ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

СОГЛАСОВАНО Научный руководитель Генеральный директор ‘‘NTRlab’’	УТВЕРЖДАЮ Академический руководитель образовательной программы «Прикладная математика и информатика», к. ф.-м. н.
_____ Н. Э. Михайловский «22» марта 2019 г.	_____ А.С. Конушин «___» _____ 2019 г.

Отчет

по исследовательскому проекту

на тему Simultaneous Localization And Mapping

Выполнил
студент группы БПМИ175
ОП ПМИ

О. А. Агапова
И.О. Фамилия
_____22 марта 2019
Подпись, Дата

Москва 2019

Введение

Визуальная одометрия (VO) — метод оценки положения и ориентации устройства с помощью анализа видеопотока с камер, установленных на устройстве (роботе). Ключевой особенностью метода является использование нескольких последовательных кадров для оценки локального (относительного) движения объекта с итеративным построением траектории. В силу этого визуальной одометрии присущи накапливаемые ошибки оценок последовательных перемещений.

В нашей работе рассматриваются в основном алгоритмы для монокулярных камер, хотя некоторые из них допускают обобщения на случай стереоизображений.

У монокулярной одометрии существует фундаментальное геометрическое ограничение: по набору кадров монокулярной камеры без дополнительной информации принципиально невозможно восстановить масштаб перемещения (Scale Ambiguity). Следствием этого является невозможность определения расстояний в абсолютных физических величинах. Существуют технологические методы преодоления этой неопределенности, например, использование сенсора глубины в RGBD камерах или стереокамер. Если же используется монокулярная камера, то инициализация и оценка карты глубин является слабым местом алгоритма.

Близким понятием к визуальной одометрии является задача т.н. Visual Simultaneous Localization and Mapping - SLAM (одновременное картографирование и навигация) — определение положения и траектории движения устройства, а также построение карты территории.

SLAM является более глобальной задачей по сравнению с визуальной одометрией. Одним из краеугольных камней SLAM является т.н. замыкание циклов (Loop Closure) — определение, находился ли ранее подвижный объект в данной области и определение его положения относительно ранее составленного участка карты. Этот прием позволяет парировать слабые стороны визуальной одометрии: неоднозначность

масштаба монокулярной визуальной одометрии и накопление ошибки абсолютного перемещения.

Можно считать, что VSLAM = VO + Построение карты + Loop Closure.

Существующие алгоритмы SLAM позволяют создать трехмерную модель пространства, которая в случае больших пространств (можно представлять себе внутренность здания аэропорта или генерирующий цех электростанции) получается непозволительно большой и не помещается в память мобильного устройства (мы говорим, в том числе, и про лучшие доступные на сегодняшний момент компьютеры для мобильных роботов, например, NVIDIA Jetson TX2). В связи с этим нужны новые алгоритмы, которые хранят 3D-модель пространства более компактно.

В данной работе мы планируем предложить алгоритмы визуальной одометрии, распознающие объекты и определяющие положение устройства относительно этих объектов (семантическая визуальная одометрия). В дальнейшем эти алгоритмы можно дополнить до полноценного SLAM. За счет того, что эти алгоритмы работают с крупными семантически значимыми объектами (например, окна, столбы и т.п.), предполагается, что объем хранения для такой трехмерной семантической карты будет заметно меньше, чем для подробной трехмерной модели того же пространства.

Обзор источников

Обзор [1] разделяет развитие алгоритмов SLAM и VO на несколько периодов.

“Классический период” предложил основные идеи вероятностных решений задач SLAM, например, оценку максимальной “похожести” (maximum likelihood estimation), а также изучал некоторые вопросы, связанные с эффективностью и устойчивостью найденных решений.

В качестве общего обзора этих задач хороши, например, публикации [2, 3], а также [4] и [5].

Следующий период - это то, что обзор [1] называет периодом алгоритмического анализа (2004-2015 гг.).Тогда были изучены фундаментальные свойства SLAM, в том числе наблюдаемость, сходимость и согласованность. В этот период была также понята ключевая роль разреженности в построении эффективных алгоритмов SLAM и были разработаны многие библиотеки SLAM с открытым исходным кодом.

Текущий период развития алгоритмов SLAM авторы обзора [1] называет периодом надежных алгоритмов. Надежность при этом понимается в нескольких аспектах:

1. надежность при длительной автономии;
2. надежность на картах больших размеров;
3. надежность в различных окружениях.

