ir a (2).

Ejemplo: Objetos Segmentados

Objetos etiquetados:

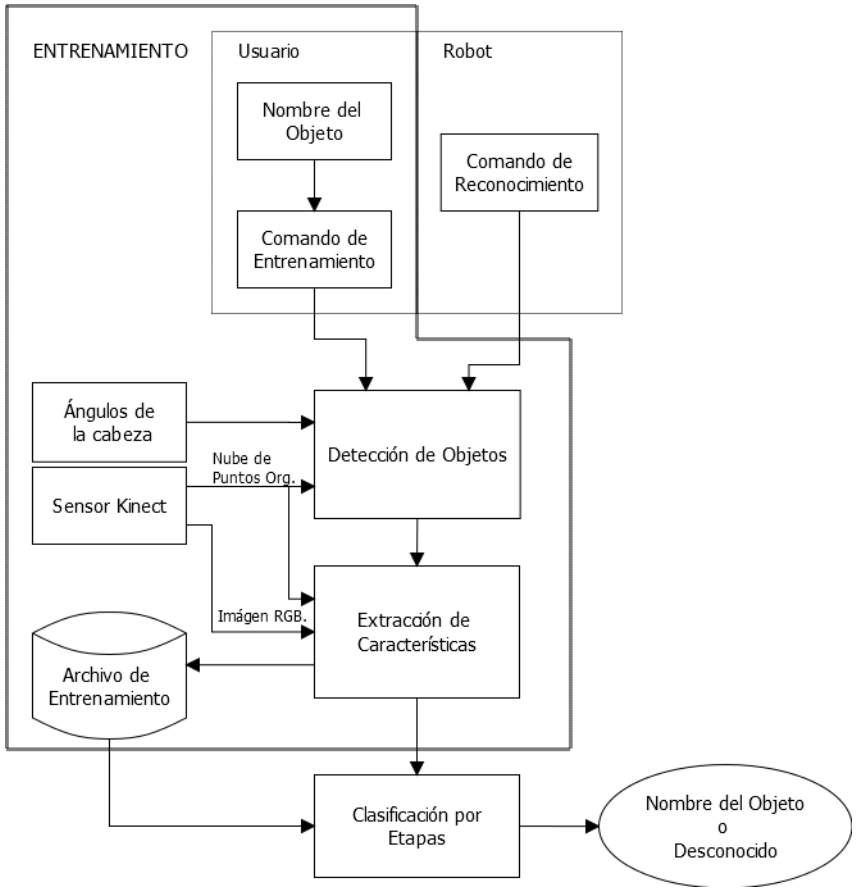


Objetos detectados:

