SLAM
Simultaneous Localization and Mapping
tinySLAM Embedded / Performance

Андрей Михалев
руководитель К.В. Кринкин

CSC

21 декабря 2016

Введение

SLAM

Simultaneous Localization and Mapping — задача построения карты и одновременной локализации робота на этой карте.

- Существует множество решений SLAM-задачи.

В рамках данного проекта работа велась над реализацией tinySLAM для ROS (Robot Operating System)

ROS

Robot Operating System — фреймворк для создания приложений робототехники.

ROS:

- Аппаратная абстракция;
- Обработка информации в узлах (nodes);
- Передача информации между нодами путём обмена сообщениями;
- Сообщения объединяются в очереди (topics);
- Открытый исходный код;
- Ориентирована на Unix-подобные системы;

tinySLAM для ROS

tinySLAM для ROS — ROS реализация исходного алгоритма tinySLAM, которая имеет некоторые улучшения.

Входные данные:

- Данные одометрии робота;
- Данные лазерного сканера;

Выходные данные:

- Положение робота (оценённое с помощью алгоритма);
- Обновлённая карта (карта с предыдущей итерации обновляется в соответствии с полученным лазерным сканом);

Для кого и зачем

Поскольку алгоритмы, решающие SLAM-задачу, выполняются на роботах или автономных транспортных средствах, то можно сформулировать следующие пожелания:

- уменьшить временные затраты на решение SLAM-задачи;
- уменьшить накладные вычислительные расходы на решение SLAM-задачи;

Перечисленные пункты обеспечат более быстрое и менее энергозатратное решение SLAM-задачи.

Задачи на семестр

1. Запустить tinySLAM на Raspberry Pi;
2. Выполнить профилирование алгоритма tinySLAM;
3. Выявить узкие места алгоритма и провести оптимизацию;
4. После внесения изменений измерить производительность и сравнить полученные результаты с baseline;

Окружение

Тестовое окружение:

- Desktop (2.3GHz dual-core Intel Core i5 2410M CPU, 6GB RAM);
- Operating System - Ubuntu 14.04 + ROS Jade;

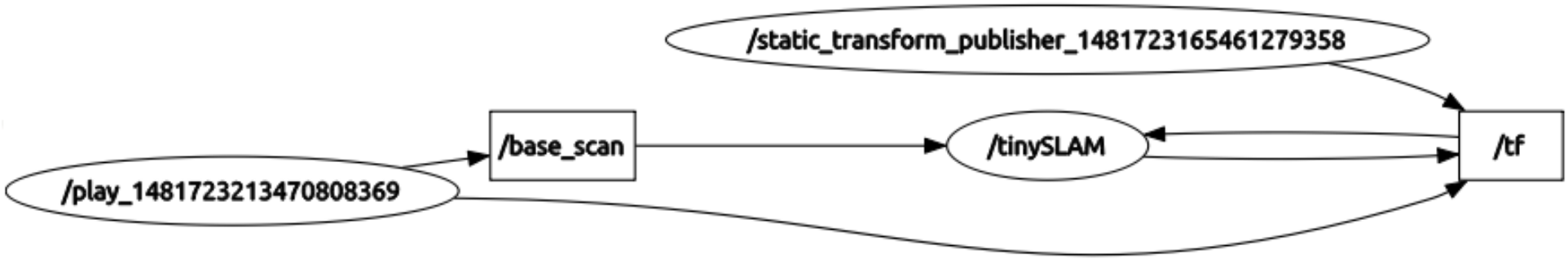
Рабочее окружение:

- Raspberry Pi 2B (900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM)
- Operating System - Ubuntu 16.04 + ROS Kinetic

Запуск tinySLAM

Запускаются 3 ноды и 2 топика:

- *Nodes (обмениваются сообщениями):*
  - /play;
  - /tinySLAM;
  - /statistic_transform_publisher;
- *Topics (очереди сообщений):*
  - /base_scan;
  - /tf;



Входные данные

*Нода **/play**:*

- отправляет входные данные для алгоритма tinySLAM;
- имеет определённый параметр –rate;

*Входные данные - ***.bag** файл:*

- Последовательность сообщений, отправляемых tinySLAM для обработки;
- Имеет определённую длительность;
- Параметр **rate** позволяет регулировать скорость передачи сообщений;

Результаты профилирования

Первый этап профилирования выявил большие накладные расходы при вызове lambda-функций, инициализирующих shared_ptr

- Выделение памяти под shared_ptr;
- Создание shared_ptr;

Профилировщик	Узкое место	% общего времени
valgrind, rate 0.05	vector(vector const&)	11.14%
	_M_release()	13.48%
gperftools, rate 1.6	vector(vector const&)	29.2%
	_M_release()	18.0%
gprof, rate 1.8	vector(vector const&)	53.36%
	_M_release()	25.35%

Результаты профилирования

Второй этап профилирования выявил накладные расходы при работе алгоритма Брезенхема

