RTABMAP Report

A simple investigation and things learned

Lihang Li

Jan.30 2015

RTABMAP Report

A simple investigation and things learned

1. Papers

(1) Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation

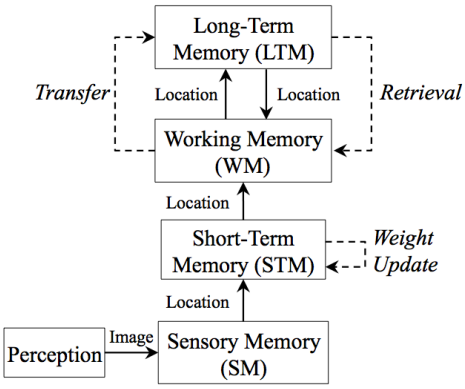


Fig. 2. RTAB-Map memory management model.

Algorithm 1 RTAB-Map

$time \leftarrow$ TIMENOW() ▷ TIMENOW() returns current time
$I_t \leftarrow$ acquired image
$L_t \leftarrow$ LOCATIONCREATION(I_t)
if z_t (of L_t) is a bad signature (using T_{bad}) **then**
    Delete L_t
else
    Insert L_t into STM, adding a neighbor link with L_{t-1}
    Weight Update of L_t in STM (using $T_{\text{similarity}}$)
    if STM's size reached its limit (T_{STM}) **then**
        Move oldest location of STM to WM
    end if
    $\boldsymbol{p}(S_t|L^t) \leftarrow$Bayesian Filter Update in WM with L_t
    Loop Closure Hypothesis Selection ($S_t = i$)
    if $S_t = i$ is accepted (using T_{loop}) **then**
        Add loop closure link between L_t and L_i
    end if
    Join $trash$'s thread ▷ Thread started in TRANSFER()
    RETRIEVAL(L_i) ▷ LTM $\rightarrow$ WM
    $pTime \leftarrow$ TIMENOW() $- time$ ▷ Processing time
    if $pTime > T_{\text{time}}$ **then**
        TRANSFER() ▷ WM $\rightarrow$ LTM
    end if
end if

Algorithm 2 Create location L with image I

procedure LOCATIONCREATION(I)
  $f \leftarrow$ detect a maximum of $T_{\text{maxFeatures}}$ SURF features from image I with SURF feature response over T_{response}
  $d \leftarrow$ extract SURF descriptors from I with features f
  Prepare nearest-neighbor index (build kd-trees)
  $z \leftarrow$ quantize descriptors d to vocabulary (using kd-trees and T_{NNDR})
  $L \leftarrow$ create location with signature z and weight 0
  return L
end procedure

Algorithm 3 Retrieve neighbors of L from LTM to WM

procedure RETRIEVAL(L)
  $L_r[] \leftarrow$ load a maximum of two neighbors of L from LTM (with their respective signatures $z_r[]$)
  Add references to $L_r[]$ for words in $z_r[]$ still in vocabulary
  Match old words (not anymore in vocabulary) of $z_r[]$ to current ones in vocabulary
  Not matched old words of $z_r[]$ are added to vocabulary
  Insert $L_r[]$ into WM
end procedure

Algorithm 4 Transfer locations from WM to LTM

procedure TRANSFER( )
  $nwt \leftarrow 0$ ▷ number of words transferred
  $nwa \leftarrow$ number of new words added by L_t and retrieved locations
  while $nwt < nwa$ **do**
    $L_i \leftarrow$ select a transferable location in WM (by weight and age), ignoring retrieved locations and those in recent WM (using T_{recent})
    Move L_i to $trash$
    Move words w_i which have no more references to any locations in WM to $trash$
    $nwt \leftarrow nwt + \text{SIZE}(w_i)$
  end while
  Start $trash$'s thread to empty $trash$ to LTM
end procedure

(2) Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM

See paper 1.

