Computer Vision - IPD421

Felipe Vera

9 de julio de 2014

1. Resumen del curso

Está enfocado más en investigación que en desarrollo de problemas, de modo de saber *interpretar* un artículo científico relacionado con Visión por Computadora.

1.1. Temario

1. **Introducción y estructura general del análisis de video**
2. **Segmentación de movimiento y actualización de fondo**: Segmentar la imagen de modo de obtener la información interesante.
3. **Representación de objetos**
4. **Seguimiento (tracking)**
5. **Geometría Proyectiva y stereo vision**
6. (Si es que queda tiempo) **Reconocimiento y aprendizaje de eventos**: Está basada más que nada en la experiencia personal del profesor.

Los más fundamentales son los puntos 1-4 para el desarrollo de algoritmos. Geometría Proyectiva es necesario para inferir información desde imágenes tridimensionales.

1.2. Evaluación

- **Certamen 1 (20 %)** - 6 mayo 2014
- **Certamen 2 (20 %)** - 4 junio 2014
- **Certamen 3 (20 %)** - 16 julio 2014

- **Desarrollo de algoritmo (25 %)**
  - Elegir algún algoritmo descrito en un paper.
  - Desarrollar un poster (un esquema en donde, en un papel gigante se describe el proyecto) del algoritmo estudiado.
  - **Presentación** del algoritmo a desarrollar - 7 de mayo
  - Presentaciones finales y entrega de código, junto con un manual de usuario para programadores y descripción de dificultades prácticas de desarrollo - 17 de julio
- **Clase (15 %)**
  - Es una clase preparada por alumnos de 40 minutos + 5 minutos de preguntas. Se propone algún paper a libre elección.
  - **1,2,8 y 9 de julio**

1.3. Material

- Grupo Google: `http://groups.google.com/group/CV-2014-1`. Para incorporar, enviar un correo a `marcos.zuniga@gmail.com`
- Página profesor: `http://profesores.elo.utfsm.cl/~mzuniga/CV-2014-1`
- Texto Guía: *Learning Computer Vision with OpenCV Library - Gary Bradsky, Adrian Kaebier*

1.4. Proyecto tentativo

En las canchas de todo el mundo está el interés de conocer el rendimiento global e individual de los jugadores de un equipo. Los problemas son:

- Detección del balón
- Reconocimiento de jugadores
- Cambio repentino de luminosidad (Nubes, inclinación del sol)
- Sombras y movimiento

2. Visión general

Def. Computer Vision Área de la Inteligencia Artificial centrada en el procesamiento y análisis de la información obtenida a través de medios visuales.

Es un área multidisciplinaria.

- **Procesamiento de imágenes**: Capturar movimiento, reconocimiento, reconstrucción, captura de características y elementos de base.
- **Reconocimiento de patrones**: Modelado y reconocimiento de elementos.
- **Aprendizaje automático**: Aprendizaje de modelos y redes neuronales.

2.0.0.1. Sobre el tiempo real Muchas veces el tiempo de respuesta es un GRAN aspecto a considerar. Incluso muchas veces, como en aplicaciones en línea o de respuestas, se requiere una restricción de **tiempo real.** Esto define que el tiempo de respuesta sea apenas perceptible por una persona (espacio de fracciones de segundo)

2.1. Algunos papers

2.1.0.2. Sobre análisis de video Hay algunos papers de reconocimiento de movimiento y de posturas ("*Applying 3D Human Model in a Posture Recognition System*")

2.1.0.3. Sobre aprendizaje de eventos Se emplean Cadenas de Markov para poder detectar control de tránsito ("*Abnormal Event Detection from Surveillance...*"). No se va a ver mucho en este curso, son muy poco flexibles.

2.2. Motivación

La visión humana es MUY buena para procesar y obtener información. El cerebro humano procesa aproximadamente 60 cuadros por segundo, de aproximadamente un millón de puntos por imagen, y la mitad del cerebro está dedicado a procesar esta información visual. La finalidad es emular la visión humana, pero hay muchos vacíos y falta mucha investigación en ese campo.

- **Identificación y biometría** De características únicas (rostros, huellas digitales, forma de caminar...)
- **Análisis de comportamiento** Tanto individual como grupal (*crowd*)
- **Cuidado de personas a distancia** Detección de comportamientos anormales y modelado de un comportamiento normal.

- **Videovigilancia asistida** Como muchas aplicaciones actuales dependen de la concentración del operador, se necesita aquí en detección de comportamientos anormales, control de tráfico, acceso, conteo de personas, etc.
- **Interactividad** Para nuevos equipos de entretenimiento, entre otras cosas.

2.2.1. AVITRACK

Corresponde a un sistema de vigilancia asistida en aeropuertos, el cual genera alarmas si hay situaciones no deseadas (como un avión colisionando con un vehículo de carga). Dado que los procedimientos son muy estandarizados y siguen protocolos, el modelo es relativamente sencillo de implementar.

2.2.2. CASSIOPEE

Videovigilancia asistida en agencias bancarias que activa alarmas en situaciones predefinidas de robo (el modelo activa la alarma si dos personas van demasiado juntas al lugar de la caja fuerte)

El programa se confunde bastante en reconocimiento, pero con la información que necesitaba y estimaba bien, acertaba situaciones de robo 8 de cada 10 veces.

2.2.3. GERHOME

Es un programa de cuidado de ancianos a distancia, que genera alertas en caso de comportamientos anormales si el anciano se encuentra en una situación de riesgo (inmobilidad).

3. Tareas típicas

- **Segmentación de movimientos**: Detección de clusters/blobs de pixeles a partir de la imagen que capta una cámara.
- **Clasificación e identificación**: Detección de los blobs que realmente interesen en el análisis, e identificación del resto para entrar en algún modelo.
- **Seguimiento de objetos**: Asociar estos blobs a algún modelo creado previamente.
- **Reconocimiento de eventos**: Reconocer comportamiento y actividades que realizan los objetos.

3.1. Segmentación

La segmentación consiste en particionar una imagen en múltiples segmentos (conjuntos de píxeles)

Consiste en segmentar las regiones que corresponden a objetos móviles. A priori se hace necesario modelar qué es fondo y qué es primer plano. Hay que tener atención de que los errores producidos en esta etapa pueden convertirse en errores de arrastre que impida un buen rendimiento de los pasos que requieran imágenes segmentadas. Este paso separa al **primer plano (*foreground*)** del **segundo plano (*background*)**

3.1.1. Problemas

- Cambios de iluminación (luz que entra por una ventana, focos fluorescentes en mal estado por ejemplo)
- Sombras.
- "Fantasmas" (cambios permanentes en el fondo, que obligan a cambiar el modelo aprendido)
- Árboles en movimiento, bajo contraste, ocultación estática y errores en los dispositivos.

