v0.1

CAPÍTULO 1

Benchmark

Durante el desarrollo del proyecto se hizo presente la necesidad de contar con herramientas de medida del desempeño objetivas para la validación de los algoritmos que se utilizaron y desarrollaron. En particular se desea evaluar los algoritmos asociados a la detección y estimación de pose. Estos son LSD, filtrado de segmentos, determinación de correspondencias y Posit Coplanar.

El enfoque elegido para la validación de dichos algoritmos consiste en la generación de una "verdad de piso" o *ground truth* de forma de poder comparar la detección y resultados obtenidos con los valores reales o verdaderos de los mismos. Este proceso es lo que se denomina en ese proyecto como *Benchmark* o Comparativa. De esta forma se pueden generar un número suficiente de datos de salida de forma de obtener algunas estadísticas de interés para la evaluación de los algoritmos.

1.1. Generación de imágenes sintéticas: Blender

La generación de imágenes sintéticas consiste en producir una imagen 2D en base a una modelo 3D de una escena y una pose de la cámara respecto a esa escena o viceversa. Este proceso se denomina *Rendering* y el mismo trata en el Capítulo **??** para la herramienta para dispositivos móviles *ISGL3D*.

Para realizar el Benchmark se desarrolló un entorno de generación de imágenes sintéticas que utiliza herramientas gratuitas y de código abierto que permiten capacidades de *scripting* y ejecución en forma "batch" de forma de serializar la producción de renders en un proceso automático. En el proceso de estos renders fue realizada mediante el uso del software Blender.

Blender es un *suite* de creación de contenidos gratiuito y de código abierto. Esta disponible para los principales sistemas operativos, Windows, Mac, Linux; bajo la Licencia Pública General GNU[37]. Algunas de sus prestaciones son modelado, *shading*, animación, renderizado, e iteracción 3D. También presenta herramientas de *scripting* mediante el uso del lenguaje *Python* las cuales se explicarán mas adelante.

1.1.1. Modelo 3D y escena

Para lograr hacer un render es necesario contar con una escena 3D y en particular con un modelo 3D del objeto de interés. Para realizar las tareas de detección se utiliza un marcador plano que se describe en el Capítulo *ch:marcador*.

Se manejaron algunas alternativas para la generación del modelo 3D del marcador entre las cuales se encuentran, VTK, Paraview, Autodesk 3ds Max y el mismo programa utilizado por este

proyecto para rendering en una PC, Blender. Finalmente debido a la simplicidad del marcador, y principalmente a que es un marcador plano, se utilizó un dibujo vectorial del marcador (formato svg) generado en Inkscape. Este dibujo vectorial puede ser importado a Blender y con el generar un modelo 3D que será un actor en la escena. Dentro de la interfaz de Blender se ajustó un factor de escala para que las medidas reales del marcador se correspondan con las unidades de Blender. En la Figura 1.1 se muestra la vista de *Perspectiva de usuario* en donde se reflejan algunos de los cambios que se realizan sobre la escena. Se busca construir una escena mínima y controlada en la

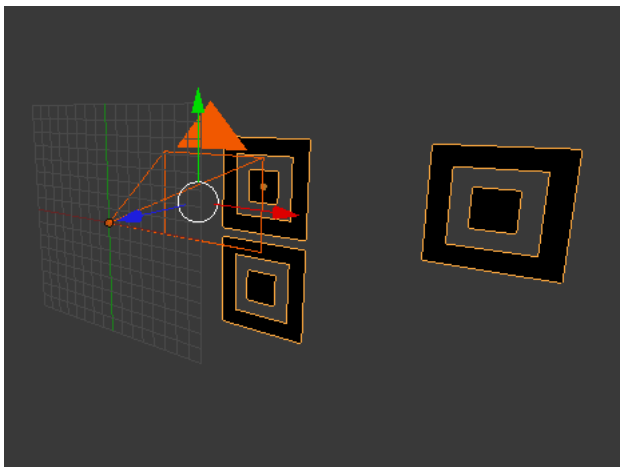


Figura 1.1: Vista de perspectiva de usuario para el modelo 3D del marcador y la cámara en la escena.

