Руководство

по лабораторной работе №3

программирования робототехнических комплексов

Оглавление

1. Подготовка карты

1.1. Добавление сенсора

В результате выполнения предыдущих имеется робот (см. рис. 1.1.).

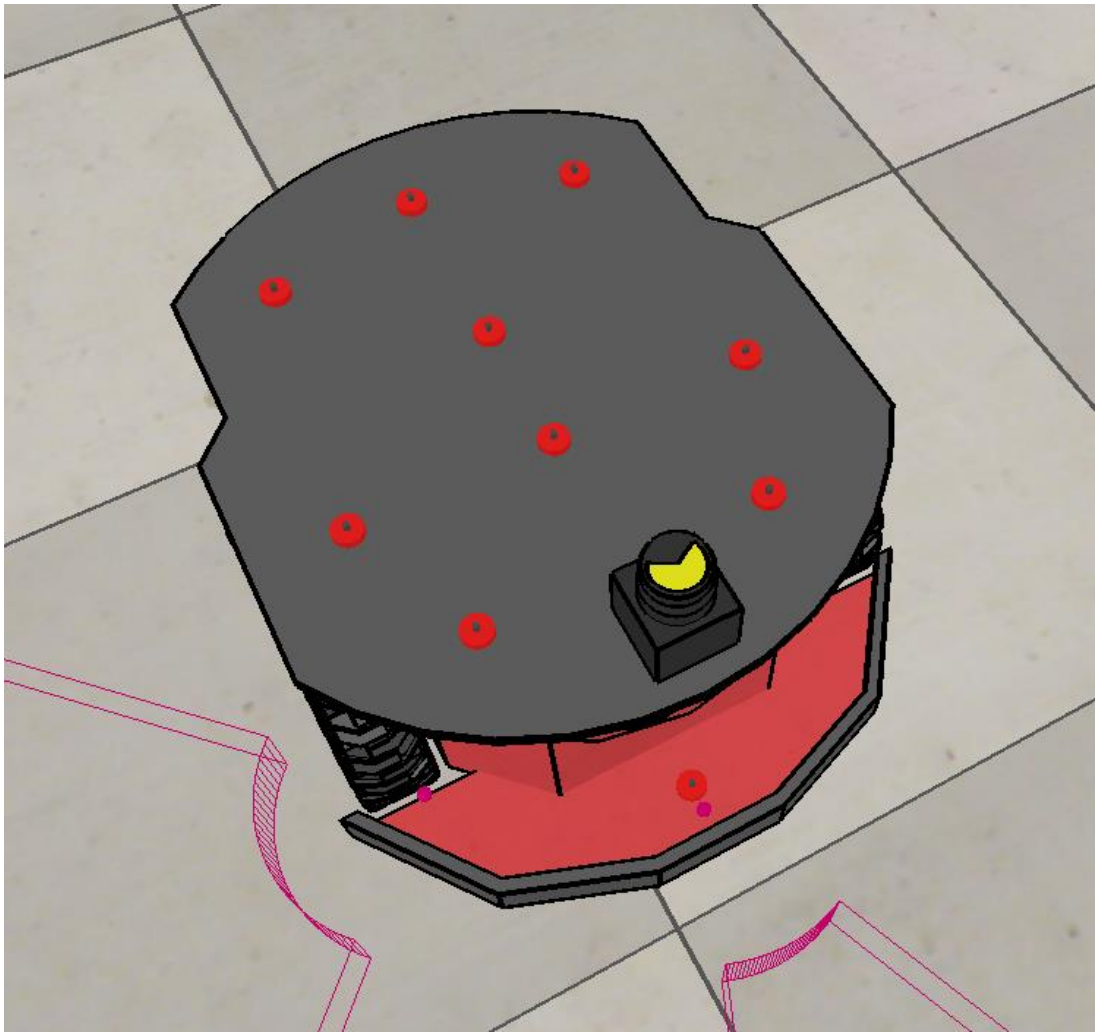

Рис. 1.1. Начальный робот.

Для добавления сенсора: ПКМ -> Add -> Vision sensor -> Perspective type. Для того, чтобы сенсор перемещался вместе с роботом, в иерархии сцены сенсор перемещают в узел робота (рис. 1.2.). Если окно иерархии сцены было случайно закрыто: на верхней панели Tools -> Scene hierarchy.