3.2. Modelamiento y clasificación

- A más complejidad del modelo, mejor precisión pero más lentitud de cálculo.
- Los modelos complejos son menos generales, más específicos en los objetos que se quieren identificar, pero favorecen la identificación de los objetos. Los simples son más rápidos de calcular pero más generales, al poder representar a más de un tipo de objeto.
- Son concatenables. A partir de un modelo general se puede desembocar a uno más específico.

3.2.1. Algunos modelos importantes

- **Modelo de punto:** Aplicable a objetos pequeños, de unos pocos píxeles de radio. Son muy rápidos de obtener.
- **Modelo de forma 2D:** Son de baja precisión y calidad descriptiva, pero son rápidos de obtener y pueden representar a más de un tipo de objetos.
- **Modelo de forma 3D:** Tienen una precisión y calidad media. Su velocidad de cálculo es media y pueden representar a más de un tipo.
- **Articulado:** Es muy específico para un tipo, pero son muy precisos y tienen mucha calidad (ej: esqueleto de una persona) - *Ver: "Modelo humano con 23 parámetros"*
- **Representación de controno:** Reconocimiento específico para un tipo y postura, y es una representación 2D, sin embargo es de una precisión y calidad alta; y es adaptable al objeto.
- **Clasificadores:** Consta en entrenar un clasificador, utilizando un conjunto de entrenamiento. Es bastante rápido de calcular pero el proceso de entrenamiento puede ser lento.
- **Descriptores locales:** Busca características elementales en una vecindad limitada. Es demasiado lenta si se buscan demasiadas características, pero puede ser bastante robusto y rápido si se busca sólo la cantidad necesaria de características.
  - SIFT (determinístico)
  - SURF (determinístico)
  - FERNS (aleatorio)

3.3. Seguimiento de objetos

Consiste en asociar un objeto a un objeto dinámico a una etiqueta y lograr hacer que dicha etiqueta permanezca en los siguientes cuadros. Hay distintos problemas y soluciones a estos.

- Ocultación dinámica (que un objeto se cruce al frente de otro)
- Seguimiento multiobjeto.
- Tiempos de respuesta.

Las soluciones propuestas son las siguientes:

- Manejo multihipótesis, pueden o no estar basadas en la información pasada del objeto.
- Actualización de los modelos para los atributos.
- Particle filtering (muestreo Montecarlo, condensation...)

3.4. Reconocimiento y aprendizaje de eventos

Es un mundo en sí mismo (*machine learning*)

3.4.0.1. Def: Estado Un conjunto de atributos válido en un instante dado o estable en un intervalo de tiempo.

3.4.0.2. Def: Evento Transiciones entre estados.

3.4.0.3. Def: Evento compuesto Combina estados y eventos.
Los problemas es cruzar el puente entre la información numérica a conceptos abstractos.

- Interacción entre objetos.
- Concepción de métodos generales.

Algunas soluciones.

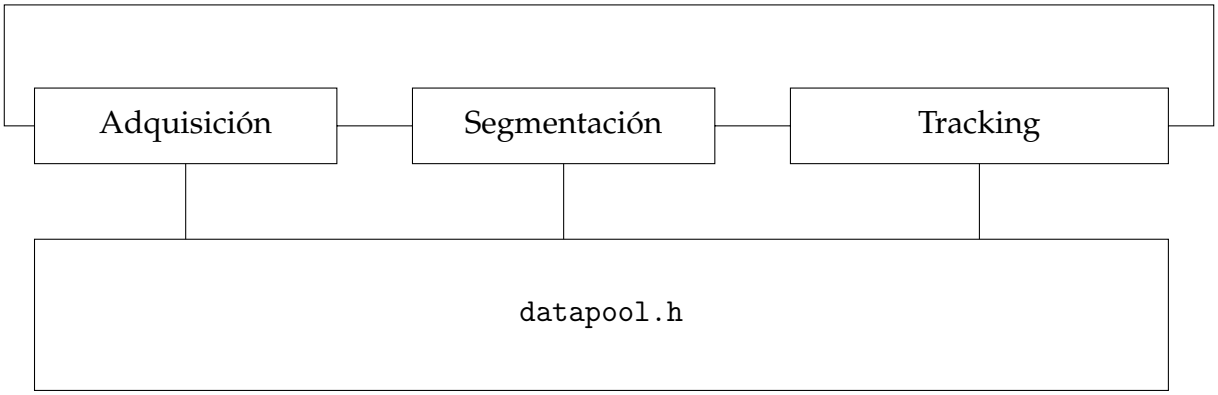
- Aprendizaje de eventos (Redes bayesianas dinámicas (DBN), cadenas de Markov ocultas (HMM), etc.)
- Modelos de evento pre-definidos.

3.5. Trabajo actual

- Utilizar un *benchmarking* más objetivo que el ojo humano para comparar la fiabilidad de los algoritmos.
- Eliminar la brecha entre lo objetivo y lo subjetivo (numérico $\rightarrow$ conceptual)
- Mejoramiento del rendimiento en tiempo de ejecución.
- Generalizar una solución de problemas de segmentación.
- Hacer menos indispensable el conocer muy a priori modelos de clasificación.
- Resolver el GRAN problema de **ocultación dinámica** (objetos que se ocultan uno detrás de otro)
- Mejoramiento de métodos para generación y control de hipótesis.
- El marco teórico de *Machine Learning*.

3.6. VAT: Video Analysis Tool

Es una herramienta bastante potente para Computer Vision basado en QT. Eso permite recoger y depositar datos en un archivo cabecera llamado `datapool.h`. Estos datos corresponden a y para **adquisición, segmentación y seguimiento.** Funciona así:



3.6.1. Flujo

1. **Al partir**, el programa llama a un `setParameters` global, el cual llama iterativamente al listado de `setParameters` para cada módulo de visualización. Los parámetros se pueden pasar mediante un archivo de configuración XML o mediante macros, de modo que no se vuelva a recompilar el programa cada vez que se necesita usar. Ver cada XML para cada detalle.
2. **Inicialización**, ejecuta las rutinas `init` de cada uno de los módulos.
3. **Execute**, ejecuta las rutinas `run` de cada módulo en distintas hebras. Es cíclica, y a menos de que se envíe una señal, continúa corriendo.

