Тема проекта:
«Электронный поводырь»

Выполнили:
Хисалиев Джалил Дилшадович
Лошкарёв Илья Васильевич
Мышливец Данил Ильнурович

Научный руководилеть:
Васильев Игорь Иванович

Инженерный Лицей КНИТУ-КАИ

Описание устройства

В качестве метода обнаружения препятствия мы использовали 2 дальномера HC-SR04. Проект работает следующим образом: при приближении незрячего человека к препятствию (к примеру, стена) значения расстояния с дальномеров начинают уменьшаться. При уменьшении расстояния вибромоторы начинают вибрировать. Устанавливается следующая зависимость: чем ближе человек к потенциальному препятствию, тем сильнее вибрируют вибромоторы. Причем, у устройства два дальномера, и расположены они под определенным углом для увиличения диаграммы направленности. Пары дальномер – вибромотор работают асинхронно. То есть, если дальность уменьшается слева, то и левый вибромотор вибрирует сильнее.

Мы используем метод передачи информации за счёт вибрации, так как у этого метода есть преимущество: мы не задействуем слуховые рецепторы. Так, как у незрячего человека отсутствует зрение(как ни странно), то нельзя лишать его и воспринимать звуковые сигналы(например, писк светофора при переходе). Если бы мы использовали акустический метод передачи информации, незрячий человек концентрировался бы на наши сообщения и не смог анализировать окружающую обстановку.

Для определения местоположения человека мы использовали инерционно – навигационную систему. Значение ускорения и угловой скорости получаются из акселерометра adxl345 и гироскопа l3g4200d. Частота передачи данных – 62 Гц.

В качестве параллелограмма, который нужно было взять, был использован красный кусок бумаги. Программа работала следующим образом: изначально детектировался красный прямоугольник. Если он не был найден, вибромоторы вибрировали с частотой 5 Гц. В нашем же случае, прямоугольник был замечен с первого раза. Далее я вожу рукой перед камерой, чтобы закрыть доступ прямоугольника к камере. Если объект был виден, но перестал быть видимым, вибромоторы вибрируют монотонно. Так и был взят квадрат.

Изначально, устройство состояло из трех дальномеров. Третий был нужен для детектирования неровной поверхности (ступени, поребрики). Но, в процессе тестирования оказалось, что если поставить основной блок с двумя дальномерами под углом в 45 градусов к земле, то система работает стабильно. В следствии этого изменения мы смогли избавится от третьего дальномера.

При перемещении человека траектория была наикратчайшая. Для определения направления, куда нужно идти человеку, мы рассчитывали угол между точкой назначения и текущей позицией используя теорему косинусов. Если человек отходит от курса, в следствии чего угол между человеком и точкой назначения превышает 8 градусов, то вибромотор стороны, в которую нужно повернуть человеку, чтобы вернутся на курс вибрирует с частотой в 2 Гц.

Для понимания картины опишем работу устройства при приближении незрячего к помехе в качестве стула, при переходе к точке назначения:
1. Человек начинает движение к точке назначения.
2. На расстоянии в два метра от стула, вибромоторы начинают вибрировать, говоря о препятствии.
3. Человек начинает обходить предмет справа.
4. Угол между человеком и точкой назначения увеличивается, в следствии чего начинает вибрировать левый вибромотор с частотой в 2 Гц.
5. Человек поворачивается, и идёт, пока вибромотор не перестанет вибрировать.
6. В итоге, человек обошёл предмет, и вернулся на траекторию маршрута.

Корпус устройства был напечатан на 3D принтере. Для увеличения прочности была применена ацетоновая баня. В качестве вычислительного устройства используется одноплатный микрокомпьютер Raspberry Pi 3 model B.

На фотографии сверху представлен модуль с двумя дальномерами ,модулем акселерометра/гриоскопа и аккумулятором внутри. Raspberry Pi висит в корпусе на поясе пользователя. Дальномеры, модуль акселерометра/гриоскопа и шина питания подключается к RPI про проводам. Выбран именно такой концепт из — за решения, чтобы было меньше выделяющих человека элементов на теле. Равенство.

