DreamVu™

Autonomous Localization & Mapping using PAL

DreamVu Inc.

October 6, 2020

Abstract

DreamVu's PAL is a revolutionary omnidirectional camera that enables key technical improvements in the sphere of machine vision sensing. The PAL camera uses a single CMOS sensor and is therefore a cost and power effective camera for machine vision. The PAL camera has a one-of-a-kind optical design that captures seamless 360° stereoscopic views of the scene within a single image, and can be used for computing dense scene-depth information. The PAL camera has no blind spots in the visual or depth field, even if a target at an infinitesimal distance from the camera. This is a unique attribute enabled by PAL's optical design and a quality that sets PAL apart from any other depth sensor. This article introduces the PAL family of cameras and provides an outline for easily integrating PAL cameras into generic software stacks for autonomous robot localization and scene mapping.

1 PAL for Autonomous Robotics

Autonomous Robot Navigation relies on a stream of inputs from different sensors. The types of sensors used in this space are either visual sensors, range-scanners and inertial sensors. Visual sensors estimate scene structure and pose from images of the scene. These may be single-image cameras that rely purely on computer vision and machine learning or multi-image cameras use additional geometric information such as parallax. Visual sensors are cheap and capture pixel-rich scene information. Their performance is thereby dependent on the texture and illumination in the scene. Range Scanners use specialized equipment such as infrared projector-camera combination, time of flight cameras, laser scanning and ranging etc. These sensors are typically fast and provide accurate measurements with very little compute overhead. However, these sensors are limited to a narrow vertical field-of-view, contain moving parts, capture sparse information about the scene and lack the semantics present in visual information. Additionally, these sensors are prone to interference and are expensive in general. Inertial Sensors provide odometry information that helps the robot localize itself within the environment. Inertial sensors can be mechanical such as accelerometers, gyroscopes, rotary encoders or visual-inertial that compute optical odometry. These are standalone sensors in the sense that they can

operate in a variety of scene and environment conditions. However, these sensors are adversely affected by interference, calibration errors and drift over time.

The PAL camera is an ideal visual sensor for robot autonomy. The PAL camera, shown in Figure 1, uses an optical mirror arrangement that can simultaneously capture 360° stereoscopic images onto a single CMOS sensor. Using the visual and depth feed of the PAL camera, a robot can not only measure the shape of the environment around it but also the position of the camera within the environment. These are integral features for a vision sensor that are quintessential for scene mapping and robot localization. The key technical attributes of the camera that enable precise odometry and mapping are *stereoscopic sensing*, *a large field of view* and *zero minimum depth distance*. The PAL camera fits the blueprint of an *ideal* machine vision sensor due to these attributes. Stereoscopic sensing enables the PAL camera to identify the distance and the shape of the objects around it. The large and seamless field of view enables the PAL camera to localize and estimate its pose more accurately and with fewer rotations than any other limited focal length cameras. A zero minimum depth distance enables the PAL camera to be robust to dynamic obstacles and also allows the camera to be mounted at the periphery of any robot, further enabling the full 360 coverage that the camera can support.

The field-of-view captured from the PAL camera is shown in Figure 2. The image shows the left and right stereoscopic views captured by the camera and the dense depth information generated from the camera. The panoramas are seamless and do not have any blind spots within the angular operating range of 360° × 110°. The PAL panoramas are compute-friendly in the sense that they are not computed by stitching different views like other multi-camera 360° solutions. The large context in PAL panoramas reduces the dependency on texture in the scene that is required by typical visual stereo sensors. There is a pixel-wise correspondence between the image and depth frames that enables the PAL camera to create textured omnidirectional point clouds. The point clouds from laser scanners are devoid of visual information while those from other stereo cameras have a limited field of view.

A combination of these salient features - large field-of-view and dense depth estimation - makes PAL a singular sensor in its own right. The only other way to replicate the features of a PAL camera is to use a multitude of different sensors together. This results in complex system building and integration, requires precise calibration, requires synchronization of all sensors and is an expensive ordeal. The PAL camera can replace the need for multiple sensors for robot autonomy, while providing quantitative 360° scene information at interactive frame-rates.

