3D Object Recognition and Relative Localization using a 3D sensor Embedded on a Mobile Robot

Gopi Krishna Erabati*†, Frédéric Lerasle*‡ and Jorg Fransisco Madrigal Diaz*
*LAAS-CNRS, 7 Avenue de Colonel Roche, F-31400 Toulouse, France
†Univ de Bourgogne, 71200 Le Creusot, France
‡Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France

***Abstract*— This work presents a framework for object pose estimation using 3D data of scene acquired from 3D sensors (e.g. Kinect, Orbec Astra Pro among others). The 3D data has an advantage of independence from object texture and invariance to illumination. The proposal is divided into two phases : An offline phase where the 3D model template of the object (for estimation of pose) is built using Iterative Closest Point (ICP) algorithm. And an online phase where the pose of the object is estimated by aligning the scene to the model using ICP, provided with an initial alignment using 3D descriptors (like Fast Point Feature Transform (FPFH)).The approach we develop is to be integrated on two different platforms : 1)Humanoid robot 'Pyrene' which has Orbec Astra Pro 3D sensor for data acquisition, and 2)Unmanned Aerial Vehicle (UAV) which has Intel Realsense Euclid on it. The datasets of objects (like electric drill, brick, a small cylinder, cakebox) are acquired using Microsoft Kinect, Orbec Astra Pro and Intel RealSense Euclid sensors to test the performance of this technique. The objects which are used to test this approach are the ones which are used by robot. This technique is tested in two scenarios, firstly, when the object is on the table and secondly when the object is held in hand by a person. The range of objects from the sensor is 0.6 to 1.6m. We present both qualitative and quantitative evaluations on the dataset we acquired using the 3D sensors.**

I. INTRODUCTION

The technology in current research scenario is marching towards automation for higher productivity with accurate and precise product development. Vision and Robotics[4] are domains which work to create autonomous systems and are the key technology in quest for mass productivity. The automation[5] in an industry can be achieved by detecting interactive objects and estimating the pose to manipulate them. Therefore the object localization (i.e., pose), which includes position and orientation, has profound significance. The application of object pose estimation varies from industry automation to entertainment industry and from health care to surveillance. The objective of pose estimation of objects is significant in many cases, like robots manipulating objects, accurate rendering of Augmented Reality (AR) among others.

As a part of automation in industries, we use robots to serve in many tasks which deals with various objects in its environment. In an industry the arm type robots are common, which are used to manipulate the objects. So, the robot should be capable of recognizing the objects and estimating the pose of them with respect to its location in such a way that it can do a particular task such as pick and place, parts assembly, amongst others.

This paper deals with the problem of 3D object recognition and relative pose estimation using depth information for object-robot interaction. The combination of *position* and *orientation* is referred to as the pose of the object. This work on pose estimation is to be integrated on two different platforms: 1) Robot Pyrene, a humanoid robot for picking the objects like drill, brick (typically, which are laid on pavements), small cylindrical object and 2) Unmanned Aerial Vehicle (UAV) for picking the objects like brick (in shape of cuboid). This task brings challenges such as, noise in depth information, occlusion of objects in the scene, message delay between the robot and a remote system due to communication lag among others.

The pose estimation stated is a visual-based approach using a RGB-D sensor, embedded in the robot, which provides 2D and 3D cues. The conventional RGB or the mono-color image is a 2D data and 3D data is fetched from the depth map. In computer vision, a depth map is a single channel image that contains the distances of the surfaces of observed objects from a view point. Here, pure 3D information is used to detect and estimate the pose of the objects in the scene as depth information enhances the accuracy of pose estimation. The depth- based algorithms have an advantage of independence from object texture and invariance to illumination. This approach is tested with Microsoft Kinect v1 and Orbec Astra Pro 3D sensors.

In order to estimate the pose of an object, first, it is required to create a 3D model of the target object in an off-line phase, and second, pose estimation of the object can be calculated by aligning the current object point cloud to the model point cloud with initial alignment and refining the pose with Iterative Closest Point (ICP) technique in an on-line phase.

This work is in collaboration with other group in the lab who develop the path planning and control of humanoid that will interact with the objects. In the context of perception, this work gives them the best accurate pose of the object to grasp.

We have tested this approach on the datasets we developed with different objects (such as drill, brick among others) recorded with 3D sensors (like Microsoft Kinect). We present both qualitative and quantitative evaluations of our approach on these datasets. This paper has the follwoing structure: we

present the related work in Section 2. The formulation of our methodology for pose estimation is given in Section 3. Section 4 presents qualitative and quantitative results. Last, Section 5 describes conclusions and future work.

II. Related Work

The topic of object recognition and pose estimation has been widely researched in the last decade, but there are many unresolved issues and challenges. The methods based on sparse feature[6] have shown good results for accurate pose estimation. Their popularity declined in the scenario of robotic applications because they rely on texture. In Hough-voting based methods, all pixels cast vote into a quantized prediction space. The cell with majority of votes is taken as the winner. Hough voting scheme for 2D object detection and pose estimation is used in [7].

Template based methods [8], [9] are also applied to estimate the pose. The template is scanned across the image and a distance metric is computed at each position to find the best match. A multi-modal template matching approach is proposed by Hinterstoisser *et al.* [10] which is able to detect textureless objects in highly cluttered scenes. A distance based approach is proposed by Lai *et al.* [11] for object classification and detection. Ruotao He *et al.* [12], proposed a template matching based on LINEMOD for object detection and pose estimation.

With the advent of 3D acquisition systems (e.g Microsoft Kinect), there is a growing interest in 3D object recognition and pose estimation using depth data. Many advantages, like easier segmentation, robust pose estimation and new geometrical features has made the researchers to shift their focus from 2D approach to the analysis of RGB-D data. A method for object recognition in cluttered scenes using Point clouds has been developed by Papazov and Burschka[13].

A lot of work has also been done in head pose estimation. An approach using key-frames with offline learning and online pose estimation using ICP is presented in [14]. Li *et al.* [15], proposed 3D head pose tracking using ICP with online face template reconstruction. Generalized Adaptive View based Appearance Model (GAVAM)[16] presents probabilistic framework which integrates all three pose estimation paradigms dynamic/motion based approaches, static user-independent, static user-dependent.

Pose estimation as a energy minimization with global hypothesis is presented in [17]. R. Brigier *et al.* [18], proposed symmetry aware evaluation of 3D object detection and pose estimation. A local bundle adjustment technique based on key frames and previous frames is proposed to avoid drift and jitter [19]. Object detection using multimodal point pair features and geometric edge extractor and pose estimation using local voting scheme, clustering and ICP is presented in [20]. W. Czajewski *et al.* [21], proposed a VFH based object detection and Camera Roll Histogram (CRH) descriptor based pose estimation with ICP as pose refinement. Single image object detection and pose estimation using deformable parts model classifier based on gPb contours with the help of superpixel and chordiogram matching is proposed by M. Zhu *et al.* [22]. Probabilistic classifier based on maximum likelihood, by defining Probability Density Function (PDF) for the pose of the object, given correspondences between scene and model features is presented in [23].

With the recent trend and its success in the field of 2D image analysis, the deep convolutional neural networks emerges as a natural consequence in application to 3D object recognition. Due to the complexity of 3D information, approaches based on depth data could not tackle all the challenges although it has been used over its RGB counterpart. To jointly perform depth prediction and surface normal estimation, an adaptive multi-scale CNN architecture was proposed in [24]. The performance of deep learning methods is better than most of classical approaches in object detection and model alignment.

Although deep learning methods started to outperform the conventional techniques, they require very huge data sets and expensive processing units for computation. Even on powerful GPUs the training time requires several days for CNNs. The evaluation of different approaches should not be based exclusively on their pure performance but also on the effort put during the training phase and its cost. If the only goal is accuracy CNNs will definitely be the answer, but for cost-effective solutions, classical approaches should still be considered.

Considering the state-of-art techniques, we proceed further with the pose estimation using the depth data, due to the fact of it being invariance to illumination and independent from object texture. We use the classical ICP algorithm to estimate the pose of objects considering the fact that it does not require very huge dataset and expensive processing units like deep learning methods.

A. Framework

RGB-D cameras have been widely used in many computer vision and robotics applications such as pose estimation [25], 3D reconstruction [26], visual SLAM [27], amongst others. We used Microsoft Kinect v1 and Orbec Astra Pro sensors for this approach and open source libraries such as OpenCV[28], PCL[29] and ROS[30] to realize this work.

III. Methodology

This work presents the pose estimation in two folds: 1) offline phase - to form 3D model of the object and, 2) online phase - pose estimation by aligning object and model. A general diagram of the proposed pose estimation system is shown in Figure 1.

An RGB-D image captured by a sensor is firstly converted into a point cloud through the camera intrinsic parameters. Then, the point cloud is filtered and segmented into clusters. Below is a quick overview of the two steps of our method:

First, in an offline phase, we acquire a set of RGB-D images of the object in different views. Then, we convert them to point clouds and perform an initial filtering to remove those points that have a depth greater than a set threshold. We place the objects on a rotating platform (a plane) to reconstruct their 3D model. We can estimate the

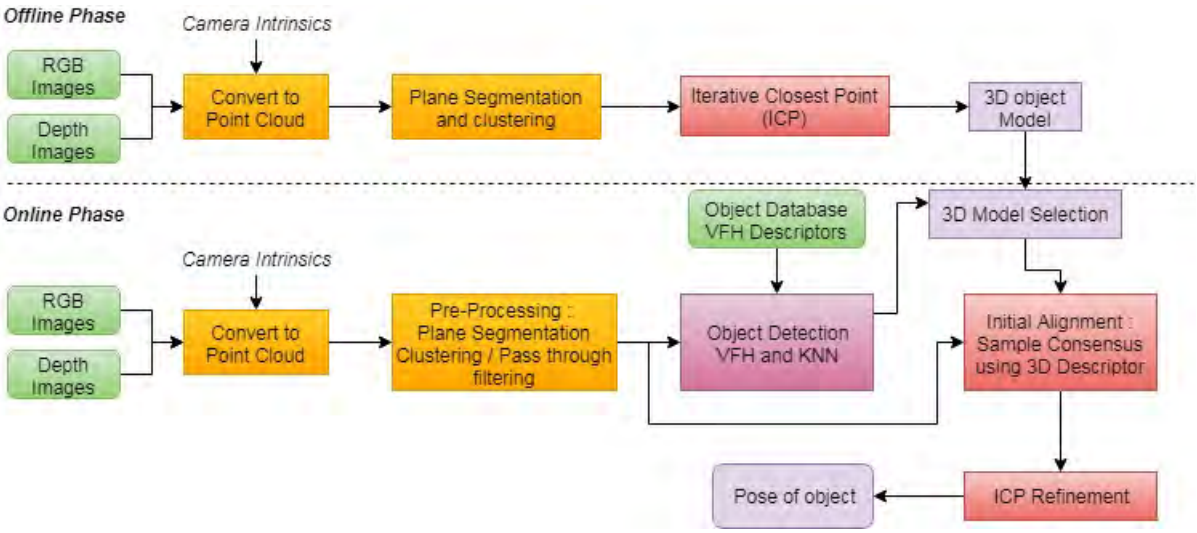


Fig. 1: Overview of the proposed vision system

plane and segment the object on the plane. By this way we could segment it from the scene and acquire the point cloud of object in different viewpoints by rotating the platform and performing the steps as described above. As we get point clouds of object in different viewpoints, we apply Iterative Closest Point (ICP) over all the point clouds to register them and then we form a 3D representation of the object, as shown by the red block in offline phase in Figure 1, which is later used in an online pose estimation phase . The point clouds of the object in different views can also be used to create a part of dataset for object detection. We compute View Point Feature Histogram (VFH)[31] descriptor for the point clouds of the object and represent the VFH descriptors as a kd-tree for efficient search with K-Nearest Neighbors (k-NN), in online phase for object detection.

Second, in an online phase, we acquire the RGB-D image of the scene to detect and estimate the pose of the object. We convert the RGB-D image into a point cloud and then we apply a pre-processing step that suppress flat areas, i.e. the table, and segment the objects by clustering them. The segmented clusters on the planar surface are described with View Point Feature Histogram (VFH). The VFH descriptor of the segmented clusters are compared against the descriptors of the target object using k-Nearest Neighbors. The clusters are classified according to number of matches and the labels of the corresponding object is assigned, we choose the 3D model of that particular object and we proceed to estimate the pose. We apply an initial alignment between the object and the model using sample consensus and a 3D descriptor like FPFH. Finally, we refine the alignment of the object and model using Iterative Closest Point (ICP) algorithm to estimate accurate pose of the object.

A. 3D Model Reconstruction - Offline phase

We reconstruct the 3D model of the object by acquiring the depth frames of the object (placed on a plane) in different view angles, converting them to point clouds followed by clustering of object to segment it from plane. Then, we apply ICP to register all the frames and reconstruct the 3D model of the object.

1) Plane Segmentation and Clustering: To reconstruct the 3D model, we have kept the objects on a rotating plane to observe the object from all points of view. In order to form the 3D model of the object, it is indeed necessary to remove the irrelevant points from the scene trying to keep only the points corresponding to the object. This is done by the plane segmentation and clustering module - the idea is to form a plane model using RANSAC (Random Sample Consensus)[32].

2) 3D Model - Registration using ICP: We do pairwise registration of 3D point clouds, i.e., aligning two point clouds with overlap by estimating the transformation between both.

The problem of registration tries to find the correspondences between source and target point and estimate a transformation. The transformation when applied on source point cloud, aligns all pairs of corresponding points with target point cloud. Here, the corresponding points are usually not known and needed to be determined.

Typically, the registration consists as follows :

1) *Selection:* Sampling of point clouds to speed up the process.
2) *Matching:* Estimating the correspondences between points in subsampled point clouds.
3) *Rejection:* Filtering the false correspondences.
4) *Alignment:* Assigning an error metric and minimizing it to find the optimal transformation.

The ICP algorithm was developed by Besl and Mckay[34] as an iterative registration method. There, the closest points in Cartesian space are considered correspondences of each others. ICP searches the closest point correspondences (*matching*) and align the found point pairs (*alignment*). These two steps are repeated until convergence thereby iteratively refining the alignment between source and target point cloud.

If there is perfect overlap[1], the alignment converges to global minimum, i.e., optimal alignment. But the main drawback is that the algorithm may get caught in local minima if there is very less overlap or if initial alignment is bad. In order not to get caught in local minima, we have to suppress the false correspondences (*rejection*) as they affect negatively the registration results.

We used uniform sampling to sample the point cloud as it preserves the geometry of the object. The correspondence estimation basing on the normal information is applied in this approach taking the advantage of normals to correctly estimate the correspondences. Invalid correspondences can negatively affect the registration results, therefore correspondence rejection is a vital step in the registration pipeline. It filters the point pairs, matched in the correspondence estimation stage, in order to facilitate the transformation estimation and improve convergence towards global minimum. Correspondence rejectors based on distance threshold and normal compatibility are used in this work. The transformation estimation is based on minimization of error metrics. There are two main error metrics to be minimized that have been considered: *point-to-point* (Eq. 1) and *point-to-plane* (Eq. 2).

$$E_{point-to-point}(T) = \sum_{k=1}^{N} ||Tp_k - q_k||^2 \tag{1}$$

$$E_{point-to-plane}(T) = \sum_{k=1}^{N} ((Tp_k - q_k).n_{q_k})^2 \tag{2}$$

where (p_k, q_k) is the k-th of the N pair correspondences from the source cloud to the target cloud.