Установите сенсор над крышей робота. Перемещение объекта не должно вызывать трудностей (кнопка Object/item shift). С поворотом (кнопка Object/item rotate) немного сложнее, потому что после всех манипуляций может выясниться, что изображение перевернуто. Можете просто задать углы поворота как на рис. 1.4.

Если хотите экспериментировать, то поставьте изначально сенсор чуть выше (рис. 1.3.), чтобы при поворотах и горизонтальных перемещениях он не прятался под текстуру. Смотрите видео с сенсора с помощью плавающего окна: ПКМ->Add->Floating view. Выбрать в иерархии сенсор, затем ПКМ по новому плавающему окну->View->Associate view with selected vision sensor.

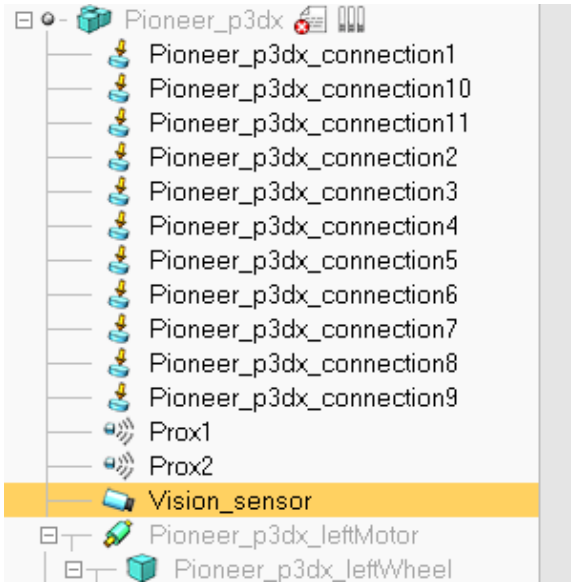

Рис. 1.2. Иерархия сцены.

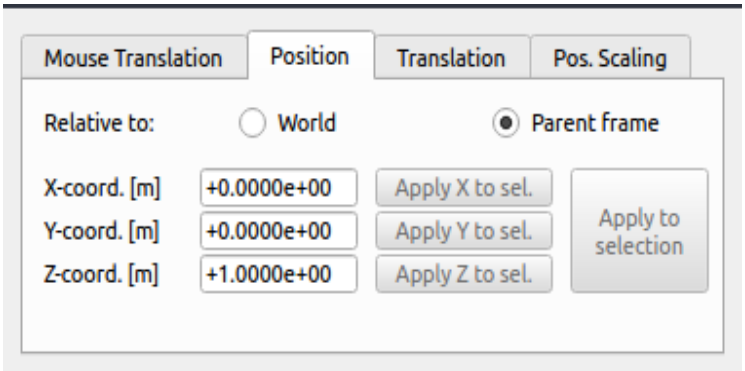

Рис. 1.3. Окно изменения положения.

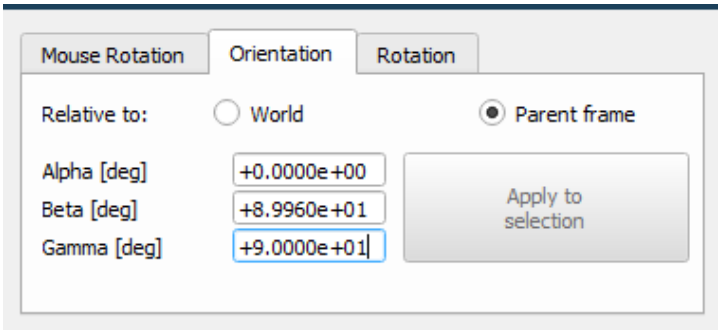

Рис. 1.4. Углы поворота сенсора.

1.2.Добавление объектов распознавания

Чтобы добавить простую фигуру: ПКМ->Add ->Primitive shape. Для остальных фигур можно воспользоваться режимом редактирования формы (Shape edit mode), преобразовав примитивную форму в нужную.

Для примера, преобразуем куб в пирамиду с квадратом в основании. Создадим куб нужного размера, выделим его и активируем режим редактирования формы (рис. 1.5.).

