View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by CiteSeerX

Cloud-Based Realtime Robotic Visual SLAM

Patrick Benavidez, Mohan Muppidi, Paul Rad, John J. Prevost, Ph.D., and Mo Jamshidi, Ph.D.Lutcher Brown Endowed Chair Professor
Autonomous Control Engineering Lab
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio
USA
patrick.benavidez@gmail.com, [mohan.muppidi, paul.rad, jeff.prevost]@utsa.edu, moj@wacong.org

***Abstract*—Prior work has shown that Visual SLAM (VSLAM) algorithms can successfully be used for realtime processing on local robots. As the data processing requirements increase, due to image size or robot velocity constraints, local processing may no longer be practical. Offloading the VSLAM processing to systems running in a cloud deployment of Robot Operating System (ROS) is proposed as a method for managing increasing processing constraints. The traditional bottleneck with VSLAM performing feature identification and matching across a large database. In this paper, we present a system and algorithms to reduce computational time and storage requirements for feature identification and matching components of VSLAM by offloading the processing to a cloud comprised of a cluster of compute nodes. We compare this new approach to our prior approach where only the local resources of the robot were used, and examine the increase in throughput made possible with this new processing architecture.**

Keywords—cloud, cooperative VSLAM, indoor robot, VSLAM

I. INTRODUCTION

There are many approaches to robot navigation. Global Positioning System (GPS) is most often the approach of choice when the robot is operating in a theatre where a GPS signal is present. Often times, however, using GPS for robot navigation and localization is not possible due to lack of signal. This occurs when the robot is operating inside a structure, or building, that blocks the reception of the GPS signals. In these situations, algorithms such as video based Simultaneous Localization And Mapping (VSLAM) can allow robots to track and keep local maps of their relative positions within their environment. VSLAM works by using a camera mounted on the robot to periodically take pictures of their immediate surroundings and extracting key features from the images. The robot can determine where it is in the local environment by comparing features to a database of images taken of the environment during prior passes by the robot.

There are many algorithms such as Scale Invariant Feature Tracker (SIFT) [1], Speeded-Up Robust Features (SURF) [2], Features from Accelerated Segment Test (FAST) [3], and Oriented FAST and Rotated BRIEF (ORB) [4] that are typically used for feature keypoints detection. Each of these algorithms are capable of detecting multiple robust features for use in VSLAM. Typically, VSLAM using these feature detection algorithms require storage of hundreds, or possibly thousands, of images to be able to properly ascertain the location of a robot in a local environment. This creates processing difficulties for the robot because the key features must be extracted then compared with the images in the database in a realtime operation.

The rest of this paper is organized as follows. Section II presents our survey of existing approaches. Section III covers the proposed algorithm and computing model. Section IV presents comparative results for our algorithm versus existing well-known algorithms. Conclusions are presented in Section V.

II. BACKGROUND

The processing speed was examined in prior work by the authors of this research [5] and a mechanism for adequately processing by limiting the image size at runtime was developed. The effect of the processing algorithms proposed was experimentally tested and shown to allow for proper image recognition within realtime operation constraints.

Fuentes-Pacheco et al. [6] discussed the problem of dynamic environments in SLAM. The authors mentioned the importance of having reliable algorithms with appreciable performance under conditions like variable light, occlusions, featureless regions and other unforeseen situation. Intensity variations are problematic indoors as there are many sources of light that can project uneven light intensities on the captured scene. If a feature detector cannot work well under varying lighting conditions then mismatches will occur and the resulting pose error will be high.

