

# Investigación en autolocalización visual

Marcos Piersas Sagardoy



4 de julio de 2016

# Índex

## 1 Introducción

## 2 Algoritmos de visualSLAM

- MonoSlam
- PTAM
- Métodos directos
- SVO
- LSD-SLAM
- ORB-SLAM

## 3 Conclusiones

# Introducción

# Introducción

## Autolocalización visual

Consiste en la determinación de la posición y orientación de la cámara utilizando para ello únicamente las imágenes recibidas de ésta.

- Aplicaciones en robótica y en realidad aumentada.



(a) Dyson



(b) HoloLens

# Introducción: Robótica

Un robot para navegar debe realizar estas tareas:

- Percepción
- Localización
- Cognición
- Control del movimiento

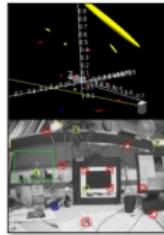
Para la **localización** hay sensores específicos como:

- IMU
- Navegación satélites
- Escáneres

Entre ellos destacan las cámaras.

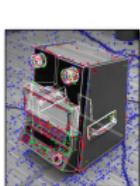
# Introducción: Visión artificial

## Autolocalización visual



Consider natural features

No *a priori* 3D knowledge



Consider natural features

Use some 3D knowledge



Make use of visual markers

Use some 3D knowledge



# Algoritmos de visualSLAM

# MonoSlam: Introducción

- Primer algoritmo de slam monocular en tiempo real.
- Basado en *Filtering, Extended Kalman Filter*.
- Basado en características, FAST.
- Reconstrucción dispersa de unos 100 puntos.
- Para robótica, enfatizar localización.

# Fucionamiento: *Tracking*

- Vector de estado:

$$\hat{x} = (\hat{x}_v \quad y_1 \dots y_n)^T$$

$$\hat{x}_v = (r^W \quad q^{WR} \quad v^W \quad \omega^R)^T$$

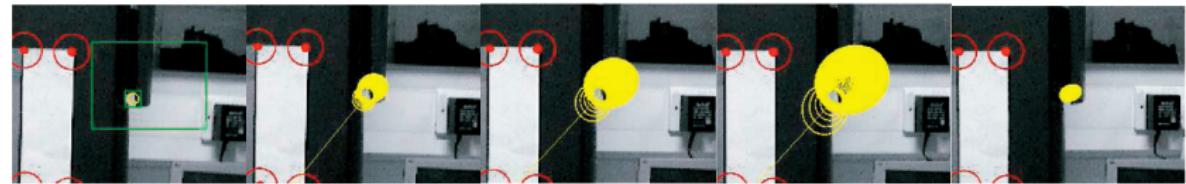
- Predicción: Modelo de movimiento y posición anterior.

$$f_v = \begin{pmatrix} r_{nueva}^W \\ q_{nueva}^{WR} \\ v_{nueva}^W \\ \omega_{nueva}^R \end{pmatrix} = \begin{pmatrix} r^W + (v^W + V^W)\Delta t \\ q^{WR} \times q((\omega^R + \Omega^R)\Delta t) \\ v^W + V^W \\ \omega^R + \Omega^R \end{pmatrix}$$

- Actualización: Modelo de observación

$$z_i = h_t(\hat{x}_t, y_i) \qquad \Sigma_{IN} = H\hat{P}_t + R$$

# Funcionamiento: *Mapping*



# Conclusiones

- Consigue localización precisa en tiempo real.
- Requisito tiempo real falla para más de 100 puntos.
- No permite relocalización.
- Entornos pequeños.
- Representación poco densa.



# PTAM

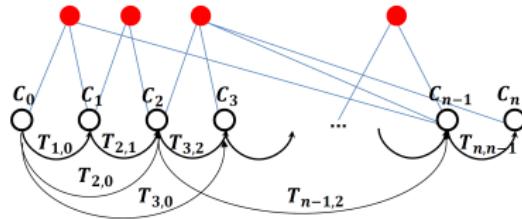
- PTAM: *Parallel tracking and mapping*.
- Separación en dos hilos del *tracking* y *mapping*.
- *Keyframe* con *Bundle adjustment*.
- Basado en características.
- Ideado para realidad aumentada.
- Su arquitectura se convirtió en estándar.