B. Pose Estimation - Online Phase

In order to estimate the pose of the object in the online phase, the RGB and depth image of the scene is acquired using 3D sensor (e.g., Kinect, Astra Pro). The RGB and depth images are converted to point clouds. If the object is placed on a plane surface (i.e., a table), we do the plane detection and object clustering as described in section III-A.1. Then the object point cloud is aligned with the respective model using initial alignment and the pose is refined by final alignment using ICP.

1) Object Recognition: In this application, we use a global recognition method based on Viewpoint Feature Histogram (VFH)[31] descriptor, as global features have the ability to represent an entire object with a single feature vector, the recognition process is faster and robust to noise, which is important for near real-time applications.

We firstly build the training set by acquiring the point clouds of objects from different viewpoints. We compute the VFH feature descriptor for every point cloud and convert it to Fast Library for Approximate Nearest Neighbors (FLANN) format. As kd-tree is one of the efficient search structures, we form a kd-tree of the training data and store if for further testing.

In testing phase, the target object is clustered and its VFH descriptor is computed, matching is performed using k-Nearest Neighbor search from the FLANN[35] library with a Chi-Square metric using the kd-tree built in the training phase.

2) Initial Alignment: Due to its greedy nature, ICP algorithm requires a reliable initial alignment to avoid converging to local minima. Therefore, we apply an initial alignment between the scene and model point clouds before refining the alignment with ICP.

We use SAmple Consensus Initial Alignment (SAC-IA) which tries to maintain the same geometric relations between correspondences without having to try all combinations, it is limited to set of correspondences. Here we use Fast Point Feature Histogram (FPFH) descriptor as a similarity metric for the correspondence pairs. FPFH is a simplified version of Point Feature Histogram (PFH) to reduce the computational complexity.

SCA-IA. This approach is more detailed in [36] and it considers all possible matching pairs for initial alignment. But in SCA-IA, the matching pairs are sampled and we follow following scheme:

1) We select p sample points from source cloud with distance between any pair greater than a distance threshold.
2) For each of points selected in previous step, we find a list of points in target cloud whose histograms are similar to source cloud sample points histogram. Out of all matched histogram, we select one randomly and form correspondence.
3) Thus, we define an error metric to compute quality of transformation and compute the rigid transformation defined by sample points and their correspondences.

These three steps are repeated until convergence, and the transformation that resulted is used to roughly align the source to target cloud.

3) Final Alignment - ICP: With the help of initial alignment described in section III-B.2, the scene and model point clouds are aligned coarsely which are needed to be aligned finely using ICP.

ICP requires an initial alignment due to its greedy nature, this is provided by SCA-IA. Now, we can perform ICP on the initially aligned model and scene point cloud to get the accurate pose of the object in scene.

As described in section III-A.2, we register (align) the pair scene-model point clouds.

IV. Results and Discussion

The process of initial alignment described in Section III-B.2 is computationally expensive (Table I) , we follow a strategy described below to estimate the pose of the objects in real time.

In order to align the model and object, we perform the SCA-IA method for the first frame, this provides a good

[1]The overlap that has majority of corresponding pairs between source and target point cloud

initial guess of object pose which drastically reduce the computational time. The next frames only need a fine alignment, by ICP, considering the previous estimation. We place our model cloud at a predefined position and orientation. We estimate the pose of the object in first frame using SCA-IA and ICP then we align the model with the object in scene and leave the model at the aligned place. In the next frame,we use the previous estimation as the initial guess. This is based on the assumption that - the movement of objects between two consecutive frames is small. With this approach, we are not always required to compute the SCA-IA for every frame, thereby reducing the computational complexity. However, we can recover the rigid body transformation between the initial model and the model aligned to previous frame using Singular Value Decomposition (SVD) and one-to-one corresponding pairs of the model.

A. Datasets

The humanoid 'Pyrene' is needed to grasp some objects, like an electric drill, a brick and a cylinder like object as shown in Figure 2. We acquired our own datasets of these objects in such a way that we can apply a reconstruction method that generates 3D models of each object. Those models are used later to test pose estimation in different scenarios.

We acquired different datasets of the objects for 3D reconstruction and pose estimation. We acquired two types of datasets to test. One, a dataset which had object on a table with changing poses. Second, a dataset in which object is held in hand by a person and changing the pose of object to estimate the pose. For the quantitative evaluation of pose estimation, we have used a commercial Motion capture system (Mo-cap) to recover the ground truth data of the pose of the objects.

B. Qualitative Evaluations

This work of pose estimation is oriented as a module that is used over two different robotic platforms, each with a different sensor: 1)Humanoid 'Pyrene', which has an embedded Orbec Astra Pro 3D sensor and 2) UAV which has a Intel Realsense Euclid sensor on board. We have also tested this approach on Microsoft Kinect v1, for initial testing before integrating on the robotic platforms.

1) Results of pose estimation of objects acquired using Microsoft Kinect v1: The pose of three different objects (brick, drill and yellow cylinder), acquired using Microsoft Kinect sensor, is estimated as detailed in Section III-B, by aligning the respective object models created in Section III-A to the objects in the scene.

We have tested our pose estimation approach on two scenarios, viz., 1) the object is placed on a table (a plane) and 2) the object is held in hand by a person.

Object - Brick. The pose estimation of the brick with partial occlusion (from a hand) is shown in Figure 3a. We can see the red color point cloud depicting the 3D model of the brick which is aligned with the point cloud of brick in the scene. ICP results in a proper alignment of the model - and the object in scene, we can recover the final transformation from the ICP which is the pose estimation of the object.

We had observe during rapid movements of the object in the scene that the alignment of the model and the object in scene for pose estimation - takes more time than usual as the object needed to be initially aligned by SCA-IA.

During the major occlusion of the brick (only small part of brick is visible) the ICP fails to align the model as shown in Figure 3c. We can also see in the Figure 3d that the brick in the scene and the model is realigned after the major occlusion.

A short video of pose estimation of brick (held by hand) is provided as supplemental material at this link[2].

A short demonstration of pose estimation of brick on the table is provided as a supplemental material at this link[3].

Object - Yellow Cylinder. The proper alignment between the model and object results in a accurate pose estimation as shown in Figure 4.

In spite of been occluded with both hands, we can see a correct alignment of the object with the model in Figure 4. In the case of the cylinder, when there are rapid movements of the object, the alignment takes more time than usual as it needs to undergo the process of initial alignment. We also observed that the pose estimation of the yellow cylinder is bad when the object is occluded in majority.

A short demonstration of the pose estimation of the yellow cylinder is provided as a supplemental material at this link[4].

Object - Drill. The pose is estimated when the drill is held in hand as seen in Figure 5.

We observed some problems with the alignment of drill with the model when this is viewed with major occlusion (only some part of drill is viewed in sideways) as shown in Figure 5.

A short demonstration of pose estimation of drill is provided as a supplemental material at this link[5].

2) Results of pose estimation of objects acquired using Orbec Astra Pro: The Orbec Astra Pro 3D sensor is embedded on the robot 'Pyrene' [37], which is connected to a remote computer on WiFi network. Due to this type of connection, there is communication latency between robot and computer to recover the ROS topics related to point clouds. Instead, we tried to recover the depth image and we convert it into a point cloud using intrinsic parameters. Due to latency, the RGB and depth images are not synchronized, so we could not use the color information in the creating of the point clouds.

Object - Brick. The pose estimation of a brick held in hand is shown in Figures 6a and 6b. We observed that, if the brick

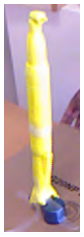
[2] https://www.youtube.com/watch?v=HoIW_qzToWU&feature=youtu.be

[3] https://www.youtube.com/watch?v=tJQIqIBMq-4&feature=youtu.be

[4] https://www.youtube.com/watch?v=OPh9xTyCC4c&feature=youtu.be and https://www.youtube.com/watch?v=L1omC3XFNWA&feature=youtu.be

[5] https://youtu.be/VY4g3-xFZSM and https://www.youtube.com/watch?v=ZPO-CSAq5tI&feature=youtu.be

(a) Brick



(b) Cylinder

(c) Drill

Fig. 2: Objects for pose estimation

(a) (b) (c) (d)

Fig. 3: Pose Estimation of Brick with hand occlusion(a), Pose Estimation of Brick with ICP before major occlusion (b), during major occlusion (c) and after major occlusion (d).

Fig. 4: Pose Estimation of yellow cylinder (left and middle), with partial occlusion (right)

Fig. 5: Pose Estimation of drill (left and middle), Improper alignment of drill with the model during major occlusion(right)

is partially viewed the model would not properly align with the brick, which leads to wrong pose estimation.

A short demonstration of pose estimation of brick is provided as a supplemental material at this link[6].

Object - Yellow Cylinder. The pose estimation of yellow cylinder is shown in Figures 6c and 6d. When the yellow cylinder is occluded in majority the model could not align with the object as seen in this video[7].

Object - Drill. The pose estimation of drill held in hand is shown in Figures 6e and 6f. A short demonstration of pose estimation can be seen at this link[8].

[6] https://www.youtube.com/watch?v=8ptkc-3vFcw&feature=youtu.be and https://www.youtube.com/watch?v=R8IOfDtlKPA&feature=youtu.be

[7] https://www.youtube.com/watch?v=HcS91xX4Naw&feature=youtu.be

[8] https://www.youtube.com/watch?v=Hhuj3Cwn7hs&feature=youtu.be

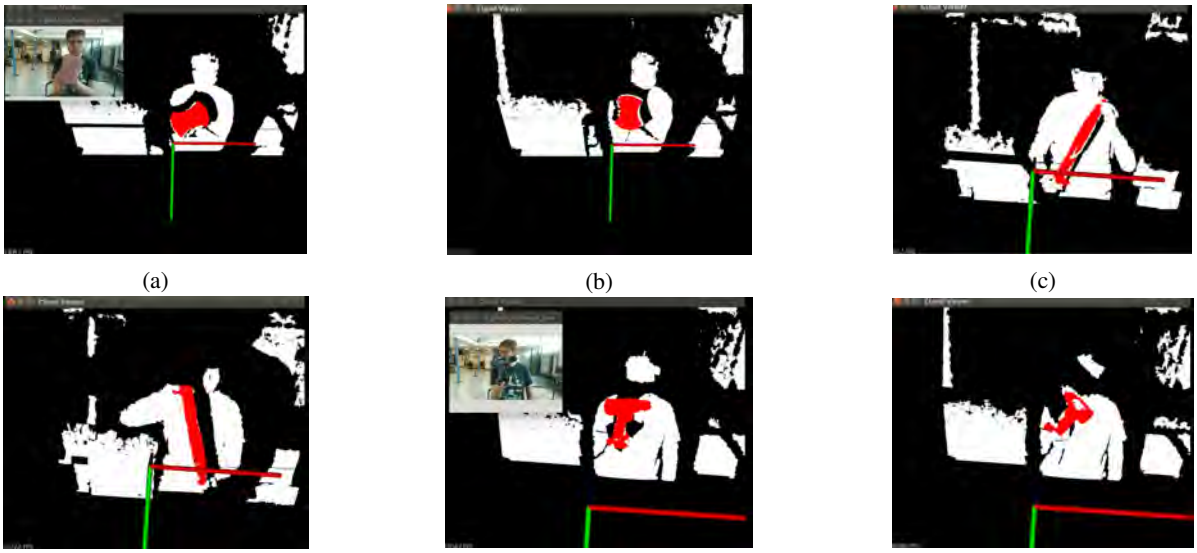


Fig. 6: Pose estimation of brick held in hand (a-b). We use only the depth information to estimate the pose. A color image of the scene is depicted at the top left. Pose estimation of yellow cylinder held in hand (c-d). Pose estimation of drill held in hand (e-f)

C. Quantitative Evaluations

The average time taken by the various modules in the approach is as shown in Table I. The first frame need to have initial alignment, this takes around 1.3 sec to process and followed by approximately 2 frames per second (fps) processing the rest of frames.

TABLE I: Average time taken by each module

Module	Time (in ms)
Convert to Point Cloud	10
Table Segmentation	200
SCA-IA	800
ICP	350
Total (with SCA-IA)	1360
Total (without SCA-IA)	560

For the quantitative evaluation of pose estimation, we have used a commercial Motion capture system (Mo-cap) [38] to recover the ground truth data of the pose of the objects. The Mo-cap system provides the pose of the object in its coordinate reference system. Our algorithm estimates the pose in 3D sensor coordinate system, therefore, we need to find the transformation from sensor to Mo-cap frame of reference in order to compare the estimated pose with the ground truth data.

We report the errors in roll, pitch and yaw angles of the pose estimated using our approach and the ground truth in Table II. The errors are a bit higher than what we visually see in qualitative evaluations (Section IV-B), because of the fact that the landmarks are not placed exactly on the object and also due to ambiguity caused by symmetry of objects.

TABLE II: Root Mean Square (RMS) error of orientation angles of different objects

Object	(Error in degrees)		
	Roll	Pitch	Yaw
Brick	12.09	3.12	11.89
Drill	7.93	7.94	5.42
Yellow Cylinder	16.11	13.84	7.64

V. Conclusion and Future Work

Using a RGB-D camera, an approach for pose estimation is proposed. This project is aimed to develop a pose estimation module, that is to be integrated on two robotic platforms: 1)Humanoid Robot 'Pyrene' and 2)UAV, with two different depth sensors on board. This brings us the challenges of communication latency, occlusion and noise in the data. The humanoid robot is intended to manipulate specific objects such as drill, brick and a small cylindrical object and UAV is intended to manipulate brick in the shape of cuboid.

This project solves the pose estimation of object with a two phase approach: 1)Offline phase - 3D model of the object is generated using ICP and 2)Online phase - model is aligned with the scene using an initial guess from SCA-IA algorithm, then further refined using ICP. Our approach uses the depth data taking the advantage of invariance to illumination and independence from object texture. We have tested this approach, both qualitatively and quantitatively using Microsoft Kinect v1, before the integration on the robotic platforms. The results are presented in Section IV. Our approach works well in case of partial occlusion of the object, but it is not robust in case of major occlusions of the object.

The offline phase for 3D model generation can be im-

proved to work online to simultaneously learn the model and estimate its pose, with a probabilistic pose prediction of object. We could improve the segmentation of the object and scene using the color information for more robust alignment of model and scene. We can improve the object recognition pipeline for more robust object recognition. We can add more rejectors (e.g., rejection of self-occluded normals) to reject the false matching pairs to increase the speed and robustness of the system.

REFERENCES

[1] J.G.F. Francis, The QR Transformation I, *Comput. J.*, vol. 4, 1961, pp 265-271.

[2] H. Kwakernaak and R. Sivan, *Modern Signals and Systems*, Prentice Hall, Englewood Cliffs, NJ; 1991.

[3] D. Boley and R. Maier, "A Parallel QR Algorithm for the Non-Symmetric Eigenvalue Algorithm", *in Third SIAM Conference on Applied Linear Algebra*, Madison, WI, 1988, pp. A20.

[4] G. Bugmann, M. Siegel, and R. Burcin. A role for robotics in sustainable development? In AFRICON, 2011, pages 14, Sept 2011.

[5] G. C. Fernandez, S. M. Gutierrez, E. S. Ruiz, F. M. Perez, and M. C. Gil. Robotics, the new industrial revolution. IEEE Technology and Society Magazine, 31(2):5158, Summer 2012.

[6] M. Martinez, A. Collet, and S. S. Srinivasa. Moped: A scalable and low latency object recognition and pose estimation system. In 2010 IEEE In- ternational Conference on Robotics and Automation, pages 20432049, May 2010.