Глобализация проекта и выход за рамки технического задания

Для практичности устройства(не зря же его делали) мы решили выйти за рамки технического задания. Была реализована работа с GPS модулем по протоколу UART на Raspberry Pi, распознавание речи при помощи Yandex SpeechKit(для того, чтобы задать точку назначения), построение маршрута и синтез человеческим голосом алгоритма перемещения от точки А к точке Б при помощи того же Yandex SpeechKit (для синтеза голоса) и Yandex Maps API(для построения маршрута). При первой просьбе оргкомитета, мы можем выслать исходники версии проекта, предназначенного для перемещения по городу.

Так же, рассматривалось предложение использовать систему стереозрения, используя ROS. Но оно сразу отпало из-за того, что у RPI не хватит вычислительной мощности на такую задачу. А при отправке видиопотока на сервер, где данные будут обрабатываться, получится большая задержка.

Время автономной работы будет зависеть только от аккумулятора. Точно можно сказать, что потребление устройства не превышает 700 mA/h.

Конкуренты

Существует немало прототипов "Электронного поводыря". К примеру The vOICe (seeingwithsound.com). Но у этого проекта есть свой большой недостаток — передача информации с помощью звука. Проблема описывалась выше. Есть также и реализованные варианты, но зачастую у них высокая цена- 10 тыс. рублей и выше, например, UltraCane от компании Sound Foresight Technology Ltd. Себестоимость нашего поводыря- 3300 рублей.
Также, плюс нашего проекта — широкая диаграмма направленности из-за конструкции корпуса, что позволяет детектировать больше препятствий.

Описание работы инерционно-навигационной системы.

Как следует из названия, ИНС обеспечивает измерение и обработку инерционных ускорений, поэтому полученные измерения, сделанные акселерометром представляют собой сумму эффектов, обусловленных линейным ускорением, ориентацей платформы относительно вектора силы тяжести, угловое ускорение платформы при движении по кривой, эффекты силы Кориолиса и, наконец, центробежные ускорение, вызванное вращением Земли вокруг себя.

Игнорируя последние два компонента ускорения можно считать, что измерения, сделанные акселерометром, фактически являются суммой эффектов линейного ускорения, ориентация платформы относительно вектора силы тяжести и углового ускорения платформы. Однако акселерометры будут в основном чувствительны к элементу гравитационного вектора.

Гироскопы являются устройствами измерения угловой скорости. Они измеряют угловую скорость вокруг трех ортогональных осей. Первое интегрирование угловой скорости приводит к углам Эйлера, которые необходимы для описания ориентации. Углы Эйлера обычно даются в аэронавигационном терминале как Pitch, Roll и Yaw

3-осевое ускорение и данные с гироскопа могут быть объединены в нелинейное матричное уравнение, чтобы дать информацию о местоположении и ориентации.

ИНС может использовать данные ориентации, получая данные с гироскопа и интегрируя их, или использование акселлерометра и магнитометра. В первом случае в процессе интегрирования получается большая ошибка, вследствии чего, использование единственного гироскопа недопустимо. Во втором случае использование является долгосрочным, но при появлении крупных металлических предметов, магнитометр сбивается, что приводит к ошибке системы. Но можно использовать оба метода, используя алгоритм Себастьяна Маджвика. Данный алгоритм принимает на вход значения с акселлерометра и гироскопа. На выходе алгоритм выдаёт кватернион. Я не вижу смысла описывать данный алгоритм математически, так как всё это сделано за меня(https://habr.com/post/255661/).

