

## Práctica 3: Mapa Denso de Disparidad Estéreo

Cree el programa *stereo\_disparity* que recibe como entrada una imagen estéreo, el fichero de calibración estéreo, y como resultado calcula la disparidad entre las imágenes en cada pixel. Para aquellos puntos con disparidad válida, genera como salida un fichero con formato [OBJ](#) que podrá visualizar con el programa [MeshLab](#). Para probar puede utilizar las imágenes en el [siguiente enlace](#).

Uso:

```
./stereo_disparity stereo_image.jpg calibration.yml out.obj
```

Para realizar el proceso, deberá

- 1) Cargar la imagen estéreo
- 2) Rectificar las imágenes
- 3) Utilizar la clase `cv::StereoBM` para realizar el cálculo de la disparidad.
- 4) Convierta la disparidad obtenida a valores 32bits flotantes

```
// Converting disparity values to CV_32F from CV_16S
disp.convertTo(disparity,CV_32F, 1.0);
disparity=disparity/16.f;
```
- 5) Para aquellos puntos con disparidad  $> 10$ , triángule usando las ecuaciones básicas del par estéreo:  $Z = |T| * f / d$ ;  $X = (x - c_x) * Z / f$ ;  $Y = (y - c_y) * Z / f$ ;
- 6) Guarde los puntos a formato obj. Puede utilizar la siguiente función para ello:

```
void writeToOBJ(std::string path, std::vector<cv::Point3f> points){

    std::ofstream file(path, std::ios::binary);
    for(auto p:points)
        file<<"v "<<p.x<<" "<<p.y<<" "<<p.z<<endl;
}
```

En el [siguiente enlace](#) tenéis el fichero resultado para la primera image reconstruction/m001.jpg



Ahora bien, si en lugar de usar la imagen con el tamaño original, la reducimos a la mitad obtenemos la [siguiente reconstrucción](#).



Para hacerlo, se deberá reducir a la mitad también el  $f, c_x$  y  $c_y$ .