[7] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(11):21882202, Nov 2011.

[8] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of tex-tureless objects. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(5):876888, May 2012.

[9] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Compar-ing images using the hausdorff distance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(9):850863, Sep 1993.

[10] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture- less objects in heavily cluttered scenes. In 2011 International Conference on Computer Vision, pages 858865, Nov 2011.

[11] K. Lai, L. Bo, X. Ren, and D. Fox. Sparse distance learning for object recog- nition combining rgb and depth information. In 2011 IEEE International Conference on Robotics and Automation, pages 40074013, May 2011.

[12] Ruotao He, Juan Rojas, and Yisheng Guan. A 3d object detec-tion and pose estimation pipeline using RGB-D images. CoRR, abs/1703.03940, 2017.

[13] Chavdar Papazov and Darius Burschka. An efficient ransac for 3d object recognition in noisy and occluded scenes. In Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto, editors, Computer Vision ACCV 2010, pages 135 148, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[14] JORGE FRANCISCO MADRIGAL DIAZ, Fr ed eric Lerasle, and Andr e Monin. 3D Head Pose Estimation enhanced through SURF-based Key- Frames. In WACV 2018, 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), page 9p., Reno, Nevada, United States, March 2018.

[15] S. Li, K. N. Ngan, R. Paramesran, and L. Sheng. Real-time head pose tracking with online face template reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(9):19221928, Sept 2016.

[16] L. P. Morency, J. Whitehill, and J. Movellan. Generalized adaptive view- based appearance model: Integrated framework for monocular head pose estimation. In 2008 8th IEEE International Conference on Automatic Face Gesture Recognition, pages 18, Sept 2008.

[17] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchyn- skyy, and C. Rother. Global hypothesis generation for 6d object pose estima- tion. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 115124, July 2017.

[18] R. Brgier, F. Devernay, L. Leyrit, and J. L. Crowley. Symmetry aware evaluation of 3d object detection and pose estimation in scenes of many parts in bulk. In 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), pages 22092218, Oct 2017.

[19] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(10):13851391, Oct 2004.

[20] B. Drost and S. Ilic. 3d object detection and localization using multi-modal point pair features. In 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission, pages 916, Oct 2012.

[21] Witold Czajewski and Krzysztof Koomyjec. 3d object detection and recog- nition for robotic grasping based on rgb-d images and global features. Foun- dations of Computing and Decision Sciences, 42(3):219 237, 2017.

[22] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3d object detec-tion and pose estimation for grasping. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 39363943, May 2014.

[23] Jonathan M. Huntley Harshana G. Dantanarayana. Object recognition in 3d point clouds with maximum likelihood estimation. Proc.SPIE, 9530:9530 9530 7, 2015.

[24] David Eigen and Rob Fergus. Predicting depth, surface normals and se-mantic labels with a common multi-scale convolutional architecture. CoRR, abs/1411.4734, 2014.

[25] Xiaoqin Wang, Y. Ahmet ekerciolu, and T. Drummond. A real-time dis- tributed relative pose estimation algorithm for rgb-d camera equipped visual sensor networks. In 2013 Seventh International Con-ference on Distributed Smart Cameras (ICDSC), pages 17, Oct 2013.

[26] K. Wang, G. Zhang, and H. Bao. Robust 3d reconstruction with an rgb-d camera. IEEE Transactions on Image Processing, 23(11):48934906, Nov 2014.

[27] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In 2014 IEEE Interna- tional Conference on Robotics and Automation (ICRA), pages 15241531, May 2014.

[28] Itseez Intel Corporation, Willow Garage. Open source computer vision li- brary. https://opencv.org/, 2018.

[29] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In 2011 IEEE International Conference on Robotics and Automation, pages 14, May 2011.

[30] Robot operating system. www.ros.org.

[31] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 21552162, Oct 2010.

[32] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and auto- mated cartography. Commun. ACM, 24(6):381395, June 1981.

[33] Euclidean cluster extraction. http://www.pointclouds.org/documentation/tutorials/cluster_extraction.php.

[34] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2):239256, Feb 1992.

[35] Fast approximate nearest neighbor search (flann). https://www.cs.ubc.ca/research/flann/.

[36] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 33843391, Sept 2008.

[37] PAL Robotics. Talos : High performance humanoid robot. https://www.pal-robotics.com/en/products/talos/.

[38] Motion capture (mo-cap). https://en.wikipedia.org/wiki/Motion_capture.

Mixed RFID-Vision Architecture for High Precision Free-Flow ETC

Sepideh Hadadi, VIBOT, University Bourgone-Franche Comte

***Abstract*— Current free-flow Electronics Toll Collection (ETC) systems may fail or fall out of calibration under certain circumstances. These failures at crowded highways will cost a lot for road owners and governments. The author will propose an architecture based on traditional RFID and imaging sensor to increase the reliability and the precision of the ETCs. The vision system employs CCD and IR sensor to grab images and detect and classify vehicles, extract License Plate(LP), segment and recognize PL content by the state-of-the-art object detection algorithm. The configuration of the RFID system including reader and tag antenna, effective detection range, frequency bandwidth selection will be discussed. Then, the dedicated camera module and a algorithm for vehicle detection and LPs analysis will be presented. Experimental result show high precision (above 95%) under normal condition and 80%-90% precision for the worse case scenarios. in addition, the proposed architecture is able to process incoming images at 30-47FPS on NVidia GPUs. This result proofs that combination of the vision system and traditional RFID can solve above reliability issue and fill precision gap of existing free-flow ETCs.**

I. INTRODUCTION

the precision of available ETC systems must be significantly improved to be deployable in crowded highways and convince road owners to replace tollbooths with such a system. On the other hand, recent improvement in image processing algorithm for unmanned vehicle put forward reliable solutions for old problems such as object detection and classification, tracking, image analysis and so on, which can be used in road monitoring and surveillance systems as well. The influence of the algorithms developed based on artificial intelligence such as neural network is unavoidable because they can imitate human vision and outperform previous algorithms in terms of speed and precision. thus constant improvement of the ETC systems is a routine work.

A. *Problems and challenges of current free-flow ETCs*

Available ETCs are mainly relaying on RFID tags and UHF technology for reading these tags. Fig.1 shows an example of free-flow ETC. There are certain problems in the RFID-based solution: 1- if due to any reason RFID system is failed there is no other means to keep them functioning in full precision 2- there is no way to track down toll evaders 3- the RFID ETCs are working in different wavelength thus it is hard to embed them in traffic monitoring system 3- a support team is needed to issue RFID tags and control their existence constantly, which imposes extra cost.

This work was jointly supported by Instrument Technology and Bourgogne Regional Council

S. Hadadi is with Department of Emerging Technology, Instrument Technology Group, London & UK- University bourgogne- Franche Comte, Dijon, France Sepideh_Hadadi@etu.u-bourgogne.fr

Fig. 1. overhead mounted RFID detector and vision based ETC

B. *computer vision solution for ETC improvement*

At a furtive glance, vision system is a strong choice to overcome above problems and increase the precision of the RFID-based ETCs. The combination of the RFID and vision system has two advantages: 1- if RFID system fails vision system will keep identifying vehicles and collecting their road toll without interruption. 2- it is possible to track down toll evader easier when ETC is equipped with vision as it records images, which can be dispatched to toll agent for further analysis and necessary actions.

the main contribution of the paper is to show how the ETC architecture can be established using RFID and vision module and state-of-the-art image processing algorithm. The paper is organized as follows: in section II, the proposed architecture will be presented and the RFID and vision component will be details. the ALPR algorithm based on YOLO object detection will be explained in section III. then, some experimental result will be presented and compared with other available solutions. the paper will end up to conclusion and proposing future works.

II. PROPOSED ETC ARCHITECTURE USING MIXED VISION AND RF COMPONENTS

The ETC system must be able to perform following task as minimum: 1)Vehicle identification, 2)Toll calculation, 3)Bank transaction and messaging, 4)Issuing receipt and record 5)vehicle registration. among them, Vehicle identification plays vital role because without correct identification non of the above can be completed. Fig. 2 shows the proposed architecture for ETC system. The ETC system runs the following sequences as shown in Fig. 2: 1- RFID transponder initiates the input module and will remain the main source of vehicle ID 2-ALPR process provides supervisory system for RFID and will be replaced by RFID if it fails. The RFID module has 4 main components ((Fig. 3)), i.e., RFID tag, Antenna, transmitter and receiver, processing

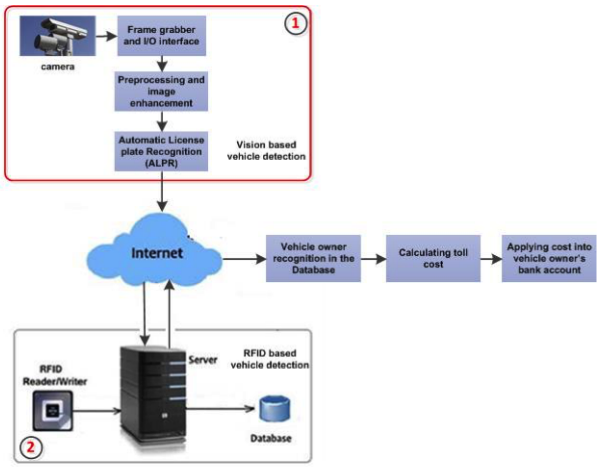


Fig. 2. overall architecture of a complete ETC system

unit. The ALPR module has three components (Fig. 4),, i.e., ALPR camera, termination box including processing unit, and database of interest. A CCD and an IR sensor with compact laser projector are encapsulated inside a waterproof and dust tight enclosure. The termination box provides easy access to all hardware interfaces and power supply to deploy camera and lighting units. Back office software system and database of interest are firmly connected together and support the central repository of all license plates along with tools to support data analysis, queries and reporting. the ETC platforms are capable to port data from both source in parallel.

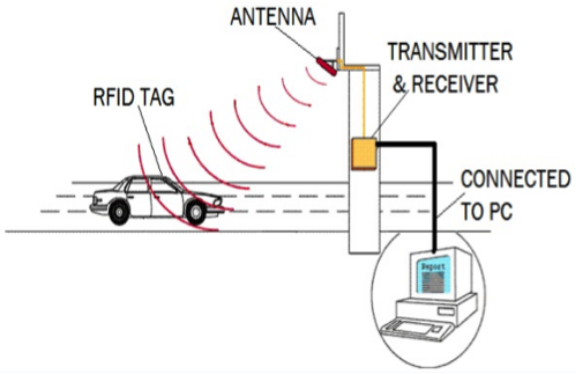


Fig. 3. example of RFID system used in ETC configuration

A. Vehicle classification and identification by ALPR

Almost all highly cited publications of object detection adapted approaches based on Faster R- CNN [5], [2], [1]. Finding regions with target object are a corner stone in all of the above algorithms. However, in YOLO [17], region finding and classification are integrated into one single stage.You Only Look Once (YOLO) algorithm is a state-of-the-art algorithm which uses Convolutional Neural Network to detect an object. Specific CNNs will be used for each ALPR stage. Thus, we can tune the parameters separately

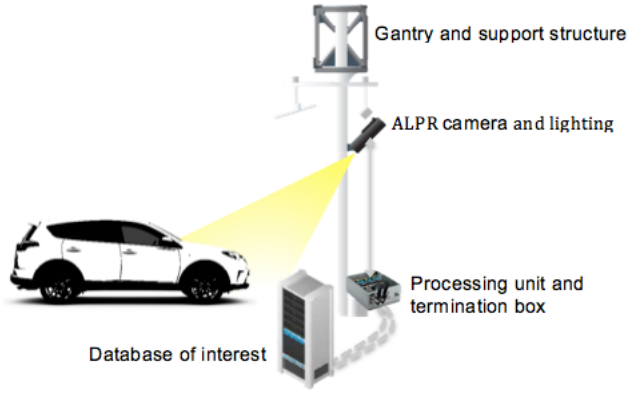


Fig. 4. ALPR based ETC system

in order to improve the performance for each task. The models used are: YOLOv3, Fast-YOLO, YOLOv2 and CR-NET [16], a CNN architecture trained using Fast- YOLO algorithm for character segmentation and recognition[16], [7] .

1) Vehicle and LP Detection: Two CNNs are trained in this stage: one for vehicle detection in the input image and other for LP detection in the detected vehicle.The Fast-YOLO, YOLOv2 and YOLOV3 algorithms were applied at this stage to be able to evaluate their performance in more realistic dataset. For simpler scenarios, the Fast-YOLO is able to detect the vehicles and their LPs correctly in much shorter time. However, for more realistic scenarios it might not be deep enough to perform these tasks with high precision. In order to use both YOLOv2 and Fast-YOLO, the number of filters in the last convolutional layer needs to change to match the number of classes. YOLO uses A anchor boxes to predict bounding boxes (A = 5) each with four coordinates (x, y, w, h), confidence and C class probabilities [14], so the number of filters is given by $Filters = (C + 5) \times A$ In the original dataset, only one class needs to be detected in both vehicle and LP detection (first the car and then its LP), so the number of filters in each task has been reduced to 30. On the other hand, the more complex dataset includes images contains cars and smaller vehicles such as motorcycles so the number of filters in the vehicle detection task must be 35 because the number of class is two (small and big vehicles). The entire image and the vehicle coordinates are used as inputs to train the vehicle detection CNN and the vehicle patch (with a margin) and the coordinates of its LP are used to learn the LP detection CNNs. By default, YOLO only returns objects detected with a confidence of 0.25 or higher. Different thresholds were checked and it turns out that the threshold can be set any values in 0.2-0.5 intervals in order to detect all vehicles having the lowest false positive rate. For LP detection, threshold set equal to 0, as there might be cases where the LP is detected with very low confidence (e.g., 0.1). Then, only the detection with the largest confidence will be kept where more than one LP is detected, since each vehicle has only one LP.

2) Character Segmentation: Once the LP has been detected, the proposed CNN by Montazzolli and Jung [16] (CR-NET) for character segmentation and recognition is applied. However, instead of performing both stages at the same time through architecture with 35 classes (0-9, A-Z, where the letter O is detected jointly with the digit 0), we chose to first use a network to segment the characters and then another network to recognize them [7]. The character segmentation CNN is trained using the LP patch as detailed above and the characters coordinates as inputs. If more than target characters is detected the selection of the character will be defined based on confidential score. If there are no overlaps (Intersection over Union $(IoU) > 0.25$), the ones with the lowest confidence levels will be discarded. Otherwise, the union between the overlapping characters will be performed to turn them into a single character.

3) Character Recognition: As mentioned in character segmentation, two networks will be used for character recognition. The characters and their labels are passed to network as input data during the training. The first four layers of the character segmentation CNN were removed and the remaining layers kept for digit recognition, this is because the test shows with and without them the results are similar. Since many characters might not be perfectly segmented, containing missing parts, and as each character is relatively small, even one pixel difference between the ground truth and the prediction might impair the character's recognition. By evaluation different padding values it turns out that 1-3 pixel length padding leads to high precision.

B. Vehicle identification using RFID tag

1) Principle of remote Radio Frequency (RF) sensing: In this paper, only Ultra-High Frequency RFID (UHF RFDI) will be discussed. More detail information in this regard can be found in the relate literatures. A typical passive RFID tag is made of a chip and an antenna. The energy needed to drive such a circuit is obtained from the electromagnetic wave transmitted by a RFID reader antenna. In a passive UHF RFID system as seen in Fig. 5, the reader antenna transmits a modulated signal with periods of un-modulated carry wave, which is received by the antenna of the tag. When the chip of the tag is turned on by the power received from the reader antenna, it will send back its identification by modulating the ID information into backscattered signal [4].