Полученные данные с фильтра Маджвика мы преобразовываем в углы Эйлера, из которых мы можем создать матрицу вращения, обозначаемую R.

$$R(W,\theta,k+1)=\begin{pmatrix}\cos\Delta\theta(k+1) & -\sin\Delta\theta(k+1) & 0\\ \sin\Delta\theta(k+1) & \cos\Delta\theta(k+1) & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$R(V,\phi,k+1)=\begin{pmatrix}\cos\Delta\phi(k+1) & 0 & \sin\Delta\phi(k+1)\\ 0 & 1 & 0\\ -\sin\Delta\phi(k+1) & 0 & \cos\Delta\phi(k+1)\end{pmatrix}$$

$$R(U,\alpha,k+1)=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\Delta\alpha(k+1) & -\sin\Delta\alpha(k+1)\\ 0 & \sin\Delta\alpha(k+1) & \cos\Delta\alpha(k+1)\end{pmatrix}$$

Δθ обозначает инкрементный угол вокруг оси X,
Δα обозначает инкрементный угол вокруг Y-оси.
Δφ обозначает инкрементный угол вокруг оси Z.
Δt обозначает временной шаг.
k - индекс времени.

Итоговая матрица вращения выглядит следующим образом(W = X, V = Y, U = Z)

$$Rotation\ (k+1) = R(W,\theta,k+1)\cdot R(V,\phi,k+1)\cdot R(U,\alpha,k+1)$$

Матрица вращения содержит информацию о ориентация ИНС и порядок вращения тогда, пока эта матрица может быть «достигнута».

Матрица ориентации описывает ускорения без вектора силы тяжести. Она может быть получена следующем образом:

$$Orientation\ (k+1) = Rotation\,(k+1)\cdot Orientation\,(k)$$

Где Orientation матрица определяется как:

$$Orientation^{-1} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

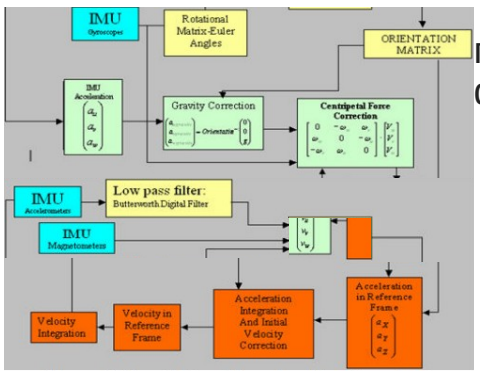
Итоговое ускорение при любом угле вращения устройства вычисляется как:

$$\begin{pmatrix} a_{ugravity} \\ a_{vgravity} \\ a_{wgravity} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}$$

Итак, ускорения по X, Y и Z без вектора силы тяжести вычисляется как:

$$\begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix} = \frac{1}{g} \cdot \begin{pmatrix} a_{ugravity} \\ a_{vgravity} \\ a_{wgravity} \end{pmatrix}$$

Далее эти данные интегрируются два раза. В первый раз итерации интегрирования получается скорость, при второй — получается позиция.

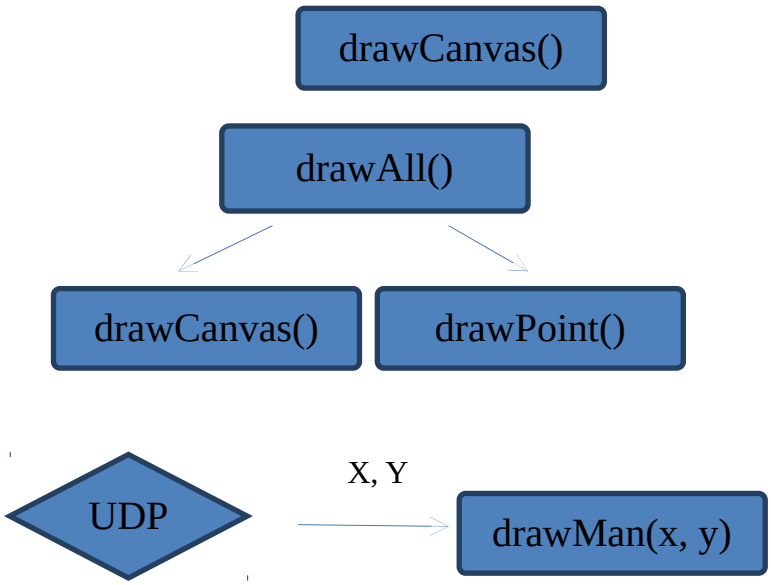


Итоговая схема работы ИНС показана ниже(белый квадрат — фильтр Маджвика)

