Application Report

Efficient Visual Localization on TDA4VM

Dilshan Godaliyadda, Hrushikesh Garud, Deepak Poddar, Soyeb Nagori, and Tarkesh Pande

ABSTRACT

As vehicles and robots move towards higher levels of automation, accurate localization within a map has become vital. However, since localization is one of many algorithms that need to run concurrently on an embedded platform, it is essential that accuracy does not come at the cost of efficiency. In this application report, an implementation of an efficient localization algorithm on a TDA4VM device, that meets these market requirements is described. The TDA4VM System-on-Chip (SoC) is the first commercially available device in the Jacinto™ 7 family of SoCs, a family of SoCs designed from the ground up specifically for automotive and similar robotics applications. This family of SoCs comprises of two primary variants, the TDA4x family designed for ADAS and similar perception tasks in robotics, and the DRA8x family designed for cloud-connected gateway systems. The localization algorithm described here is implemented on the TDA4VM device, a device from the TDA4x family, because the feature extraction portion of the algorithm is based on a Deep Neural Network, and the TDA4VM is equipped with one of the industry's most power efficient deep learning engines.

Table of Contents

Trademarks

Jacinto™ is a trademark of Texas Instruments.
Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
All trademarks are the property of their respective owners.

1 Introduction

Accurate ego-location within a map is an essential requirement for autonomous navigation. In the ADAS and Robotics communities this problem is referred to as the localization problem. Typically, when a vehicle or robot is outdoors, localization can be handled to some extent by the Inertial Navigation System (INS), which uses Global Positioning System (GPS) data together with measurements from an Inertial Measurement Unit, or IMU, to localize the vehicle/robot. However, an INS can only communicate with a GPS satellite when there is no obstruction between the two, that is, when there is a clear line of sight (LOS) towards the satellite. When the vehicle or robot is in a garage, warehouse, or tunnel, the accuracy of GPS location degrades substantially, because the LOS to the satellite is obstructed. Furthermore, even when GPS is available, it can only position a vehicle within approximately a 5 meter radius [1]. This error coupled with errors in the IMU result in noisy localization that may not be sufficiently accurate for high complexity ADAS or robotics tasks.

Visual localization is a popular method employed by the ADAS and robotics community to meet the stringent localization requirements for autonomous navigation. As the name implies, in visual localization, images from one or more cameras are used to localize the vehicle or robot within a map. Of course, for this task, a map of the environment needs to be constructed and saved prior to localization. In the field of localization, the more popular solutions thus far have been based on LiDAR, because LiDAR measurements are dense, and precise. However, though LiDAR based localization is highly accurate, it is cost prohibitive for the every day vehicle, because high precision LiDARs are typically in the order of thousands of dollars. Thus, it is critical that a cheaper alternative such as visual localization is made available.

In robotics and in automotive, the computations for localization as well as for other tasks, need to be performed within the vehicle or robot. Therefore, it is critical that the vehicle or robot is equipped with high-performance embedded processors that operate at low power. The Jacinto 7 family of processors by TI, was designed from the ground up with applications such as visual localization in mind. The Jacinto 7 family is in fact the culmination of two decades worth of TI experience in the automotive field, and many decades of TI experience in electronics. These processors are equipped with deep learning engines that boast one of the best power-to-performance ratios of any device in the market today, together with Hardware Accelerators (HWAs) for specific Computer Vision (CV) tasks, and also Digital Signal Processors (DSPs) that can efficiently perform related CV tasks.

Figure 1-1. Why Localization?

2 The Visual Localization Problem

In the simplest sense, visual localization, as the name implies, is the problem of determining the location of a vehicle or robot by matching *key-points* in a stored map with *key-points* extracted from images captured by a camera mounted on a vehicle/robot. A *key-point* is a unique or distinctive point in space from which a *descriptor* can be extracted. A *descriptor* is a set of values (a vector) that holds information about a key-point, that will help distinguish said key-point from others. The method that is used to compute these features is described in the next section.