Reconocimiento de objetos

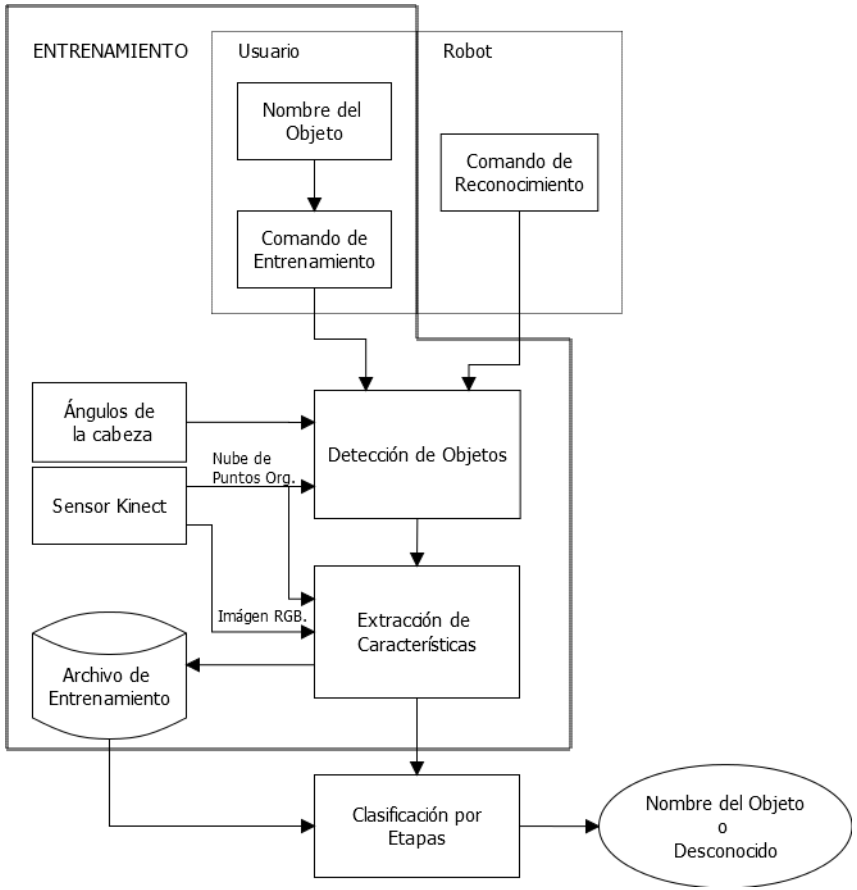
Arquitectura del sistema

- Objetos segmentados (nube de puntos y RGB).
- A los objetos detectados se les pueden extraer características.
  - Tamaño
  - Forma
  - Color
- Las características pueden ser guardadas (base de entrenamiento) con un nombre.
- Las caracteristicas pueden ser comparadas con la base de entrenamiento.



Arquitectura del sistema

- Objetos segmentados (nube de puntos y RGB).
- A los objetos detectados se les pueden extraer características.
  - Tamaño
  - Forma
  - Color
- Las características pueden ser guardadas (base de entrenamiento) con un nombre.
- Las caracteristicas pueden ser comparadas con la base de entrenamiento.



Características simples pero muy descriptivas, que tienen cualquier objeto !

Tamaño

Tamaño

Idea:

- Caja delimitadora orientada en 3D muy costosa: $\mathcal{O}(n^3 \log n)$ [**?**].
- Simplificación en 2D: rectángulo delimitador orientado [**?**] + altura.

Obtención de $V = \{w, d, h\}$:

1. Proyectar puntos 3D en el plano que lo sustenta (sólo eliminar componente z).
2. Obtener rectangulo delimitador orientado en 2D. (ancho y profundidad).
3. Obtener altura máxima de los puntos, es decir, distancia de los puntos al plano. (altura): $h = \max_{i=1\ldots n} \left(\frac{|ap_x^i + bp_y^i + cp_z^i + d|}{\sqrt{a^2+b^2+c^2}} \right)$

Error de tamaño

Error de tamaño estara dado por:

$$e_{tamaño} = |V_T - V_i|$$

Si los objetos son muy asimétricos, usar solo la altura:

$$e_{tamaño} = |h_T - h_i|$$

Forma

Forma:

Idea:

- Objetos siempre son vistos por el robot desde arriba.
- La idea nació de diferenciar cajas de cilindros. También otras formas.
- Reconocer objetos por su "sombra en el plano" (proyección).
- Describir contornos: 7 Momentos de Hu [**?**]. Invariantes a escala, rotación, traslación.

Obtención de H:

1. Proyectar puntos 3D en el plano que lo sustenta (sólo eliminar componente z).
2. Obtener envolvente convexa de los puntos en 2D.
3. Obtener Momentos de Hu H del polígono (envolvente convexa).

Ejemplo: Sombra de los objetos

Momentos de Hu

Cálculo de los momentos de Hu:

$$H[1] = \eta_{20} + \eta_{02}$$
$$H[2] = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$
$$H[3] = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - 3\eta_{03})^2$$
$$H[4] = (\eta_{30} + \eta_{21})^2 + (\eta_{21} + \eta_{03})^2$$
$$H[5] = (\eta_{30} + 3\eta_{12})(\eta_{30} + \eta 12)((\eta_{30} + \eta_{21}^2 - 3(\eta_{21} + \eta_{03}^2) + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$
$$H[6] = (\eta_{20} - \eta_{02})((\eta_{30} - \eta_{12})^2 - (\eta_{21} - \eta_{03})^2) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} - \eta_{03})$$
$$H[7] = (3\eta_{21} + \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + 3\eta_{12})^2 - 3(\eta_{21} + \eta_{03}^2)) - (\eta_{30} + 3\eta_{12}^2)(\eta_{21} + \eta_{03})(3(\eta_{31} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$

Momentos de Hu (cont.)