Одним из методов достижения надежности на картах больших размеров, как мы уже упоминали, возможно, является использование семантических подходов к SLAM.

Нужно понимать, что эти термины новые и не вполне устоявшиеся, поэтому каждый автор понимает под ними что-то свое, сходясь с остальными только в общих формулировках.

Понятие Semantic SLAM вводится вместе с обзором современного положения дел в SLAM в статье [6]. Суть задач этого направления в том, чтобы научить устройство понимать, какой смысл несут те предметы, которые его окружают, и от набора значимых точек в памяти устройства, лишенных семантического смысла, перейти к осмысленным и знакомым нам объектам. Карты, создающиеся таким устройством, будут нести не только геометрическую информацию, но и семантическую, а хранение таких семантических карт, особенно трехмерных, будет требовать меньшей памяти, чем в случае карт из облаков точек или треугольных граней.

.

Диссертация [7], в значительной степени опирающаяся на статью [8], видит в семантическом SLAM способ получить трехмерную модель мира с качеством плотной и объемом хранения разреженной. Эти работы опираются на RGBD-сенсоры и используют для распознавания объектов их трехмерные модели,.

Статья [9] предлагает опираться на распознавание предметов с помощью нейронных сетей и строить алгоритмы определения положения не относительно значимых точек, а относительно распознанных предметов. Там же вводятся понятия объектно-ориентированного и семантического SLAM.

Работа [10] предлагает вместо значимых точек использовать специальные точки на семантически значимых частях распознанных объектов, например, центр колеса автомобилей.

Близкий к нашему подход описан в неоконченной публикации [12]. В этой публикации авторы используют нейронные сети для распознавания объектов на 2D изображениях. Затем создается граф, в котором вершины - это положение + ориентация камеры, либо объект, а ребра - расстояния (подробно об идее такого графа можно прочитать в статье [13]). Предположения об объектах, основанные на таким образом полученных данных, позволяют эффективно решать проблему loop closure. В отличии от нас, авторы статьи поставили задачу разработать алгоритм полноценного SLAM, в то время как мы, для начала, хотим предложить только алгоритм визуальной одометрии для распознавания образов (то есть, попытаться улучшить только половину решения задачи, а не весь

SLAM). Кроме того, авторы так же локализуют семантически значимые части распознанных объектов.

Что касается общего подхода к Semantic SLAM, авторы довольно давней работы [11] (2011 год) разделили решение задачи Semantic SLAM на четыре основных подзадачи: выделение характерных значимых точек, классификация и хранение, семантический анализ, и локализация. В этих терминах мы собираемся выполнить первые две задачи. Несмотря на то, что со временем подход к Semantic SLAM мог измениться, основные составляющие (получение и обработка данных, и составление карты на их основе) остались те же, и мы уделим особенное внимание первой из них.

Для целей этой работы мы изучили также статью [14], в которой авторы разрабатывают решают задачу распознавания объектов на основе данных, полученных единственной камерой (а не multi-view). Такая система поможет эффективно угадывать, что это за объект, имея несколько его ракурсов.

Кроме этого мы изучили ряд статей, в числе которых статьи [15] и [16]. В этих работах общим является предположение о том, что для улучшения качества распознавания можно представлять объекты как квадрики (поверхности второго порядка), наиболее похожие по форме на эти объекты. С помощью такого обобщения авторы хотят более компактно хранить информацию о размере, положении и ориентации объектов.

Таким образом, формулировки и решения целого ряда задач, связанных Semantic SLAM и визуальной одометрией, уже освещены в литературе, идея обработки полученных данных нейронной сетью уже упоминалась в некоторых работах, но устойчиво успешных результатов работы таких систем пока не получено. В нашей работе мы попытаемся улучшить эти результаты.

Исследование

Общая схема алгоритма

1. Распознавание неподвижных объектов на 2D-изображениях, полученных с камеры
2. Выявление значимых точек на 2D-изображениях
3. Сопоставление значимых точек на кадрах одной и той же сцены с разных ракурсов
4. Выявление смещения камеры между парами таких кадров и построение 3D-карты местности

Рассмотрим подробнее каждый из пунктов.