2 PAL Autonomous Robotics with ROS

The PAL RGB and depth output is supported at four native resolutions of 9.5 megapixels at 10fps, 4.3 megapixels at 20fps, 0.6 megapixels at 40fps and 0.15 megapixels at 100fps. The camera supports an angular resolution of 0.068° degree in the horizontal direction and 0.060° degree in the vertical direction at the highest resolution. Since the PAL camera is a full 360° camera, once it is rigidly mounted onto a robot, the user may specify a one-time parameter that maps the start position of the horizontal field-of-view with respect to the robot and the span of the vertical field-of-view based on the height PAL on the robot.

2.1 PAL's Camera Coordinate System

PAL follows a right-handed cartesian co-ordinate system illustrated in Fig 3 and defined as follows : -

- The Y-axis is PAL's central axis and points vertically downwards.
- The Z-axis points radially outwards in a horizontal plane at the start of horizontal FoV.

Figure 1: DreamVu PAL Camera

Figure 2: PAL Stereoscopic Panoramas and Depth Map

- The X-axis is orthogonal to Z-axis and lies in the horizontal plane.
- Yaw is denoted by the rotation about Y-axis. Counter-clockwise rotation is considered positive.

2.2 Mounting and Initialization

The PAL camera can be flat-mounted on the robot (in absence of pitch and roll) in a horizontal plane i.e base of the PAL is parallel to the floor-plane as shown in Fig 3. The pose of the PAL's base frame in robot's coordinate system is specified by the translational offset parameters (`x_offset`, `y_offset`, `z_offset`) and rotational offset parameter (`yaw_rot`) as shown in Fig 3. These parameters are used to compute the transform between robot's base frame and PAL's base frame. In addition, the height of the PAL's base w.r.t the ground-plane is specified in the `pal_height` parameter as shown in Fig 3. The aforementioned five variables are given as inputs to the PAL's SDK in a default file named **mounting.yaml**. A user can easily integrate the PAL camera with their robot by editing the default values of these variables.

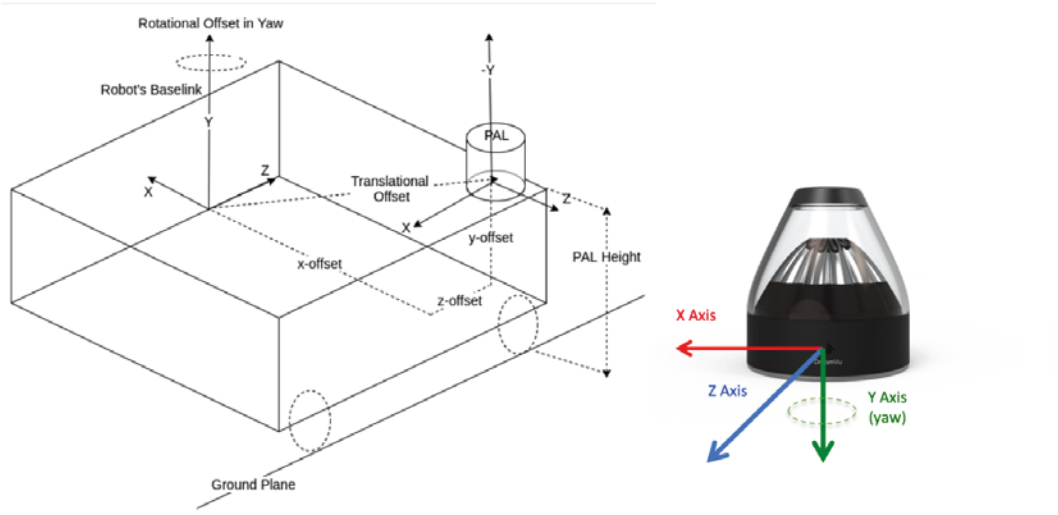

Figure 3: A flat-mounted PAL camera on the base of Robot. The transformation between the robot's co-ordinate frame and the PAL camera is specified by the (x_offset, y_offset, z_offset, yaw_rot, pal_height) parameters. The front of the PAL camera - indicated by the DreamVu logo - corresponds to the start position of the horizontal field of view.