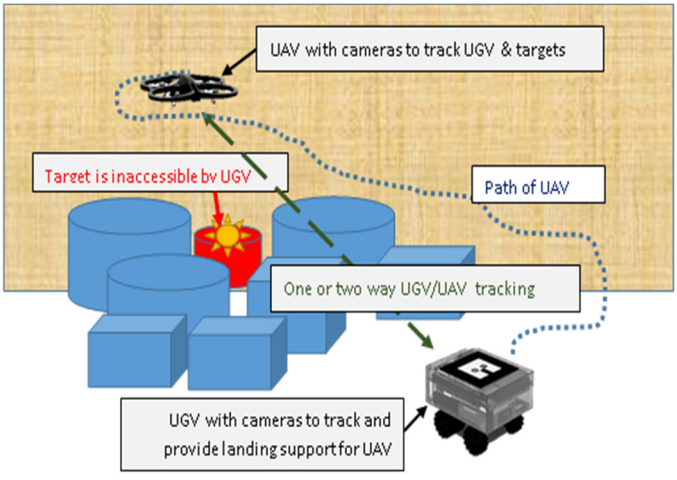

Fig. 1. Indoor cooperative robots to utilize realtime VSLAM

Variation in camera pose is an issue in indoor robotics as images can be taken from about any direction while navigating in small enclosed spaces. The problem is magnified when taking images from a heterogeneous mix of cameras with different fields of views and calibrations. This is especially the case as shown below in Fig. 1 with two camera sources. An example indoor system for deploying VSLAM is depicted in Fig. 1, from [7], where an Unmanned Aerial Vehicle (UAV) is cooperating with an Unmanned Ground Vehicle (UGV). With image information coming from multiple poses within the environment from both camera sources, as in Fig. 1, lighting and pose variation are problematic. Differences in intrinsic properties of cameras will provide some additional variation in captured image quality, thus affecting the resulting feature matching. To achieve sufficient processing of environmental data for a limited range of working conditions, tradeoffs and sacrifices are often made. For example, processing speed must be made a high priority or the resolution of processed sensory data will suffer. Often times, data compression or dimensionality reduction methods are employed to increase processing speed.

One of the primary algorithms within VSLAM is feature matching. Once the key features have been extracted, they can be compared to a known database to determine matches. New topologies of computer architectures can implement the database search with one or more nodes distributed either locally or connected via a network. Parallelization of the database search is therefore possible and can either occur at the processor or the network level. Clipp et al. in [8] presented a multi-process CPU and GPU-based system for performing SLAM. Their system exploited the computing power within the video card to perform operations on images.

In [9], Hunziker et al. presented a real-time cloud-based implementation of SLAM running in the cloud. Much emphasis in their work, however, was not to show the performance of SLAM but the topology of their configuration that enabled the parallelization.

The contribution made in this paper is the implementation of a ROS-based VSLAM algorithm hosted in a multi-node cloud. We show that the sacrifices made in our prior work to achieve the desired performance metrics are no longer necessary and that an optimal result is obtained when the processing is actually distributed across both the local robot resources as well as the scalable resources available in the cloud. A discussion of the parallelization potential of the VSLAM algorithm is also presented to quantify the benefits of employing a cloud-based distributed processing approach.

III. THEORY

A. *Proposed Algorithm*

Assuming that well-defined database of landmark images is obtainable, search in the neighborhood of the previous location of the robot can get us its current location. Our algorithm relies on ORB features and brute force matching to find the best landmark match for current image input.

Fig. 2 depicts the flow chart of the search algorithm. Features of current image input are extracted and then compared with features of landmark images to get the current location of the robot. In our algorithm we propose to use parallel neighborhood search to expedite the search process. The neighborhood of the present location is divided into 9 different parts and search in each part is carried out by a cloud node. In the case of failure to find the proper match, the neighborhood is extended and now this extended neighborhood is again divided into 9 parts to carry out the search operation. Ideally this process can carried out until the best landmark is found. But due to practical constraints after 2 or 3 successive failures complete database search is carried out to the get the present location. The process continues for each data point processed.

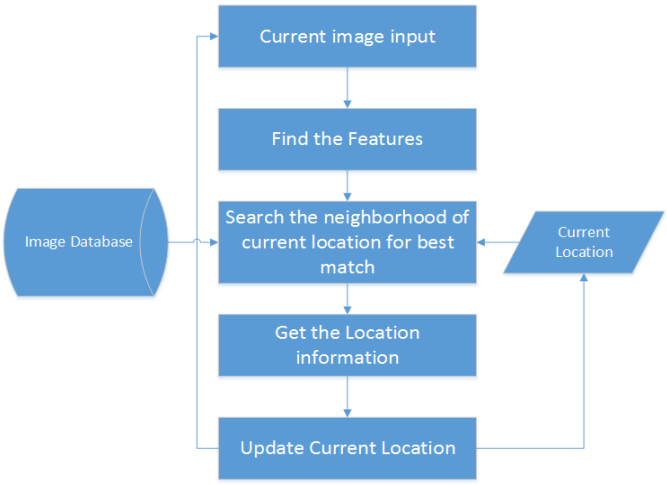

Fig. 2. Flow chart for proposed method

B. *Network Setup*

A wireless switch configuration is made using a D-LINK DIR-655 wireless router. In the configuration, the router has a reserved IP on the building network LAN (10.x.x.x address space), its DHCP disabled, and a building network LAN cable connected to one of its switched connections. This configuration allows wireless clients to connect to the network with IP addresses on the building network LAN (10.x.x.x address space), allowing them to be directly addressable by the local campus cloud. Fig. 3 depicts the network configuration of the system.