The first step in localization is extracting key-points from the image. Then, the extracted key-points, which are on a 2D image plane, need to matched with a 3D sparse map held in memory. To create the 3D sparse map, features need to be extracted and stored together with their corresponding locations in some arbitrary but known coordinate system. This task is typically achieved by driving a vehicle equipped with a high precision differential GPS and camera along all the paths that make up the map. In order to ensure the features are not biased by the time of day, or day of the year, information is gathered throughout the year to refine the map. Then, when the position of the vehicle/robot needs to be estimated, key points extracted from an image are matched with key points in the sparse 3D map, and using the point correspondence the pose of the vehicle/robot is estimated. This process is described in more detail in the next section.

The entire localization process is shown at a high level in Figure 2-1.

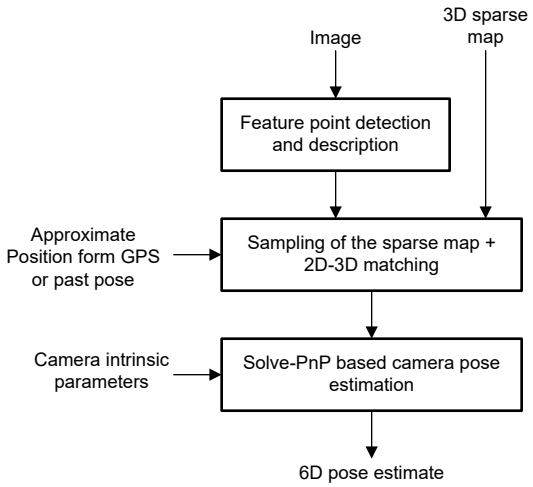

Figure 2-1. High-Level Block Diagram on Visual Localization

In the next two sections, the implementation of the steps that make up visual localization, namely key point extraction, descriptor computation, feature matching and pose estimation, are described in more detail.

2.1 Key Point Extraction and Descriptor Computation

There are a variety of techniques used by the Computer Vision community to extract key-points. These techniques fall in to one of two categories – traditional Computer Vision based feature extraction methods such as SIFT, SURF [2], KAZE [3], or Deep Neural Network (DNN) based feature extractions methods.

An important advantage of DNN based key-point extraction is that the process can be performed using a generic Deep Learning accelerator. In contrast, for traditional CV based key-point extractors, one either needs to design specialized hardware accelerators (HWAs) or use general purpose processor cores. The former limits the types of features the customer can use, and the latter is prohibitively inefficient, and as a consequence DNN based key-point extraction becomes the more practical solution.

This document describes a DNN-based feature extraction method for localization. In particular, the algorithm described here learns feature descriptors similar to KAZE [3] in a supervised manner using DNNs and is therefore named DKAZE or Deep KAZE. Using the DKAZE framework, one can extract both key-points and the corresponding descriptors as shown in [3]. More details on this algorithm can be found here. Once key-points are extracted, the next step is to match the extracted features with features in the stored 3D map, to thereby estimated the pose of the vehicle/robot. The network structure of the DKAZE DNN is shown in Figure 2-2.