2.3 PAL Bringup

The following command can be used to bringup the PAL camera in ROS and visualize the published topics in Rviz:

```
roslaunch dreamvu_pal_camera pal_rviz.launch
```

This will open an rviz window, with the following topics being published.

- /PAL/left - Stereo Left Image
- /PAL/right - Stereo Right Image
- /PAL/depth - Depth image registered in Left Image's frame
- /PAL/pointCloud - Dense Point Cloud in PAL's camera coordinate system
- /PAL/depthscan - Laser Scan in PAL's camera coordinate system
- /PAL/visualodom - Visual Odometry computed from temporal sequence of images
- /PAL/occupancy - Dynamic Occupancy grid computed from Depth image
- /tf - Transform tree containing the robot's base-link, PAL's camera frame, Odometry Frame and the Map frame(if any).

It may be noted that the odometry published from PAL is converted to follow the standard SLAM convention of Left(X), Forward(Z) and Up(Y). The topics published by PAL can be listed using

```
rostopic list
```

2.4 Mapping

Mapping is defined as the process of creating a map of a known or an unknown environment. A map of the environment is either represented in a topological or a geometric manner. In most robotic applications, geometric maps are common because the robot requires a metric understanding of the environment to be able to navigate around it. Geometric maps are further classified as feature-based maps (landmarks) or grid-based maps (occupancy grid). Feature maps are sparse and are created by isolating specific landmarks in the scene. Grid maps - that define the environment as a metric grid - are denser and more flexible in comparison to feature maps. There is no explicit need to pre-compute features beforehand for grid maps unlike feature based maps. Grid maps have therefore become very popular for creating large-scale maps for robotic navigation applications.

Gmapping is a popular grid based mapping package available in ROS. Gmapping creates accurate 2D maps of the environment using only the laser-scan and odometry information. Gmapping is based on a probabilistic framework and is very robust to sensor or odometry noise. Typically, laser-scans are commonly obtained from Sonar (sparse and very noisy), Lidar (sparse and very expensive), or other sensors based on echo (range-sensing) phenomena. More and more of these sensors have recently been replaced with RGBD cameras such as Intel Real-Sense, Microsoft Kinect, Orbbec ASTRA, etc, which can provide dense depth-scans (in-principle equivalent of a laser-scan) but a narrow field of view. The PAL camera not only alleviates the problem of limited field of view but also provides dense depth-scan information. This makes PAL an ideal camera for mapping with dense depth-scans supported by visual context. The depth-scan provided by the PAL camera is more information rich than that of a Lidar laser-scan as it includes textured pixels for every scene point. The depth-scan from PAL is more flexible than a Lidar laser-scan as the user may choose the desired horizontal and vertical region of interest.

The PAL camera can be used to map a large environment by moving a robot in the scene. We provide a detailed outline for integrating the PAL camera with the ROS gmapping stack. The gmapping algorithm relies on the pose of the robot (odometry) and a corresponding laser-scan at every location that the robot navigates to. To launch the gmapping software with PAL, a launch file **gmapping.launch** is provided with the PAL SDK. The PAL camera can also be invoked into a custom launch file by remapping the topics **laser-scan** and