3.6.2. Curiosidades

- `datapool.h` contiene referencias a la imagen actual, la anterior, en código Gray, a un fondo, una máscara a gusto, dos imágenes diferenciales... Contiene un mapa de probabilidades, vector de blobs, objetos del módulo de seguimiento

3.6.3. Customizando

Para crear un módulo, simplemente se crea una clase que herede de `ModuleInterface`, se implementa su método `setParameters`, su `init` y `run` de solo ESE módulo, para no recompilar todo el código de nuevo.

3.7. Ground Truth

Es la información que se obtiene "al ojo", cuadro por cuadro, que representa el resultado esperado del ejecución de algoritmos de Computer Vision, y se usa como base de comparación. Con eso se permite hacer que una persona realice, por ejemplo, una segmentación a mano. De dicho modo se puede usar mediciones estadísticas de error, o si la información es cualitativa hay cuatro evaluaciones:

- **Verdadero-positivo (TP)**: Es un acierto del algorimo, detectando una situación que también está marcada en la *Ground Truth*.
- **Falso-negativo (FN)**: Al algoritmo se le pasó desapercibida una situación indicada en la *Ground Truth*.
- **Falso-positivo (FP)**: El algoritmo creyó haber detectado una situación que en realidad no existe en la *Ground Truth*.
- **Verdadero-negativo (TN)**: Otro acierto del algoitmo, no detectando una situación que tampoco está marcada.

3.7.1. Medición de efectividad

3.7.1.1. Cualitativo

- **Precisión** (Lo correcto v/s lo equivocado de lo detectado): $\frac{\text{TP}}{\text{TP} + \text{FP}}$
- **Sensitividad** (Lo detectado v/s lo no detectado de lo que se debería detectar): $\frac{\text{TP}}{\text{TP} + \text{FN}}$

3.7.2. Matriz de Confusión

Se usa en Machine Learning aplicado, se aplica en algorimos de clasificación binaria. La forma de evaluarla es una matriz cuadrada, de modo que si en las diagonales los resultados son mayores, el algoritmo lo está haciendo bien. Funciona de la siguiente manera. Columnas: Ground truth, Filas: algoritmo.

Alg — GT	Clase 1	Clase 2	Clase 3
Clase 1	80	10	3
Clase 2	11	76	6
Clase 3	17	21	92

En otras palabras, muestra en qué manera el algoritmo confunde objetos de una clase 1 con una clase 2, entre otras.

3.7.3. Curvas ROC

Son una curva entre Falsos positivos y verdaderos positivos aplicada a clasificadores binarios. Estos clasificadores van cambiando su umbral.

4. Segmentación

Una segmentación consiste en particionar una imagen en distintos conjuntos de pixeles.

4.1. Segmentación de movimiento: Separación background (BG) con primer plano (FG)

La **segmentación de movimiento** separa los pixeles que se mueven del resto de la imagen. Esto debería dar como resultado una zona blanca donde ocurre movimiento y una zona negra donde no ocurre. Esto es útil para diferenciar el segundo plano (*background*) del primer plano (*foreground*).

Lo problemático del asunto es la ambigüedad de las definiciones. El fondo es un concepto *ill-defined*. Es decir, depende de la aplicación, es decir que si en una algoritmo algún objeto se considera como primer plano, el otro se deberá considerar como segundo plano.

Sin embargo, segmentar "lo que se mueve con lo que no se mueve" no es la solución perfecta para segmentar, si es que, por ejemplo, hay una bandera, árboles o cosas que se muevan con el viento, por dar un ejemplo.

Al menos la segmentación funciona como un primer filtro. Es mejor tener algunos falsos positivos que se puedan filtrar en etapas posteriores que descartar datos correspondientes a Falsos verdaderos en esta etapa.

4.1.1. Método de umbral

Son una forma sencilla de separar información de imagen. Un ejemplo sencillo es una pelota roja en una mesa verde: Convertimos la información RGB a HSV (Matiz-Saturación-Valor), y aplicamos un umbral alrededor del color rojo (340^o a 20^o), de modo de que la parte roja se convierta en blanco y el resto en negro.

Luego se ejecuta un algoritmo para detectar qué píxeles están conectados (tomo un pixel al azar de la pelota y voy dilatándolo) y se tiene la pelota roja como un objeto.

4.1.2. Umbral con histéresis

Se usa también histéresis. Un pixel de FG puede bajar a BG con umbral de histéresis menor al cual subió.

4.1.3. Actualización mediante tasa de aprendizaje

Es un método antiguo de actualización de fondo. (*Heikkila & Silven, 1999*). Actualiza la información a una tasa de aprendizaje $\alpha \in [0, 1]$, en donde B_t es el fondo en el instante t, e I_t significa la imagen en el instante t.

$$Bg_{t+1} = \alpha I_t + (1 - \alpha) Bg_t$$

Generalmente se usa una máscara para ocultar en I_t píxeles que se muevan mucho.

4.1.3.1. Función en OpenCV

```
void cvRunningAvg (const CvArr* image, CvArr* acc, double alpha,
                   const CvArr* mask = NULL);
```

4.1.4. Ventana temporal

Es una técnica de *Cutler y Davis* (1998). Define si un pixel está en movimiento si:

$$\sum_{C \in \{R,G,B\}} |I_t(C) - Bg_t(C)| > K_\sigma$$

σ es una estimación offline de la desviación estándar del ruido y K es una constante de "perilleo".

4.1.5. Algoritmo Wallflower

Definen los problemas a resolver por una actualización de fondo adecuada.

- **Moved objects**: Objetos de fondo movido
- **Time of day / Light switch**: Cambios en la iluminación graduales/repentinos, respectivamente.
- **Waving trees**: Fondos con movimiento irrelevante (ej: banderas, árboles que se mueven...)
- **Camouflage**: Pixeles de objeto que pueden confundirse con el fondo al ser del mismo color.
- **Bootstrapping**: No se dispone de un tiempo de entrenamiento.
- **Foreground aperture**: Al tener un fondo con demasiado contraste, puede haber dificultad en detectar un foreground.
- **Sleeping person**
- **Walking person**
- **Shadows**

Wallflower trabaja en niveles de pixel, región y cuadro.