One of the most important criteria of performance for every UHF RFID is the read range. The maximum read range of the tag can be calculated by eq.1 as detailed in [13].

$$r = \frac{\lambda}{4}\sqrt{\frac{P_t G_t G_r \tau}{P_{th}}} \tag{1}$$

Where λ is the free space wavelength, P_t is the power transmitted by the reader, G_t is the gain of the reader antenna, G_r is the gain of the tag antenna, τ is the power transmission coefficient between the tag antenna and the chip, and P_{th} is the threshold of the chip power-up. When the power is needed for chip and the power transmitted by

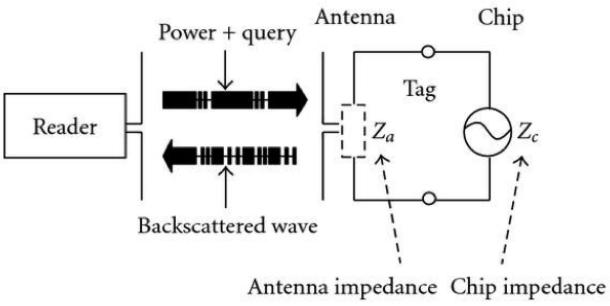


Fig. 5. Principle of passive back-scattered UHF RFID system

reader antenna is equal, G_r and τ determine the maximum read range of the UHF RFID tag. These two parameters are determined by the tag design [12]. The power transmission coefficient τ is determined by the impedance matching of the chip and the antenna, which can be calculated by eq.2:

$$\tau = \frac{4R_c R_a}{|z_a + z_c|^2}, 0 < \tau < 1 \tag{2}$$

Where, $Z_c = R_c + jX_c$ is the impedance of the tag chip, $Z_a = R_a + jX_a$ is the impedance of the tag antenna. When the impedances of the antenna and the chip are conjugate matching, the transmission coefficient τ could get the maximum value 1 and the most energy will be transmitted from the antenna to the chip when the reader calls the tag. In addition to above parameters, the bandwidth and the radiation pattern are also important for UHF RFID tag antenna design. Wide bandwidth antenna makes the tag to be read in a required bandwidth and the broadside radiation pattern makes the tag to be read in a wide direction scale [11],[9]. For instance, Fig.6 shows an antenna radiation patterns.

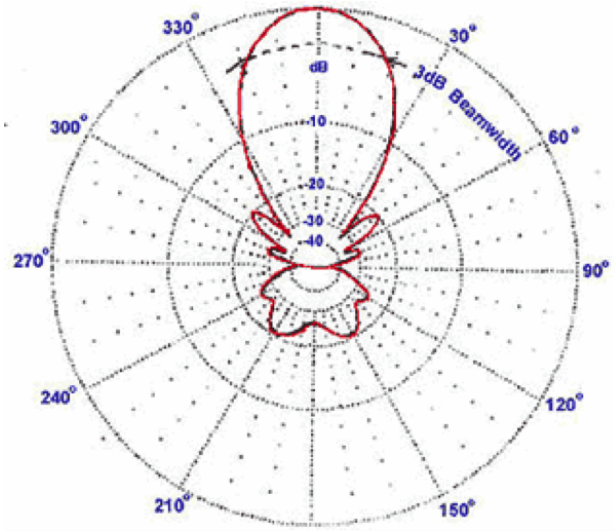


Fig. 6. This is a polar logarithmic plot of the same 10 elements radiation plot that emphasizes the shape of the major beam while compressing very low-level ($> 30dB$) side-lobes towards the center of the pattern.

III. EXPERIMENTAL RESULT

the evaluation of the the ALPR is conducted on two datasets. The first dataset (dataset I) is adapted from Menotti et al [6] and the second dataset (Dataset II) was made by enriching the first dataset further with image selected an added from CompCars dataset [18], ImageNet 1000-class [15], ImageNet 2012 [10], PASCAL VOC 2007 and 2012 [8].The dataset is split as follows: 40% to training, 40% to test and 20% for validation, using the same protocol division proposed by Goncalves et al. [6] in the original dataset. Overall results shows that YOLO algorithm is achieved high speed and at the same time high precision in both vehicle/LPs detection and LPs segmentation/recognition. 1) Vehicle Detection: Fig. 7 shows vehicle detection result for YOLOv3 on an image. The vehicles are classified into 4 main categories, i.e., cars, buses and trucks and motorcycle. Different confidence thresholds were evaluated for the vehicle detection. The confidence started with 0.7, however some vehicles were not detected. The threshold was reduced to 0.25 to detect all vehicles in the validation set. As per our experiences the threshold values must be kept between 0.2-0.3. For the test set, the threshold value was set to 0.13 and achieved a precision above 99%. Two example images were presented to the algorithm and the result shown in Fig. 8 achieved. 2) LP Detection: the LPs were completely segmented within the predicted bounding box for each vehicle in the validation set. All LPs (100% precision) were correctly detected in both validation and test sets as expected. Example of this result is shown in Figure 9. 3) Character Segmentation: the margin was set to 5%, 8% and 10% of the bounding box size in the test and the training set of the character segmentation CNN. The confidence threshold was changed from 0.5 to 0.1 with step of 0.05, achieved 99.89%, regardless. Therefore, the threshold was set to the lowest value (0.1) and 99.85% (6,712/6,722) precision was achieved. 4) Character Recognition: The padding values of 2-pixels and 1-pixel for letter and digit yield the best result. The result analyzed with and without temporal redundancy [7], [3].

Fig. 7. (left) vehicle detection and classification, (right) vehicle image

Fig. 8. (left) original images were taken under poor lighting condition and (right) high detection rate (99%) were achieved with threshold 0.13

Fig. 9. the license plate of the vehicles were detected and selected by a small green rectangular bonding box

The algorithm achieved recognition rate of 86.56% without temporal redundancy, all three letters and all four digits in the LPs were recognized in 86.32% and 98.63% of the time, respectively. The results are greatly improved with temporal redundancy information. The final recognition rate with redundancy is 93.53%, since the digits are correctly recognized in all vehicles and the letters in 95.75% of them. This result is given based on the number of frames correctly recognized versus total frames were processed. The both result were shown in Table I for comparison. Comparing two results shows 9.19% improvement with temporal redundancy. As expected, the commercial solution have achieved great recognition rates too. The great improvement in the proposed architecture for character recognition lies on separating the letter and digits recognition and performing on two networks, so each one is tuned specifically for its task.

TABLE I

RECOGNITION RATES OBTAINED BY THE PROPOSED ALPR SYSTEM, PREVIOUS WORK AND COMMERCIAL SYSTEMS ON TWO DATASETS

	≥ 6 characters		All characters	
ALPR algorithm	**Dataset I**	**Dataset II**	**Dataset I**	**Dataset II**
Result without redundancy				
Montazzolli and Jung [20]	90.55%	-	63.18%	-
Sighthound [7]	89.05%	62.50%	73.13%	47.39%
ALPR ETC	99.38%	87.33%	86.56%	64.89%
OpenALPR [28]	92.66%	54.72%	87.44%	50.94%
Result with redundancy				
Gonçalves et al. [25]	–	-	82.22%	-
Sighthound	99.13%	76.67%	84.44%	61.67%
OpenALPR	95.77%	73.33%	88.89%	75.00%
ALPR ETC	100.00%	88.33%	95.75%	83.33%

In Table II, we report the accuracy rate achieved in each ALPR stage separately, as well as the time required for the Nvidia Quadro 2000M and Nvidia Titan XP to perform each stage. The reported time is the average time spent processing all inputs in each stage, assuming that the network weights

TABLE II

PRECISION OBTAINED AND THE COMPUTATIONAL TIME REQUIRED IN EACH ALPR STAGE IN DATASET I

Hardware used		NVidia-Quadro		NVidia Titan	
ALPR Stage	Accuracy (%)	t(ms)	FPS	t(ms)	FPS
Vehicle Detection License	100.00%	12.195	82	4.082	245
Plate Detection Character	100.00%	12.048	83	4.065	246
Segmentation	99.75%	4.9260	203	1.656	604
Character Recognition	97.83%	32.258	31	11.49	87
ALPR (all correct)	85.45%	62.500	16	21.28	47
ALPR (with redundancy)	93.53%				

TABLE III

RESULTS OBTAINED AND THE COMPUTATIONAL TIME REQUIRED IN EACH ALPR STAGE IN THE DATASET II.

Hardware used		NVidia-Quadro		NVidia Titan	
ALPR Stage	Accuracy (%)	t(ms)	FPS	t(ms)	FPS
Vehicle Detection License	100.00%	28.571	35	11.158	90
Plate Detection Character	98.33%	11.765	85	3.9292	255
Segmentation	95.97%	4.4843	223	1.6548	604
Character Recognition	90.37%	37.037	27	11.559	87
ALPR (all correct)	64.89%	76.923	13	28.301	35
ALPR (with redundancy)	78.33%				

are already loaded.The average processing time for each frame was 21.31 seconds, an average of 47 FPS.

The algorithm was trained and evaluated using dataset II.

1) Vehicle Detection: the Fast-YOLO model was first evaluated, but the recognition rates achieved were not satisfactory. The confidence threshold was modified and the recognition rate was evaluated, the best precision achieved was 83.33%. This was expected since this dataset has greater variability in vehicle types and positions and images were recorded under more complex scenarios. The YOLOv2 and YOLOv3 models were tested for vehicle detection. 2) LP Detection: A small margin from the surrounding region (10%) of the LPs was selected to keep LPs completely within the predicted vehicle's bounding box. The precision attained by this method is 88.33%. the character segmentation CNN can be used to perform a post-processing in cases where more than one LP is detected. Since, the actual LP can be detected with very low confidence levels (i.e., < 0.1), many false negatives would have to be analyzed, increasing the overall computational cost of the system. 3) Character Segmentation: The precision obtained was 97.59% when disregarding the LPs not detected in the previous stage and 95.97% when considering the whole test set. 4) Character Recognition: The best results were obtained with 1 pixel of padding and data augmentation, for both letters and digits. The proposed system achieved a recognition rate of 64.89% when processing frames individually and 78.33% with temporal redundancy. Despite the great results obtained in the previous dataset, both commercial systems did not achieve satisfactory results in this dataset. OpenALPR performed better than Sighthound, attaining a recognition rate of 70% when exploring temporal redundancy information. Table I shows all results obtained for the dataset II.

The accuracy rate achieved in each ALPR stage is reported separately in Table III, as well as the time required for the proposed system to perform each stage. Despite the fact that a deeper CNN model is used in vehicle detection (i.e., YOLOv2), the ETC system is still able to process images at 35 FPS (against 47 FPS using Fast-YOLO). This is sufficient for real-time usage, as commercial cameras generally record videos at 30 FPS.

IV. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

An architecture based on RFID and vision proposed to improve the accuracy of traditional RFID ETCs. we presented ALPR algorithm based on recently proposed YOLOv3 algorithm, which uses neural network to provide real-time object detection. Evaluation of the algorithm in an experiment established for more complex scenarios demonstrates very high precision result for vehicle detection, plate extraction and segmentation as well and character recognition. Using appropriate off-the-shelf GPUs can accelerate the algorithm up to 47FPS while algorithm is still delivering high precision result, 78% of accuracy in the worse case scenarios. This result proofs that ALPR based on YOLO algorithm can be considered as strong alternative to supervise RFID-based ETCs. Besides, the output of the ALPR algorithm can be used to validate ID read by RFID antenna and decrease the overall error of the system.

B. Future Works

The RFID and ALPR section were built and tested separately. The next step is to assemble a complete free-flow ETC system using RFID components and ALPR camera module and test the system under real working condition in a highway. current datasets are not very well developed and does not contain more complex scenarios. It seems working on the datasets and recoding more images will be another task to accomplish in the near future.

V. ACKNOWLEDGMENTS

REFERENCES

[1] Openalpr cloud api. http://www.openalpr.com/cloud-api.html.

[2] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pages 354–370. Springer, 2016.

[3] Michael Donoser, Clemens Arth, and Horst Bischof. Detecting, tracking and recognizing license plates. In *Asian Conference on Computer Vision*, pages 447–456. Springer, 2007.

[4] Klaus Finkenzeller. *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. John Wiley & Sons, 2010.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[6] Gabriel Resende Gonçalves, Sirlene Pio Gomes da Silva, David Menotti, and William Robson Schwartz. Benchmark for license plate character segmentation. *Journal of Electronic Imaging*, 25(5):053034, 2016.

[7] Gabriel Resende Gonçalves, David Menotti, and William Robson Schwartz. License plate recognition based on temporal redundancy. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 2577–2582. IEEE, 2016.

[8] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[9] MingYin Lai and RongLin Li. Broadband uhf rfid tag antenna with parasitic patches for metallic objects. *Microwave and Optical Technology Letters*, 53(7):1467–1470, 2011.

[10] D Mishkin. Models accuracy on imagenet 2012 val, 2017.

[11] L Mo, H Zhang, and H Zhou. Broadband uhf rfid tag antenna with a pair of u slots mountable on metallic objects. *Electronics Letters*, 44(20):1173–1174, 2008.

[12] Ling-fei Mo, Hong-jian Zhang, and Hong-liang Zhou. Analysis of dipole-like ultra high frequency rfid tags close to metallic surfaces. *Journal of Zhejiang University-Science A*, 10(8):1217–1222, 2009.

[13] KV Seshagiri Rao, Pavel V Nikitin, and Sander F Lam. Antenna design for uhf rfid tags: A review and a practical application. *IEEE Transactions on antennas and propagation*, 53(12):3870–3876, 2005.

[14] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

[15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[16] Sérgio Montazzolli Silva and Claudio Rosito Jung. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In *Graphics, Patterns and Images (SIBGRAPI), 2017 30th SIBGRAPI Conference on*, pages 55–62. IEEE, 2017.

[17] Pavel Svoboda, Michal Hradiš, Lukáš Maršík, and Pavel Zemcík. Cnn for license plate motion deblurring. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 3832–3836. IEEE, 2016.

[18] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3973–3981, 2015.

Perspective camera models with windshields and polynomial descriptions for camera calibration

Di Meng and Tomas Pajdla

Abstract— **Camera systems used in cars are often used to detect pedestrians, vehicles and estimate the distance of other road users from the car. The correction of information collected by the camera is essential. The cameras are typically mounted in the car behind the windshield. In this contribution, the effects of refraction introduced by the windshield are analyzed. Three types of windshield models are proposed based on the geometry of optics and integrated with generalized pinhole camera. The projection equations are derived and polynomial descriptions are provided for calibration of the camera with windshields. The three shapes of windshield models are experimentally verified and the projection relations are proved to be established. The functions of projections of planar windshield camera models are provided and simulated.**

I. INTRODUCTION

Camera in the system is used to achieve functions such as pedestrian detection, guideboard detection or obstacle avoidance.[1] For instance, when an event of accident has to be avoided, the distance to the accident has to be determined by a camera. In order to achieve this, the relation between object in space and its matching pixels in image has to be estimated.

Camera model provides the projection equation which maps the points in space and the pixels in image. The initial parameters of the cameras can be obtained from camera production and then the precise parameters are from calibration process. Different cameras have different models for calibration. Unfortunately, the camera for automotive application is placed inside the car behind the windshield. The windshield has the effect of distorting the light rays from its original path. The disparity goes to the camera image which would influence the functionality of the camera. As we know, little disparity in pixel wise would cause large errors meters away. Compared with eliminating the effect of windshield in the optimization process, it is more necessary to develop a camera model with windshield which takes the effects of windshield into account.

