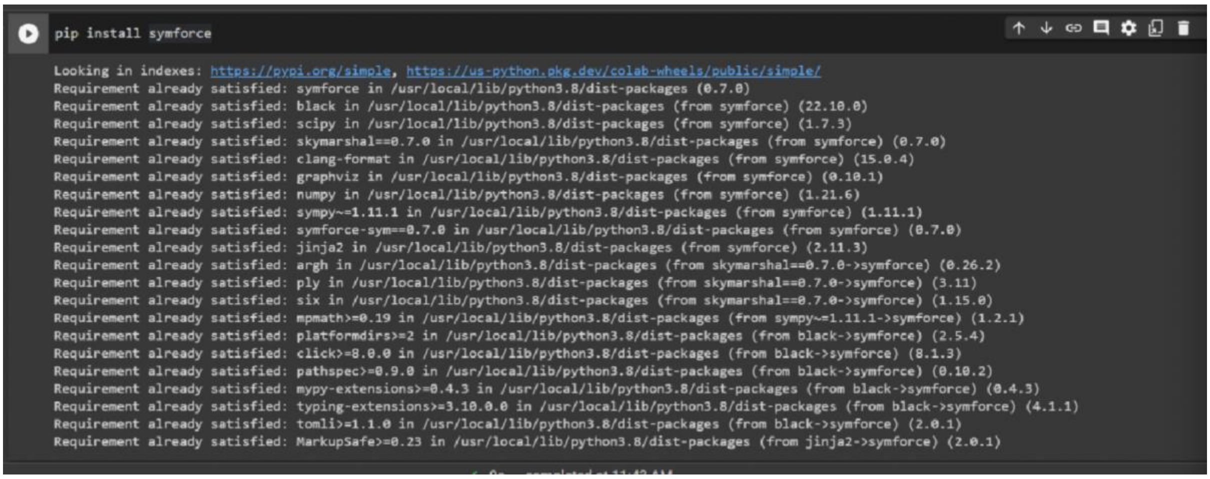
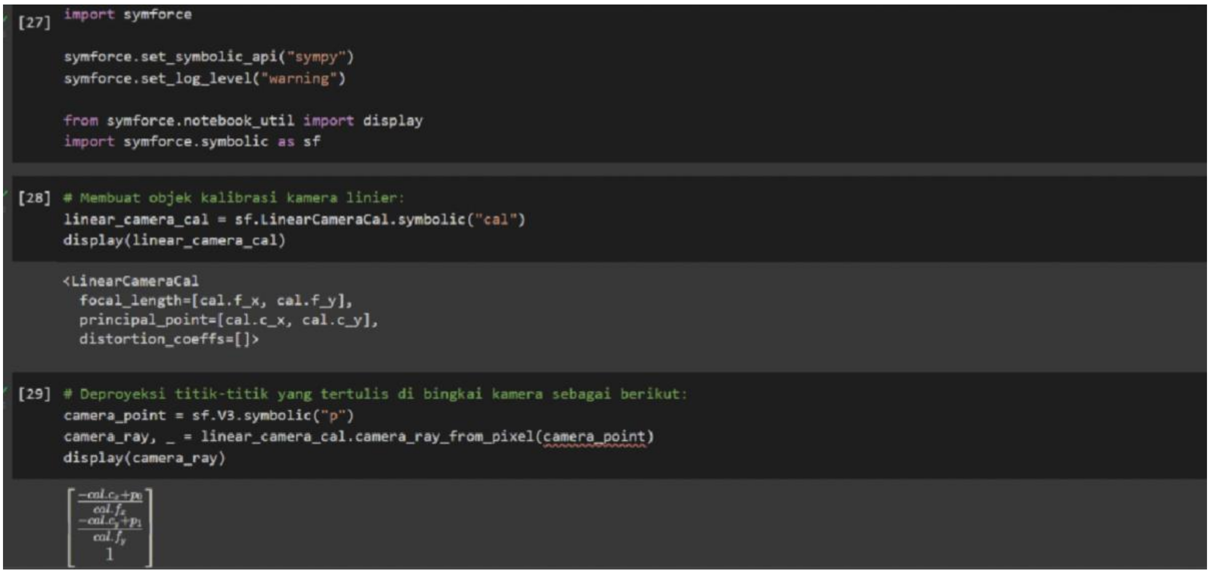


```
pip install symforce

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: symforce in /usr/local/lib/python3.8/dist-packages (0.7.0)
Requirement already satisfied: black in /usr/local/lib/python3.8/dist-packages (from symforce) (22.10.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (from symforce) (1.7.3)
Requirement already satisfied: skymarshal==0.7.0 in /usr/local/lib/python3.8/dist-packages (from symforce) (0.7.0)
Requirement already satisfied: clang-format in /usr/local/lib/python3.8/dist-packages (from symforce) (15.0.4)
Requirement already satisfied: graphviz in /usr/local/lib/python3.8/dist-packages (from symforce) (0.10.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from symforce) (1.21.6)
Requirement already satisfied: sympy~=1.11.1 in /usr/local/lib/python3.8/dist-packages (from symforce) (1.11.1)
Requirement already satisfied: symforce-sym==0.7.0 in /usr/local/lib/python3.8/dist-packages (from symforce) (0.7.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.8/dist-packages (from symforce) (2.11.3)
Requirement already satisfied: argh in /usr/local/lib/python3.8/dist-packages (from skymarshal==0.7.0->symforce) (0.26.2)
Requirement already satisfied: ply in /usr/local/lib/python3.8/dist-packages (from skymarshal==0.7.0->symforce) (3.11)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages (from skymarshal==0.7.0->symforce) (1.15.0)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.8/dist-packages (from sympy~=1.11.1->symforce) (1.2.1)
Requirement already satisfied: platformdirs>=2 in /usr/local/lib/python3.8/dist-packages (from black->symforce) (2.5.4)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.8/dist-packages (from black->symforce) (8.1.3)
Requirement already satisfied: pathspec>=0.9.0 in /usr/local/lib/python3.8/dist-packages (from black->symforce) (0.10.2)
Requirement already satisfied: mypy-extensions>=0.4.3 in /usr/local/lib/python3.8/dist-packages (from black->symforce) (0.4.3)
Requirement already satisfied: typing-extensions>=3.10.0.0 in /usr/local/lib/python3.8/dist-packages (from black->symforce) (4.1.1)
Requirement already satisfied: tomli>=1.1.0 in /usr/local/lib/python3.8/dist-packages (from black->symforce) (2.0.1)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.8/dist-packages (from jinja2->symforce) (2.0.1)
```