Momentos centralizados normalizados η_{pq} de orden $p+q$:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \quad \text{donde} \quad \gamma = \frac{p+q}{2} + 1 \quad \forall (p+q) \geq 2$$

Momentos centrales:

$$\mu_{pq} = \sum_x \sum_y (x - x_c)^p (y - y_c)^q I(x,y) \tag{1}$$

Donde $I(x,y)$ es una imagen binaria. x_c y y_c son las compoentes del centroide de la imagen:

$$x_c = \frac{\sum_x \sum_y x I(x,y)}{\sum_x \sum_y I(x,y)}$$

$$y_c = \frac{\sum_x \sum_y y I(x,y)}{\sum_x \sum_y I(x,y)}$$

Error de forma

Durante el entrenamiento, se guardan los 7 momentos de Hu de un objeto.

El error de forma e_{forma} entre dos conjuntos de momentos H_1 y H_2 esta dado por:

$$e_{forma} = \sum_{i=1}^{7} \left| \frac{1}{h_1[i]} - \frac{1}{h_2[i]} \right|$$

donde :

$$h_1[i] = sign(H_1[i]) \log H_1[i]$$
$$h_2[i] = sign(H_2[i]) \log H_2[i]$$

Color

Color

Idea:

- Color es un elemento muy distintivo de los objetos.
- RGB solo es apropiado para condiciones controladas [**?**].
- Tono (H) es apropiado para condiciones no controladas.
- Tono no describe blancos o negros. (S y V)
- Histogramas describen globalmente un objeto.

Obtención de los histogramas:

1. Pasar a espacio de color HSV.
2. Obtener Histograma del tono (H) en con alta saturación (S) y alto valor (V).
3. Contar pixeles con saturación baja (blancos) y añadirlo al histograma.
4. Contar pixeles con bajo valor (negros) y añadirlo al histograma.

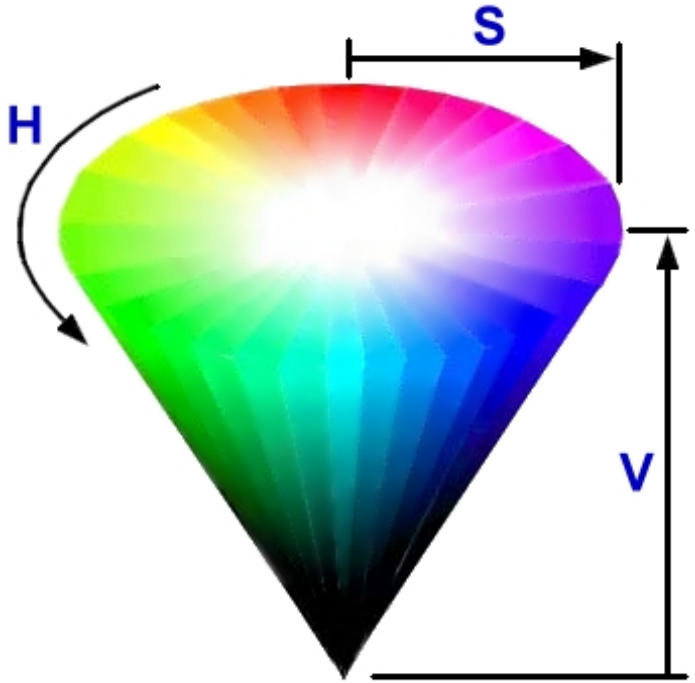
Histograma en HSV

Convertir a HSV:

$$V = \max(R, G.B)$$

$$S = \begin{cases} (V - \min(R,G,B))/V & \text{si } V \neq 0 \\ 0 & \text{si } V = 0 \end{cases}$$

$$H = \begin{cases} 60(G-B)/(V - \min(R,G,B) & \text{si } V = R \\ 120 + 60(B-R)/(V - \min(R,G,B) & \text{si } V = G \\ 240 + 60(R-G)/(V - \min(R,G,B) & \text{si } V = B \end{cases}$$



$$H = \underbrace{B_1 \cup B_2 \cup ... \cup B_N}_{H=[0,255], V=[50,255], S=[50,255]} \cup \underbrace{B_{N+1}}_{H=[0,255], V=[0,50], S=[50,255]} \cup \underbrace{B_{N+2}}_{H=[0,255], V=[50,255], S=[0,50]}$$

Error de color

Durante el entrenamiento, se guarda el histograma de un objeto.

El error se obtiene usando la *Intersección de Histogramas* propuesta por [**?**].