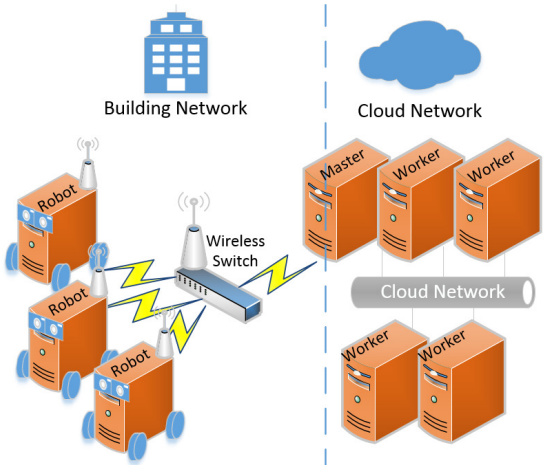

Fig. 3. Cloud Robot Network Configuration

C. Message Passing Configuration (ROS)

Throughout our system we use the Robot Operating System (ROS) middleware. ROS provides tools for managing global access to parameters via a parameter server, data via *topics* using a publisher/subscriber model, across a network of connected processes called *nodes*. Input is obtained, in an event driven manner, when subscribed topics have new data to be processed. Data packets are sent by the publisher(s) of topics when ready to be used.

A similar configuration to the popular Message Passing Interface (MPI) was developed on our local cloud using ROS. ROS *topics*, socket-based message passing that use the name of the topic as the key to connection management, and inherit distributed behavior of ROS *nodes* provided the basis for the system. A single ROS network, or collection of ROS nodes, can be distributed across multiple computers. To manage connectivity in the system, ROS utilizes a master and worker hierarchy. A master in the system contains the core ROS process, *roscore*, which facilitates communication between all nodes. To enable networked ROS functionality, both the master and worker need parameters to be set in order to specify the network configuration. On the master, the personal network information is reflected as being that of the ROS master computer. The master simply needs to indicate its own network information. Workers need to identify the IP address of the master and their own IP address. Both the worker and master must be directly addressable by one another.

Flow of data in the system is as described below. An image topic from the robot provides the input to the cloud computer system. The master processes the image for its features and publishes features to the workers. Workers publish the highest ranked matches in the database to the master. The master ranks the matches and determines the robot's position. Fig. 4 shows the configuration of ROS nodes in the system to achieve a MPI-like configuration.

D. Simulation Setup

A ROS node, running on a computer acting as a mobile robot with video input, is used to play back the contents of a dataset. Playback is performed approximately at the real-time rates the frames were taken from the camera. The purpose of the real-time playback mode is to explore the feasibility of the approach in real-time and determine what alterations are necessary to make the algorithm feasible. For validation purposes, we selected a set of freely available benchmark datasets which contain RGB-D images and an accurate ground truth trajectory taken by a VICON camera system [10].

E. Distribute Cloud Architecture Span ROS and Private Cloud Nodes

1) OpenStack Cloud Architecture

OpenStack is an open-source cloud management software, which consists of several loosely coupled services, designed to deliver a massively scalable cloud operating system for building public or private clouds. To achieve this, all of the constituent services are designed to work together to provide a complete Infrastructure as a Service (IaaS). All the services collaborate to offer flexible and scalable cloud solution using API .