(3) Memory Management for Real-Time Appearance-Based Loop Closure Detection

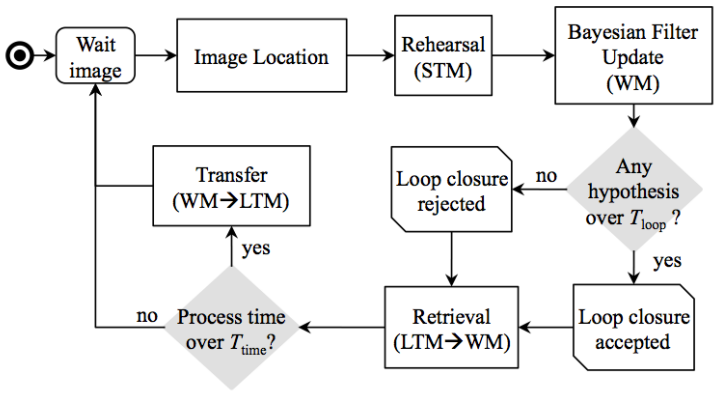


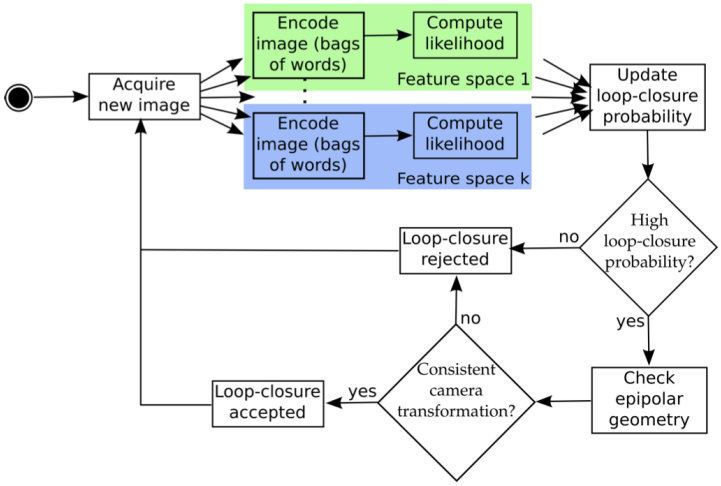
Fig. 1. Flow chart of our memory management loop closure detection processing cycle.

(4) Video Google: A Text Retrieval Approach to Object Matching in Videos

An approach to object and scene retrieval which searches for and localizes all the occurrences of a user outlined object in a video. The object is represented by a set of viewpoint invariant region descriptors so that recog- nition can proceed successfully despite changes in view- point, illumination and partial occlusion.

The analogy with text retrieval is in the implementation where matches on descriptors are pre-computed (using vec- tor quantization), and inverted file systems and document rankings are used. The result is that retrieval is immediate, returning a ranked list of key frames/shots in the manner of Google.

(5) A Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words



2. Methods

(1) Bag-of-words

http://people.csail.mit.edu/fergus/iccv2005/bagwords.html

(2) Vector Quantisation

http://www.data-compression.com/vq.html

http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Vector_quantization.html

(3) Recursive Bayesian Filtering

http://en.wikipedia.org/wiki/Recursive_Bayesian_estimation

(4) Pose Graph Optimisation

http://www2.informatik.uni-freiburg.de/~stachnis/toro/

(5) FLANN

http://docs.opencv.org/trunk/modules/flann/doc/flann_fast_approximate_nearest_neighbor_search.html

http://www.cs.ubc.ca/~lowe/papers/09muja.pdf

3. Source Code

Directory	Functionality
app	main.cpp
corelib	The RTABMAP backend
guilib	The Qt frontend
utilite	Utilities include logger, thread manager, event manager,etc. Independent project at https://code.google.com/p/utilite/.

(1) main.cpp

The application entry.

http://qt-project.org/doc/qt-4.8/qapplication.html#exec

http://qt-project.org/doc/qt-4.8/qcoreapplication.html#processEvents

RtabmapThread starts as an backend here.

(2) Rtabmap.h/Rtabmap.cpp

The main class which hold the 'process' method, the core function of RTABMAP.

(3) CameraRGBD.h/CameraRGBD.cpp

- Image I/O, abstract for RGB and RGBD even Scanner sensors, for RGBD sensors, support OpenNI, OpenNI2, OpenNICV, Freenect, etc.
- Store sensor data by RtabmapThread with std::list<SensorData> _dataBuffer

(4) UThread.h

Thread management abstraction layer.

- Posix(UThreadC.h)
- Win32(UThreadC.h & UWin32.h)

Specific Threads include:

- Main Thread
- RtabmapThread
- CamearThread(Triggered by **actionStart**, slot is **startDetection**)
- OdometryThread(**OdometryOpticalFlow** or **OdometryBOW** or **OdometryICP**)
- DBReader
- CompressionThread
- DBDriver
- qimageThread
- qimageLoopThread
- qdepthThread
- qdepthLoopThread

(5) UEvent*

Event management abstraction layer.

- UEvent
- UEventHandler
- UEventManager
- UEventSender

(6) Odometry.h/Odomery.cpp

Visual Odometry.

(7) VMDictionary.cpp

Visual words dictionary.

(8) Memory.cpp

STM(Short Term Memory) & WM(Working Memory).

(9) BayesFilter.h/BayesFilter.cpp

Implementation of the recursive bayesian filter.

(10)EpipolarGeometry.cpp

Used to check loop closure hypothesis.

4. Things learned

(1) RTABMAP is a good framework to start with Graph Based SLAM which uses Qt, PCL, OpenCV, OpenNI, etc

(2) The popular approach currently seems to build a pose graph first, then optimise after loop closure detection, finally obtain the point cloud

(3) However, the open source implementation address the problem of loop closure detection, not much to Mapping, so it does not fit well to do Object Reconstruction using Kinect

(4) Other investigations need to be done, like RGBDSLAM & Washington University's RGBD Mapping project