Также прошу заметить, что в видео при повороте устройства, во время интерирования учитывалось и вектор силы тяжести. Сейчас этот баг исправлен. Причина была в том, что неправильно составлялась матрица поворота.

Описание модуля визуализации проекта

Модуль визуализации проекта представляет собой локальный сайт, на котором происходит отрисовка местонахождения пользователя и точки назначения.
Все это работает следующим образом:
Сначала отрисовывается рабочая плоскость (drawCanvas()), представляющая собой комнату. Дальше, кликом мыши мы отмечаем точку назначения (drawPoint()) и перерисовываем плоскость. Эти координаты отсылаются на Raspberry Pi. А та в свою очередь, каждые 10 милисекунд отправляет координаты пользователя и угол между целью и тем, куда пользователь смотрит. Далее, вызывается функция drwMan(x,y) с координатами человека, что отрисовывает его на плоскости. При каждой итерации, перерисовывается вся плоскость.



Потом происходит вывод на экран местоположения, угла и координат пользователя и точки назначения. Также, просчитывается расстояние между точкой назначения и местонахождением человека.
Процесс обмена данными осуществлен с использованием протокола UDP и ЯП Python.

Описание модуля компьютерного зрения проекта

Для определения объекта данного цвета используется компьютерное зрение с использованием OpenCV3. Кадры для обработки поступают с камеры. Далее применяется перевод кадра в цветовую модель HSV с помощью функции cvtColor(frame, cv.COLOR_BGR2HSV). К полученному кадру применяется цветовой фильтр inRange(hsv, hsv_min, hsv_max), где мы определяем, какой цвет объекта должен быть. После этого происходит размытие Гаусса для кадра GaussianBlur(tresh, (5, 5), 0)
Гаусово размытие с радиусом *r* считается по формуле

$$y(m,n) = \frac{1}{2\pi r^2} \sum_{u,v} e^{\frac{-(u^2+v^2)}{2r^2}} x(m+u, n+v)$$

Далее детектируются края с помощью детектора границ Кенни Canny(tresh, 100, 300). Детектор границ выполняет следующие функции:
— Убрать шум и лишние детали из изображения
— Рассчитать градиент изображения
— Сделать края тонкими (edge thinning)
— Связать края в контура (edge linking)
После этого кадр проходит через морфологическое преобразование morphologyEx(edged, cv.MORPH_CLOSE, kernel)
Кадр в результате преобразования расширяется, а затем сжимается, это используется для уменьшения шумовых выбросов на границах регионов.
Теперь в кадре находятся контуры объектов нужного цвета с помощью функции findContours(closed.copy(), cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE). Данная функция использует алгоритм Сатоши Сузуки: топологический структурный анализ оцифрованных двоичных изображений по границе.
На выходе получается массив контуров, которые проходят циклы обработки аппроксимации approxPolyDP(c, 0.03 * peri, True), где мы указываем сколько точек в контуре нам нужно детектировать. В данном случае 4 точки, что является параллелограммом. Аппроксимации используется для более точного определения контура по точкам, что повышает эффективность нахождение. Также полученный массив проходит через фильтр площадей контуров contourArea(c), где отсекаются мелкие объекты похожей формы по нахождению их площади по контуру. На выходе мы получаем условие детектирования if len(approx) == 4 and area > 1000:
По данной программе можно находить определенные объекты по точкам и цвету.