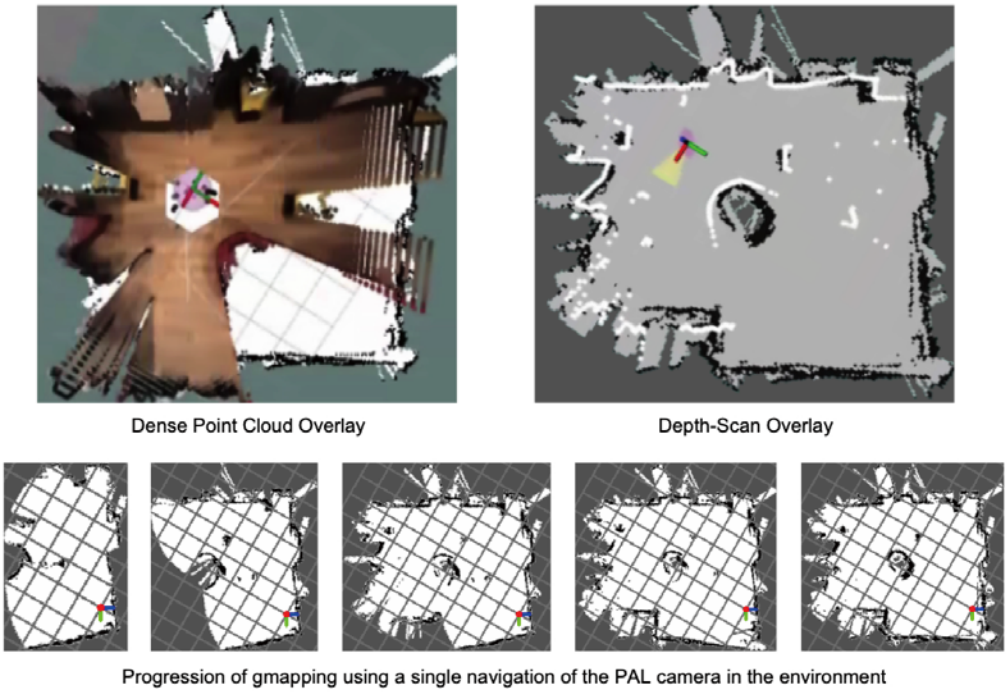

Figure 4: Progression of the gmapping stack using the PAL camera. Each grid cell corresponds to a size of 100cm×100cm. Top Left: The textured dense point cloud from the PAL camera is overlaid onto the map at one position of the robot, illustrating the range of mapping from PAL. Top-right: The depth-scan created from this location is illustrated. The points in gray are free space, the points in black indicate occupied regions of the map and the points in white represent the current depth-scan. Bottom Row: From left to right, the progression of the map is shown as the robot circles around the central obstacle and returns to its original position. Even at the initial frame, it may be noted that PAL has already mapped a large section of the environment.

odom to PALs **depthscan** and **visualodom** respectively. Additional sources for odometry may be specified directly in the **odometry.yaml** file. The default parameters and best practices for the gmapping SLAM framework with reference to PAL are provided in Section 3.

On bringup, the gmapping package can be launched using the command

```
roslaunch gmapping.launch
```

and the robot can be manually navigated in the environment to be mapped. For best practices on gmapping parameter settings and mounting suggestions for the PAL camera, please refer to Section 3. Once the mapping is completed the map can be saved using the command

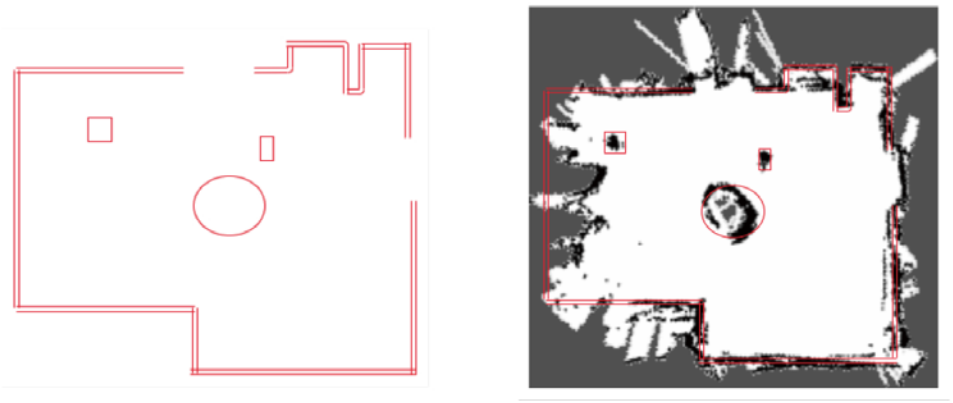

Figure 5: A comparison of the ground truth metric representation of the map of the environment and the map generated using the gmapping stack. The gmapping stack uses both the odometry and depth-scan information directly from the PAL camera without any additional sensors.

```
rosrun map_server map_saver -f <filename>
```