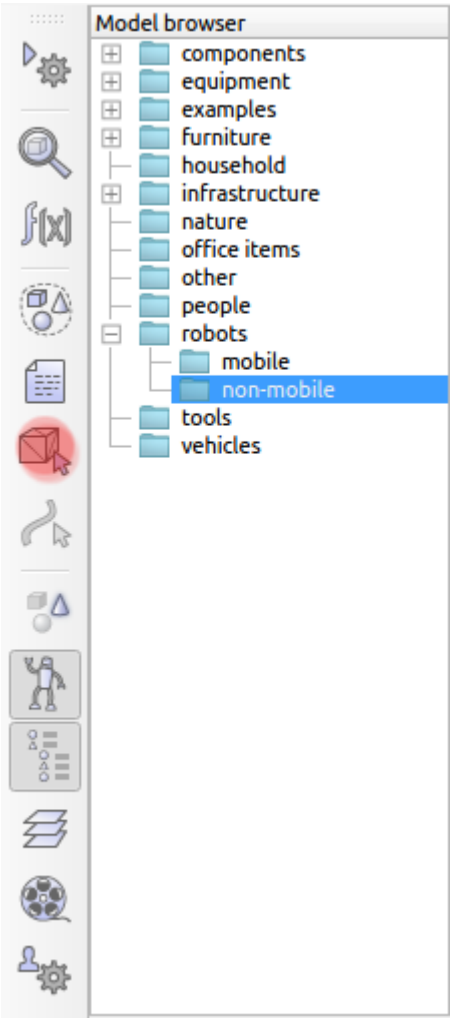

Рис. 1.5. Режим редактирования формы.

Можно заметить, что поверхность фигуры разделена на треугольники. У треугольников есть вершины и ребра. На эти объекты можно взаимодействовать довольно разными способами. Для построения пирамиды в режиме редактирования треугольников один раз разделим большие треугольники (рис. 1.6.).

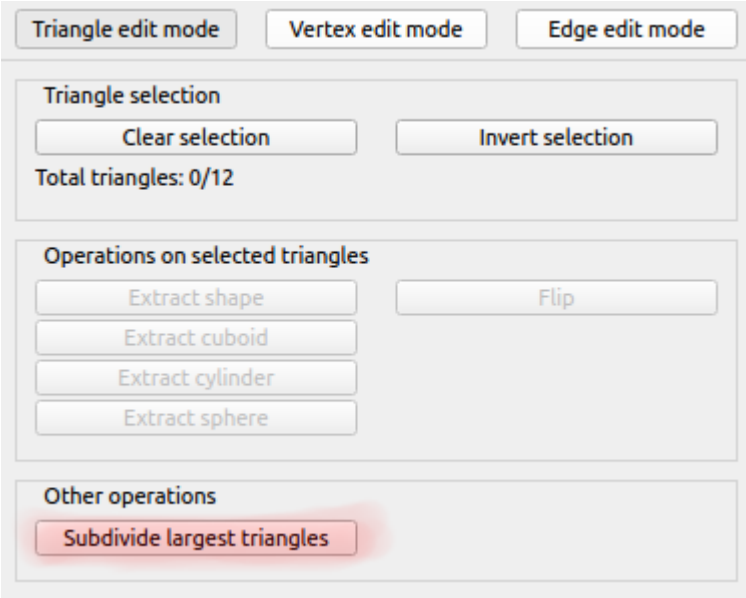

Рис. 1.6. Кнопка разделения треугольников.

Видно, что в центре верхней грани появилась точка пересечения диагоналей. Удалим все треугольники, кроме тех, которые находятся в основании (рис. 1.7.).