Распознавание объектов (object recognition) -- задача, с которой в настоящее время успешно справляются нейронные сети. Мы решили выбрать для написания экспериментального кода датасет (набор картинок, полученных с камер, как правило, эти картинки снимаются и используются специально для таких задач) и модель нейронной сети, уже предварительно обученную на других данных.
Мы выбрали The KITTI Dataset в качестве шаблонного набора картинок. Команда KITTI (Технологический Университет Карлсруэ) использует самоуправляемые автомобили, оснащенные камерами, чтобы производить шаблонные данные для задач компьютерного зрения. Результат их работы -- наборы последовательных (то есть, напоминающих нарезку кадров с видео) фотографий разных местностей, в том числе городской. Среди их фотографий мы нашли наиболее, на наш взгляд, насыщенную сцену европейского города (KITTI 2011_09_26_drive_0005): 160 последовательных фотографий, снятых в движении по улице, на которых видны деревья, машины, пешеходы, велосипедисты, дома и прочие типичные уличные объекты. Такие объекты мы и хотели распознавать, однако не все, а только неподвижные, ведь дальше мы собирались выявлять значимые точки на каждой сцене, а опираться на такой объект,

как, например, машина или пешеход, бессмысленно -- на кадре с одного ракурса такой объект есть, а на кадре с другого ракурса уже нет (уехал).

Далее нам понадобилась обученная модель нейронной сети для распознавания на картинках из датасета.

Нейронную сеть можно представлять как несколько слоев (возможно -- один) нейронов, между которыми передается информация (от каждого нейрона i-го слоя к каждому нейрону i+1-го). У каждой такой передачи информации от нейрона А к нейрону Б имеется некий вес, который определяет, насколько информация у нейрона А значима. Тогда каждый нейрон следующего слоя получает линейную комбинацию: $\sum_{i=1}^{n} x_i w_i$, где x_i -- это значения i-го нейрона на предыдущем слое, а w_i -- вес связи между i-м нейроном и тем, в который переходит информация. При инициализации нейронной сети веса присваиваются случайным образом. Основная задача обучения нейросети -- “настроить” веса в сети так, чтобы ошибка распознавания (или классификации, или любой другой задачи, для которой используется нейросеть) была минимальной.

Обучать нейронную сеть (с учителем) -- значит при помощи специальных функций “настраивать” веса так, чтобы на обучающей выборке, для которой правильные ответы уже известны, она решала задачу с удовлетворительной точностью. Модель, обученная на какой-либо выборке -- это нейронная сеть с уже расставленными весами, то есть та, для которой задача обучения уже решена.

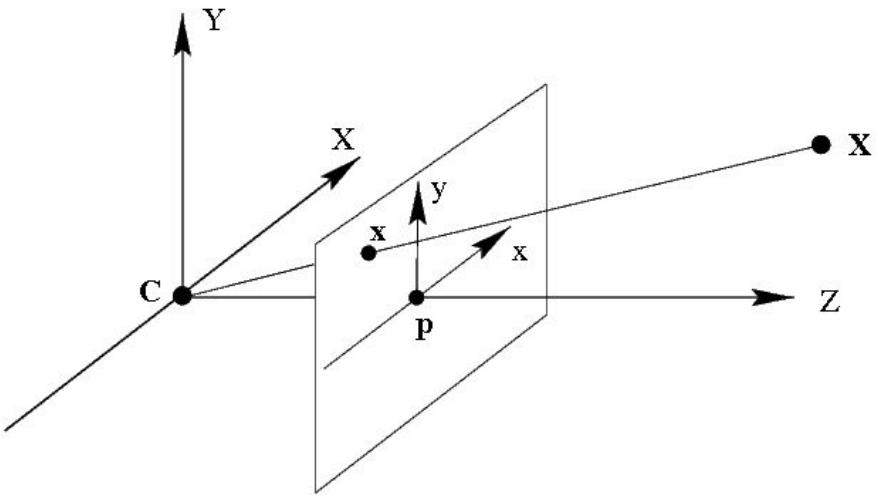
Сначала мы решили использовать модель нейронной сети с архитектурой Mask R-CNN, обученную на датасете COCO. Этот выбор был обусловлен популярностью данной нейронной сети и конкретно этой модели -- она была представлена как демо-версия для быстрого ознакомления с работой Mask R-CNN. Однако вскоре, проведя несколько тестов, мы поняли, что эта модель не подходит для нашей задачи: модель может распознавать всего 81 тип объектов, и среди них почти нет неподвижных. Нас устроили бы такие объекты, как “дверь”, “окно”, “растение”, “здание”, “билборд”.