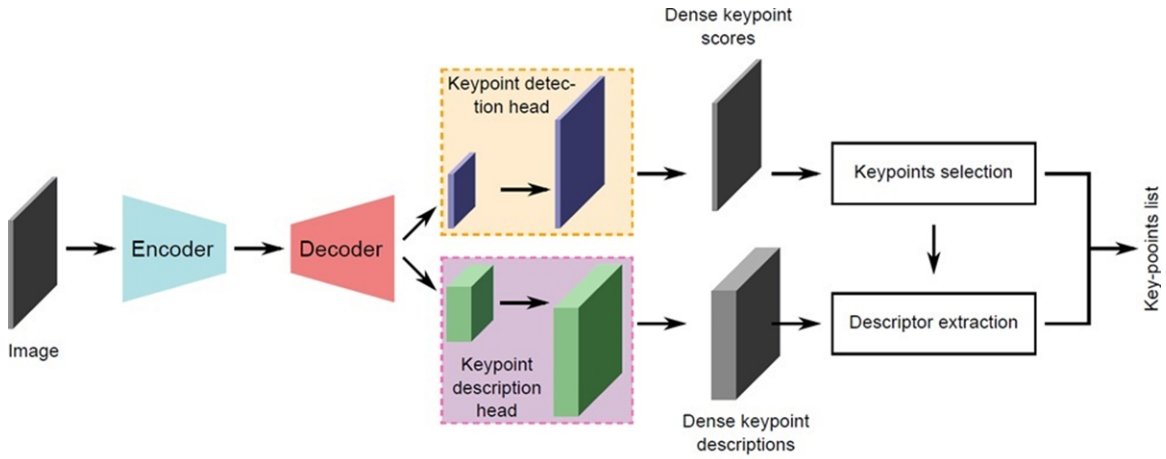

Figure 2-2. DKAZE Network Structure

2.2 Feature Matching and Pose Estimation

Feature matching is the process of matching *M* 2D key points from an image, to *N* stored 3D key points. The *goodness* of a match between two key points is computed using the descriptors that correspond to each point. In the implementation described here, the Sum of Absolute Different, or SAD, is used as the measure of how well two descriptors match. In particular, a smaller SAD score corresponds to a better match. However, since computing *MxN* SAD scores is computationally prohibitive, the SAD scores are only computed between *M* 2D points from the image and *n*<*N* 3D points from the sparse 3D map. These *n* points are selected based on the estimated position of the vehicle/robot. Then, from these scores, the correspondences that give the lowest cumulative SAD score is selected.

Once the feature correspondences are computed, the next step is to compute the pose of the vehicle/robot. In this implementation, a 6D pose is computed, that is, rotation as roll, pitch and yaw and translation as X, Y, Z. In particular, the Perspective-n-Point or PnP method is used for pose estimation. PnP is the problem of estimating the pose of a calibrated camera, given a set of n 3D points in the world, and their corresponding 2D projections in the image plane. In this implementation, the P3P scheme [4] is used in the RANdom SAmple Consensus (RANSAC) framework [5]. At a high level, this is an iterative process where point correspondences are randomly selected in each iteration to refine the pose estimate.

In the next section, the subtasks that make up the visual localization algorithm are mapped to the different components TDA4VM SoC, to showcase the seamless mapping that can be achieved with the SoC.

3 Visual Localization on TDA4VM

This section describes how each of the subtasks that make up the visual localization algorithm described here maps seamlessly to the TDA4VM device. This application consists of three primary steps: Image pre-processing, DKAZE feature extraction and localization. Since the TDA4x family of devices were designed with applications like this in mind, each of these subtasks can be mapped to specialized hardware within the device, that ensures efficient and accurate execution of the tasks. A diagram of the TDA4VM device, the first variant of the TDA4x family available to customers, is shown in Figure 3-1.

The diagram shown below as Figure 3-1 is a block diagram that details the key components that make up the TDA4VM SoC. These include, a Deep Learning hardware accelerator coupled to a C7x DSP, a few general purpose Arm® cores, a vision pre-processing hardware accelerator and hardware accelerators designed specifically for certain widely used CV tasks. Next, the sub-tasks that make up the algorithm are mapped to the different components of the SoC in Figure 3-2.