The survey[2] shows that as the angle of incidence increased, a number of optical effects occur that are unfavorable to vision. Some of these effects are caused by the increased thickness of the transparent material through which the light must pass.

This work was partially supported by OP Research, development and education project IMPACT No. CZ.02.1.01/0.0/0.0/15_003/0000468

D. Meng is with MSc Erasmus Mundus in Vision and Robotics (VIBOT), University of Burgundy, 71200 Le Creusot, France di_meng@etu.u-bourgogne.fr

T. Pajdla is with Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 16000 Prague, Czech Republic pajdla@cvut.cz

Assistance systems use in many cases stereo cameras to observe the environment in front of the car.[3] The influence of a car windshield on depth calculation with a stereo camera system is analyzed.[4] The calibration is performed for the identical stereo camera system with and without a windshield in between. The base lengths are derived from the relative orientation in both cased and are compared. Distance values are calculated and analyzed. It shows that the difference of the base length values in the two cases is highly significant.

Camera calibration is also important for vehicle cameras in order to enable reliable results of object detection in image.[5] Automatic calibration while the car is driven is often performed by extracting road markings from the image. [6]use the extraction method to calibrate an embedded camera. Road lanes are detected and road boundaries derived. Therefore, a flat road is assumed for a plane-to-plane mapping with projection error minimization.[7] To link the camera orientation to a car coordinate system, the position of the car relative to the calibration pattern has to be measured with high precision.

Generic theory of camera models and projection formation is presented in [8]. It poses the problem in terms of polynomial equations constraining the image projection and camera parameters. In order to use this approach, it is important to develop practical polynomial models of image formation. A parameter free radial camera distortion calibration is developed in [9]. It might be used to correct additional local distortions but we believe that it needs to be combined with a physically motivated and exact refraction projection model to be practical.

Multiple flat refraction projection model is developed in [10]. It deals with multiple parallel glasses and addresses a calibration rather than projection computation. The resulting model is close to but still did not bring to a polynomial system even for a single parallel glass. More advanced models, e.g. a non-parallel surfaces glass model, are not considered. In the thesis of David Franklin Neralla,[11] he presented a flat windshield model for a single parallel glass. However, the model is only an approximation leading to a cubic equation solving for a projection. We will show an exact model, leading to a polynomial of degree four, can be developed in this thesis.

An iterative procedure for projection computation for a single parallel glass has been presented in [12]. The procedure is used in a larger optimization of complete camera calibration. Our projection model can replace their procedure by a formula with closed form or by a eigenvalue computation, which is better understood.

The contribution of this work is introducing different shapes of windshield models along with generalized pinhole camera, providing polynomial constraints for camera calibration in automotive application and formulating the projection functions.

II. Methodology

Three shapes of windshields are modeled which are planar glass, non-parallel surfaces glass and spherical glass. Camera parameters are not taken into account when describing the physical nature of optical path of a light ray from object in space to a camera through a windshield. The windshield models are described by parameters such as the vertical distance from the camera origin, the thickness of the glass and the refractive index of the inner medium. For the planar glass model, we obtained exact formula of solution for the forward projection equations. For non-parallel surfaces and spherical glass models, we provide polynomial constraints for computing the solutions instead of formulas. Which means that given the location of the point in space, the corresponding projection in image can be obtained by further knowing the camera parameters. The parameters of the introduced windshield model can be estimated from a sample set of windshields. With the windshield parameters fixed, there are only parameters of the perspective camera to be adjusted.

A. Planar windshield model

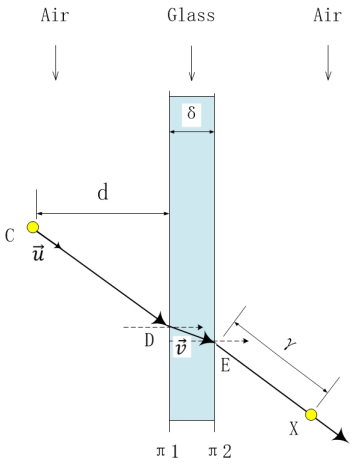


Fig. 1: Model of the planar glass and projection geometry

The planar windshield model is a transparent glass composed of two flat surfaces($\pi_1 and \pi_2$) parallel to each other. We assume the inner medium is homogeneous so that its refractive index is a constant value. The model is proposed mainly for automotive application which the practical environment is in the air. Thus, we set the refractive index of the outer medium to one which is the refractive index of the air. Figure 1 shows the sketch of the planar glass. This model is described by four parameters:

- The vertical distance from camera origin to surface π_1: $\boldsymbol{d}$ $(0 < d \in \mathbb{R})$
- The thickness of the glass: $\boldsymbol{\delta}$ $(0 < \delta \in \mathbb{R})$
- The normal of the glass: $\overrightarrow{\boldsymbol{N}}$ $(\|\overrightarrow{N}\| = 1)$
- The refractive index of the inner medium: $\boldsymbol{n}$ $(0 < n \in \mathbb{R})$

Point C is the camera origin. Point X is the location of point in space with γ distance far from the plane π_2. From the viewpoint of optics, the light ray starts from object X going through the glass with two times refraction and then projects on the camera. It is not doable to derive the forward light propagation directly. We firstly put effort to express the Point X by given pixel location. Then formulating the forward projection equations by solving the back projection equations. All the expressions are with respect to the world frame.

1) Derivation of Back projection with planar glass: As shown in the figure 1, $\overrightarrow{u}$ is a unit vector pointing at the pixel location in the normalized image plane. It goes from the camera origin and intersects at the point D with plane π_1, the left side surface of the model. Th perpendicular distance between camera origin C and the plane π_1 is *d*. We assume the angle between incident vector $\overrightarrow{u}$ and the normal vector of the glass $\overrightarrow{N}$ is α. The path length from C to D is $\frac{d}{\cos\alpha}$. Since we know the unit vector(direction) and how far it propagates, the intersection D can be expressed as

$$\overrightarrow{D} = \overrightarrow{C} + \frac{d}{\cos\alpha}\overrightarrow{u} \tag{1}$$

The projection ray refracts when passing through the plane $\pi 1$ from the air into glass medium. The path doesn't go along with the direction as from the camera origin. According to the Snell's law, the scalar form of the refraction is

$$\sin\alpha = n\sin\beta \tag{2}$$

The vector form of Snell's law is

$$\overrightarrow{v}_{refract} = \frac{1}{n}\overrightarrow{u} + (\cos\beta - \frac{1}{n}\cos\alpha)\overrightarrow{N} \tag{3}$$

The restriction of using the vector form of Snell's law is that the incident vector $\overrightarrow{u}$ and the normal vector $\overrightarrow{N}$ should both be unit vectors. To be noticed that, the obtained refracted vector is also a unit vector.

$$\overrightarrow{v} = m\overrightarrow{u} + \overrightarrow{N}\left(\sqrt{1 - m^2 + m^2(\overrightarrow{u}\cdot\overrightarrow{N})^2} - m\overrightarrow{u}\cdot\overrightarrow{N}\right) \tag{4}$$

The refracted vector which goes into the glass medium is expressed as in equation 4, where $m = \frac{1}{n}$ for simplicity. We assume that β is the angle between the refracted vector $\overrightarrow{v}$ and the normal of the glass. The two surfaces of the glass are flat, the planes $\pi 1$ and $\pi 2$ are parallel. The refracted ray intersects at location E with the second plane $\pi 2$. Since the thickness of the glass slab is δ, the path distance from intersection D to E is $\frac{\delta}{\cos\beta}$. Therefore, the vector from D to E is represented as

$$\overrightarrow{DE} = \frac{\delta}{\cos\beta}\overrightarrow{v} \tag{5}$$

Moreover, the intersection E can be derived

$$\overrightarrow{E} = \overrightarrow{D} + \overrightarrow{DE} = \overrightarrow{C} + \frac{d}{\cos\alpha}\overrightarrow{u} + \frac{\delta}{\cos\beta}\overrightarrow{v} \tag{6}$$

After going through the second plane π_2, the projection ray is again refracted. It is proved that the outgoing ray from intersection E is aiming at the same direction as the incident ray of $\overrightarrow{u}$. Thus for the object which is located at X and has the distance γ from the plane $\pi 2$, the representation is

$$\overrightarrow{X} = \overrightarrow{E} + \gamma \cdot \overrightarrow{u} \tag{7}$$

Substituting the above equations to 7, we get

$$\overrightarrow{X} = \overrightarrow{C} + \frac{d}{\cos\alpha}\overrightarrow{u} + \frac{\delta}{\cos\beta}\overrightarrow{v} + \gamma \cdot \overrightarrow{u} \tag{8}$$

which is the relationship between 3D point location and the 2D image pixel. In the equation, the angles α and β are not measured and not the parameters of the model. They are further derived.

$$\cos\alpha = \frac{\overrightarrow{u}\cdot\overrightarrow{N}}{\|\overrightarrow{u}\|\|\overrightarrow{N}\|} = \overrightarrow{u}\cdot\overrightarrow{N} \tag{9}$$

The denominator in the equation 9 can be removed for the magnitude of the unit vector is equal to one. $\cos\beta$ can also be expressed as $\overrightarrow{v}\cdot\overrightarrow{N}$. However, to achieve a simpler representation of $\cos\beta$, we derive it from $\cos\alpha$ by the relationship of Snell' law(equation 2).

$$\cos\beta = \sqrt{1 - m^2 + m^2(\overrightarrow{n}\cdot\overrightarrow{N})^2} \tag{10}$$

The equation 8 is extended

$$\begin{aligned}\overrightarrow{X} =& \overrightarrow{C} + \frac{d}{\cos\alpha}\overrightarrow{u} + \frac{\delta}{\cos\beta}\overrightarrow{v} + \gamma\overrightarrow{u} \\ =& \overrightarrow{C} + \frac{d\overrightarrow{u}}{\overrightarrow{u}\cdot\overrightarrow{N}} + \frac{\delta}{\sqrt{1 - m^2 + m^2(\overrightarrow{u}\cdot\overrightarrow{N})^2}}\cdot\big(m\overrightarrow{u} \\ &+ \overrightarrow{N}(\sqrt{1 - m^2 + m^2(\overrightarrow{u}\cdot\overrightarrow{N})^2} - m\,\overrightarrow{u}\cdot\overrightarrow{N})\big) + \gamma\overrightarrow{u}\end{aligned} \tag{11}$$

Moreover, the denominator is eliminated. The complete equation which links the 3D point and its projection with a planar glass is

$$\begin{aligned}\overrightarrow{u}\cdot\overrightarrow{N}\sqrt{1 - m^2 + m^2(\overrightarrow{u}\cdot\overrightarrow{N})^2}(\overrightarrow{X} - \overrightarrow{C}) = \\ d\overrightarrow{u}\sqrt{1 - m^2 + m^2(\overrightarrow{u}\cdot\overrightarrow{N})^2} + \delta m(\overrightarrow{u}\cdot\overrightarrow{N})\overrightarrow{u} + \\ \delta(\overrightarrow{u}\cdot\overrightarrow{N})\overrightarrow{N}\sqrt{1 - m^2 + m^2(\overrightarrow{u}\cdot\overrightarrow{N})^2} - \\ \delta m(\overrightarrow{u}\cdot\overrightarrow{N})\overrightarrow{N}(\overrightarrow{u}\cdot\overrightarrow{N}) + \\ \gamma(\overrightarrow{u}\cdot\overrightarrow{N})\overrightarrow{u}\sqrt{1 - m^2 + m^2(\overrightarrow{u}\cdot\overrightarrow{N})^2}\end{aligned} \tag{12}$$

2) Formulating the forward projection of planar glass: Forward projection of the light ray through a planar glass is necessary and of significance for camera calibration applied for windshield. Given the 3D point in space, the calibration of the camera can be done by acquiring the exact pixel position in 2D image. In order to obtain the pixel coordinate given the object location in space, the forward projection is investigated by solving the equation of back projection 12, since it provides the relationship between 3d point and its projection.

The aim of this section is to derive $\overrightarrow{u}$ which represents the pixel location in world coordinate from the equation 12. We observe that each term in equation 12 contains $\overrightarrow{u}$ and it is unachievable to extract it in the vector form. Therefore, to investigate the solvable possibility, the vectors in the equation are extended by elements.

We initiate that

$$\overrightarrow{C} = \begin{pmatrix}0\\0\\0\end{pmatrix}, \overrightarrow{N} = \begin{pmatrix}0\\0\\1\end{pmatrix}, \overrightarrow{u} = \begin{pmatrix}u1\\u2\\u3\end{pmatrix}, \overrightarrow{X} = \begin{pmatrix}x1\\x2\\x3\end{pmatrix}$$

Here, the camera origin is set as the origin of the world frame and specialize the normal vector of the planar glass to $[0,0,1]'$ which is a unit vector along the z axis. $\overrightarrow{u}$ and $\overrightarrow{X}$ are generic. And let

$$w = \sqrt{1 - m^2 + m^2(\overrightarrow{u}\cdot\overrightarrow{N})^2} \tag{13}$$

In order to make sure the equations are solvable, new variable w is involved to make the equation polynomial.

$$1 - m^2 + m^2(\overrightarrow{u}\cdot\overrightarrow{N})^2 - w^2 = 0 \tag{14}$$

By extending the vectors in equation 12, a polynomial equation set is obtained

$$-\delta m u_1 u_3 - \gamma u_1 u_3 w - d u_1 w + u_3 w x_1 = 0 \tag{15}$$

$$-\delta m u_2 u_3 - \gamma u_2 u_3 w - d u_2 w + u_3 w x_2 = 0 \tag{16}$$

$$-\gamma u_3^2 w - d u_3 w - \delta u_3 w + u_3 w x_3 = 0 \tag{17}$$

$$-m^2 u_3^2 + m^2 + w^2 - 1 = 0 \tag{18}$$

By solving this equation set, we saw that there were several solutions and only one of them is what we are looking for. There are some solutions containing $u_1 = 0, u_2 = 0$ since the equations 15 16 17 are without constant terms. In equation 17, each term contains u_3 and w. We then divide the equation 17 by u_3 and w. w is nonzero value for it is originally in the denominator. The equation 17 derives as:

$$-\gamma u_3 - d - \delta + x_3 = 0 \tag{19}$$

Therefore, the incorrect solutions which contains $u_3 = 0$ are removed by introducing the equation 19. The process can be proceeded to this step because the specialization of vector $\overrightarrow{N}$. The equation 17 would be more complicated and not all the terms contain u_3 and w if $\overrightarrow{N}$ was assigned randomly, which could not be simplified. Express u_3 from equation 19 and substitute it to equations 15 16 18, we obtain the equation set

$$\begin{aligned}\big((\delta u_1 - u_1u_3)\gamma - x_1d - x_1\delta + x_1x_3\big)w + \\ \delta m u_1 d + \delta^2 m u_1 - \delta m u_1 x_3 = 0\end{aligned} \tag{20}$$

$$\begin{aligned}\big((\delta u_2 - u_2u_3)\gamma - x_2d - x_2\delta + x_2x_3\big)w + \\ \delta m u_2 d + \delta^2 m u_2 - \delta m u_2 x_3 = 0\end{aligned} \tag{21}$$

$$\gamma^2w^2 + (m^2-1)\gamma^2 - m^2(d+\delta-x_3)^2 = 0 \tag{22}$$

By solving the above three polynomial equations, two solution sets are obtained whose w are with different signs. We know that $w = \cos\beta$ where β is the angle between the projection ray and the normal of the plane and is between 0π degrees, so that the constraint $w > 0$ established. A unique solution can be computed then. The followings are the components of $\overrightarrow{u}$ which is the formula of the solution.

$$u_1 = \frac{A_1}{B_1} \tag{23}$$

where

$$\begin{aligned}A_1 = &-(-dx_1 - \delta x_1 + x_1x_3)(d^2m^2 + 2d\delta m^2 - 2dm^2x_3 \\ &+ \delta^2m^2 - 2\delta m^2x_3 - \gamma^2m^2 + m^2x_3^2 + \gamma^2)^{\frac{1}{2}}\end{aligned} \tag{24}$$

$$\begin{aligned}B_1 = &\gamma\big((\delta - x_3)(d^2m^2 + 2d\delta m^2 - 2dm^2x_3 + \delta^2m^2 - 2\delta m^2x_3 \\ &- \gamma^2m^2 + m^2x_3^2 + \gamma^2 + d\delta m + \delta^2 m - \delta m x_3)^{\frac{1}{2}}\big)\end{aligned} \tag{25}$$

$$u_2 = \frac{A_2}{B_2} \tag{26}$$

where

$$\begin{aligned}A_2 = &-(-dx_2 - \delta x_2 + x_2x_3)(d^2m^2 + 2d\delta m^2 - 2dm^2x_3 \\ &+ \delta^2m^2 - 2\delta m^2x_3 - \gamma^2m^2 + m^2x_3^2 + \gamma^2)^{\frac{1}{2}}\end{aligned} \tag{27}$$

$$\begin{aligned}B_2 = &\gamma\big((\delta - x_3)(d^2m^2 + 2d\delta m^2 - 2dm^2x_3 + \delta^2m^2 - 2\delta m^2x_3 \\ &- \gamma^2m^2 + m^2x_3^2 + \gamma^2 + d\delta m + \delta^2 m - \delta m x_3)^{\frac{1}{2}}\big)\end{aligned} \tag{28}$$

$$u_3 = \frac{-d - \delta + x_3}{\gamma} \tag{29}$$

The vector pointing to the pixel location from camera origin is formed as $\overrightarrow{u} = [u_1, u_2, u_3]'$;. It can be further normalized by dividing the third element u_3 to get the pixel location which is the first two elements.