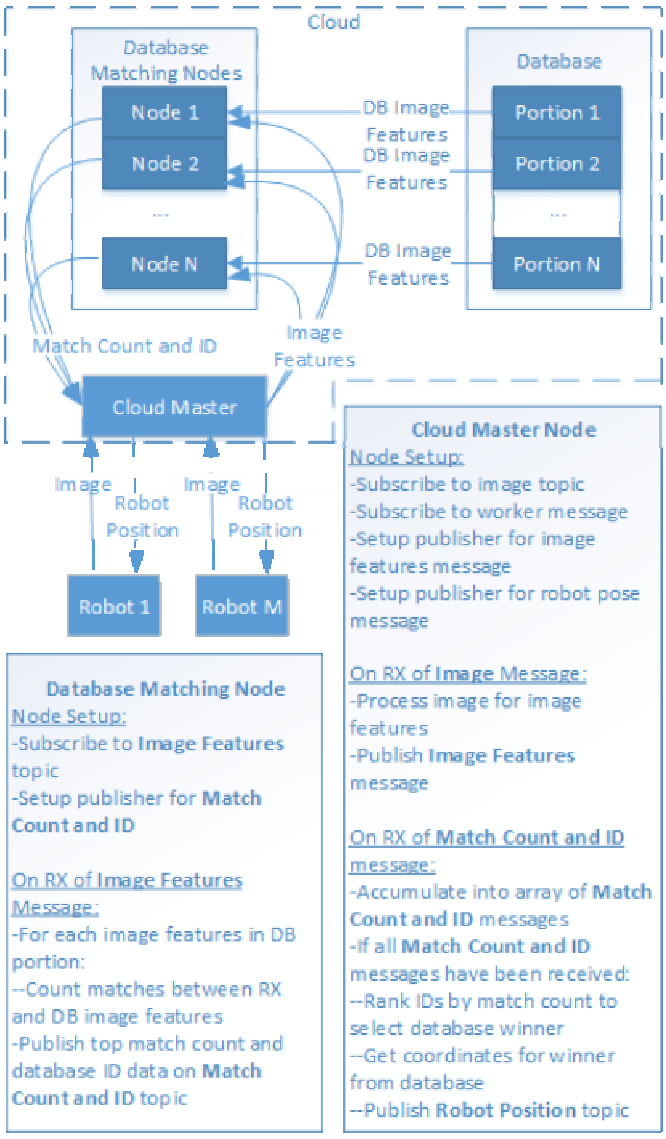

Fig. 4. Detailed ROS Node Configuration

Rackspace and NASA announced the OpenStack project in July of 2010. The OpenStack software consists of several loosely coupled services with well-defined APIs. While these APIs allow each of the services to use any of the other services, it also allows an implementer to switch out any service as long as they maintain the API.

The implementation described in this paper is based on the Havana release of the OpenStack distribution. The OpenStack services set up in our experiments:

- OpenStack Identity Management ("Keystone") manages a directory of users, a catalog of OpenStack services, and a central authentication mechanism across all OpenStack components.
- OpenStack Compute ("Nova") provides virtual servers upon demand. Nova controls the cloud computing fabric, the core component of an infrastructure service.

- OpenStack Network ("Neutron") provides a pluggable, scalable, and API-driven system for managing networks and IP addresses.
- OpenStack Block Storage ("Cinder") provides persistent block storage that compute instances use.
- OpenStack Image Service ("Glance") provides a catalog and repository for virtual disk images used in OpenStack Compute.

To support a distributed processing infrastructure for robotic research, we implemented a backend robotic cloud infrastructure as shown in Fig. 5. The structure enables distributed processing and scheduling among ROS and backend robotic cloud with respect to ROS and backend robotic compute node's compute capacity.

2) Experiment Setup

To evaluate network properties and performance of the proposed cloud, we set up a robotic cloud cell under the UTSA Open Cloud Laboratory. The robotic cloud cell for the evaluation comprises of 64 high-throughput AMD's SeaMicro SM15000 nodes. The nodes are linked via a high-performance fabric, called Freedom™ Fabric. All servers run Ubuntu with Linux kernel 2.6 and KVM hypervisor kvm-kmod-3.2. We use the Havana version of OpenStack, whereby 8 servers are configured as OpenStack compute servers as shown in Fig. 4.

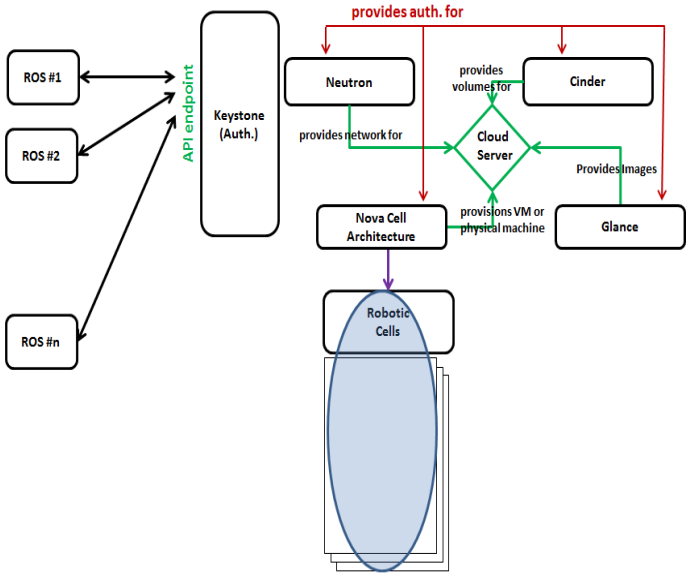

Fig. 5. Cloud Distributed Processing using OpenStack and ROS

IV. Results

The execution of the algorithm has been divided into two phases. The first phase is to create a landmark database. The second phase is to use the landmark database to determine the location of the robot. Initially a landmark database has been created for 2nd floor hall way of UTSA Bio Sciences and Engineering building, and this database has been used later to determine the position of the robot. For this experiment we have used Pioneer 2 mobile robot with a camera mounted on it. The database has been created on the laptop attached to the robot.

