```
import glob
import os
import sys
import random
import time
import numpy as np
import cv2

try:
    sys.path.append(glob.glob('../carla/dist/carla-*%d.%d-%s.egg' % (
        sys.version_info.major,
        sys.version_info.minor,
        'win-amd64' if os.name == 'nt' else 'linux-x86_64'))[0])
except IndexError:
    pass

import carla

IM_WIDTH = 640
IM_HEIGHT = 480
actor_list = []

def image(image):
    matrix_representational_data = np.array(image.raw_data)
    reshape_of_image = matrix_representational_data.reshape((IM_HEIGHT, IM_WIDTH, 4))
    live_feed_from_camera = reshape_of_image[:, :, 3]
    cv2.imshow("", live_feed_from_camera)
    cv2.waitKey(1)
    return

try:
    client = carla.Client('127.0.0.1', 2000)
    client.set_timeout(10.0)
    world = client.get_world()
    get_blueprint_of_world = world.get_blueprint_library()
    car_model = get_blueprint_of_world.filter('model3')[0]
    spawn_point = random.choice(world.get_map().get_spawn_points())
    dropped_vehicle = world.spawn_actor(car_model, spawn_point)

    dropped_vehicle.apply_control(carla.VehicleControl(throttle=1.0, steer=1.0))
    simulator_camera_location_rotation = carla.Transform(spawn_point.location,
spawn_point.rotation)
    simulator_camera_location_rotation.location += spawn_point.get_forward_vector() * 30
    simulator_camera_location_rotation.rotation.yaw += 180
    simulator_camera_view = world.get_spectator()
    simulator_camera_view.set_transform(simulator_camera_location_rotation)
    dropped_vehicle.set_transform(spawn_point)
    actor_list.append(dropped_vehicle)

    ## camera sensor
    camera_sensor = get_blueprint_of_world.find('sensor.camera.rgb')
    # change the dimensions of the image
    camera_sensor.set_attribute('image_size_x', f'{IM_WIDTH}')
    camera_sensor.set_attribute('image_size_y', f'{IM_HEIGHT}')
    camera_sensor.set_attribute('fov', '110')
    # Adjust sensor relative to vehicle
    sensor_camera_spawn_point = carla.Transform(carla.Location(x=2.5, z=0.7))
    # spawn the sensor and attach to vehicle.
    sensor = world.spawn_actor(camera_sensor, spawn_point, attach_to=dropped_vehicle)
    # add sensor to list of actors
    actor_list.append(sensor)
    # do something with this sensor_camera_spawn_point
    sensor.listen(image)

    time.sleep(1000)
finally:
    print('destroying actors')
```

```
    for actor in actor_list:
        actor.destroy()
    print('done.')
```