Moreover, we investigated a way to eliminate the parameters γ and d, for the distance between object and the glass cannot be measured and provided in practical application. Looking back to the polynomial equation set 15 to 17, each equation contains parameters γ and d. u_1 u_2 u_3 are the unknowns we are aiming to solve. Hence, the skew-symmetric cross-product matrix of $\overrightarrow{u}$ is applied here to eliminate γ and d.

$$[u]_\times = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$$

Multiplying the polynomial equations 15 16 17 by $[u]_\times$, the parameters γ and d are eliminated and we got a new polynomial equation set.

$$\begin{aligned}&\begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \begin{bmatrix} -\delta m u_1 u_3 - \gamma u_1 u_3 w - d u_1 w + u_3 w x_1 \\ -\delta m u_2 u_3 - \gamma u_2 u_3 w - d u_2 w + u_3 w x_2 \\ -\gamma u_3^2 w - d u_3 w - \delta u_3 w + u_3 w x_3 \end{bmatrix} \\ &= \begin{bmatrix} \delta m u_2 u_3 - \delta u_2 w + u_2 w x_3 - u_3 w x_2 \\ -\delta m u_1 u_3 + \delta u_1 w - u_1 w x_3 + u_3 w x_1 \\ u_1 x_2 - u_2 x_1 \end{bmatrix}\end{aligned} \tag{30}$$

The newly obtained polynomial set contains three equations but two of them are linearly independent. Since we have three unknowns, one more equation is needed to solve u_1 u_2 and u_3. As we know, $\overrightarrow{u}$ is a unit vector. By adding one more equation that the norm of $\overrightarrow{u}$ is equal to one, we got a new polynomial equation set to describe our glass model.

$$\delta m u_2 u_3 - \delta u_2 w + u_2 w x_3 - u_3 w x_2 = 0 \tag{31}$$

$$-\delta m u_1 u_3 + \delta u_1 w - u_1 w x_3 + u_3 w x_1 = 0 \tag{32}$$

$$u_1 x_2 - u_2 x_1 = 0 \tag{33}$$

$$u_1^2 + u_2^2 + u_3^2 - 1 = 0 \tag{34}$$

$$-m^2u_3^2 + m^2 + w^2 - 1 = 0 \tag{35}$$

After setting up the equations, we solve them using the Groebner Basis algorithm.[13] The monomials of unknowns are ordered as $u_3 > u_2 > u_1 > w$, which means after Gaussian elimination w is solved firstly and been substituted to the equation of u_1 to further solve u_1 and so forth for the u_2, u_3.

$$P(a_1)w^8 + P(a_2)w^6 + P(a_3)w^4 + P(a_4)w^2 + P(a_5) = 0 \tag{36}$$

$$P(b_1)w^7 + P(b_2)w^5 + P(b_3)w^3 + P(b_4)w + P(b_5)u_1 = 0 \quad (37)$$

$$P(c_1)w^7 + P(c_2)w^5 + P(c_3)w^3 + P(c_4)w + P(c_5)u_2 = 0 \quad (38)$$

$$P(d_1)w^7 + P(d_2)w^5 + P(d_3)w^3 + P(d_4)w + P(d_5)u_3 = 0 \quad (39)$$

The above equation set illustrates the form of Groebner Basis. $P(x_i)$ represents polynomials including parameters δ and m. As we noticed, the equation 36 contains only one variable w. The highest degree of w is 8 but the w only has even degrees. Hence the degree of the equation is four, there are four solutions for w. From the computation, two of the solutions for w can be removed for they are complex numbers. The solutions of w which are real numbers are kept and then substituted to the equations 37 to 39 to derive u_1 u_2 and u_3. We got two sets of solutions of $\overrightarrow{u}$, one is of positive elements and the other one is of negative elements. They are the vectors lying on one straight but point to opposite directions. Our vector $\overrightarrow{u}$ is always point to the camera origin, so that the positive one is the unique solution we are looking for.

B. Non-parallel surfaces windshield model

In reality, it is possible for the windshield not to be as planar as ideal. The surface from each side of the glass may be uneven or vary in angles. For that case, the projection rays do not exactly follow the path as we modeled in the last section. Thus, a glass model with non-parallel surfaces is introduced. Similar with the planar glass model, it consists of two flat surfaces. The two surfaces are with a certain angle. The inner medium is homogeneous.

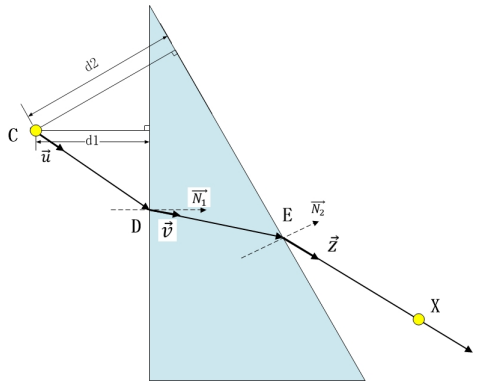


Fig. 2: Model of the non-parallel surfaces glass and projection geometry

This model is described by five parameters:

- The normal of the first surface: $\overrightarrow{\boldsymbol{N_1}}$ $(\|\overrightarrow{N_1}\| = 1)$
- The normal of the second surface: $\overrightarrow{\boldsymbol{N_2}}$ $(\|\overrightarrow{N_2}\| = 1)$
- The vertical distance from camera origin to the first surface: $\boldsymbol{d_1}$ $(0 < d_1 \in \mathbb{R})$
- The vertical distance from camera origin to the second surface: $\boldsymbol{d_2}$ $(0 < d_2 \in \mathbb{R})$
- The refractive index of inner medium: $\boldsymbol{n}$ $(0 < n \in \mathbb{R})$

1) Derivation of Back projection with non-parallel surfaces glass: We use the same method to derive the back projection equation as we did for planar glass. The difference is that the length of propagation path inside the glass is not given, we introduce a parameter τ representing the length of vector $\overrightarrow{DE}$. $\overrightarrow{E}$ can be expressed with τ. The equation of the tilted plane can be formed by knowing a in-plane point E and the normal $\overrightarrow{N_2}$ of the plane. The second plane π_2 is

$$\pi_2 : \overrightarrow{N_2}^T(\overrightarrow{D} + \tau\overrightarrow{v}) - d_2 = 0 \quad (40)$$

The unknown parameter τ can be computed from equation (40) and point E is derived.

$$\tau = \frac{d_2 - \overrightarrow{N_2}^T \cdot \overrightarrow{D}}{\overrightarrow{N_2}^T \cdot \overrightarrow{v}} \quad (41)$$

$$\overrightarrow{E} = \overrightarrow{D} + \tau\overrightarrow{v} \quad (42)$$

Then the projection ray is refracted the second time after intersection E. The outgoing vector from second surface is

$$\overrightarrow{z} = n\overrightarrow{v} + \overrightarrow{N_2}\Big(\sqrt{1 - n^2 + n^2(\overrightarrow{v} \cdot \overrightarrow{N_2})^2} - n\overrightarrow{v} \cdot \overrightarrow{N_2}\Big) \quad (43)$$

Combining the above equations, a long single equation is obtained to represent $\overrightarrow{X}$. After eliminating its denominator, we obtain

$$\begin{aligned}(\overrightarrow{u} \cdot \overrightarrow{N_1})(\overrightarrow{N_2} \cdot \overrightarrow{v})(\overrightarrow{X} - \overrightarrow{C}) &= d_1(\overrightarrow{N_2} \cdot \overrightarrow{v}) \cdot \overrightarrow{u} \\ &+\Big(d_2\overrightarrow{u} \cdot \overrightarrow{N_1} - \overrightarrow{N_2}\big((\overrightarrow{u} \cdot \overrightarrow{N_1})\overrightarrow{C} + d_1\overrightarrow{u}\big)\Big) \cdot \overrightarrow{v} \\ +\gamma m(\overrightarrow{u} \cdot \overrightarrow{N_1})(\overrightarrow{N_2} \cdot \overrightarrow{v}) \cdot \overrightarrow{v} &+ \gamma(\overrightarrow{u} \cdot \overrightarrow{N_1})(\overrightarrow{N_2} \cdot \overrightarrow{v}) \\ \overrightarrow{N_2}\Big(\sqrt{1 - n^2 + n^2(\overrightarrow{v} \cdot \overrightarrow{N_2})^2} &- n\overrightarrow{v} \cdot \overrightarrow{N_2}\Big)\end{aligned} \quad (44)$$

where

$$\overrightarrow{v} = m\overrightarrow{u} + \overrightarrow{N_1}\Big(\sqrt{1 - m^2 + m^2(\overrightarrow{u} \cdot \overrightarrow{N_1})^2} - m\overrightarrow{u} \cdot \overrightarrow{N_1}\Big) \quad (45)$$

2) Formulating the forward projection of non-parallel surfaces glass: The relationship of the $\overrightarrow{u}$ and $\overrightarrow{X}$ through non-parallel surfaces glass model is provided by the equation 44. The goal of this section is to investigate solutions for deriving $\overrightarrow{u}$ from back projection equation.

The vectors in equation 44 are extended by elements,

$$\overrightarrow{C} = \begin{pmatrix}0\\0\\0\end{pmatrix}, \overrightarrow{N_1} = \begin{pmatrix}0\\0\\1\end{pmatrix}, \overrightarrow{N_2} = \begin{pmatrix}n_1\\n_2\\n_3\end{pmatrix}$$
$$\overrightarrow{u} = \begin{pmatrix}u_1\\u_2\\u_3\end{pmatrix}, \overrightarrow{X} = \begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}$$

With the same principle of extending strategy as we did in planar glass modeling, the camera origin is set to world origin. The normal of the plane which first intersects with the back projected ray is specialized to $[0, 0, 1]'$. The rest vectors are generic. By adding new variables, the format of square root is removed.

Here, $\overrightarrow{u}$ and two normal vectors $\overrightarrow{N_1}$ $\overrightarrow{N_2}$ are unit vectors. Let

$$m^2u_3^2 - m^2 + 1 - a^2 = 0 \tag{46}$$

$$1 - n^2 + n^2(mn_1u_1 + mn_2u_2 + an_3)^2 - b^2 = 0 \tag{47}$$

to make equations polynomial by adding more variables. Then a equation set is derived from (44)-(47),

$$\begin{aligned}&d_1(mn_1u_1 + mn_2u_2 + an_3)u_1 + (-d_1n_1u_1 - d_1n_2u_2\\&- d_1n_3u_3 + d_2u_3)mu_1 + gm^2u_3(mn_1u_1 + mn_2u_2 + an_3)u_1\\&+ gu_3(mn_1u_1 + mn_2u_2 + an_3)(-mnn_1u_1 - mnn_2u_2 -\\&ann_3 + b)n_1 - u_3(mn_1u_1 + mn_2u_2 + an_3)x_1 = 0\end{aligned} \tag{48}$$

$$\begin{aligned}&d_1(mn_1u_1 + mn_2u_2 + an_3)u_2 + (-d_1n_1u_1 - d_1n_2u_2\\&- d_1n_3u_3 + d_2u_3)mu_2 + gm^2u_3(mn_1u_1 + mn_2u_2 + an_3)u_2\\&+ gu_3(mn_1u_1 + mn_2u_2 + an_3)(-mnn_1u_1 - mnn_2u_2\\&- ann_3 + b)n_2 - u_3(mn_1u_1 + mn_2u_2 + an_3)x_2 = 0\end{aligned} \tag{49}$$

$$\begin{aligned}&d_1(mn_1u_1 + mn_2u_2 + an_3)u_3 + (-d_1n_1u_1 - d_1n_2u_2\\&- d_1n_3u_3 + d_2u_3)a + gmu_3(mn_1u_1 + mn_2u_2 + an_3)a\\&+ gu_3(mn_1u_1 + mn_2u_2 + an_3)(-mnn_1u_1 - mnn_2u_2\\&- ann_3 + b)n_3 - u_3(mn_1u_1 + mn_2u_2 + an_3)x_3 = 0\end{aligned} \tag{50}$$

$$1 - n^2 + n^2(mn_1u_1 + mn_2u_2 + an_3)^2 - b^2 = 0 \tag{51}$$

$$m^2u_3^2 - m^2 + 1 - a^2 = 0 \tag{52}$$

$$mn - 1 = 0 \tag{53}$$

$$t_1u_3 - 1 = 0 \tag{54}$$

The useless solutions whose u_3 is equal to zero are removed because of the adding equation 54. Above is the polynomial equation set developed to solve u_1, u_2 and u_3 with degrees greater than four. There is usually no formula for the solution of a polynomial equation whose degree is greater than four. For our case, There are eight solutions from equation 48 to 54 and only one is selected and correct. Therefore, for the forward projection with non-parallel glass model, polynomial equations and the solver are provided.

C. Spherical-shape windshield model

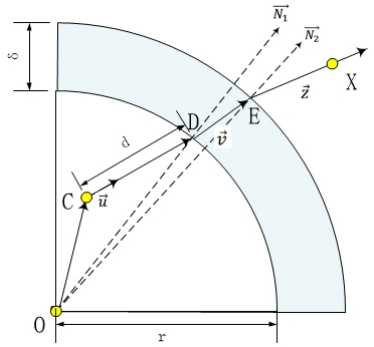


Fig. 3: Model of the spherical glass and projection geometry

In the previous sections, we saw the glass models which surfaces are flat shaped. As we know, the windshield for automotive application is the front window that consists of two curved sheets of glass. Apart from the glass models with flat surfaces, it is necessary to model a curved shaped glass to approximate the real windshield as much as possible.