- Выделение памяти под вектор точек;
- Копирование вектора точек;

Профилировщик	Узкое место	% общего времени
valgrind, rate 1	Добавление элемента в вектор	12.42%
gperftools, rate 1.6	Добавление элемента в вектор	16.4%
gprof, rate 1.8	Добавление элемента в вектор	4.01%

Внесение изменений

Version:

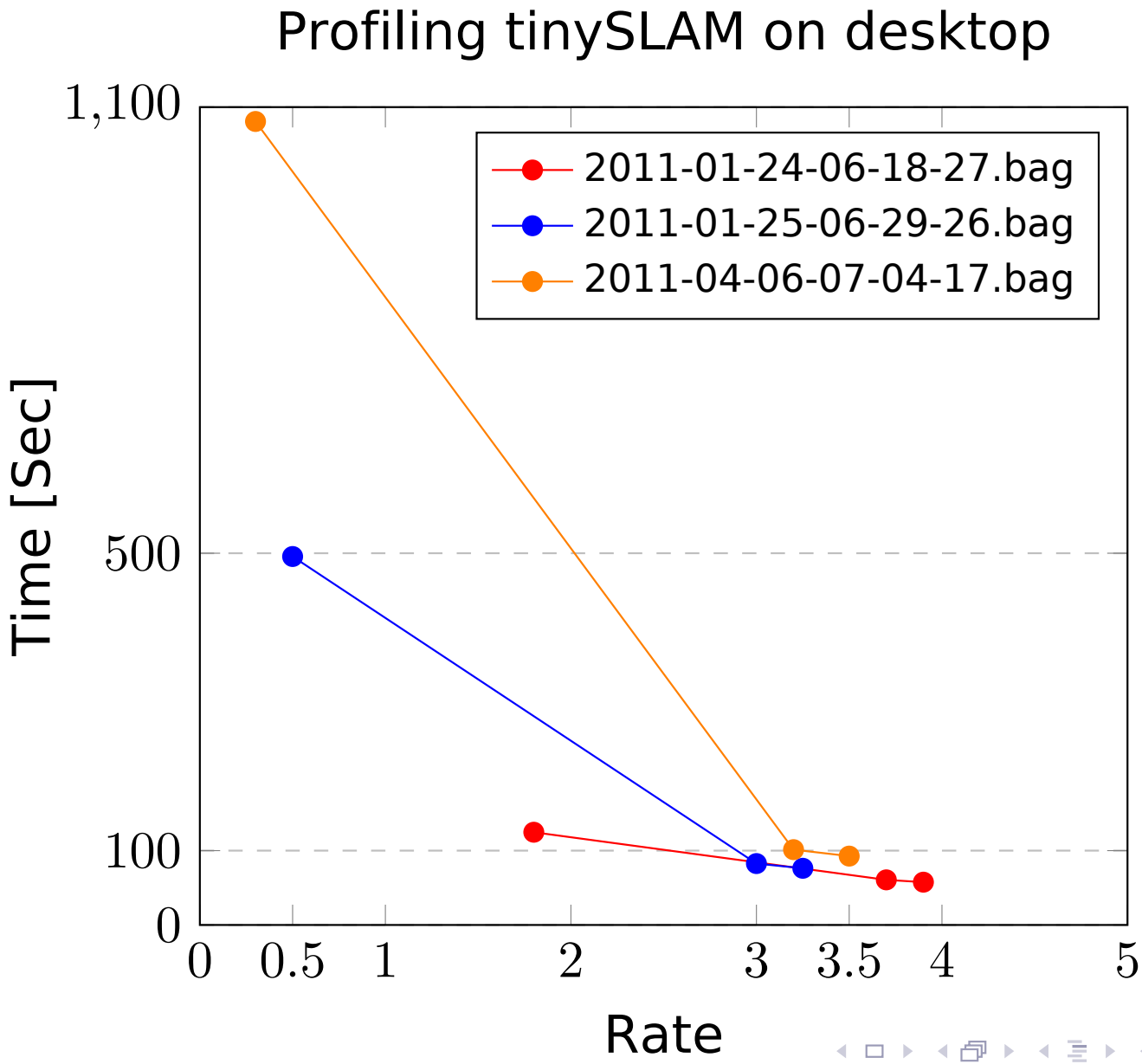
- default - исходный tinySLAM;
- Первый этап (lambda): исключил использование lambda-функций;
- Второй этап (memory): резервирование памяти;