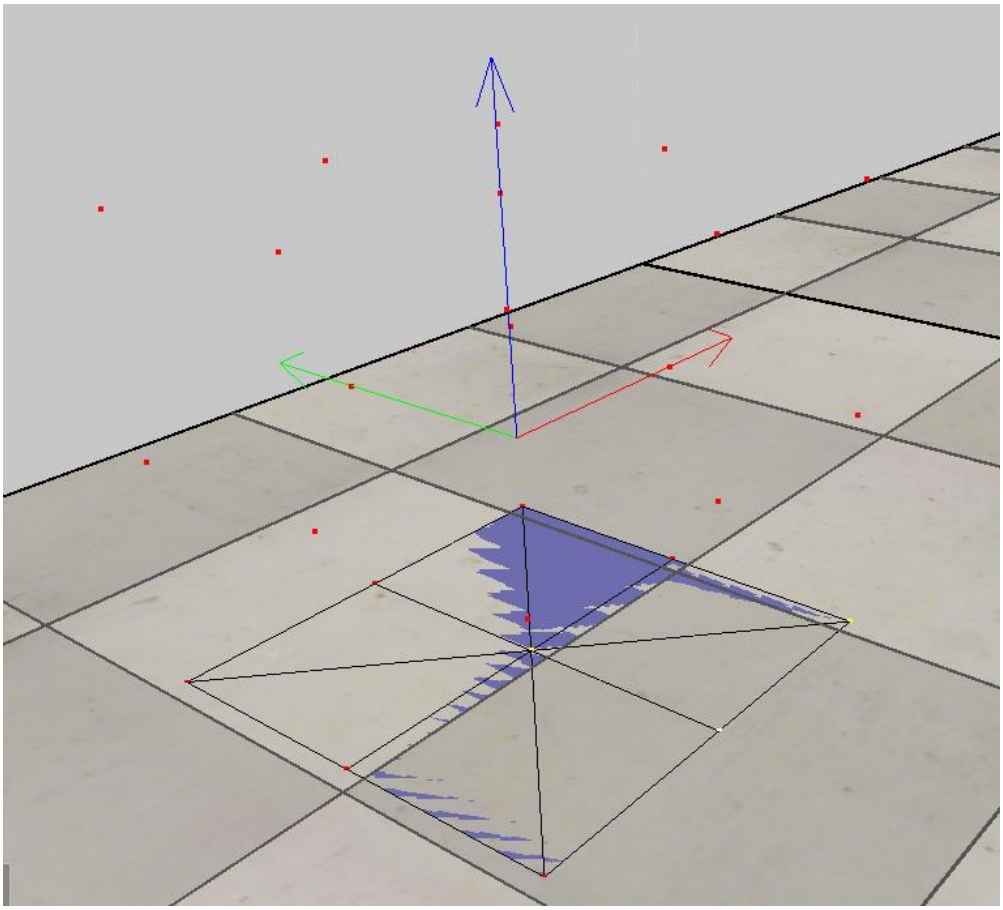

Рис. 1.7. Основание пирамиды.

Перейдем в режим редактирования вершин. Выделим три точки: соседние две в основании и полученную от пересечения диагоналей (если с выделением проблемы, запоминайте точки, а затем выделяйте их на панели слева). Добавим треугольник.

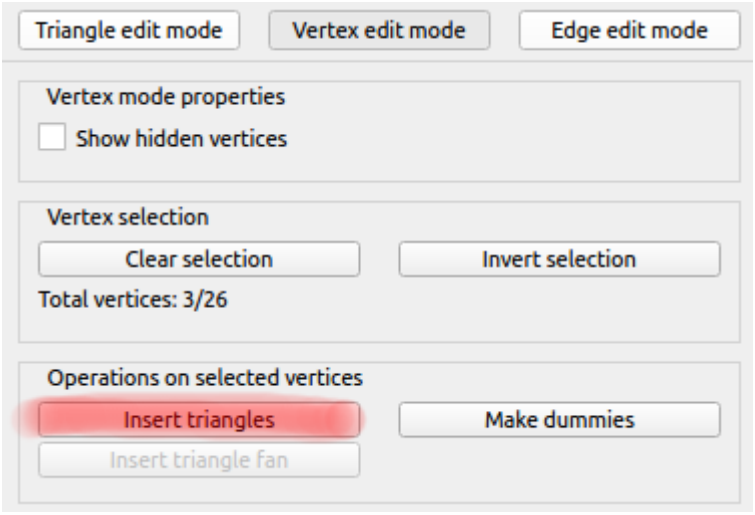

Рис. 1.8. Добавление треугольника по 3-м вершинам.

Подобным образом добавить остальные грани (рис. 1.9.).