The PAL camera creates the 2D occupancy map of a scene more efficiently than other depth sensors. Due to the full 360 horizontal FoV, the PAL camera requires fewer rotation than a narrow-FoV visual sensor and a single walk-through around the environment is sufficient for complete mapping. For the environment shown in Figure. 2 and with manual operation of the robot within the environment, the progression of the ROS gmapping stack is shown in Figure 4. The figure shows a dense texture point cloud overlay from the PAL camera at a particular location in the scene. It also illustrates the depth-scan created from the PAL camera at this location. The map can be seen growing to completion even before the camera returns to the origin location, which is enabled by the full 360° field of view from PAL. The ground truth map overlaid on the estimated scene map from gmapping using the PAL camera is illustrated in Figure 5. This map is created using PAL as the sole sensor on the robot.

2.5 Localization

Robot localization is the process of determining the location of the mobile robot within the map of the environment. Monte-Carlo localization is one of the most widely used approaches for localization of mobile robots. Monte-Carlo localization is based on a probabilistic framework and fundamentally relies on particle filtering. The Adaptive Monte Carlo Localization (AMCL) package is a framework in ROS for robot localization. The AMCL framework uses efficient strategies for distributing particles in the state-space and thereby finds much use in mobile robotics. The AMCL package requires the odometry of the robot,

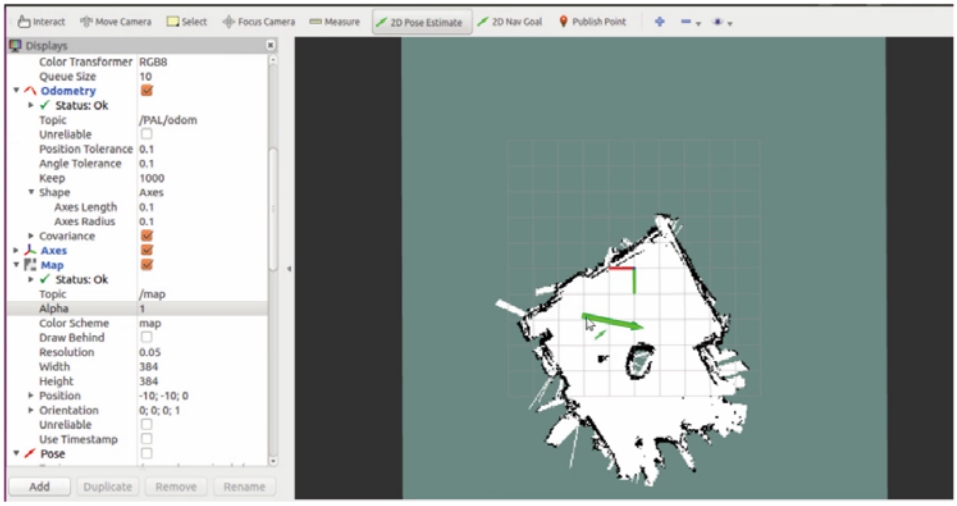

Figure 6: The start point of the green arrow is the estimate of robot's initial position and the arrow of the arrow is the estimate of the robot's yaw angle.

the map of the environment and a laser-scan (or equivalent depth-scan) from the sensor.

We provide an outline for using the PAL camera as the stand-alone sensor for robot localization. This illustration is meant to provide an overview to the user that the large field-of-view makes the PAL camera capable of accurately identifying the pose and the location of the robot. We provide a launch file for using the PAL camera with the inbuilt AMCL stack which can be invoked using:

```
roslaunch amcl.launch.
```

The path to pre-computed map of the environment is specified within the launch file. This launch file defaults to robots with motion models similar to TurtleBot 2. For holonomic or omni-directional motion models, the AMCL odometry parameters may be adjusted in the launch file. The parameters for odometry that can be tuned for better performance are discussed in Section 3.