Fig. 6 below depicts the 2nd floor hall way of UTSA Bio Sciences and Engineering building.

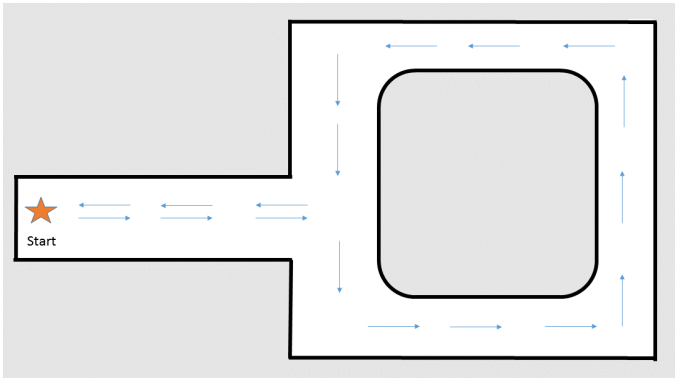

Fig. 6. Overview of robots path for Landmark image database creation

Initially a landmark database has been created for 2nd floor hall way of UTSA Bio Sciences and Engineering building, and this database has been used later to determine the position of the robot. For this experiment we have used Pioneer 2 mobile robot with a camera mounted on it. The database has been created on the laptop attached to the robot.

The images have been navigation information stamped and stored at a rate of 2 Hz. Fig. 6 depicts the route travelled by robot in 2nd floor hall way of UTSA Bio Sciences and Engineering building.

For the second phase we have taken the camera input and searched for the best match in the landmark database. The navigation information is then obtained from best match and the process is continued again. Fig. 7 shows the path obtained by the algorithm.

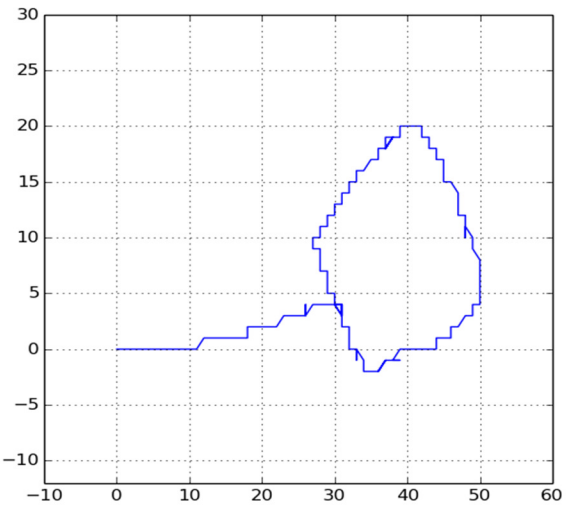

Fig. 7. Overview of robots path for Landmark image database creation

V. Conclusions and Future Research

In this paper we proposed a method to offload the VSLAM process as an effort to enhance robot performance and capabilities. With parallel computation via cloud computing

nodes, we can utilize computationally expensive algorithms without directly impacting robot on-board compute functionality. To expand the cloud paradigm to facilitate heterogeneous multi-robot VSLAM, we identified that the intrinsic and extrinsic parameters of the cameras need to be calibrated and taken into consideration into algorithm design. Future research will address network considerations such as determining network system capacity limits and the amounts of necessary overlap for image feature maps.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision,* vol. 60, pp. 91-110, 2004.

[2] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*, ed: Springer, 2006, pp. 404-417.

[3] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*, ed: Springer, 2006, pp. 430-443.

[4] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 2564-2571.

[5] P. Benavidez, M. Muppidi, and M. Jamshidi, "Improving Visual SLAM Algorithms for use in Realtime Robotic Applications," presented at the 2014 World Automation Congress, Waikoloa Village, HI, 2014.

[6] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review,* pp. 1-27, 2012.

[7] P. Benavidez, J. Lambert, A. Jaimes, and M. Jamshidi, "Landing of a Quadcopter on a Mobile Base Using Fuzzy Logic," in *Advance Trends in Soft Computing*. vol. 312, M. Jamshidi, V. Kreinovich, and J. Kacprzyk, Eds., ed: Springer International Publishing, 2014, pp. 429-437.

[8] B. Clipp, L. Jongwoo, J. M. Frahm, and M. Pollefeys, "Parallel, real-time visual SLAM," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 3961-3968.

[9] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea. (2014, 2014). Rapyuta: The RoboEarth Cloud Engine - RCE2013.pdf. Available: http://roboearth.org/uploads/RCE2013.pdf

[10] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 573-580.