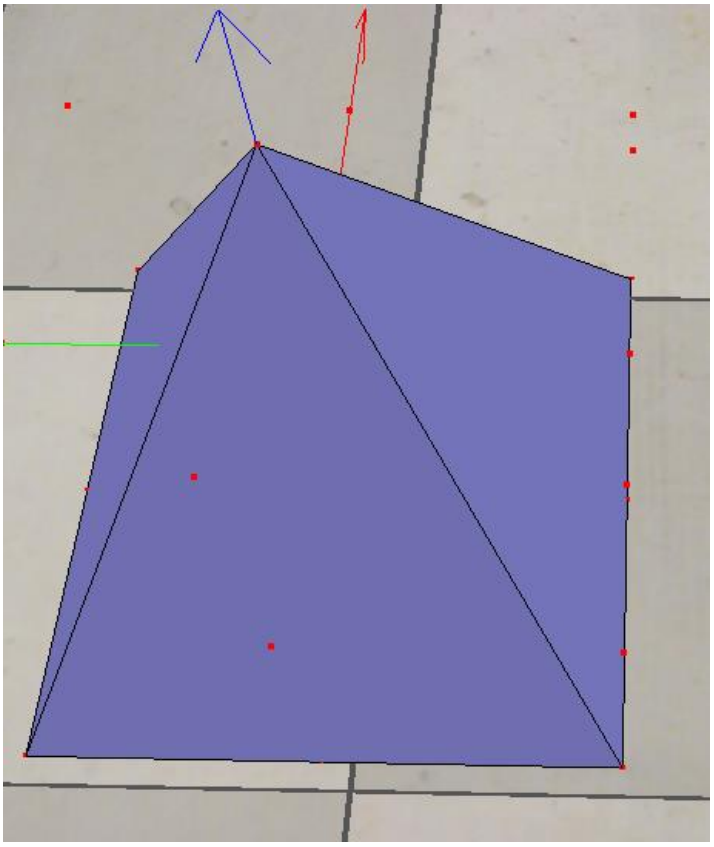

Рис. 1.9. Полученная пирамида.

1.3.Настройка объектов распознавания

У созданных объектов необходимо изменить некоторые свойства для того, чтобы сенсор мог их улавливать. Для этого нужно установить флаг в специальных свойствах объекта (рис. 1.10.).