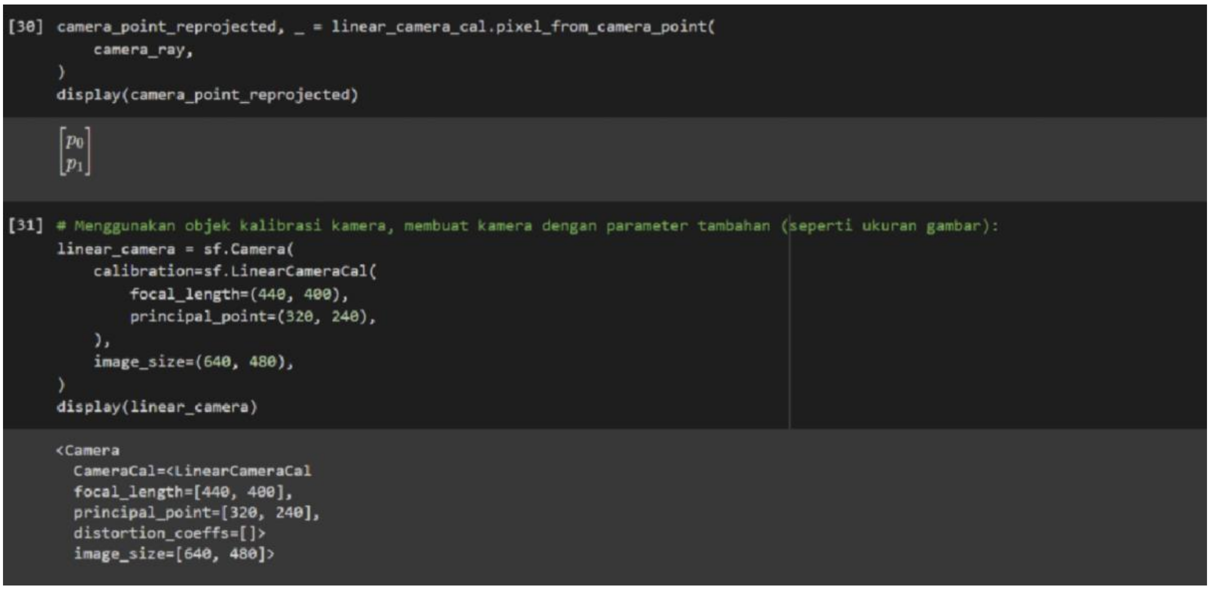
```
[27] import symforce

     symforce.set_symbolic_api("sympy")
     symforce.set_log_level("warning")

     from symforce.notebook_util import display
     import symforce.symbolic as sf
```

```
[28] # Membuat objek kalibrasi kamera linier:
     linear_camera_cal = sf.LinearCameraCal.symbolic("cal")
     display(linear_camera_cal)
```

```
<LinearCameraCal
  focal_length=[cal.f_x, cal.f_y],
  principal_point=[cal.c_x, cal.c_y],
  distortion_coeffs=[]>
```

```
[29] # Deproyeksi titik-titik yang tertulis di bingkai kamera sebagai berikut:
     camera_point = sf.V3.symbolic("p")
     camera_ray, _ = linear_camera_cal.camera_ray_from_pixel(camera_point)
     display(camera_ray)
```