Результаты

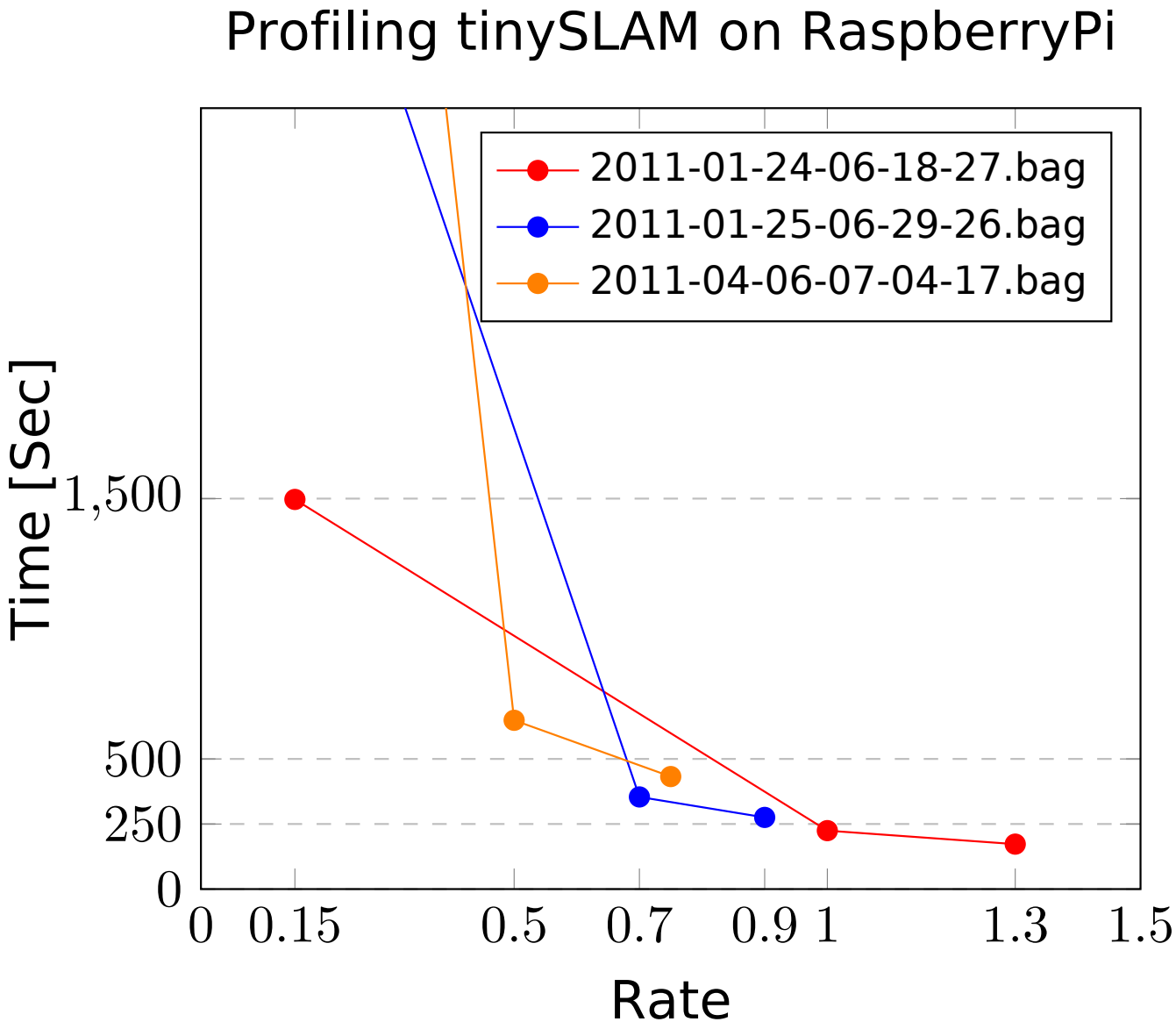
После внесения изменений достигли улучшения производительности.

Version	Desktop, rate	Optimized desktop	RaspberryPi rate	Optimized RaspberryPi
default	1.8	100%	0.15	100%
labmda	3.7	+ 105.6%	1.0	+ 566,7%
memory	3.9	+ 5.4%	1.3	+ 30%

Результаты для Desktop



Результаты для RaspberryPi



Приобретённые навыки

- Работа с ROS;
- Профилирование с помощью различных утилит;
- Работа с RaspberryPi;
- Работа с C++11;
- Работа с ssh + remote desktop;
- Работа с doxygen;
- Настройка IDE Clion для простой работы с ROS;

Используемые технологии и исходный код

Исходный код и входные данные

- ROS реализция tinySLAM;
- Mit stata center data set;

Технологии

- Robot Operating System (ROS);
- allinea;
- gperftools;
- gprof;
- Oprofile;
- valgrind ;
- C++11;

Перспективы

- Кэширование (табличное представление) sin/cos;
- Избавление от std::shared_ptr;
- Замена vector< vector<T> > на двумерный массив;

Вопросы

Вопросы?

Сложности

- Разработка под ROS без IDE;
- Запуск профилировщиков требовал определённого подхода;
- Профилировщики требовали длительного эмпирического подбора **rate**;
- Профилирование на RaspberryPi было затруднительным;
- tinySLAM объёмный проект, в котором порядка 20 классов;
- Поиск узких мест в исходном коде занимал много времени;