Para dos histrogramas I y M, con n numero de clases, el error de color e_{color} entre estos esta dado por:

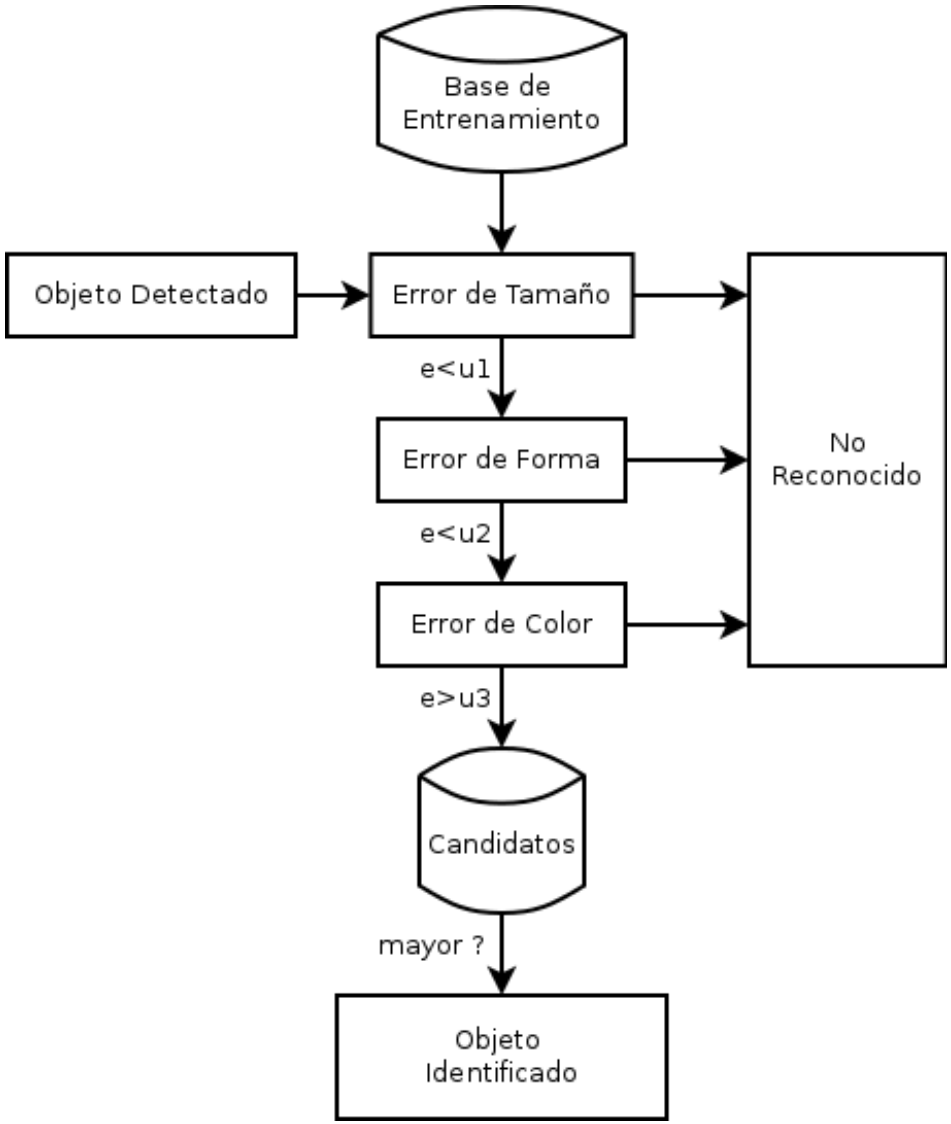
$$e_{color} = H(I, M) = \frac{\sum_{j=1}^{n} \min(I_j, M_j)}{\sum_{j=1}^{n} M_j}$$

Reconocimiento por etapas

Reconocimiento por etapas

Características:

- Combinación de los caracácteristicas: tamaño, forma y color.
- De lo más general a lo más particular.
- En cada etapa se reducen candidatos.
- Umbrales obtenidos experimentalmente.
- En la ultima etapa, se escoge el que tenga color más similar.



Experimentos y resultados

Descripción de los experimentos

Se trató de reproducir la prueba de RoCKin, moviendo y rotando el objeto entre cada toma, usando dos técnicas diferentes: el sistema propuesto y el anterior basado en SIFT.

- 15 objetos.
- 20 tomas para cada objetos.
- Distancia del robot a la mesa: 0.5 m.
- Ángulo de cabeceo: -40 deg.
- Angulo de guiñada: 0 deg.
- Altura de la mesa: 1.2 m.