cual solo actúa el marcador. Con este fin es que se decide por una escena que contenga únicamente el marcador y una cámara. No se añaden luces ya que el efecto de estas en el marcador para alguna pose puede resultar en artefactos de iluminación indeseados. La no inclusión de luces en las imagen esta ligada a que el modelo 3D del marcador es en realidad hueco en las zonas blancas y negro en las zonas negras por lo tanto el negro permanece negro y la parte hueca asume el color del fondo. En otro caso se podría utilizar algún tipo de iluminación de ambiente que sea pareja en todas las direcciones y lograr un efecto similar.

La escena conteniendo el modelo 3D del marcador se resume en el archivo `Marker.blend`.

1.1.2. Ajuste de escena y parámetros

El software Blender permite el ajuste de la escena y los parámetros y características asociados al render, la cámara y el objeto mediante su interfaz gráfica. Estos tienen como resultado final la imagen renderizada. Los parámetros y ajustes que fueron realizados y resultan ser los más relevantes para la reproducción del proceso de renderizado del marcador se explican a continuación.

- **Mundo y fondo**: Los ajustes realizados al mundo y fondo se explicaron en cierta medida anteriormente. Al no utilizar iluminación en la escena es importante definir el color del fondo ya que los huecos del modelo 3D asumirán ese color. Los ajustes realizados se muestran en la Figura 1.2(a) y consisten principalmente en definir el fondo de color blanco.
- **Parametros de Render**: Los principales parámetros ajustables en cuanto a las propiedades de render son el tamaño de imagen y el porcentaje de escala sobre el tamaño de imagen el cual por defecto en proyectos nuevos se ubica en 50 % y en este caso interesa que sea 100 %. Estos ajustes se pueden ver en la Figura 1.2(b).

- **Parámetros de Cámara**: Sobre las propiedades ajustables de la cámara se encuentra un parámetro de interés como es el campo visual, *field of view*, o fov. Adicionalmente el *Shift* representa el corrimiento del centro de imagen de la misma y también se puede ajustar en *near* y el *far*. Los ajustes realizados se pueden ver en la Figura 1.2(c).
- **Parámetros de Objeto**: Las propiedades ajustables de mayor interés sobre el objeto se refieren a su pose. La locación y rotación del objeto pueden ser asociadas a la pose del objeto respecto a la cámara si se asume una interacción del tipo cámara fija, objeto movil. Se aclara aquí que un ajuste idéntico se puede realizar sobre el objeto cámara. La cámara se fija para todos los casos en el origen de coordenadas de Blender con orientación nula en las tres direcciónes. Esto resulta en una cámara que mira en el eje z. Volviendo a los ajustes del objeto marcador, se presentan los ajustes de escala ya mencionados en el ajuste de unidades del modelo 3D y también un factor de importancia como es el orden y tipo de rotación *XYZ Euler*. Todos los ajustes aquí descritos para el objeto marcadores se pueden ver en la Figura 1.2(d).

Vale la pena observar que los ajustes de Render y Cámara constituyen una forma de establecer los **parámetros intrínsecos** de la cámara. Mientras que los ajustes de Objeto para la Cámara y el Marcador representan la **pose del objeto** en el modelo de vinculación adoptado. Se aclara que los **parámetros extrínsecos** de la cámara respecto a las coordenadas de Blender es nula en rotación y traslación.

1.1.3. Python scripting

Python es un lenguaje de programación interpretado, interactivo y orientado a objetos. Incorpora módulos, excepciones, tipeo dinámico, tipos de datos de muy alto nivel y clases. Python logra combinar notáblemente poder con claridad en la sintaxis [**?**]. La mayoría de las áreas de Blender pueden ser programadas mediante *scripts*, incluyendo Animación, Renderizado, Importación y Exportación, Creación de objetos y tareas repetitivas. Para la interacción con Blender se hace uso de una API "ajustada a medida".