Тогда задача стала такой: либо найти модель, обученную на выборке с большим количеством объектов, которые можно распознать, либо найти датасет вместо KITTI с

теми объектами, которые могла распознать модель Mask R-CNN с COCO в качестве обучающей выборки.
Мы нашли модель нейронной сети с архитектурой Faster R-CNN, обученную на датасете Open Images v4 (правда, в этом случае вместо модели с уже записанными весами мы нашли код, который с самого начала проводил обучение, таким образом, хоть сами мы сеть и не обучали, сервер с GPU нам понадобился). Здесь обучающий датасет содержал в себе уже не 81, а более 600 классов объектов, что подходило нам намного больше. Такой модели мы дали на вход наши 160 картинок и распознали объекты. Среди множества классов объектов мы выделили такое подмножество: "Window", "Umbrella", "Building", "House", "Tree", "Skyscraper", "Furniture", "Traffic light", "Street light", "Door", "Tower", "Billboard". Это неподвижные объекты, которые встречались на наших изображениях, то есть те, относительно которых можно считать перемещение в пространстве (класс “umbrella” включен сюда только потому, что на наших 160 картинках встречались зонты над киосками и кафе, но не встретилось ни одного зонта, который несёт в руках человек. В общем случае, конечно, этот класс объектов к стационарным отнести нельзя).

Итак, даны кадры нескольких ракурсов одной сцены, например, когда между кадрами камера продвинулась вперед и немного повернулась (то есть кадры последовательные), или ракурсы абсолютно разные, как часто бывает при решении задач loop closure.
В данном случае перед нами стоит задача: по двум кадрам одной сцены определить, как именно сместилась камера, то есть, найти матрицу поворота и вектор перемещения (подробнее эти объекты мы опишем позже). Для этого нужно выделить на изображении точки, характеризующие объект, то есть те, которые видны с большинства ракурсов. Часто (хоть это и не лучшее приближение для самого объекта) в качестве значимых точек для объектов берут углы прямоугольников, в которые эти объекты вписаны. В нашем эксперименте мы решили начать как раз с такого выбора точек.

Для определения смещения и поворота камеры между двумя кадрами нам понадобятся соответствующие алгоритмы. Расскажем вкратце, в чем их суть.



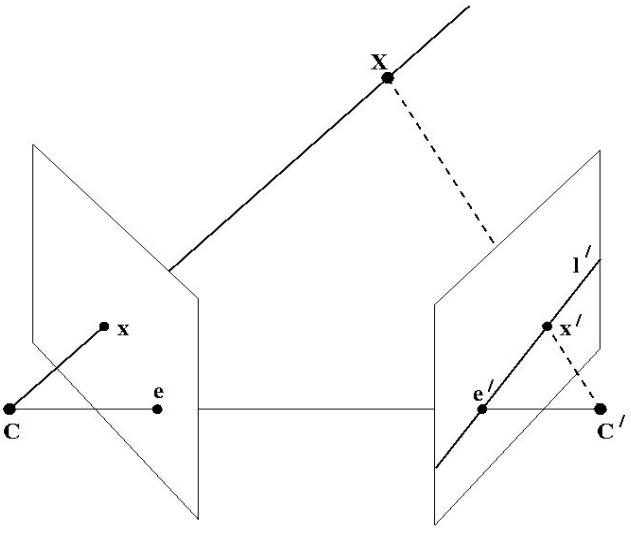
Здесь C — центр камеры. Точка X трехмерного пространства проецируется в точку x — на плоскости изображения. Представляя камеру таким образом, можно ввести понятие матрицы камеры.

Матрицей камеры называется матрица P размера 3x4, для которой верно такое равенство:

x = PX,

где x -- это однородные координаты точки на плоскости изображения, а X -- однородные координаты точки в пространстве.

Сама матрица P зависит от матрицы K внутренних параметров камеры, а так же от матрицы поворота R и вектора перемещения t. Матрица R определяет поворот камеры относительно глобальной системы координат, а вектор перемещения определяется равенством c = -Rt, если c - координаты центра камеры.



$x'^{T} F x = 0.$

Матрица F называется фундаментальной матрицей (fundamental matrix). Ее ранг равен 2, она определена с точностью до ненулевого множителя и зависит только от матриц исходных камер P и P'. Такую матрицу можно вычислить из предположения, что матрица

внутренних параметров камеры неизвестна. Однако матрица внутренних параметров нашей камеры нам известна, поэтому мы можем посчитать другую матрицу -- существенную матрицу.
Формально она определяется как $E = K'^{T} F K$. По существенной матрице мы можем восстановить матрицу поворота и вектор перемещения, и эта матрица и вычисляется в задачах, связанных с перемещением камеры.
Как вычисляется фундаментальная или существенная матрица? Для этого существуют алгоритмы, известные как 8-point algorithm и 5-point algorithm, а так же метод выбора незашумленных данных RANSAC.