The model is described by three parameters:

- The thickness of the glass: $\boldsymbol{\delta}$ $(0 < \delta \in \mathbb{R})$
- The refraction index of the material: $\boldsymbol{n}$ $(0 < n \in \mathbb{R})$
- The radius of the sphere: $\boldsymbol{r}$ $(0 < r \in \mathbb{R})$

1) Derivation of Back projection with spherical glass:

$$\begin{aligned}w^2\overrightarrow{X} = \gamma\Bigg(&nw^2\Big(m\overrightarrow{u} + R(d\overrightarrow{u} + \overrightarrow{C})\Big(w - m\overrightarrow{u}\cdot(R(d\overrightarrow{u}\\&+ \overrightarrow{C}))\Big)\Big) + S\Big(wd\overrightarrow{u} + w\overrightarrow{C} + \delta\Big(m\overrightarrow{u} + R(d\overrightarrow{u} + \overrightarrow{C})(w\\&- m\overrightarrow{u}\cdot(R(d\overrightarrow{u} + \overrightarrow{C}))\Big)\Big)\Big(w^2 - \Big(n\big(m\overrightarrow{u} + R(d\overrightarrow{u} + \overrightarrow{C})\\&(w - m\overrightarrow{u}\cdot(R(d\overrightarrow{u} + \overrightarrow{C})))\big)\Big)\Big(S\Big(wd\overrightarrow{u} + w\overrightarrow{C} + \delta\big(m\overrightarrow{u}\\&+ R(d\overrightarrow{u} + \overrightarrow{C})(w - m\overrightarrow{u}\cdot(R(d\overrightarrow{u} + \overrightarrow{C})))\big)\Big)\Big)\Bigg) + w^2d\overrightarrow{u}\\&+ w^2\overrightarrow{C} + \delta w\big(m\overrightarrow{u} + R(d\overrightarrow{u} + \overrightarrow{C})(w - m\overrightarrow{u}\cdot(R(d\overrightarrow{u} + \overrightarrow{C})))\big)\end{aligned} \tag{55}$$

where

$$w = \sqrt{1 - m^2 + m^2(\overrightarrow{u}\cdot(R(d\overrightarrow{u} + \overrightarrow{C})))^2} \tag{56}$$

2) Formulating the forward projection of spherical glass: The vectors are extended to a generic form

$$\overrightarrow{C} = \begin{pmatrix} c1 \\ c2 \\ c3 \end{pmatrix}, \overrightarrow{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}, \overrightarrow{X} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

A polynomial equation set can be obtained by substituting the above vectors into the equation (55).

The polynomial equations are of high degree. The solutions can be investigated by substituting numerical values. However, for the spherical glass model, we just provide the polynomial equation set. There is no formula solution for high degree polynomial equation in this case.

D. Perspective camera model behind windshield

The process of camera calibration with a windshield also requires the properties of optical system. In other words, given a point in space, our model has to provide the pixel location of the corresponding point in the image of the camera. To approach this, the camera model has to be integrated with the glass model. Here, we choose the generalized pinhole camera. With the parameters of the camera, we can further know the location in image where emergent ray projects. The perspective camera model is generally expressed as

$$\widetilde{u} = K[R \;\; t]\widetilde{X} \tag{57}$$

In our case, t is considered to be zero since there is no translation. R is the rotation matrix with respect to the world frame. K is the intrinsic matrix transferring the point between 3D camera frame and 2D image frame. In the previous sections of this chapter, R and K are assumed to be 3×3 identity matrices when deriving the path of projection through glass model. To approach a generic pinhole camera model with windshield, R and K are considered generic.

When deriving the forward projection equations of the planar glass model and non-parallel surfaces glass model, we specialized the normal of the glass as $\overrightarrow{N} = [0, 0, 1]'$ which implies that the normal of the glass is along with the z axis of the coordinate. The glass model is applied for the situation when glass is perpendicular to the z axis.

However, we are aiming to achieve a camera model with glass that is generic for any configuration of the glass. In other words, the camera model is always applicable even the glass is of any angle with the reference plane. Therefore, we define a coordinate system called ***Model System*** which its z axis is of same direction as the normal of the glass. This newly introduced system is dedicated for the glass models which the surfaces are flat shaped since their normal vectors are specialized. For spherical glass model, this system is not necessary. Moreover, $\boldsymbol{R'}$ is the rotation matrix of the *Model system* with respect to the world coordinate. The contribution of this section is figuring out the transformation between world frame, *Model frame*, camera frame and the image frame and deriving the relationship of pixel location in image and the point in space through glass with generalized pinhole camera.

There are four coordinate systems to be considered:

- The world frame **W**
- The camera frame **C**: Rotation matrix R w.r.t **W**
- The *Model frame* **M**: Rotation matrix R' w.r.t **W**
- The image frame **I** Intrinsic matrix K w.r.t **C**

There is a transformation between camera frame and *Model frame*. We say S is a 3×3 matrix which rotates the axes of camera coordinate system **C** onto axes of *Model system* **M**. The z axis of *Model system* is known which is the normal vector of the plane. S can be computed by two vectors of z axes provided by two coordinate systems, R and R'. For camera coordinate, z axis is the last column of the inverse of the rotation matrix R. Firstly, let's see the format of two rotation matrices of two frames.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} R' = \begin{bmatrix} r'_{11} & r'_{12} & r'_{13} \\ r'_{21} & r'_{22} & r'_{23} \\ r'_{31} & r'_{32} & r'_{33} \end{bmatrix}$$

The following is the equation that links the camera coordinate system and *Model system*.

$$R'^{-1} = S \cdot R^{-1} \tag{58}$$

Transformation matrix S can be computed with two vectors extracted from matrices R and R'. As we know, the only known vector to represent the configuration of the rotation matrix R' is its third column which is $\overrightarrow{N} = [0, 0, 1]'$. Therefore, we extract the corresponding vector in R which is also the third column of the matrix, $[r_{13}, r_{23}, r_{33}]'$ to compute S. Let

$$\overrightarrow{a} = \overrightarrow{N} = [0, 0, 0]', \quad \overrightarrow{b} = [r_{13}, r_{23}, r_{33}]' \tag{59}$$

After a and b are normalized into unit vectors, let

$$\overrightarrow{r} = \overrightarrow{a} \times \overrightarrow{b} \tag{60}$$

Then the transformation matrix S can be computed from the existing formula, Then the rotation matrix S can be obtained

$$S = I + [r]_\times + [r]_\times^2 \cdot \frac{1 - \overrightarrow{a} \cdot \overrightarrow{b}}{\|r\|^2} \tag{61}$$

Knowing the transformation between different frames and the camera parameters, we can describe the projection of the camera model with windshield which is to obtain the pixel

location in image given a point in space on the other side of the windshield. The unique corresponding projection is provided.

Given a 3D point $\overrightarrow{X}$ in world frame, it is firstly transferred to the frame centered at the camera origin.

$$\overrightarrow{X'} = \overrightarrow{X} - \overrightarrow{C} \tag{62}$$

Then it is transferred from world frame to the *Model frame*,

$$\overrightarrow{X''} = R' \cdot \overrightarrow{X'} = R \cdot S^{-1} \cdot \overrightarrow{X'} \tag{63}$$

Further, the pixel location in Model coordinate $\overrightarrow{u''}$ can be obtained by inputting the $\overrightarrow{X''}$ into the function of glass model,

$$\overrightarrow{u''} = \mathscr{M}(\overrightarrow{X''}) \tag{64}$$

where $\mathscr{M}$ represents the function of the glass model. In general,

$$\widetilde{u} = \mathscr{M}(d_1, d_2, \overrightarrow{N_1}, \overrightarrow{N_2}, n, \widetilde{X}) \tag{65}$$

The obtained pixel location is transferred into camera frame from *Model frame* by using camera extrinsic parameters,

$$\overrightarrow{u'} = R \cdot R'^{-1} \cdot \overrightarrow{u''} = R \cdot S \cdot R^{-1} \cdot \overrightarrow{u''} \tag{66}$$

By applying the intrinsic parameters of the perspective camera, the projection of 3D point in image is

$$\overrightarrow{u} = K^{-1} \cdot \overrightarrow{u'} \tag{67}$$

In summary, the projection equation through camera model with windshield is

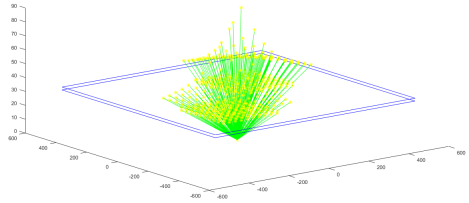
$$\overrightarrow{u} = K^{-1} \cdot \left(R \cdot R'^{-1} \left(\mathscr{M} \left(R' \cdot (\overrightarrow{X} - \overrightarrow{C}) \right) \right) \right) \tag{68}$$

So that the equation 68 links the point in space and its projection through a glass. Given the parameters of the glass model and the parameters of the generalized pinhole camera, the corresponding pixel location can be obtained knowing the location of the object. To be noticed that, the glass model parameters solely explain the refraction introduced by the windshield. With the windshield parameters fixed, our new model has only the parameters of the perspective camera to be adjusted.

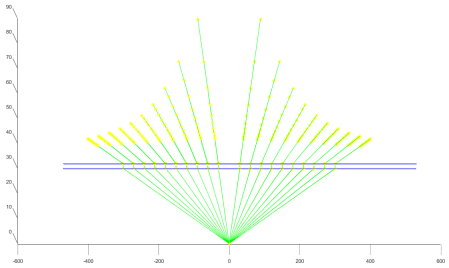
III. Experiment and Evaluation

The projection equations for each case are experimentally verified by substituting numerical values in *Maple* software. The validation of three glass models is performed by checking the consistency of the input and output through the glass models. Given a input pixel location, the output point location in space can be obtained through the back projection functions. Again input the result to the forward projection, the output is as the same as the input pixel location. Moreover, in particular, the generalized pinhole camera model with planar glass is implemented in *Matlab* software and simulated. The projection rays are visualized passing through the planar glass. The back projection and forward projection equations integrated with camera parameters are proved to be valid. The camera model with glass is able to further be experimented through real windshields.

Then, a virtual image of size 100×100 is created. Every pixel location is imported into the back projection function. We can see the performance in the figure 4.



(a) Top view



(b) Side view

Fig. 4: Simulation of perspective camera model with planar glass

As shown in figure 4, two blue planes represent the planar glass. The projection rays are in green color. All the projection rays pass through the glass with two times refraction and intersect at one single point which is the camera origin.

We then extended the refracted rays(upper side of the glass) to the reverse direction(camera side of the glass). The incident rays are parallel to the emergent rays after refraction introduced by the planar glass. However, we noticed that the reverse extensions of the refracted rays are not intersecting at one single point but a circle area(Figure 5). This phenomenon further remarks that the significance of modeling the geometry of how light rays project through a glass with

a certain thickness.

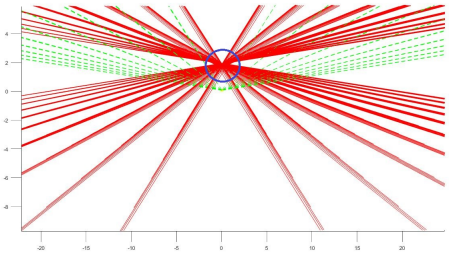


Fig. 5: The intersections of the reverse extension of the refracted rays

IV. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

Finding the precise mapping between points in space and pixels in image is essential for automotive application. However, the windshield has effects on the projection of the object outside the car. We analyzed that the refraction introduced by the windshield matters significantly on the path of light ray and camera image. Hence we narrow down our focus on the refraction effect of windshield and proposed three types of windshield models.

The newly introduced windshield models are planar glass model, non-parallel surfaces glass model and spherical glass model. The relation of correspondences between three dimension and two dimension is found and projection equations of three models are provided. For planar glass model, the polynomial descriptions are developed as well as the formula for projection equation. Moreover, the parameters of the planar windshield model are reduced by the distance from object to glass and the distance from camera to glass. For non-parallel glass model, the polynomial equations are derived and the solutions can be calculated. For spherical glass model, only polynomial description is provided. What's more, a whole process of projection of generalized pinhole camera is developed and illustrated integrated with windshield models.

The proposed three types of windshield models are experimentally verified with numerical values. The functions of back projection and forward projection of planar glass model is implemented. And a simulation is performed with planar glass model.

B. Future Works

Three types of windshield models are proposed in this paper. As we have seen, the planar windshield model is relatively fully developed. The other two models, non-parallel surfaces glass model and spherical glass model are not able to provide functions to be directly used for camera calibration. For spherical glass model, the solution should be derived from the polynomial equations. For non-parallel surfaces glass model, a solver needs to be developed for solving the polynomial equations whose degrees are greater than four. Both of them have one more issue to consider which is eliminating the parameters γ and d, the distance from object and camera to the windshield respectively.

Further, the windshield models will be tested with real set up and data. By experimenting with cameras and windshields, the parameters can be estimated. Bundle adjustment could be involved to optimize in the process of calibration. At the same time, we will find out which model fits the best for this application. Our aim is to provide windshield camera models and highly parameterize them in car industry before selling out. It works to perform a better and more precise camera calibration for automotive application.

REFERENCES

[1] Rezaei, Mahdi, and Reinhard Klette. "Vision-Based Driver-Assistance Systems." Computer Vision for Driver Assistance. Springer, Cham, 2017. 1-18.

[2] Grether, Walter F. Optical factors in aircraft windshield design as related to pilot visual performance. No. AMRL-TR-73-57. AIR FORCE AEROSPACE MEDICAL RESEARCH LAB WRIGHT-PATTERSON AFB OH, 1973.

[3] Horgan, Jonathan, et al. "Vision-based driver assistance systems: Survey, taxonomy and advances." Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on. IEEE, 2015.

[4] Hanel, A., L. Hoegner, and U. Stilla. "TOWARDS THE INFLUENCE OF A CAR WINDSHIELD ON DEPTH CALCULATION WITH A STEREO CAMERA SYSTEM." International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences 41 (2016).

[5] Ernst, Stefan, et al. "Camera calibration for lane and obstacle detection." Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on. IEEE, 1999.

[6] Ribeiro, Anderson Andr Genro Alves, Leandro Lorenzett Dihl, and Cludio Rosito Jung. "Automatic camera calibration for driver assistance systems." Proceeedings of 13th International Conference on Systems, Signals and Image Processing. 2006.

[7] Gennery, Donald B. "A Stereo Vision System for an Autonomous Vehicle." IJCAI. Vol. 2. 1977.

[8] Ramalingam, Srikumar, and Peter Sturm. "A Unifying Model for Camera Calibration." IEEE transactions on pattern analysis and machine intelligence 39.7 (2017): 1309-1319.

[9] Bergamasco, Filippo, et al. "Parameter-free Lens Distortion Calibration of Central Cameras." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[10] Agrawal, Amit, et al. "A theory of multi-layer flat refractive geometry." Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012.

[11] David Franklin Neralla."Layout and implementation of a software based framework to compensate windshild-based distortions of lenses of stereo cameras." TU CHEMNITZ, 2016.

[12] Feng, Mingchi, et al. "Global Calibration of Multi-Cameras Based on Refractive Projection and Ray Tracing." Sensors 17.11 (2017): 2494.