La API de Blender para Python se encuentra empaquetada en el módulo `bpy`. Mediante este módulo se puede acceder a los objetos y sus propiedades de la escena. De esta forma se puede de forma programática ajustar las propiedades y parámetros explicados anteriormente. De esta forma mediante la interprete de comandos de Blender se puede prescindir de la interfaz gráfica de Blender. En la Figura **??** se muestra un *screenshot* de la consola interactiva de Python en Blender.

- Obtener referencias a la escena y el marcador.

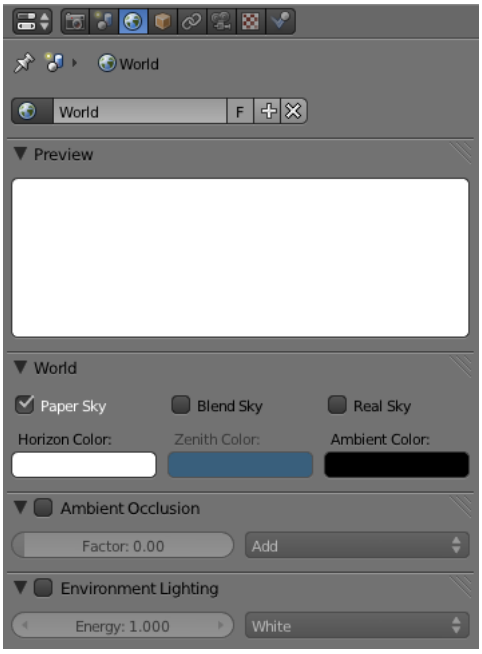
```
>>> scene = bpy.data.scenes["Scene"]
>>> marker = bpy.data.objects["Marker"]
```

- Seleccionar modo de rotación y ubicar y orientar cámara como se desea.

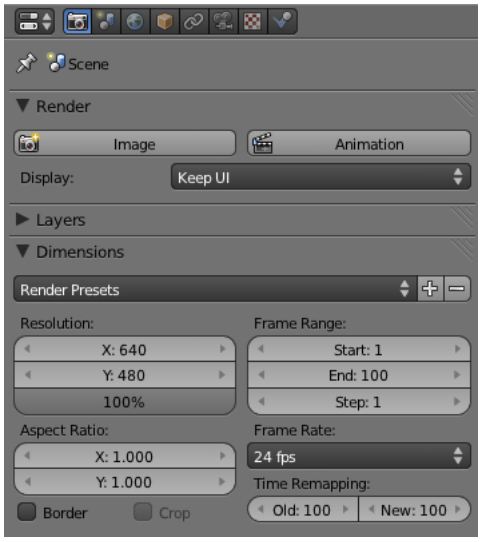
```
>>> scene.camera.rotation_mode = 'XYZ'
>>> scene.camera.rotation_euler = [0,0,0]
>>> scene.camera.location = [0,0,0]
```

- Asignación de valores asociados a **parámetros intrínsecos** de la cámara.

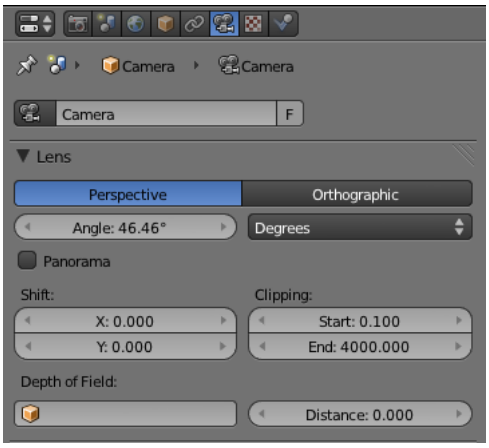
```
>>> scene.render.resolution_x = width
>>> scene.render.resolution_y = height
>>> scene.camera.data.angle = fov*(pi/180.0)
```