4.1.5.1. Nivel Pixel Genera predicciones lineales usando el método de Mínimo Error Cuadrado (filtro de Wiener). Esta predicción es guardada en un Historial de Predicciones. Genera dos predicciones, una a partir del historial y otra a partir de las imágenes reales. Si el píxel es cercano a alguna de estas predicciones, se considera BG.

4.1.5.2. Nivel Región Considera 5 pasos:

1. Calcula umbral de diferencias: $J_t(x) = 1$, si $|I_t(x) - I_{t-1}(x)| > T$.
2. Se asegura usando historial, para asegurar el fondo (corto plazo)
3. Encontrar regiones 4-conectadas R_i (conectadas en vecindad $\leftrightarrow, \updownarrow$), de modo de tener superficies lo suficientemente grandes de pixeles moviéndose.
4. Calcula un histograma normalizado H_i de cada R_i respecto a la imagen I_{t-1}

$$H_i(s) = \frac{|\{x : x \in R_i \ \wedge \ I_{t-1}(x) = s\}|}{|R_i|}$$

5. Para cada R_i computa $Fg_{t-1} \wedge R_i$ y para cada punto de intersección hace crecer las regiones 4-conectadas L_i y las agrega a F_t.

$$L_i(x) = \begin{cases} 1 & \text{si } H_i(I_t(x)) > \epsilon \\ 0 & \text{En otros casos} \end{cases}$$

Se puede usar $T = 16$, $k_{\text{mín}} = 8$ y $\epsilon = 0{,}1$

4.1.5.3. Nivel Cuadro Se emplea una fase de entrenamiento con K-Means para crear modelos de fondo, aplicado a la imagen actual. Se escoge la que produzca menos FG. Esto se hace solo con cambios drásticos en la imagen ($> 70\,\%$ de pixeles totales).

4.1.5.4. Interpretación Observa cambios drásticos en grandes porciones de la imagen. Se supone una baja cantidad de píxeles en movimiento, de modo que se entrena con k-means, y se escoge un fondo que produzca menos píxeles de *foreground* (primer plano)

4.1.6. Mixture of Gaussians

La particularidad es que cada píxel es modelado como una mezcla entre K gaussianas. Generalmente se usa $K \in \{3, 4, 5\}$. Esto debido a que el fondo no necesariamente es estable y varía sus valores en el tiempo. La idea es que cada píxel tenga un valor independiende de los demás.

4.1.6.1. Caso monocromático Se define la probabilidad de obtener un valor x en un píxel en el tiempo t como:

$$\mathbb{P}[x_t] = \sum_{i=1}^{k} w_{i,t} \cdot \mathbb{P}[\text{Norm}(\mu_{i,t}, \Sigma_{i,t}) = x_t]$$

Los parámetros de las gaussianas $\mu_{i,t}$ (media de una gaussiana de un píxel), y $\mu_{i,t}$ (matriz de covarianza de las gaussianas) se fijan en una fase de entrenamiento. Por simplicidad asumiremos que cada píxel es una variable independiente, por lo que la matriz de covarianza es cero excepto en su diagonal. Es decir:

$$\Sigma_{i,t} = \sigma_i^2 \cdot \mathbb{I}$$

Estas gaussianas se ordenan de acuerdo a su cuociente $b = \frac{w}{\sigma}$

4.1.6.2. Movimiento Un píxel se considera en movimiento si su diferencia con todas las gaussianas supera su tolerancia de desviaciones estándares. Es decir:

$$|x - \mu_i| > m \cdot \sigma_i$$

m es una constante de perilleo. Empíricamente la mejor elección es $m = 2{,}5$.

4.1.6.3. Actualización del modelo Y si el píxel no "se mueve", la gaussiana i más cercana al píxel es actualizada, usando una variable de memoria α (menor a 0,01).

$$\mu_{i,t} = (1-\alpha)\mu_{i,t-1} + \alpha x_t$$
$$\sigma_{i,t}^2 = (1-\alpha)\sigma_{i,t-1}^2 + \alpha(x_t - \mu_{i,t})^T(x_t - \mu_{i,t})$$
$$w_{i,t} = (1-\alpha)w_{i,t-1} + \alpha M_{i,t}$$

Usando $M_{i,t} = 1$ si x_t coincide con la distribución i. 0 en caso contrario.

4.1.7. Kernel Density Estimation

El foco es manejar píxeles altamente variables (como árboles en movimiento). Esto usa el modelo de estimación de densidad de Kernel, modelando con algún kernel con distribución de probabilidad.

4.1.7.1. El Kernel Su integral definida entre $-\infty$ y ∞ es 1 (distribución de probabilidad) y debe ser simétrica. Al final es muy similar a Mixture of Gaussians, pero más caro y más preciso.

4.1.8. Background Averaging (usado en OpenCV)

Es un modelo aprendido en los primeros cuadros para aceptar píxeles de fondo en un intervalo dado por la varianza de los pixeles. Corresponde a la función `accumulateBackground`.

4.1.9. CodeBook (usado en OpenCV)

Optan por usar el espacio YUV dado que es el brillo el que varía más en el fondo que el color. Se crean *codebooks* que simbolizan un rango en donde se mueven las intensidades de cada píxel, y finalmente, estos son análogos a las gaussianas, pero al tratarse de intervalos y no Gaussianas propiamente tales, se hace mucho más barato de implementar. Finalmente estos *codebooks* terminan siendo aprendidos para dar un modelo de fondos en movimiento.

4.2. Componentes Conectados

Se obtiene una imagen binaria como resultado del procedimiento anterior, por lo que ahora se debe agrupar la información obtenida. Los más populares son:

- Two-pass
- One-pass
- RLE Segmentation.

4.2.1. Two-pass

Se recorre de arriba-abajo, izquierda-derecha todos los píxeles de la imagen. Si no es un píxel de FG, se busca los vecinos etiquetados y se copia dicha etiqueta (en caso de haber varios se copia la más baja). Si no hay ninguno se crea una nueva. Si lo hay, se copia la etiqueta; y si hay un vecino con una etiqueta distinta, se guarda en alguna parte que ambas etiquetas son equivalentes. Luego se repite el procedimiento, reetiquetando las etiquetas equivalentes.

4.2.2. One-pass

El costo con respecto al anterior, depende del tipo de imagen. Pasa solo una vez por la imagen, etiquetando todo. Es similar a la búsqueda de componentes conectados usando Morfología.