$$\begin{bmatrix} \frac{-cal.c_x + p_0}{cal.f_x} \\ \frac{-cal.c_y + p_1}{cal.f_y} \\ 1 \end{bmatrix}$$

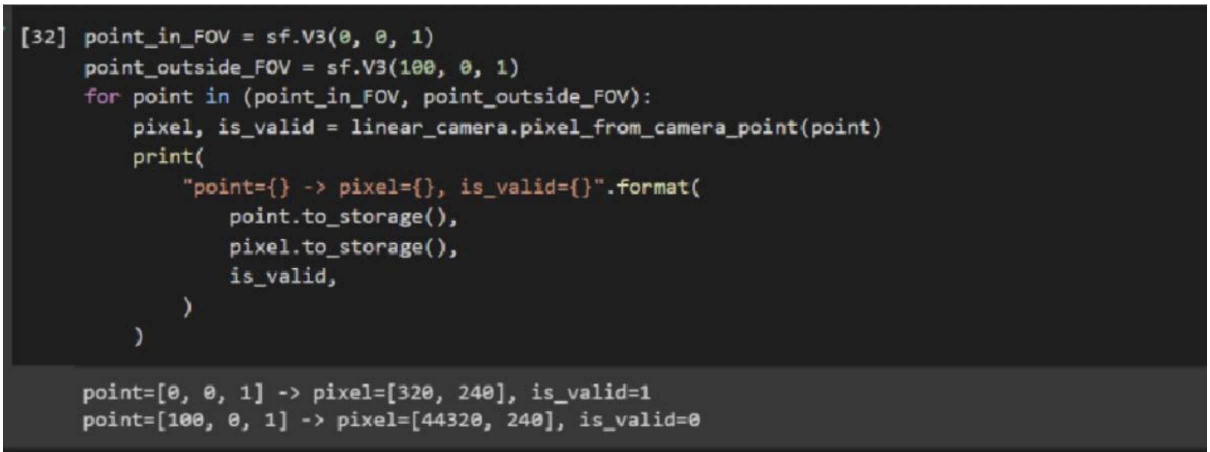


```
[30] camera_point_reprojected, _ = linear_camera_cal.pixel_from_camera_point(
         camera_ray,
     )
     display(camera_point_reprojected)
```