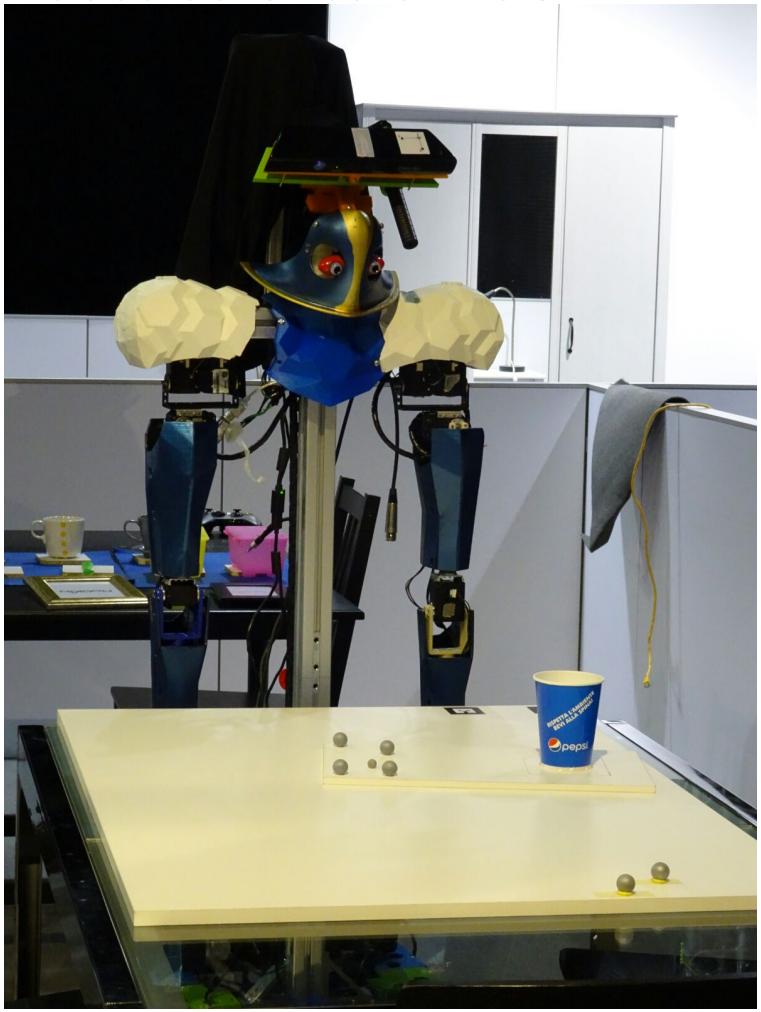
Objetos utilizados

Grupos propuestos :

- Dos portaretratos de diferente tamaño y color. (Objetos 6 y 7)
- Dos recipientes de la misma forma pero diferente color. (Objetos 14 y 15)
- Tres cajas de la misma forma pero diferente color. (Objetos 3, 4, y 5)
- Tres botes de color y forma similar pero diferente tamaño. (Objetos 1,2 y 8)
- Tres tazas del tamaño y forma similar pero diferente color. (Objetos 11, 12 y 13)
- Dos objetos del mismo color pero diferente forma y tamaño. (Objetos 9 y 10)

Ejemplo: Experimentos

Robot durante el RoCKiN:



Robot durante las pruebas:

Exp 1.- Sistema propuesto: Matriz de confusión

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	NR
1	14		-	-	-	-	-	-	-	-	-	-	-	-	-	6
2	-	20	-	-	-	-	-	-	-	-	-	-	-	-	-	0
3	-	-	20	-	-	-	-	-	-	-	-	-	-	-	-	0
4	-	-	-	20	-	-	-	-	-	-	-	-	-	-	-	0
5	-	-	-	-	19	-	-	-	-	-	-	-	-	-	-	1
6	-	-	-	-	-	17	-	-	-	-	-	-	-	-	-	3
7	-	-	-	-	-	-	10	-	-	-	-	-	-	-	-	10
8	-	-	-	-	-	-	-	20	-	-	-	-	-	-	-	0
9	-	-	-	-	-	-	-	-	18	-	-	-	-	-	-	2
10	-	-	-	-	-	-	-	-	-	19	-	-	-	-	-	1
11	-	-	-	-	-	-	-	-	-	-	20	-	-	-	-	0
12	-	-	-	-	-	-	-	-	-	-	-	20	-	-	-	0
13	-	-	-	-	-	-	-	-	-	-	3	-	17	-	-	0
14	-	-	-	-	-	-	-	-	-	-	-	-	-	20	-	0
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	20	0