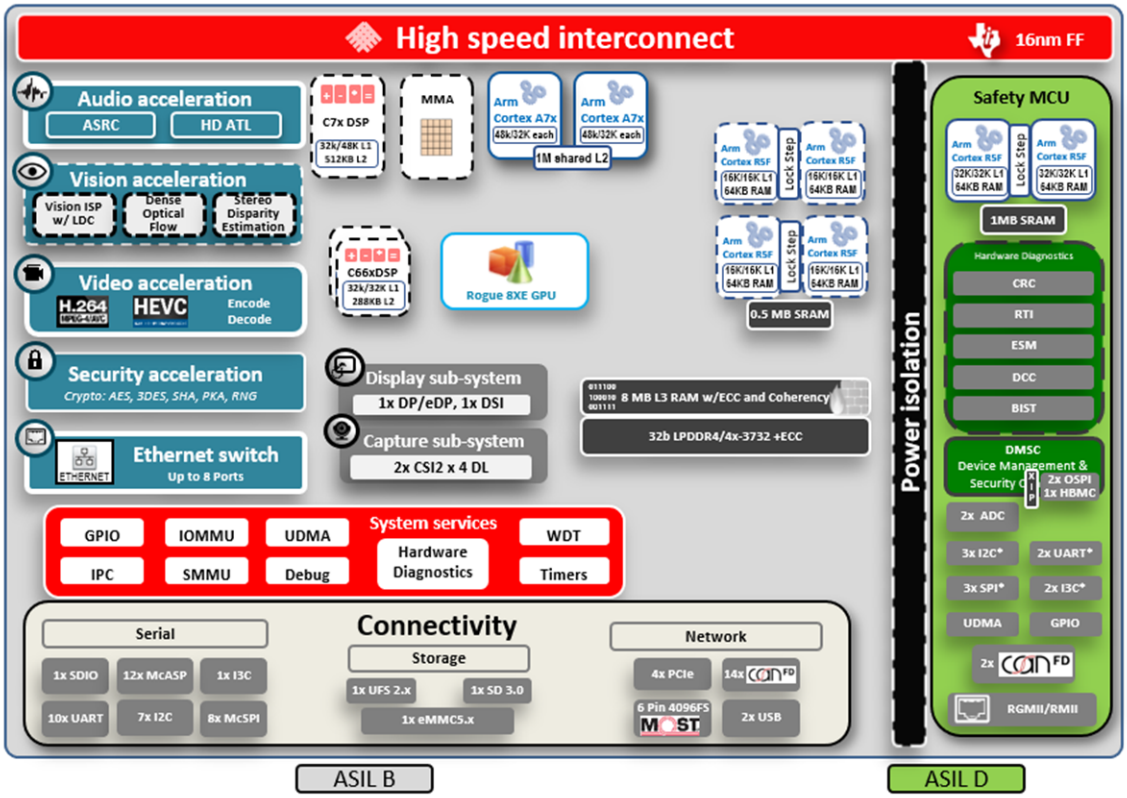

Figure 3-1. TDA4VM Diagram

The first subtask, image pre-processing, can be performed entirely on the on-chip Vision Pre-Processing Accelerator, or VPAC, that includes an Image Signal Processor, or ISP. This module takes the image from the camera, which comes across on a CSI-2 interface, and performs the pre-processing steps necessary before further processing. The VPAC module on the TDA4VM consists of a Raw Front End (RFE), a dual noise filter, a global and local tone mapping module, a flexible color processing module, lens distortion correction, and a scaling engine. More information about the VPAC can be found here.

The next sub-task, DKAZE feature extraction, can be performed using the on-chip DNN hardware accelerator, the C7x/MMA. The C7x/MMA is an HWA designed specifically to accelerate commonly used Deep Learning operations. The C7x/MMA is one of the most power efficient Deep Learning Accelerators in the market today, since it was designed with automotive and industrial application in mind, by engineers with decades of experience. The C7x/MMA module also boasts one of the best power to TOPS ratios of any device in the market today. More information about C7x/MMA can be found here.

Finally, the remaining visual localization subtasks are performed on one of the DSPs available on the SoC, either the C7x or C66x.

The subtask mapping from the visual localization algorithm to the TDA4VM device is shown as a the flow diagram below in Figure 3-2.