$$\begin{bmatrix} p_0 \\ p_1 \end{bmatrix}$$

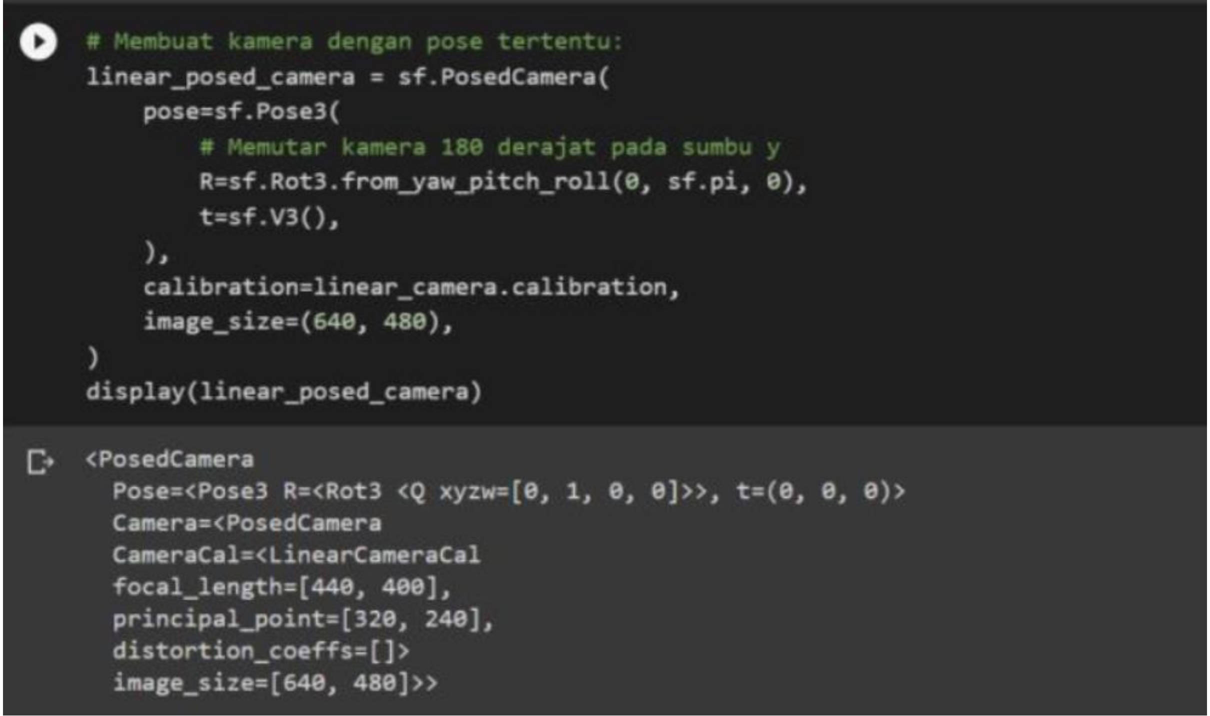
```
[31] # Menggunakan objek kalibrasi kamera, membuat kamera dengan parameter tambahan (seperti ukuran gambar):
     linear_camera = sf.Camera(
         calibration=sf.LinearCameraCal(
             focal_length=(440, 400),
             principal_point=(320, 240),
         ),
         image_size=(640, 480),
     )
     display(linear_camera)
```

```
<Camera
  CameraCal=<LinearCameraCal
  focal_length=[440, 400],
  principal_point=[320, 240],
  distortion_coeffs=[]>
  image_size=[640, 480]>
```



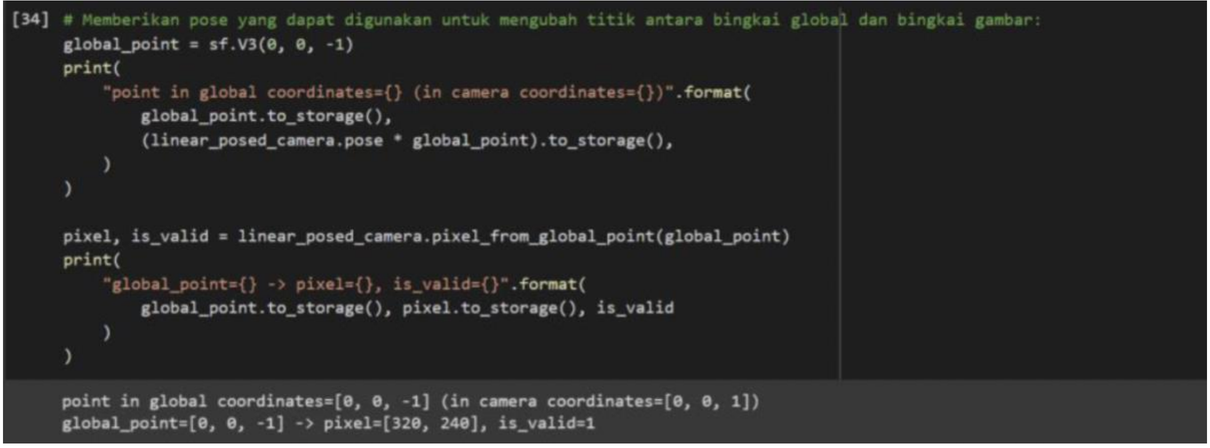
```
[32] point_in_FOV = sf.V3(0, 0, 1)
     point_outside_FOV = sf.V3(100, 0, 1)
     for point in (point_in_FOV, point_outside_FOV):
         pixel, is_valid = linear_camera.pixel_from_camera_point(point)
         print(
             "point={} -> pixel={}, is_valid={}".format(
                 point.to_storage(),
                 pixel.to_storage(),
                 is_valid,
             )
         )
```

```
point=[0, 0, 1] -> pixel=[320, 240], is_valid=1
point=[100, 0, 1] -> pixel=[44320, 240], is_valid=0
```