4.2.3. Método RLE

Se convierte la imagen a representación RLE, leyendo corridas de píxeles (offset + cantidad de píxeles juntos). Las corridas de píxeles se denotan por cada fila, y luego se hace análisis, en lugar de píxeles, por segmentos.

5. Representación de objetos

5.1. Como puntos

5.2. Como rectángulos 2D

5.3. Como cajas 3D

5.4. Como modelos articulado

La velocidad de proceso es muy baja y específica, pero puede alcanzar una muy buena precisión. Boulay en 2006 modeló una persona con 23 parámetros. Primero se debe situar al modelo en alguna posición 3D, y se compara el blob obtenido en la imagen con una colección predefinida de distintas posturas 3D (avatares de postura 3D). Luego se valida la coherencia temporal de esta silueta.

Al principio del aprendizaje se modela cada parte del cuerpo para armar siluetas a partir de ellas, graficándolas en 3D.

5.5. Esqueleto

Consiste en un modelo que permite separar un modelo en articulaciones y ejes. Para ello se debe obtener un borde.

5.5.0.1. Medial Axis Transform Transformada de eje central, una forma de obtener esqueleto.

5.6. Representación de contorno

Sirven para definir el borde del objeto con mejor precisión que los rectángulos 2D. La complejidad depende del objeto y método usado, y su precisión y calidad son altas; pero es muy específico para un tipo de objetos. Contorno de personas: Yilmaz et al., 2004 y Seguimiento de contornos: Shi & Karl, 2005.

Es compatible con un proceso de aprendizaje, para **modelar de forma estadística**.

5.7. PCA (Principal Component Analysis)

Para encontrar los ejes principales de una nube de puntos. Esto sirve para que las variables involucradas sean descorrelacionadas y tratadas como variables independientes. Con ello se puede minimizar la cantidad de parámetros a usar para representar modelos.

- Covarianza de dos coordenadas: Permite discernir si los puntos se pueden representar aproximadamente por una curva.

- Autovalores y autovectores: Recordar que el principio matemático $(A - I\lambda)\vec{x} = 0$ describe vectores $\vec{x} \neq \vec{0}$ que no son afectados por rotación, si es que A es una matriz de rotación y λ es algún escalar.

¿Pero qué es? Es una transformación lineal que permite pasar de un sistema (x, y) a otros tipos de coordenadas ortogonales para que estas estén ordenadas según la varianza de la proyección de los datos en cada eje.

Cuando la nube de puntos tienen poca correlación, PCA no es necesario dado que PCA es simplemente un descorrelacionador.

5.8. Active Shape Models

Estos contornos se van moldeando para ajustarse a la forma de una imagen. Se toma un vértice y la perpendicular, buscando el borde más fuerte en dicha línea. Esa permite reducir la operación de búsqueda a una sola dimensión. Es un modelo iterativo, buscando tras varias iteraciones el mejor ajuste del modelo de forma.

5.9. Snakes o Modelos de contorno activo

Busca una forma de *minimizar la energía* asociada al contorno actual. Tiene menor energía cuando está en los bordes.

Mezclando modelos de contorno y articulado es posible modelar a una persona caminando de lado.

6. Puntos de interés

6.1. Esquinas de Harris

Caracteriza según cambios de intensidad alguna región en particular. Al necesitar las derivadas parciales de x e y de las imágenes, se necesitan filtros de Prewitt y Sobel. Cada vez más lejos del origen los pesos $w_{u,v}$ van a hacerse más pequeños, dado que $w_{u,v}$ corresponde a una ventana.

La Matriz usada (similar a una matriz de covarianza) es, dada una imagen I se computa:

$$M = \begin{bmatrix} \sum_{u,v} w_{u,v} \cdot I_x^2(x_r, y_r) & \sum_{u,v} w_{u,v} \cdot I_x(x_r, y_r) I_y(x_r, y_r) \\ \sum_{u,v} w_{u,v} \cdot I_x(x_r, y_r) I_y(x_r, y_r) & \sum_{u,v} w_{u,v} \cdot I_y^2(x_r, y_r) \end{bmatrix}$$

Sean I_x, I_y derivadas paraciales de la imagen respecto a x y a y. $w_{u,v}$ es una función ventana (norma 1: integral en todo $\mathbb{R}^2 = 1$). Se sugiere que sea Gaussiana, es decir:

$$w_{u,v} = e^{-\frac{u^2+v^2}{2\sigma^2}}$$

6.2. Good Features to Track (Shi-Tomasi)

Se debe usar la matriz de Harris y Valores propios. Es bastante similar a Harris.

6.2.1. Diferencia de Gaussianas

Se elabora imágenes de diferencia a medida que se aplica filtros de suavizado para encontrar bordes.

6.2.1.1. Scale space Es la base teórica de la diferencia de Gaussianas. Se tiene una familia de imágenes que se convoluciona con un filtro gaussiano con un parámetro de varianza t.

$$L(x, y, t) = I(x, y) * e^{-\frac{u^2+v^2}{2t^2}}$$

Como su desviación estándar $\sqrt{\sigma} = \sqrt{t}$, detalles de tamaño menor a $\sqrt{t}$ desaparecen.

6.2.2. Hessiano rápido

Se basa en la determinante de la matriz Hessiana. Corresponde a un buen puntaje de esquinas.

$$H(x, y, \sigma) = \begin{bmatrix} \frac{\partial^2}{\partial x^2} G(\sigma) * I(x, y) & \frac{\partial^2}{\partial x \partial y} G(\sigma) * I(x, y) \\ \frac{\partial^2}{\partial x \partial y} G(\sigma) * I(x, y) & \frac{\partial^2}{\partial y^2} G(\sigma) * I(x, y) \end{bmatrix}$$

Como derivar gaussianas es muy costoso, se aproxima esto mediante filtros:

$$\begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

6.2.2.1. Imágenes Integrales Se usa como una forma rápida para sumar píxeles en una imagen. En cada píxel se escribe la suma de todos los píxeles en el cuadrado entre $(0, 0)$ y (x, y), inclusive el píxel original. Se puede calcular fácilmente haciendo sólo un barrido por la imagen.

La gracia de esto es que si se quiere sacar alguna sumatoria de la imagen (como calcular intensidad), el cálculo es en tiempo constante.