On bringup, the map is shown in an Rviz window. The initial pose of the robot is first marked using the initial pose tool in Rviz by marking the position and orientation (yaw) of the camera as illustrated in Figure 6.

Once the robot begins navigating in the environment based on the path-planner, the localization of robot can be computed using only the PAL camera as shown in Fig 7. The figure shows a metric comparison of the ground-truth localization from the wheel odometry of the robot in comparison to the visual odometry computed from PAL. The default parameters for the AMCL localization framework with reference to PAL are provided in Section 3.

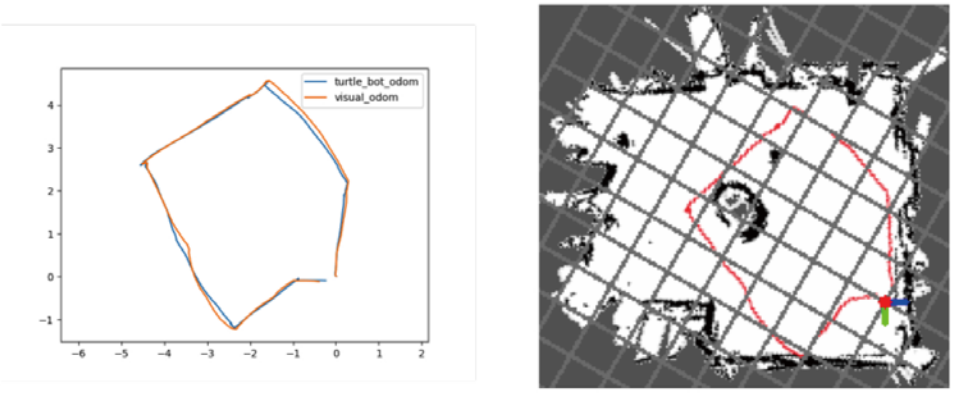

Figure 7: AMCL localization using PAL depth scan and PAL's visual odometry

This article is provided as a starting step for integrating the PAL camera with a robot for 2D mapping and localization. By design, the PAL camera creates a dense volume of 360° 3D information and this article provides a roadmap where the information from the camera can be integrated easily with existing stacks for robot mapping and localization. The dense information from the PAL camera can not only be used for 2D mapping but also 3D mapping of environments which is discussed in a separate article. The PAL SDK provides athe APIs and documentation required to integrate the PAL camera with your robot.

3 Appendix

Best Practices for Mapping and Localization:-

- The translational and angular velocity of the robot should be specified based on the selected resolution frame-rate (ranging from 10Hz to 100Hz) of the PAL panoramas. Since the PAL camera captures visual information, motion blur in the panoramas should be avoided.
- An additional loop-closure pose correction flag is provided in the **odometry.yaml** file. This module automatically corrects the pose after every loop closure. For offline mapping, it is recommended to set the loop-closure optimization flag to *True*.
- The default Gmapping and AMCL ROS parameters for PAL camera are provided with the SDK. In a typical scenario similar to the one shown in this article, these parameters work well.
- In order to ensure that the transform tree is correctly initialized, run the following command - **rosrun tf view_frames && evince frames.pdf**. This saves the tf-tree along with the other meta-information of timestamps, frequency of messages being published, etc. as a pdf document. It

Figure 8: Transform tree

may be ensured that the structure of the tf-tree is same as that shown in Figure 8.

- The following Gmapping parameters may be tuned if the robot's odometry is used: **srr**, **srt**, **str**, **stt**
- The following Gmapping parameters may be tuned for optimizing compute complexity: **linearUpdate**, **angularUpdate**, **map_update_interval**, **throttle_scans**, **particles**
- The following AMCL odometry parameters may be tuned based on robot's motion model and error covariance. **odom_model_type**, **odom_alpha1**, **odom_alpha2**, **odom_alpha3**, **odom_alpha4**, **odom_alpha5**