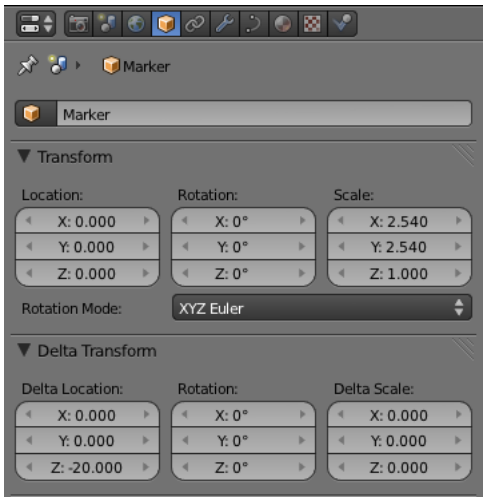
(a) Ajustes de mundo y fondo



(b) Ajustes de render



(c) Ajustes de cámara



(d) Pose del objeto

Figura 1.2: Ajustes en la escena en Blender para realizar el render.

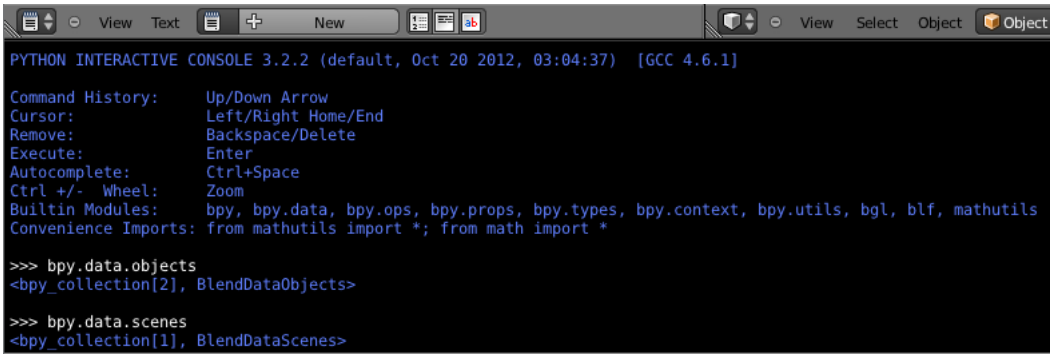


Figura 1.3: Consola interactiva de Python incrustada en Blender.

- Asignación de valores asociados a la **pose** del marcador.

```
>>> marker.rotation_mode = 'XYZ'
>>> marker.rotation_euler = [rx,ry,rz]
>>> marker.location = [tx,ty,tz]
```

- Asignación de nombre de imagen renderizada y ejecución de renderizado para guardar la imagen.

```
>>> bpy.data.scenes["Scene"].render.filepath = str(fname)
>>> bpy.ops.render.render( write_still=True )
```

1.1.4. Ejecución de consola de comandos

Bibliografía

[1] A.Etemadi. Robust segmentation of edge data. In *Proceedings of the 4th international conference on Image Processing and its applications*, 1992.

[2] Y. I. Abdel-Aziz and H. M. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Proceedings of the Symposium on Close-Range photogrammetry*, volume 1, pages 1–18, 1971.

[3] C. Avellone and G. Capdehourat. Posicionamiento indoor con señales wifi. 2010.

[4] Jean-Yves Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, November 2012.

[5] J. Brian Burns, Allen R. Hanson, and Edward M. Riseman. Extracting straight lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:425–455, 1986.

[6] F. John Canny. A Computational Approach to Edge Detection. 8(6):679–698, 1986.

[7] Stuart Caunt. Isgl3d homepage. `http://www.isgl3d.com`, November 2012.