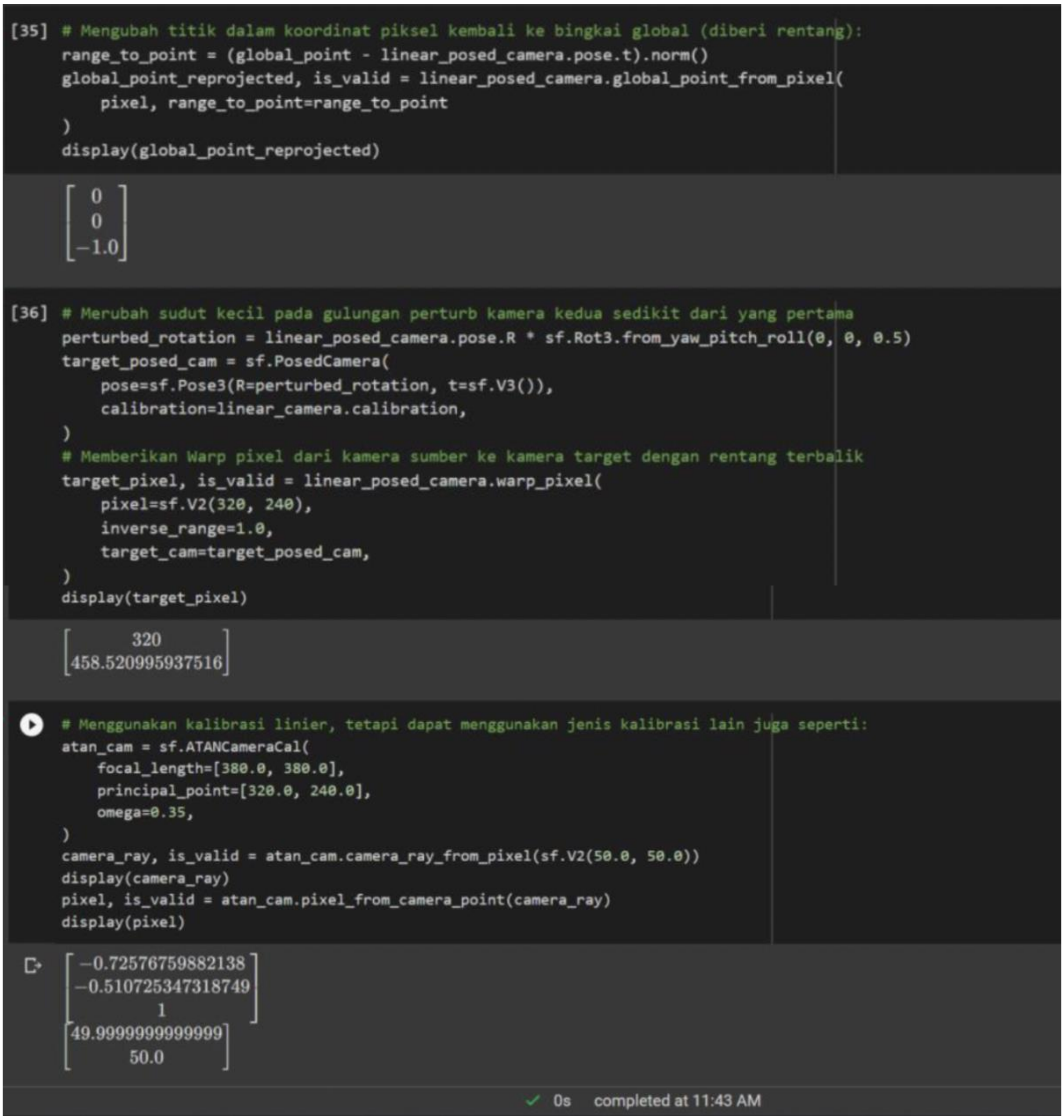
```
# Membuat kamera dengan pose tertentu:
linear_posed_camera = sf.PosedCamera(
    pose=sf.Pose3(
        # Memutar kamera 180 derajat pada sumbu y
        R=sf.Rot3.from_yaw_pitch_roll(0, sf.pi, 0),
        t=sf.V3(),
    ),
    calibration=linear_camera.calibration,
    image_size=(640, 480),
)
display(linear_posed_camera)
```

```
<PosedCamera
  Pose=<Pose3 R=<Rot3 <Q xyzw=[0, 1, 0, 0]>>, t=(0, 0, 0)>
  Camera=<PosedCamera
  CameraCal=<LinearCameraCal
  focal_length=[440, 400],
  principal_point=[320, 240],
  distortion_coeffs=[]>
  image_size=[640, 480]>>
```



```
[34] # Memberikan pose yang dapat digunakan untuk mengubah titik antara bingkai global dan bingkai gambar:
     global_point = sf.V3(0, 0, -1)
     print(
         "point in global coordinates={} (in camera coordinates={})".format(
             global_point.to_storage(),
             (linear_posed_camera.pose * global_point).to_storage(),
         )
     )

     pixel, is_valid = linear_posed_camera.pixel_from_global_point(global_point)
     print(
         "global_point={} -> pixel={}, is_valid={}".format(
             global_point.to_storage(), pixel.to_storage(), is_valid
         )
     )
```

```
point in global coordinates=[0, 0, -1] (in camera coordinates=[0, 0, 1])
global_point=[0, 0, -1] -> pixel=[320, 240], is_valid=1
```



```
[35] # Mengubah titik dalam koordinat piksel kembali ke bingkai global (diberi rentang):
     range_to_point = (global_point - linear_posed_camera.pose.t).norm()
     global_point_reprojected, is_valid = linear_posed_camera.global_point_from_pixel(
         pixel, range_to_point=range_to_point
     )
     display(global_point_reprojected)
```