# Funcionamiento: *tracking*

- Modelo de movimiento: Modelo velocidad decayente.
- Enfoque piramidal.
- Minimización del error de retroproyección con 50 puntos.
- Refinamiento iteración con 1000 puntos.
- Permite relocalización: nota sobre la calidad del *tracking*.

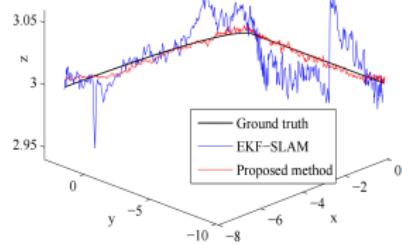
# Funcionamiento: *mapping*

- No tiene necesidad de funcionar en tiempo real.
- Guardan *keyframe* en memoria.
- Triangulación entre *Keyframes*.
- Optimización de puntos y pose con *Bundle adjustment*.

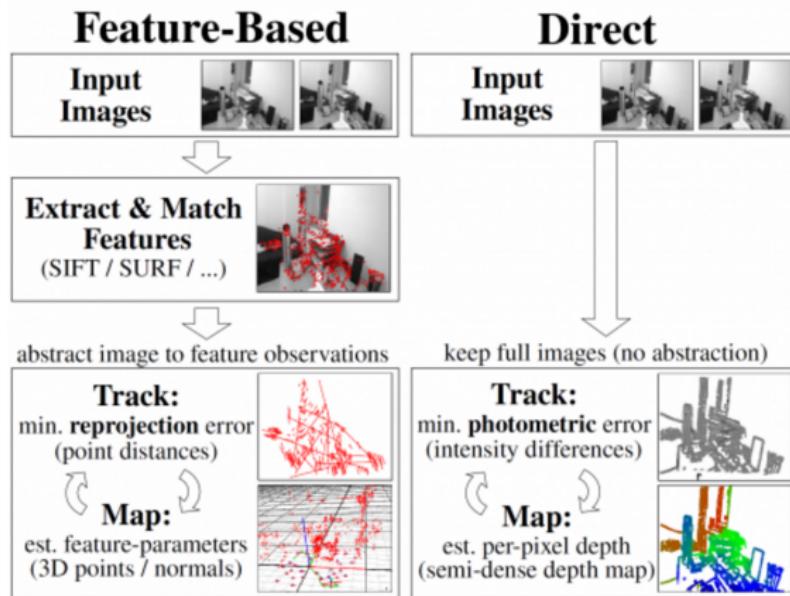


# Conclusiones

- Localización más precisa y representación más densa que MonoSlam.
- Cumple el requisito de tiempo real en el *tracking* para gran cantidad de puntos.
- No es robusto a grandes escenas.
- Permite la relocalización.