[8] Philip David, Daniel Dementhon, Ramani Duraiswami, and Hanan Samet. Simultaneous pose and correspondence determination using line features. pages 424–431, 2003.

[9] Philip David, Daniel DeMenthon, Ramani Duraiswami, and Hanan Samet. Simultaneous pose and correspondence determination using line feature. In *CVPR (2)*, pages 424–431, 2003.

[10] Daniel F. DeMenthon and Larry S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141, 1995.

[11] Daniel DeMenthon Denis Oberkampf. Posit for coplanar points. `http://www.cfar.umd.edu/~daniel/Site_2/Code.html`, 1996 - 2004.

[12] Agnès Desolneux, Lionel Moisan, and Jean-Michel Morel. Meaningful alignments. *International Journal of Computer Vision*, 40(1):7–23, 2000.

[13] A. Etemadi, J-P. Schmidt, G.Matas, J.Illingworth, and J. Kittler. Low-level grouping of straight-line segments. In Peter Mowforth, editor, *Processings of the British Machine Vision Conference*. Springer-Verlag, 1991.

[14] Mark Fiala. Artag revision 1, a fiducial marker system using digital techniques. `http://www.artag.net/`, November 2004.

[15] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[16] B. Furht. *The Handbook of Augmented Reality*. 2011.

[17] Martin Giupponi. Kalman robusto aplicado en segmentación de videos con texturas dinámicas. Tratamiento Estadístico de Señales. Facultad de Ingeniería, Universidad de la República, 2009.

[18] Rafael Grompone von Gioi, Jérémie Jakubowicz, J.-M. Morel, and Gregory Randall. Lsd: a line segment detector. *Image Processing Online*, mar 2012.

[19] Rafael Grompone von Gioi, Jéremie Jakubowicz, J.-M. Morel, and Gregory Randall. On straight line segment detection. *Journal of Mathematical Imaging and Vision*, 2008.

[20] Robert M. Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *Int. J. Comput. Vision*, 13(3):331–356, December 1994.

[21] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[22] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[23] Monson H. Hayes. *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1996.

[24] Jane Heikkilä and Olli Silvén. A four-step camera calibration procedure with implicit image correction. In *1997 Conference on Computer Vision and Pattern Recognition (CVPR 97), June 17-19, 1997, San Juan, Puerto Rico*, page 1106. IEEE Computer Society, 1997.

[25] Martin Hirzner. Marker detection for augmented reality applicactions. October 2008.

[26] Dr. Hirokazu Kato. Artoolkit: a software library for building augmented reality (ar) applications. `http://www.hitl.washington.edu/artoolkit/`.

[27] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

[28] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[29] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[30] Denis Oberkampf, Daniel F. DeMenthon, and Larry S. Davis. Iterative pose estimation using coplanar feature points. *Comput. Vis. Image Underst.*, 63(3):495–511, may 1996.

[31] J. García Ocón. Autocalibración y sincronización de múltiples cámaras plz. 2007.

[32] Matias Tailanian and Santiago Paternain. Autoposicionamiento 3d. `http://sites.google.com/site/autoposicionamiento3d/`, Julio 2011.

[33] R.Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[34] Daniel Wagner and Dieter Schmalstieg. Artoolkitplus for pose tracking on mobile devices. 2007.

[35] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, pages 666–673, 1999.

[36] Helmut Zollner and Robert Sablatnig. Comparision of methods for geometric camera calibration using planar calibration targets. In W. Burger and J. Scharinger, editors, *Digital Imaging in Media and Education, Proc. of the 28th Workshop of the Austrian Association for Pattern Recognition (OAGM/AAPR)*, volume 179, pages 237–244. Schriftenreihe der OCG, 2004.

[37]

[38] Image processing on line - lsd: a line segment detector. `http://www.ipol.im/pub/art/2012/gjmr-lsd/`, nov 2012.

[39] Vlfeat homepage. `http://www.vlfeat.org`, nov 2012.