Суть "point"-алгоритмов в том, что из уравнений (для каждой точки) вида $x'^{T} F x = 0$, а так же алгебраических свойств матриц F или E можно составить систему линейных уравнений и решать ее относительно элементов искомой матрицы. Утверждается, что для достоверного предположения этой матрицы достаточно 8 точек в случае фундаментальной матрицы и 5 -- в случае существенной. "Point"-алгоритмы позволяют предположить несколько гипотез о том, как выглядит фундаментальная или существенная матрица (в силу неоднозначности решения систем линейных уравнений), в то время как RANSAC позволяет выбрать из гипотез наиболее вероятный вариант. Дальше с помощью SVD-разложения можно выделить матрицу поворота и вектор перемещения. Вариантов набора из матрицы поворота и вектора перемещения может быть четыре (одна и та же матрица и вектор, но с по-разному расставленными знаками), и выбор правильного варианта нужно делать, опираясь на смысл матрицы поворота и вектора перемещения (знаки связаны с направлением поворота и смещения).

Для того, чтобы найти смещения камеры в нашем случае, мы использовали уже реализованные алгоритмы в библиотеке OpenCV. Мы использовали следующие функции:

1. findEssentialMat, которая находит существенную матрицу (ведь наша камера была откалибрована, и матрица К нам была известна) по массивам значимых точек с пары картинок. Эта функция применяет 5-точечный алгоритм, а затем каким-либо из двух методов (RANSAC или LMEDS) выбирает наиболее

вероятную гипотезу. Мы попробовали работу с обоими методами, и особой разницы не заметили, поэтому остановились на RANSAC.

2. decomposeEssentialMat, которая использует SVD-разложение и выдает два предположения о том, как выглядит матрица поворота.
3. recoverPose, которая использует decomposeEssentialMat, а затем выбирает лучший из двух вариантов матрицы R.

Используя эти функции, а также вышеописанные методы и модель нейронной сети, мы на последовательности из 70 картинок (в нашей выборке были фотографии движения вперед, затем поворота и снова движения вперед, и мы взяли именно ту подпоследовательность, которая содержит поворот) выделили неподвижные объекты, записали в массив координаты углов этих объектов, затем для каждой последовательной пары картинок применили функцию findEssentialMat. По найденным матрицам восстановили матрицу поворота и вектор перемещения (мы попробовали сделать это двумя способами: сначала с помощью decomposeEssentialMat, а матрицу из двух выбирали наиболее похожую на предыдущую; а затем с помощью recoverPose. Результат был схожий).

Далее мы решили проверить наши вычисления: посчитать траекторию движения камеры, исходя из полученных нами данных о поворотах и смещениях, и сравнить их с реальной траекторией камеры.

Мы посчитали предполагаемые координаты положения камеры относительно первой фотографии для каждой из последовательных фотографий: имея матрицы поворота и вектор перемещения, для каждой следующей точки мы умножали матрицу на вектор координат предыдущей точки и прибавляли вектор перемещения.

Мы нарисовали график предполагаемых точек и встретились с трудностями: результат оказался не похож на реальную траекторию камеры. Точки группировались в районе одной прямой с небольшим разбросом, а траектория не была похожа на поворот, хоть и была наклонной.

На текущий момент мы находимся в поиске ошибки и улучшения условий нашего эксперимента. Несоответствие наших результатов реальным данным, в принципе, объяснимо. Изложим несколько факторов, которые могли влиять на это:

1. Мало объектов + нейронная сеть обучена неподходящим образом. Наша модель нейронной сети распознает небольшое количество объектов: в общем случае её “устраивают” только неподвижные объекты, которые мы определили довольно грубо: всего 11 классов из 600 вряд ли покрывают все неподвижные объекты. Но даже среди 600 подходящих нам неподвижных объектов немного. В итоге на картинках распознается от 2 до 8 объектов, таких стандартных, как “дерево” или “дом”. **Что делать?** Для решения этой проблемы мы должны найти другую модель или обучить свою на подходящем датасете.
2. Мало точек для использования point-алгоритмов. Столкнувшись с проблемой, мы изучили некоторые существующие эксперименты с использованием 8-point algorithm и выяснили, что, несмотря на теоретический смысл алгоритма, на практике для похожего на реальность результата требуется гораздо больше точек -- чем больше, тем лучше. **Что делать?** Для решения этой проблемы мы можем выделить больше значимых точек, например, по периметру bounding box. Такое решение намного лучше уточнит положение объекта, с которым мы работаем.
3. Слишком грубый выбор значимых точек. Углы bounding box-а -- не самое лучшее приближение к объекту. Углы, как мы и упоминали, часто используются в качестве значимых точек, но в нашем и без того грубом случае лучше определить границы объекта более точно. **Что делать?** Поискать другие способы выделения значимых точек для объекта. Вариантов немало.

Вывод

Таким образом, на текущий момент у нас есть гипотеза о том, что алгоритм будет хорошо справляться с задачами семантической визуальной одометрии, что потом может быть применено в решении задач Semantic SLAM, и сейчас мы находимся в стадии проверки этой гипотезы. У нас есть очень грубый одометр и план на дальнейшее развитие проекта (устранение ошибок и неточностей в том, к чему мы пришли на данный момент).

Список литературы

[1] C. Cadena and L. Carlone and H. Carrillo and Y. Latif and D. Scaramuzza and J. Neira and I. Reid and J.J. Leonard, “Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age”, in IEEE Transactions on Robotics 32 (6) pp 1309-1332, 2017

[2] H. F. Durrant-Whyte и T. Bailey, “Simultaneous Localisation and Mapping (SLAM): Part I”, in IEEE Robotics & Automation Magazine 13(2) pp99 - 110, July 2006

[3] H. F. Durrant-Whyte и T. Bailey, “Simultaneous Localisation and Mapping (SLAM): Part II”, in IEEE Robotics & Automation Magazine 13(3) pp108 - 117, October 2006

[4] F. Fraundorfer and D. Scaramuzza. Visual Odometry. Part II: Matching, Robustness, Optimization, and Applications, in IEEE Robotics & Automation Magazine 19(2), pp78-90, June 2012

[5] C. Stachniss, S. Thrun, and J. J. Leonard, “Simultaneous Localization and Mapping”, from book Robotics in Construction (pp.1153-1176), January 2016

[6] Dominik Maximilián Ramík, Christophe Sabourin and Kurosh Madani, “On Human Inspired Semantic SLAM’s Feasibility”, 6th International Workshop on Artificial Neural Networks and Intelligent Information Processing pp 99-108, January 2010

[7] Renato F. Salas-Moreno, “Dense Semantic SLAM”, PhD Thesis, Imperial College London, October 2014

[8] F. Salas-Moreno, Renato & A. Newcombe, Richard & Strasdat, Hauke & Kelly, Paul & J. Davison, Andrew. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and

Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 1352-1359, 2013

[9] Sünderhauf, Niko & Dayoub, Feras & McMahon, Sean & Eich, Markus & Upcroft, Ben & Milford, Michael. “SLAM -- Quo Vadis? In Support of Object Oriented and Semantic SLAM”, 2015

[10] Nikolay Atanasov, Sean L. Bowman, Kostas Daniilidis, and George J. Pappas. “A Unifying View of Geometry, Semantics, and Data Association in SLAM”, 2018

[11] Choon Ling Tan, Simon Egerton, Velappa Ganapathy. “A Semantic SLAM Model for Autonomous Mobile Robots Using Content Based Image Retrieval Techniques”, 2011

[12] Xiu Li, Yuwang Wang, Yuhan Liu, and Qionghai Dai. “Monocular Semantic SLAM using Object-pose-graph Constraints”, 2016

[13] Beipeng Mu, Shih-Yuan Liu, Liam Paul, John Leonard, and Jonathan P. How. “SLAM with Objects using a Nonparametric Pose Graph”, April 2017

[14] Sudeep Pillai and John J. Leonard. “Monocular SLAM Supported Object Recognition”, 2015

[15] Lachlan Nicholson, Michael Milford, Niko Sünderhauf. “QuadricSLAM: Dual Quadrics From Object Detections as Landmarks in Object-Oriented SLAM”, April 2018

[16] Mehdi Hosseinzadeh, Yasir Latif, Trung Pham, Niko Sünderhauf, Ian Reid, “Structure Aware SLAM using Quadrics and Planes”, April 2018