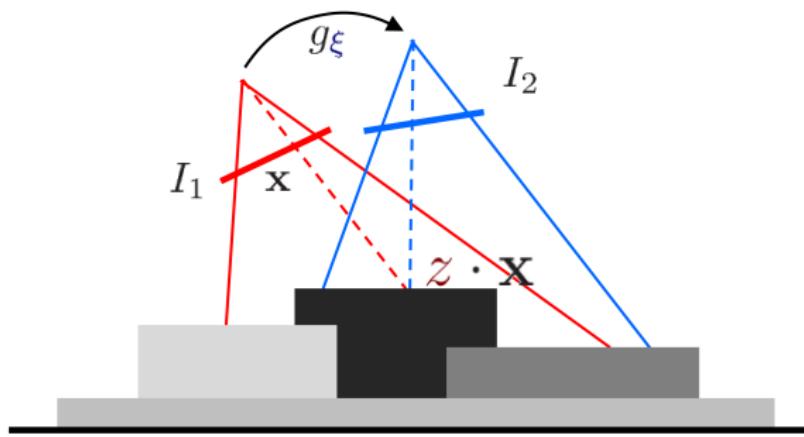


# Métodos directos

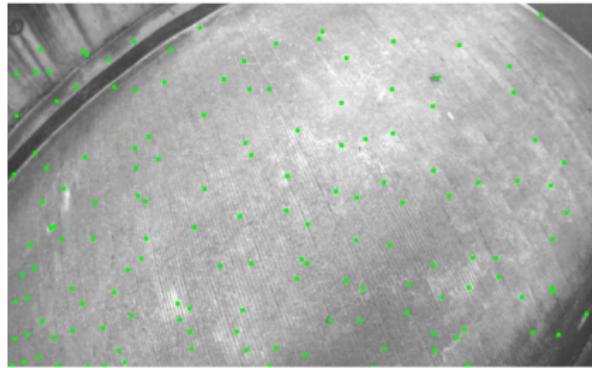


# Métodos directos

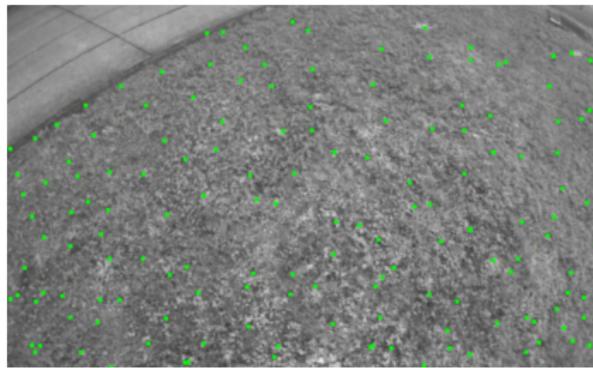
$$l_1(x) = l_2(\pi(g_\xi(z * x)))$$



# Métodos directos: Ejemplos



(j) Textura de alta frecuencia.

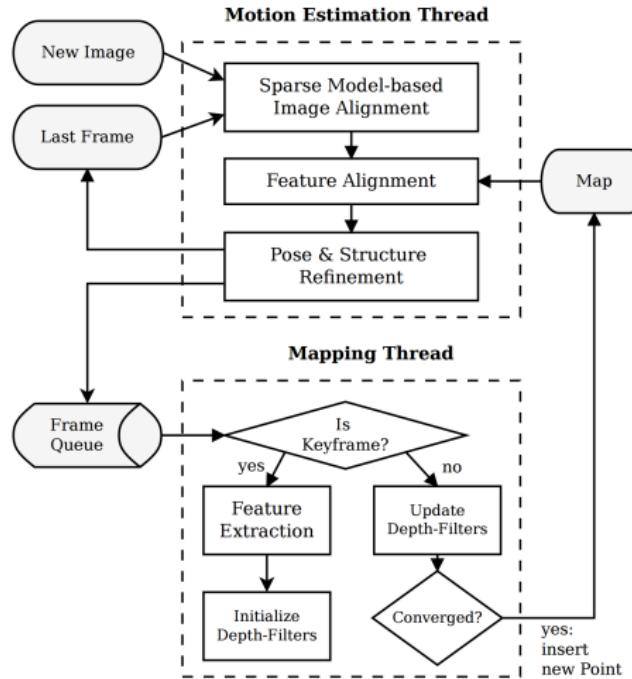


(k) Textura de alta frecuencia.

# SVO

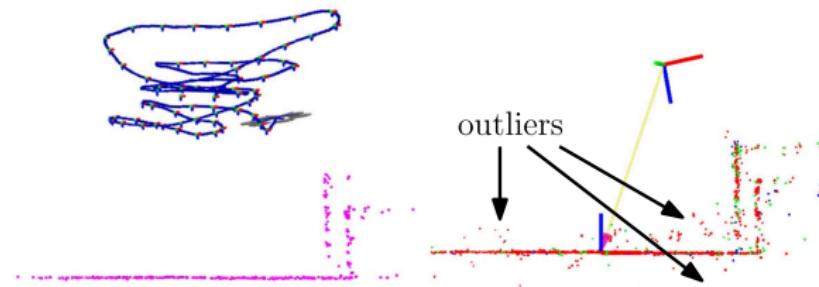
- Siglas de *Semi-direct visual odometry*.
- Odometría visual, la reconstrucción poco densa.
- Basado en métodos híbridos.
- Separación *tracking* y *mapping*. Uso de *Keyframe* con *Bundle adjustment*.
- Marco de trabajo probabilístico para estimar la profundidad.
- Para sistemas embebidos.