$$\begin{bmatrix} 0 \\ 0 \\ -1.0 \end{bmatrix}$$

```
[36] # Merubah sudut kecil pada gulungan perturb kamera kedua sedikit dari yang pertama
     perturbed_rotation = linear_posed_camera.pose.R * sf.Rot3.from_yaw_pitch_roll(0, 0, 0.5)
     target_posed_cam = sf.PosedCamera(
         pose=sf.Pose3(R=perturbed_rotation, t=sf.V3()),
         calibration=linear_camera.calibration,
     )
     # Memberikan Warp pixel dari kamera sumber ke kamera target dengan rentang terbalik
     target_pixel, is_valid = linear_posed_camera.warp_pixel(
         pixel=sf.V2(320, 240),
         inverse_range=1.0,
         target_cam=target_posed_cam,
     )
     display(target_pixel)
```

$$\begin{bmatrix} 320 \\ 458.520995937516 \end{bmatrix}$$

```
     # Menggunakan kalibrasi linier, tetapi dapat menggunakan jenis kalibrasi lain juga seperti:
     atan_cam = sf.ATANCameraCal(
         focal_length=[380.0, 380.0],
         principal_point=[320.0, 240.0],
         omega=0.35,
     )
     camera_ray, is_valid = atan_cam.camera_ray_from_pixel(sf.V2(50.0, 50.0))
     display(camera_ray)
     pixel, is_valid = atan_cam.pixel_from_camera_point(camera_ray)
     display(pixel)
```

$$\begin{bmatrix} -0.72576759882138 \\ -0.510725347318749 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 49.9999999999999 \\ 50.0 \end{bmatrix}$$

✓ 0s completed at 11:43 AM