6.2.3. Algoritmo FAST

Acrónimo para *Features from Accelerated Segment Test*. El algoritmo trabaja en un círculo discretizado alrededor de un punto candidato p. Con árboles de decisión (machine learning) se pudo acelerar el algoritmo, reduciendo su complejidad.

El problema de FAST es que no indica qué tan fuerte es la característica. Para hacerlo hay que aplicar *non-maximum suppression*.

6.3. Descriptores locales

Luego de identificar puntos de interés, esto permite identificarlos entre imágenes.

Permiten representar características de una vecindad reducida o una región de interés. Sirven para detectar características fuertes y dominantes en una imagen y ser capaz de encontrarlas en las siguientes (seguimiento) Cuidado, pueden ser no-únicas, pero se puede contar con otras para el seguimiento.

Un tracker funciona mejor vienen más que de características intrínsecas, depende de una característica que sobreviva por todo el video, o por la mayor parte de él.

Los más populares son Patch de Imagen, SIFT, SURF y FERNs.

6.3.1. Patch de Imagen

Simplemente toma la vecindad cuadrada del pixel tomado. Luego se hace varias aproximaciones para acercarse al estado del arte.

6.3.2. SIFT

Es invariante a la escala (Scale Invariant). Se crea una imagen basada en un conjunto de vectores de características, de modo que estos sean inmunes a la mayoría de transformaciones lineales (translación, rotación y escala euclídeos) y parcialmente invariables a cambios de iluminación y robustos ante distorsión geométrica local.

Se sacrifica mucha velocidad por robustez.

Pasos:

1. Construcción de Espacio de escalas (Scale Space). 6.2.1.1
2. Diferencia de Gaussianas. 6.2.1
3. Asignación de orientación: *Se genera un histograma de orientación, de modo que los peaks del histograma sean las direcciones dominantes de los gradientes locales.*
4. ...

6.3.3. SURF

Es mucho más rápido que SIFT al ocupar algunas características como Imágenes integrales, Hessiano rápido y filtros de Haar. En algunas transformaciones es más robusto que SIFT. Tiene muchísimas implementaciones disponibles. Al contrario que SIFT, el Espacio de escalas en lugar de empequeñecer la imagen, se agranda.

Se usa el filtro Haar para encontrar imágenes diferenciales (matrices -1, 1 horizontales y verticales), haciéndolo con filtros box. La respuesta del filtro se pudera usando una Gaussiana según el punto de interés, generando un diagrama polar, con una ventana de ángulo $\pm 30^0$

y luego los puntos encontrados ahí se les saca el promedio de su ángulo. El descriptor final saca un 4x4 desde una ensalada de vectores. Es invariante a offset de iluminación.

Devuelve una tupla $(\sum dx, \sum |dx|, \sum dy, \sum |dy|)$

6.3.4. Comparación SIFT y SURF

Los espacios de búsqueda de SURF son de 64 en lugar de SIFT de 128. Además es más robusto al ruido. Los gradientes SIFT también detecta ruido, pero SURF se mantiene intacto a pesar del ruido.

6.3.4.1. SURF master race

7. Seguimiento / Tracking

7.0.4.2. En qué consiste Es una definición ambigua, pero permite sacar conclusiones en relación al movimiento de objetos detectados mediante una secuencia temporal de imágenes.

Uno de sus mayores propósitos es estimar a priori en dónde estará el objecto rastreado en la próxima imagen. **Esto es costoso**, por lo que lo ideal es tener un modelo o una región acotada para hacerlo. Si la cámara o el objeto no se mueven muy rápido, se puede acotar el espacio de búsqueda, agilizando el rastreo.

Sus variedades son:

- **Flujo óptico / Optical flow**: Recupera directamente el movimiento de la imagen a partir de variaciones en el brillo.
- **Seguimiento / Tracking**: Asocia objetos modelados y detectados anteriormente en la presente imagen. Generalmente usa modelos dinámicos.
- **Comparación de características / Feature Matching**: Consiste en comparar características de dos objetos.

  - Filtro de Kalman
  - Filtro de Partículas
  - Feature Matching: FLANN y KD-trees

7.1. Flujo óptico

Estima movimiento de los objetos entre dos imágenes consecutivas sin conocimiento previo. Una forma de pensarlo es asociar un vector de velocidad a cada píxel en la imagen (*dense optical flow*).

Es problemático hasta para una persona detectar el flujo óptico. Es de alto costo (hay que seguir todos los píxeles), y no siempre arroja buenos resultados (por ejemplo, en una hoja de papel blanco los píxeles se parecen mucho)

Una alternativa es el **Sparse Optical Flow**

7.1.1. Sparse Optical Flow

Sigue un conjunto de puntos clave a seguir. Esto corresponde a esquinas detectadas sea con Shi-Tomasi, SIFT, SURF u otros. Estos se siguen en un espacio local (unos 5 píxeles alrededor del *Feature to track*) para no sobrecargar el sistema.

7.1.2. Lucas-Kanade (LK) Optical Flow

Sigue un conjunto de puntos (*Good features to track*) localmente. Hace algunos supuestos para seguir una región.

7.1.2.1. Hipótesis

- **Brillo constante** Se supone que entre una imagen y otra no cambia el brillo (significativamente).
- **Persistencia temporal** Los movimientos de la cámara u objeto son lo suficientemente suaves como para que entre un cuadro y otro permanezca en la misma vecindad. Y tampoco desaparece de la cámara.
- **Coherencia espacial** Los píxeles de una misma vecindad pertenecen a una misma superficie (estos puntos tienen aproximadamente el mismo movimiento al mover la cámara).

7.1.2.2. Ecuaciones Si u y v representan el desplazamiento relativo a la primera imagen...

- **Persistencia temporal**: Se hace la ecuación de consistencia de brillo: El valor de I (imagen de brillo) no cambia y su derivada respecto al tiempo es 0.

$$I_x u + I_y v + I_t = 0$$

  Dado que tenemos una ecuación y dos incógnitas u y v, necesitamos otra ecuación y elaborar otro supuesto.