# Funcionamiento



# Conclusiones

- Algoritmo muy rápido. 300 fps en sobremesa. 55 fps sistema embebido.
- Reconstrucción poco densa, pero más precisa.



# LSD-SLAM

- LSD-SLAM, siglas de *Large scale direct slam*
- Semi denso, solo reconstruye píxeles con gradiente.
- Estimación probabilística de la profundidad.
- Robustez largas trayectorias, realiza *loop closure*.
- Parametriza la escala.
- En CPU, anterior DTAM en GPU.

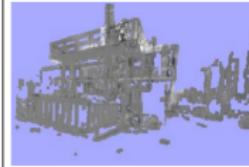
# Tracking

## Tracking

New Image  
(640 x 480 at 30Hz)



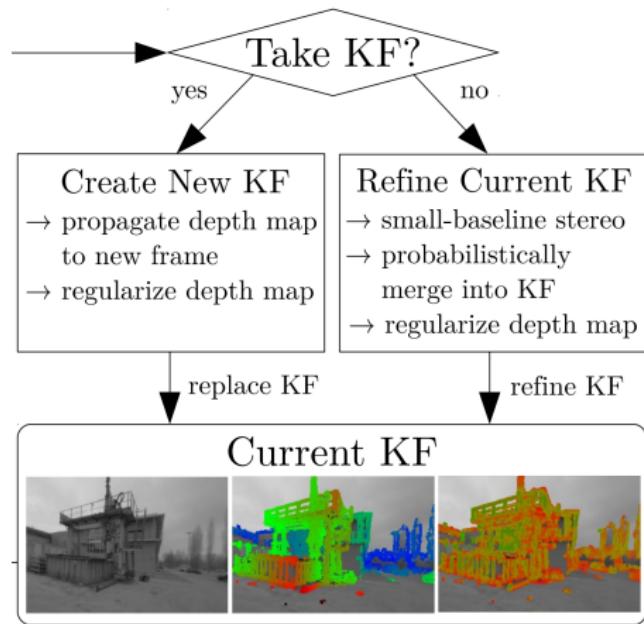
Track on Current KF:  
→ estimate SE(3) transformation



$$\min_{\xi \in \mathfrak{se}(3)} \sum_{\mathbf{p}} \left\| \frac{r_p^2(\mathbf{p}, \xi)}{\sigma_{r_p(\mathbf{p}, \xi)}} \right\|_{\delta}$$

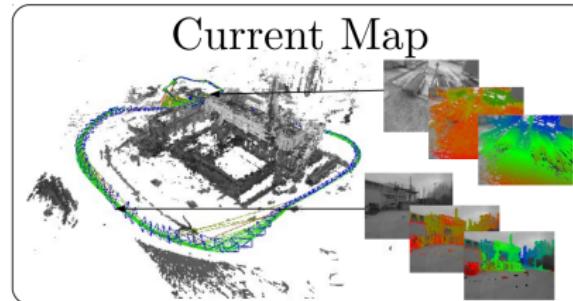
# Mapping

## Depth Map Estimation



# Loop closure

## Map Optimization

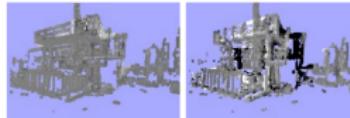


↑ add to map

### Add KF to Map

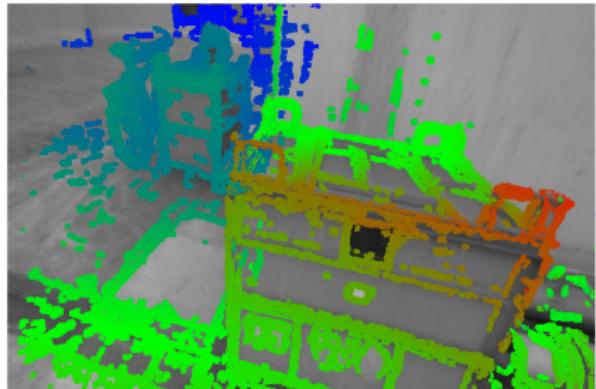
- find closest keyframes
- estimate  $\text{Sim}(3)$  edges

$$\min_{\xi \in \text{sim}(3)} \sum_{\mathbf{p}} \left\| \frac{r_p^2(\mathbf{p}, \xi)}{\sigma_{r_p}^2(\mathbf{p}, \xi)} + \frac{r_d^2(\mathbf{p}, \xi)}{\sigma_{r_d}^2(\mathbf{p}, \xi)} \right\|_{\delta}$$