Exp 1.- Sistema propuesto: Resultados

Objeto	% Reconocido	% No reconocidos	% Errores
(1) Bote azul	70	30	0
(2) Bote blanco	100	0	0
(3) Caja azul	100	0	0
(4) Caja café	100	0	0
(5) Caja verde	95	5	0
(6) Portaretratos café	85	15	0
(7) Portaretratos dorado	50	50	0
(8) Lata plateada	100	0	0
(9) Plato verde	90	10	0
(10) Shampoo	95	5	0
(11) Taza crema	100	0	0
(12) Taza negra	100	0	0
(13) Taza roja	85	0	15
(14) Tupper azul	100	0	0
(15) Tupper naranja	100	0	0
TOTAL	**91.333**	**7.666**	**1.000**

$$\tau_{avg} = 0.283[s]$$

Exp 2.- Sistema basado en SIFT: Matriz de confusión

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	NR
1	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	20
2	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	20
3	-	-	18	-	-	-	-	-	-	-	-	-	-	-	-	2
4	-	-	-	20	-	-	-	-	-	-	-	-	-	-	-	0
5	-	-	-	-	16	-	-	-	-	-	-	-	-	-	-	4
6	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	20
7	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	19
8	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	15
9	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	20
10	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-	20
11	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	20
12	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	18
13	-	-	-	-	-	-	-	-	-	-	-	-	4	-	-	16
14	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	20
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	20

Exp 2.- Sistema basado en SIFT: Resultados

Objeto	% Reconocido	% No reconocidos	% Errores
(1) Bote azul	0	100	0
(2) Bote blanco	0	100	0
(3) Caja azul	90	10	0
(4) Caja café	100	0	0
(5) Caja verde	85	15	0
(6) Portaretratos café	0	100	0
(7) Portaretratos dorado	5	95	0
(8) Lata plateada	25	75	0
(9) Plato verde	0	100	0
(10) Shampoo	0	100	0
(11) Taza crema	0	100	0
(12) Taza negra	10	90	0
(13) Taza roja	20	80	0
(14) Tupper azul	0	100	0
(15) Tupper naranja	0	100	0
TOTAL	**22.33333333**	**77.66666667**	**0**

$$\tau_{avg} = 0.903[s]$$

Análisis de los resultados

- Menor tiempo de ejecución.
- Mayor porcentaje de reconocimiento.
- Más del 50% de los objetos, 8 de 5, se reconocieron el 100% de las veces.
- Mayoria de los objetos tinenen una taza de reconocimiento mayor al 85
- Presentó errores (falsos negativos), mientras que SIFT no.
- Errores debido a mala segmentación (Caja verde y plato verde)
- Errores debido al color: 3 (taza crema)
- Error de forma: 10 (portaretratos)

Conclusiones y trabajo futuro

Conlcusiones

- Se logro diseñar un algoritmo para detectar objetos usando solamente la nube de puntos organizada.
- Se logro modificar RANSAC para obtener planos de manera rápida usando diferentes heurísticas.
- Segmentación de objetos aún muy cercanos sin creación de árboles k-dimensionales. (rápido)
- Se consiguió aumentar la taza de reconocimiento casi 4 veces en objetos con poca o nula textura, reduciendo el tiempo de ejecución en un 32%.
- Se propuso una nueva técnica basada en 3D para reconocer objetos a partir del contorno de las proyecciónes de los puntos.
- Se comprobó la efectividad de la Intersección de Histogrmas para identificar dos objetos.
- Los objetos deben de estar muy bien segmentados para utilizar el reconocedor propuesto.

Trabajo a futuro

- Usar Kinect 2. Menos ruido y más resolución. No se espera una incremento considerable de los tiempos de ejecución.
- Refinar algoritmo de segmentación para objetos concavos.
- Detectar objetos no sólo en planos horizontales.
- Probar diferentes métodos para comparar contornos y figuras.
- Proyectar los puntos de los objetos en otros planos (p.e. caras de la caja delimitadora).

Video

Video mostrando el sistema en ejecución.

Gracias.