- **Movimiento coherente**: Si los píxeles cercanos se mueven coherentemente:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \end{bmatrix}$$

  Como ahora sobran ecuaciones se usa mínimos cuadrados, de modo que:

$$\begin{bmatrix} \Sigma I_x I_x & \Sigma I_x I_y \\ \Sigma I_y I_x & \Sigma I_y I_y \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \Sigma I_x I_t \\ \Sigma I_y I_t \end{bmatrix}$$

7.1.3. Filtro de Kalman

Trata de estimar el estado real de una cosa $\mathbf{a}_i$ a partir de varias mediciones ruidosas de esta $\mathbf{x}_i$ (sujetas a ruido gaussiano). Este modelo se va actualizando a medida que se toman más mediciones. Básicamente se tiene un estimado $\hat{\mathbf{a}}_i$ que es un ponderado entre la estimación anterior $\hat{\mathbf{a}}_{i-1}$ y la medición actual $\mathbf{x}_i$, ponderándolo con el factor K_i que se va actualizando constantemente.

$$\hat{\mathbf{a}}_i = K_i\,\mathbf{x}_i + (1 - K_i)\,\hat{\mathbf{a}}_{i-1}$$

Por eso se debe buscar una ganancia de Kalman óptima K_i. Para eso se toma una matriz-modelo de observación H para describir cómo va a ser nuestra medición.

$$d$$

[Fill]

7.2. Feature Matching

Busca un objeto cuyo modelo ya está conocido por el sistema. El enfoque es la rapidez. Utiliza como base el método *kd-trees*.

Dos implementaciones son BBF y FLANN.

7.2.1. KD-Trees

Sirve para organizar puntos en un espacio k-dimensional. En este ámbito las k-dimensiones pueden salir de vectores de características de acuerdo al modelo que se quiere rastrear. Su aplicación en *feature matching* es buscar cuáles son las características más cercanas dentro de los modelos guardados.

Lo que se quiere evitar es obtener la distancia euclideana de un punto a buscar con todos los puntos del modelo. Con esto reducimos la cantidad de distancias para calcular, de modo que para cada una descartemos la mitad de los puntos del modelo.

7.2.1.1. Buscar Básicamente se va buscando dentro del árbol para abajo si en un modelo la distancia se minimiza. Si en ninguna de los hijos la distancia baja, entonces esa es la característica que mejor acomoda.

7.2.2. Best Bin First (BBF)

Es una solución similar a KD-Trees. Se busca el vecino más cercano, pero en ciertos casos no asegura precisión en los peores casos. Se arma, a diferencia de un árbol binario, una cola de prioridad basada en cercanía. Como se dijo la precisión merma debido a que **no se revisan todos los puntos**. En algunas pruebas resultó ser 14 veces más rápido que KD-Trees.

7.2.3. FLANN

No es algo nuevo sino que fija un criterio de selección para saber en qué casos es más conveniente usar KD-Trees o BBF. En su implementación usa:

- Randomized KD-Trees
- Hierarchical K-Means Tree

7.3. Medidas de fiabilidad

Se propone un modelo dinámico que integre medidas de fiabilidad, para saber qué tan útil y confiable es la medida obtenida y ponderarlas para la actualización de un modelo. Algunos atributos 3D contemplan la ocultación dinámica, los errores en píxeles para objetos cercanos y lejanos, etc.

7.3.1. Fiabilidad de coherencia espacio-temporal $RC_a(t_c)$

En el tiempo t_c, con $a_{\text{máx}}$ y $a_{\text{mín}}$ predefinidos para a.

$$RC_a(t_c) = 1{,}0 - \text{mín}\left(1{,}0, \frac{\sigma_a(t_c)}{a_{\text{máx}} - a_{\text{mín}}}\right)$$

En esta desviación estandar se considera una versión incremental (ver diapos del profesor)
Inspirado fuertemente en filtros de Kalman, dado que está basado en estimadores. Sin embargo esta medida incluye función de enfriamiento.

7.4. Multitarget Tracking (MTT)

Consiste, en como dice su nombre, seguir más de un objeto sin perderlos ni confundirlos.
Consiste en dos enfoques: probabilista y determinístico.

- **Probabilista** (*Monte Carlo, Particle Filtering, Condensation*): Se busca modelar una distribución probabilística de los atributos de los objetos seguidos (distribución multi-modal, varias modas).
- **Determinístico** (*Multi Hypotesis Tracking*): Múltiples hipótesis de correspondencia para cada objeto en cada cuadro.

7.4.1. Algoritmo CONDENSATION (CONditional DENSity estimATION)

Dado que un filtro de Kalman solo modela una hipótesis (la distribución de probabilidad es gaussiana y solo tiene una moda). Y CONDENSATION supone varias modas, lo que permite ser usado para seguimiento de varios objetos.

Un ejemplo es el seguimiento de una persona detrás de un árbol. Puede que simplemente vaya caminando como jugando a las escondidas, entonces el CONDENSATION permite modelar ambos casos y seguir siguiendo al objeto.

CONDENSATION funciona de manera similar a Kalman: hay un paso determinístico de transformación, añade ruido para estimar la medición y luego, a diferencia de Kalman, opera localmente.

7.4.1.1. Forma de operar Se busca un vector de estados de N dimensiones $\mathbf{x}_N \in \mathcal{X}$. Se tiene mediciones $\mathbf{z}_N$, tal como en Kalman. Las t últimas mediciones se conocen y se designan como un vector $\mathbf{Z}_t$.

[...]

El costo de estas operaciones son demasiado altos en muchos casos (debido a la cantidad de muestras necesarias para hacer la hipótesis), debido a las siguientes operaciones:

- **Filtro bayesiano recursivo** Debido a su complejidad y al no ser posible evaluarlo en forma *cerrada*.
- **Muestreo por factores (Factored Sampling)** Muy pocas veces se puede tener soluciones analíticas al problema.
- **Espacio de muestras multidimensional**

7.4.2. MHT Multi-Hypotesis Tracking

El algoritmo típico de MHT es asociar cada evidancia visual con un objeto.

1. Hay que ser cuidados con los objetos cercanos, dado que las asociaciones entre ellos aumentan. (Hipótesis relacionadas)
2. Se actualiza la información trayectoria para cada objeto.