# Conclusiones

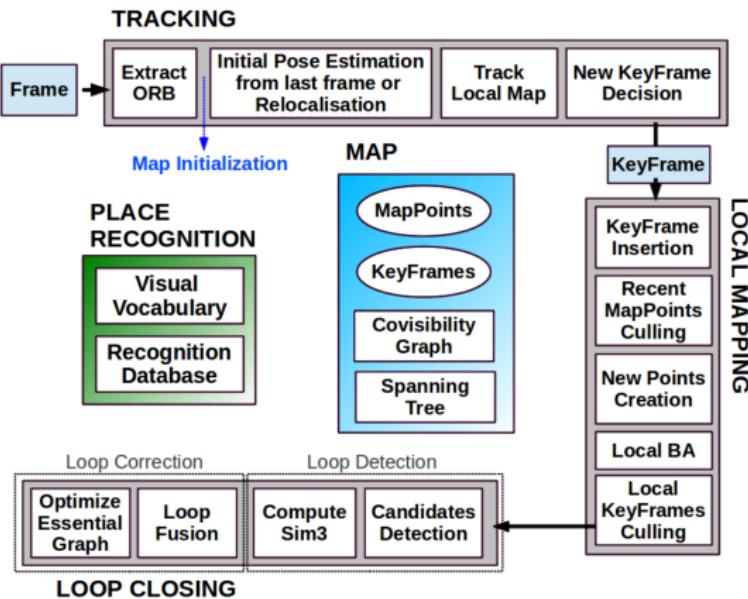
- Reconstrucción semidensa.
- Largas trayectorias.



# ORB-SLAM

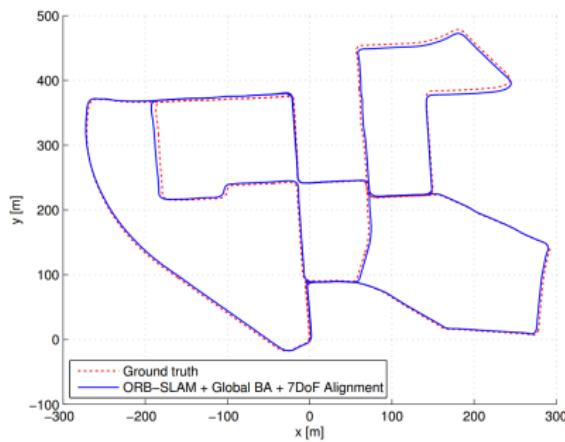
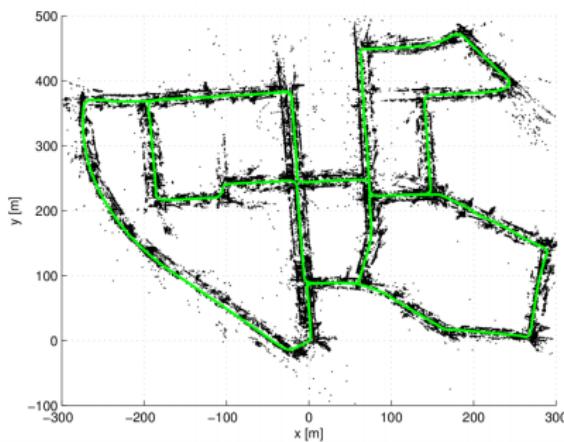
- Basado en características.
- Separación *tracking*, *mapping* y *loop closure*.
- *Keyframe* con *Bundle adjustment*.
- Reconocimiento de escena para realizar *Loop closure*.
- Relocalización.

# Funcionamiento



# Conclusiones

- Hasta la fecha es el más preciso. Supera a PTAM, LSD-SLAM y RGBD-SLAM.
- Tiempo real.



# Conclusiones

# Conclusiones

- MonoSlam, primer algoritmo monocular slam en tiempo real.
- PTAM, separación *tracking* y *mapping*, PTAM supera a MonoSlam
- Métodos directos: SVO,LSD-SLAM
- Basados en características: ORB-SLAM

# Gracias por su atención