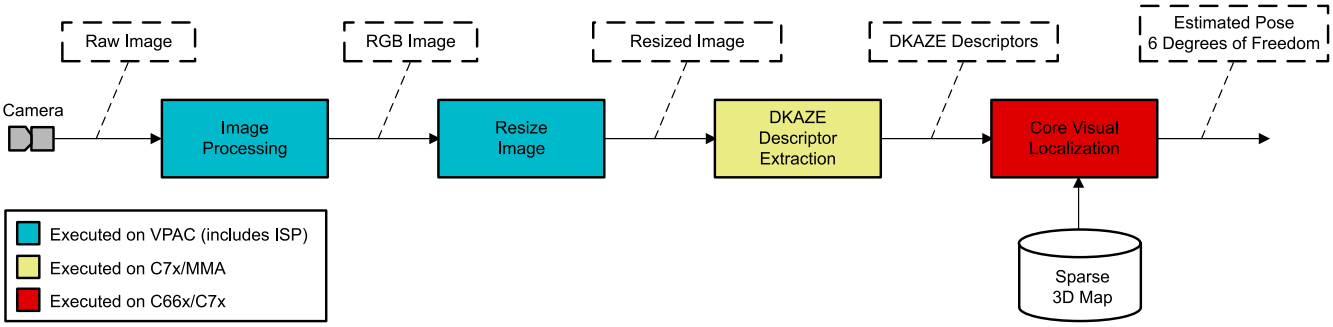

Figure 3-2. Visual Localization Algorithm Flow on TDA4VM

4 Example Visual Localization Application

A visual localization application that is built on the subtasks described above is included with the SDK that accompanies the TDA4x family of devices. Since this application was designed to run on simulated data, as it allows for comparison with the ground truth, the data does not need to be processed using VPAC. The application instead consumes saved simulated image data. Detailed documentation of this application can be found here.

Figure 4-1 shows an example output from this application. Here, the computed and actual paths are overlaid on a bird-eye view of the scene. The blue markings here show the computed trajectory, and the white shows the ground truth. The utilizations of the different compute cores at this instance are shown on the bottom right corner. Detailed results can be found here.

Figure 4-1. Example Output From Visual Localization Application

4.1 Optimized Building Blocks for Your Own Visual Localization Pipeline

It is important to note that the visual localization application described above is optimized end-to-end. However, if one wishes to construct their own localization pipeline, they can take advantage of the optimized building blocks contained in this pipeline for certain compute heavy tasks within their own pipeline. The optimized building blocks that are provided as part of the TIADALG component package are listed below:

- **Two Way Descriptor Matching** - This API can be used to carry out two way matching between 2 sets of descriptors.
- **Sparse Up-sampling** – This module can up-sample features generated at lower resolutions. For example, this function up-samples the features generated by DKAZE, which are at 1/4th the original resolution, to the full image resolution.
- **Recursive** non-maximum separation (NMS). This is a recursive method to clean up duplicate features within localized neighborhood.
- **Perspective N Point Pose estimation, a.k.a. SolvePnP** - After 2D-3D correspondences are computed this API can solve the PnP problem to estimate the 6-D camera pose.

More details can be found here.

5 References

1. https://www.gps.gov/systems/gps/performance/accuracy/
2. Rublee, Ethan, et al. "ORB: An efficient alternative to SIFT or SURF." 2011 International conference on computer vision. IEEE, 2011.
3. Alcantarilla, P.F., Bartoli, A. and Davison, A.J., “KAZE features”. In European Conference on Computer Vision, pp. 214-227, 2012.
4. Gao, X.-S., Hou, X.-R., Tang, J. and Cheng, H.F., “Complete Solution Classification for the Perspective-Three-Point Problem.” IEEE Transactions on Pattern Analysis and Machine Intelligence. Volume 25, Issue 8, pp. 930–943, August 2003.
5. M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun.ACM 24 (6) (1981) 381–395.

Table 5-1. Other Important Links

Reference Name	Link
TDA4VM product information:	https://www.ti.com/product/TDA4VM
Evaluation Board Purchase/Request	Common Board: https://www.ti.com/tool/J721EXCPXEVM TDA4VM Processor SOM: https://www.ti.com/tool/J721EXSOMXEVM
Download SDK	https://www.ti.com/tool/PROCESSOR-SDK-DRA8X-TDA4X
SDK Documentation	https://software-dl.ti.com/jacinto7/esd/processor-sdk-rtos-jacinto7/latest/exports/docs/psdk_rtos/docs/user_guide/index.html
Jacinto 7 Video Training series	https://training.ti.com/jacinto7-platform
Technical Support from E2E Processor Forum:	https://e2e.ti.com/support/processors/f/791
PTK API Guide (library used in second application)	https://software-dl.ti.com/jacinto7/esd/processor-sdk-rtos-jacinto7/latest/exports/docs/perception/docs/ptk_api_guide/index.html

6 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (April 2021) to Revision A (April 2021)	Page
• Updated the numbering format for tables, figures and cross-references throughout the document	2
• Update was made in Section 3	5

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265