8. Geometría Proyectiva

Se tiene las siguientes transformaciones:

- **Transformación Afín**: La matriz $[A]$ tiene parámetros restringidos, para permitir cizallamiento, escalamiento, rotación y traslación. [1]

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = [A] \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Transformación Proyectiva**:

$$\begin{bmatrix} x_t \\ y_t \\ t \end{bmatrix} = [P] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad x' = \frac{x_t}{t} \qquad y' = \frac{y_t}{t}$$

- **Homografía**: Plano 2D $\rightarrow$ Plano 2D

$$\begin{bmatrix} x_t \\ y_t \\ t \end{bmatrix} = [H] \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad x' = \frac{x_t}{t} \qquad y' = \frac{y_t}{t}$$

8.1. Modelo de cámara *pinhole*

Se modela una cámara de modo que tiene un agujero de tamaño infinitesimal (punto focal) en donde converge toda la luz. Luego de este agujero hay una retina que convierte esto en una imagen 2D. ¿Pero cuál es el problema? Esta imagen 2D puede estar formada por rayos de luz cuya distancia recorrida no se conoce.

Desventajas: El problema es que la cámara pinhole captura muy poca luz (dado este punto infinitesimal). Por lo tanto se usan lentes para captar más luz, pero se sacrifica la linealidad al usar lentes convexos.

8.2. Modelo de geometría euclideana y proyectiva

La diferencia entre ellas es que en la geometría euclideana no existe un punto en común entre dos rectas paralelas. En geometría proyectiva ese punto sí existe, y es el **punto en el infinito** (pensarlo como dos líneas de tren que se encuentran en el horizonte). OJO: La recta en el infinito también está definida.

[1] El componente "1" de los vectores sirve para proporcionar el componente de traslación a la transformación.

Dado que puede haber varios puntos en el infinito (y el infinito no está definido en los reales), agregamos una nueva coordenada t para parametrizar el punto en el infinito (haciendo $t = 0$). Con esto tenemos las **coordenadas homogéneas**, definidas por una tupla (x_t, y_t, t).

Punto de fuga: Proyección de un punto en el infinito encima de una imagen. Corresponde al punto en el cual convergen todos los pares de líneas paralelas en el mismo *ground-plane*.

8.2.1. Ecuación de la recta

Para definir la recta homogénea, pasamos de la ecuación euclideana:

$$\underset{\text{Euclideana}}{Ax + By + C = 0} \qquad \Rightarrow \qquad \underset{\text{Proyectiva}}{Ax + By + Ct = 0}$$

Esto sirve para:

- Geometría euclideana: $t = 1$
- Puntos en el infinito: $t = 0$
- Recta en el infinito: $A, B = 0$, haciendo que la ecuación de una recta en el infinito sea $t = 0$.

8.2.2. Propiedades con puntos y rectas

Dado que un punto y una recta se expresan como:

$$\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T \qquad \mathbf{R} = \begin{bmatrix} A & B & C \end{bmatrix}^T$$

8.2.2.1. Incidencia

$$\mathbf{p}^T \cdot \mathbf{R} = 0$$

8.2.2.2. Colinealidad y concurrencia Tres puntos forman parte de una misma recta si ocurre que:

$$\begin{vmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 \end{vmatrix} = 0$$

Asimismo tres rectas se cortan en el mismo punto si:

$$\begin{vmatrix} \mathbf{R}_1 & \mathbf{R}_2 & \mathbf{R}_3 \end{vmatrix} = 0$$

8.2.2.3. Recta entre dos puntos La recta que forman dos puntos se puede expresar como:

$$\mathbf{R} = \mathbf{p}_1 \times \mathbf{p}_2$$

8.2.2.4. Punto de intersección entre dos rectas Corresponde a:

$$\mathbf{p} = \mathbf{R}_1 \times \mathbf{R}_2$$

8.2.3. Transformación de homografía

La colinealidad de cualquier conjunto de puntos se preserva. Y dado que el escalamiento en una transformación homográfica no importa (para eso está el factor t, se tiene 8 incógnitas para encontrar la matriz $[H]$. Por eso, entonces, para encontrar la matriz $[H]$ de una homografía, se necesita al menos 4 puntos. Hay métodos si se consideran más de 4 puntos.

8.2.3.1. Null Space Búsqueda de todos los vectores $\vec{x}$ tal que $M\vec{x} = 0$. Esto se logra mediante Singular Value Decomposition (SVD). [2]

8.3. Transformación mundo real $\rightarrow$ cámara

Primero se tiene una transformación de coordenadas 3D del mundo real (altura, ancho y profundidad) a coordenadas 3D de la cámara (con eje z normal al plano de la imagen proyectada). Estos son los **parámetros extrínsecos**

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [Rt] \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Luego los haces de luz se proyectan en un plano a una distancia f del pinhole de la cámara (largo focal). Esto aún es en sistema métrico.

$$\begin{bmatrix} u_t \\ v_t \\ t \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Y al final, transformamos de metros a píxeles.

$$\begin{bmatrix} u_{pix} \\ v_{pix} \\ 1 \end{bmatrix} = [P] \begin{bmatrix} \frac{u_t}{t} \\ \frac{v_t}{t} \\ 1 \end{bmatrix} \qquad \text{Con } [P] = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

(Con γ casi siempre es cero)

Las últimas dos matrices de parámetros se pueden combinar para formar la *matriz intrínseca* $[I]$.

[2] * en las diapos significa "Conjugado (imaginario) transpuesto"

8.3.1. Calibración

Es el proceso para encontrar los datos necesarios para convertir las imágenes 2D captadas a 3D del mundo real. Entonces:

$$\mathbf{u}_{2D} \rightarrow \mathbf{x}_{3D}$$

Para ello debemos encontrar los doce parámetros que nos permiten hacer esto:

$$\begin{bmatrix} x_t \\ y_t \\ t \end{bmatrix} = \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{\text{Parámetros}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Para que el problema sea homogéneo suponemos que $p_{34} = 1$, con lo que tenemos un sistema de 11 ecuaciones. Sean (x, y, z) los puntos 3D, y (u, v) los puntos 2D que corresponde...

$$\left[\begin{array}{cccc|cccc|ccc} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_1x_1 & -v_1y_1 & -v_1z_1 \\ \hline x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v_2x_2 & -v_2y_2 & -v_2z_2 \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}\right] \cdot \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ \vdots \\ p_{32} \\ p_{33} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ \vdots \\ u_N \\ v_N \end{bmatrix}$$

Como la matriz no es cuadrada, se debe recurrir a la **pseudoinversa** para poder encontrar los parámetros. Se puede usar Singular Value Decomposition o algún otro método.

A. Herramientas matemáticas

A.1. Least Square Mean

Se debe buscar un vector $\mathbf{x}$ tal que, con una matriz A se minimice la siguiente norma:

$$\mathbf{x} = \min_{\mathbf{x}}(\|A\mathbf{x}\|)$$