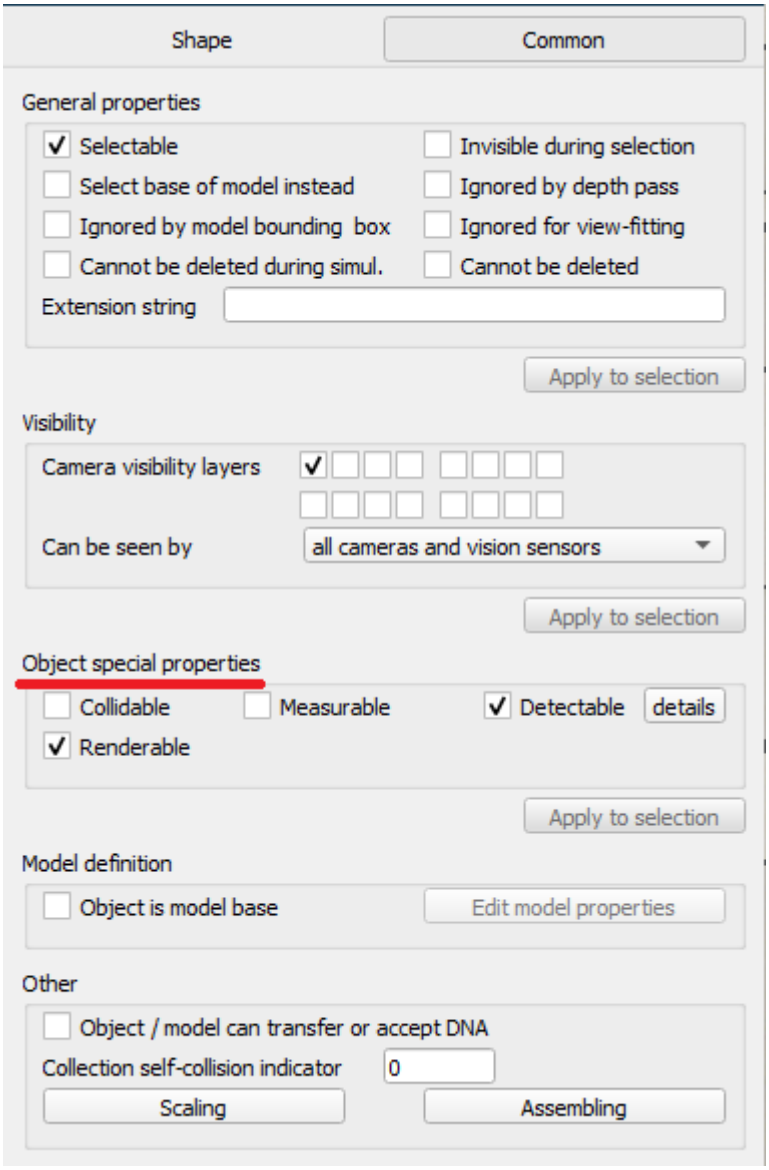

Рис. 1.10. Специальные свойства объекта

2. Установка библиотек и получение изображения из V-REP

2.1.Установка библиотек

Основная библиотека для работы – OpenCV. Не помешают также imutils и Pillow.

Самый простой способ установить библиотеки – использовать менеджер пакетов. Стоит помнить, что OpenCV не полностью открыта. В стандартной версии могут быть не доступны такие возможности как SIFT, SURF и прочие. Если они нужны, то можно установить версию contribute. Для этого удалите обычную версию:

```
pip uninstall opencv-python
```

Затем установите новую:

```
pip install opencv-contrib-python
```

2.2.Получение изображения из V-REP

С помощью функции simxGetVisionSensorImage получаем код возврата, разрешение и само изображение.

```
number returnCode,array resolution,array image = simxGetVisionSensorImage(number clientID,number sensorHandle,number options,number operationMode)
```

Copellia рекомендуют первое использование функции делать в режиме simx_opmode_streaming, все остальные – в simx_opmode_buffer (подробнее о режимах работы функций в [3]).

Полученное изображение будет представлять из себя одномерный массив длиной h*w*c, где h – высота, w – ширина изображения, а c = 3, так как именно столько переменных нужно для задания цвета в RGB формате. Необходимо трансформировать одномерный массив в 3-х мерный, чтобы в последствии было удобно работать с изображением. Можно воспользоваться функцией reshape из библиотеки numpy. При проверке станет ясно, что изображение перевернуто и содержит отрицательные значения, чего быть не должно. Для решения проблемы: 1) до reshape’а перевернём массив и трансформируем его в np.array из типов np.uint8; 2) после reshape’а горизонтально развернем изображение.

Полный код можно посмотреть здесь [4] в функции simulate класса Robot. Там же можно посмотреть, как можно выводить видео с помощью OpenCV. Стоит отметить, что, если аргумент функции waitKey ≤ 0, ожидание будет вечным.

3. Построение примитивного решения

Решим задачу в простом варианте (код в источнике [4], функция imageProcessing), то есть без подсчета всех фигур на карте.

Выделим, для начала, области нужного цвета. Ясно, что понятие «зелёный» - довольно абстрактное. Определим цвет в отрезке от [0, 180, 0] до [50, 255, 56] в формате RGB. С помощью функции cv2.inRange получим маску.

На маске найдем все контуры с помощью функции cv2.findContours. Если контур похож на контур куба, тогда выделим этот контур синим прямоугольником (подробнее в [5]).

4. Возможные проблемы

4.1. Не работает cv2.imshow на Ubuntu / Debian

Может возникать ошибка такого вида с комментарием «The function is not implemented…». Самый простой способ решения проблемы – установка OpenCV версии 4.1.0 [6].

5. Дополнительные ресурсы

1) Расшифровка всех параметров сенсора http://www.coppeliarobotics.com/helpFiles/en/visionSensorPropertiesDialog.htm
2) Расшифровка кодов ответа http://www.coppeliarobotics.com/helpFiles/en/remoteApiConstants.htm#functionErrorCodes
3) Описание того, как работают API функции http://www.coppeliarobotics.com/helpFiles/en/remoteApiModusOperandi.htm

4) https://github.com/P4ll/robotics/blob/master/3/src/robot.py
5) https://www.pyimagesearch.com/2016/02/08/opencv-shape-detection/
6) https://stackoverflow.com/questions/14655969/opencv-error-the-function-is-not-implemented