[13] Sturmfels, Bernd. "WHAT IS... a Grobner Basis?." NOTICES-AMERICAN MATHEMATICAL SOCIETY 52.10 (2005): 1199.

Application of state of the art Machine Learning to various Industrial Visual Inspection problems

Nour Islam Mokhtari
nourislam.mokhtari@gmail.com

***Abstract*— In this work, we applied machine learning techniques to common problems in industrial visual inspection. We first addressed a classification problem by developing a pipeline. This pipeline can handle the data on one hand and classify this data on the other hand. For choosing the appropriate classifier, we first started by experimenting with *HOG* features with *SVM*. After analyzing the results, we decided that the next logical step to improve the accuracy would be to use a *bag of visual words*. We chose *ORB* features detector for extracting features that were lately used for building our *dictionary of visual words*. We were able to improve the accuracy by a large margin. We then tried a mixed of deep learning and SVM. The solution we proposed for classification tasks was to use better features that were extracted by passing an image through a pre-trained convolutional neural network. We used *Inception* network to do this. Using these features we trained a *SVM* and we obtained very high accuracies. We also worked on an object detection problem where we developed a machine learning agent that could detect 7 different types of objects (7 classes). In industrial setups, both accuracy and speed are important. Therefore, for this detection task, we chose to work with *SSD* (Single Shot Multibox Detector) with *Inception v2* as a feature extractor. We also made use of *data augmentation* techniques to generate more images from a small set of images by applying different transformations such as zooming and shearing. We obtained some very promising results in both classification and object detection tasks.**

I. INTRODUCTION

In industry, we are often faced with classification and object detection problems. In most of the cases, we don't have enough data to solve our problems. In this work, we will present some of the approaches to tackle classification and object detection tasks while having few data to start with. We will show in the next section, how we solved some common classification problems. We will discuss how we experimented with multiple solutions to develop a classification pipeline. We will also show how we used *data augmentation* techniques to generate new images from a small data set. In the same section, we will show how we solved an object detection task by using deep learning techniques.

II. METHODOLOGY

In this section, we have 2 parts : the first part is for demonstrating how we solved common classification tasks found in industry. The second part is dedicated for an object detection task. For both parts, we will examine real cases found in industry. We will show the difficulties that came with every task and how we overcame them by proposing different and more powerful techniques.

A. Classification

To address common classification tasks in industry, we followed a pipeline scheme. Part of the pipeline is to handle the data, another part is to train a classifier and the last part is for using trained classifiers for prediction. We chose this scheme because it allows us to try different classifiers and test them. To start off, we tried a commonly used technique for classification, namely : SVM with HOG descriptor. We chose this as our first approach to the classification problem because it has been a popular solution to such tasks [1], [2], [9] and [10].

SVM with HOG

We performed many tests by changing the SVM parameters and testing on a dataset of 285 images where there are 142 images with screw, 127 images without a screw and 16 unknown cases. To evaluate the trained models we used *accuracy* as a metric. Accuracy is defined as *the number of correctly classified cases over the number of all the cases*. For SVM classifier, we have 2 main parameters that we can change to adjust the performance.

- C : a parameter that controls overfitting for SVM. A high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors. A low C makes the decision surface smooth.
- γ : defines how far the influence of a single training example reaches, with low values meaning far and high values meaning close.

To choose the best parameters C and γ for our classifier, we used *cross-validation*. In cross-validation we split our data set into 2 subsets, usually into 80% for training and 20% for testing, or 90% for training and 10% for testing. We train the bigger subset with our chosen parameters and then we test the trained classifier on the test set (the smaller set). This approach helps us determine the parameters C and γ that maximize both the training accuracy and the testing accuracy. It also helps us address the over-fitting problem, since we are testing the trained classifier on a subset that was not seen before. Which will subsequently give us a good idea on how well our classifier will perform on new data in the future. The tests are listed in the tables I and II.

In the table I we were changing the SVM parameter C while fixing γ to 1. In the table II we were changing γ while fixing C to 16. Then we do cross validation. We realized that

SVM parameters	Accuracies (for train set and test set)	
$C = 0.1$	train set = 48.62%	test set = 56.52%
$C = 0.5$	train set = 50%	test set = 50%
$C = 1$	train set = 100%	test set = 50%
$C = 2$	train set = 100%	test set = 60%
$C = 4$	train set = 100%	test set = 43.48%
$C = 8$	train set = 100%	test set = 45.65%
$C = 16$	train set = 100%	test set = 50%

TABLE I

DIFFERENT TESTS USING SVM WITH HOG (C IS VARIABLE, γ IS FIXED)

SVM parameters	Accuracies (for train set and test set)	
$\gamma = 0.1$	train set = 100%	test set = 58.69%
$\gamma = 0.5$	train set = 100%	test set = 56.52%
$\gamma = 1$	train set = 100%	test set = 50%
$\gamma = 2$	train set = 100%	test set = 52.17%
$\gamma = 4$	train set = 100%	test set = 41.3%
$\gamma = 8$	train set = %	test set = 32.61%
$\gamma = 16$	train set = 100%	test set = 54.35%

TABLE II

DIFFERENT TESTS USING SVM WITH HOG (γ IS VARIABLE, C IS FIXED)

there is a certain accuracy threshold that we just couldn't surpass which is around 60%. When we see the images that contain screws and the ones that don't contain screws we can understand why the features descriptor is not discriminating enough, since both classes look the same, they both look like they have a small circle surrounded by a bigger circle and *HOG* descriptors rely on the gradient which means that there won't be a big difference between the gradients of a circle and those of a hexagone.

What we started to think of next is, we either need to look for a better features descriptor or we can try to improve the way we are using these features descriptors. Since, we tried one of the best features descriptors that is used in state of the art classification tasks, we thought it's better to try the latter approach, that is, we try to improve on the way we are using features descriptors, the way to do this is by using an algorithm called : *Bag of visual words*.

Bag of visual words with SVM

Introduction

Bag of visual words [3] is a famous unsupervised machine learning technique. In unsupervised learning, our data set is comprised of data points and no labels. The *bag of visual words* technique constructs a *visual dictionary* from features that were extracted from images. The way to use this algorithm for classification tasks is by following these steps:

- We first extract features from images. These could be any good features such as SIFT, HOG, SURF, ...etc. In our case we used ORB features (Oriented FAST and Rotated BRIEF). The reason we used ORB features is because they are very similar to SIFT features and they are free to use unlike SIFT and SURF.
- We use a clustering algorithm to construct a *bag of visual words* or a *dictionary*. In our case we used *k-means* algorithm.
- Then, for every image (in the training set) we construct a *histogram*. This histogram is not for pixel intensities. It actually represents the frequency of occurrence of each entry in the dictionary in an image. This works by computing distances of each features vector extracted from the image to each cluster's centroid (*visual word*) in the dictionary.
- We then train a classifier using the constructed histograms. In our case we chose SVM.

The previous steps could be summarised in diagram 1.

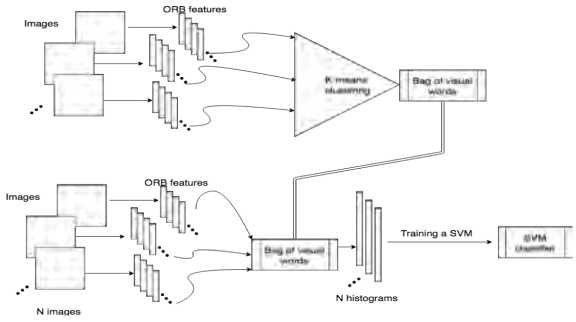


Fig. 1. Bag of visual words with SVM

The implementation of the *bag of visual words* algorithm can be summarized in the following manner :

• Extracting features from an image I_i that belongs to a set of N images (we used ORB features in our case):

$$X_i = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ f_i^1 & f_i^2 & \vdots & f_i^{N_i} \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}_{d \times N_i}$$

N_i : number of descriptors in image I_i.

So for image I_i we have a matrix X_i associated with it which represents a set of column vectors, these vectors are ORB descriptors. d represents the dimension of the ORB features vector (in our case $d = 32$) and N_i represents the number of vectors.

• For all the images we have :

$$X = \begin{pmatrix} X_1 & X_2 & ... & X_N \end{pmatrix}_{d \times M}$$

which can then be written as :

$$X = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ f_1 & f_2 & \vdots & f_M \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}_{d \times M} \quad ; \quad M = \sum_{j=1}^{N} (N_i)$$

Now, to find the dictionary we minimize the following quantity:

$$\min_{D} \sum_{m=1}^{M} \min_{(k=1...K)} \|f_m - d_k\|^2$$

$D = [d_1, \ldots, d_k]$ is the visual dictionary. K : number of clusters centers to be found.

• For every image, we encode the local features by assigning them to their closest word in the dictionary:

$$A = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ a_1 & a_2 & \vdots & a_{N_i} \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}_{K \times N_i}$$

K : number of clusters.

The final representation of image I_i is :

$$\tilde{X}_i = \frac{1}{N_i} \sum_{l=1}^{N_i} a_l$$

So for every image we will have a histogram that represents the frequency of occurrences ($\tilde{X}_i$). We can then train a SVM with $\tilde{X}_i$ from every image.

It makes sense to use bag of words as a next step because we are not just extracting features from the image, but we are actually trying to find among these features, which ones are similar to each other and which ones are not, by constructing the visual dictionary. This additional step made a big difference and we will show some results a little later. In order to create a good visual dictionary we had to *augment* the dataset.

Data augmentation

Data augmentation is a technique that has the purpose of creating new data from an already existing dataset. Many machine learning algorithms require a good amount of data to train a good classifier. To generate more data from existing data, what we do is apply different transformations on the pre-existing data. These transformations could be : rotation, translation, skewing, zooming, shearing, cropping...etc. We can also have a combination of these transformations so we can easily imagine how powerful this technique is. Some examples of images before and after data augmentation are shown in the following figures.

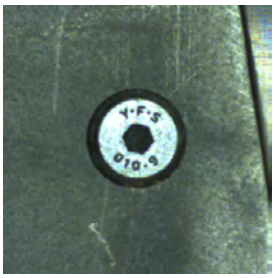


Fig. 2. Example with a screw (before augmentation)

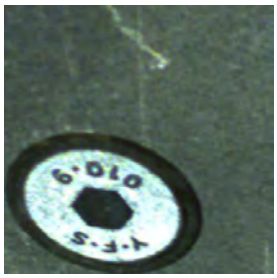


Fig. 3. Example with a screw (after augmentation)

Fig. 4. Example without a screw (before augmentation)

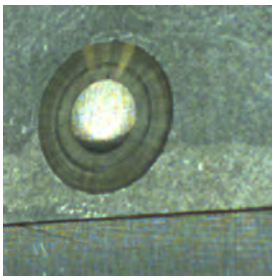


Fig. 5. Example without a screw (after augmentation)

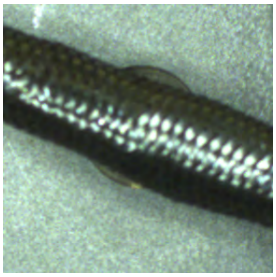


Fig. 6. Example of unknown case (before augmentation)

Fig. 7. Example of unknown case (after augmentation)

We went from a dataset of 381 images to a dataset of 1826 images (672 images that contain a screw, 673 that don't contain screws and 517 images where it's an unknown case). The transformations that we applied were mainly a combination of zooming (to account for scale), skewing (to account for different camera poses) and cropping. Most of these transformations are affine transformations which means we used the following equations :

$$x' = a_x.x + b_x.y + c_x$$
$$y' = a_y.x + b_y.y + c_y$$

Where x and y are coordinates of the pixel from the input image and x' and y' are the coordinates of the output pixel (the transformed pixel). It is also important to note that when doing data augmentation we often end up with some bad images that do not belong to the right class. An example of this would be an image with a screw. If we want to create more images from this image using cropped parts then we might crop an area of the image that doesn't contain a screw, so it is important to *clean* the data set after data augmentation.

Experimentation

For experimentation, we changed some parameters and computed a new prediction model every time and then we computed training and testing accuracies to see how they change with respect to those parameters. The parameters that we can change are :

- For clustering (k-means algorithm):
  - K : the number of clusters. When we run k-means algorithm to cluster our extracted feature vectors

into different groups, we can provide the number of these groups as a parameter.
  - max_iter : the number of iterations that k-means algorithm will use.
  - epsilon : the error tolerance allowed for the algorithm.
- For SVM classifier, we have as previously the parameters C and γ. We also have another parameter :
  - max_iter : maximum number of iterations that SVM will take while training.

We will mainly focus on two parameters because they have the most influence on the results. These parameters are : C and K. We will use the same cross-validation approach to choose the best parameters. We will split our data set into 2 subsets. 80% of the data is for training and 20% is for testing. The tables III, IV and V show some results before and after data augmentation when using *bag of visual words* with *SVM*.

Clustering(k-means)	SVM	Accuracy(80% train / 20% test)
K=75 max_iter=150 epsilon=1	C=1 $\gamma = 1.3$ max_iter=2000 kernel=gaussian	on training set=95.59% on testing set=92.86%
K=90 max_iter=200 epsilon=1	C=1 $\gamma = 1.3$ max_iter=2000 kernel=gaussian	on training set=95.59% on testing set=91.07%
K=50 max_iter=200 epsilon=1	C=1 $\gamma = 1.3$ max_iter=2000 kernel=gaussian	on training set=95.15% on testing set=94.67%

TABLE III
BEFORE DATA AUGMENTATION (WHILE CHANGING K)

We can notice from the table IV that changing the number of clusters affects the accuracy. It's not about choosing a large or small number of clusters. It's about finding the best value for a specific application. For our case, with $K = 50$ we see that we get the best results in terms of both training and testing accuracies. In the table V, we fixed the number of clusters to 50 and changed the parameter C for SVM and we saw that it affected the accuracy as well. As we can see, both the number of clusters and the SVM parameter C can have a major impact on our final model.

CNN features with SVM

Introduction

Using the bag of visual words, we were able to improve the results significantly. But as we can see from the previous results, we are still making false predictions 6% to 10% of the time. This means that in every 1000 images we have 60 to 100 misclassifications which is a lot, especially by industry standards. This led us to think about using something more powerful such as deep learning. In the paper [7], they reported that using pre-trained networks such as

Clustering(k-means)	SVM	Accuracy(80% train / 20% test)
K=10 max_iter=700 epsilon=0.01	C=1 $\gamma = 1$ max_iter=2000	training=83.88% testing=82.97%
K=30 max_iter=700 epsilon=0.01	C=1 $\gamma = 1$ max_iter=2000	training=86.71% testing=90.33%
K=50 max_iter=700 epsilon=0.01	C=1 $\gamma = 1$ max_iter=2000	training=88.56% testing=87.81%
K=60 max_iter=700 epsilon=0.01	C=1 $\gamma = 1$ max_iter=2000	training=86.16% testing=85.43%
K=80 max_iter=700 epsilon=0.01	C=1 $\gamma = 1$ max_iter=2000	training=88.55% testing=86.81%
K=100 max_iter=700 epsilon=0.01	C=1 $\gamma = 1$ max_iter=2000	training=88.41% testing=87.36%

TABLE IV
AFTER DATA AUGMENTATION (WHILE CHANGING K)

Clustering(k-means)	SVM	Accuracy(80% train / 20% test)
K=50 max_iter=700 epsilon=0.01	C=1 $\gamma = 1$ max_iter=2000	training=88.56% testing=87.81%
K=50 max_iter=700 epsilon=0.01	C=2 $\gamma = 1$ max_iter=2000	training=90.34% testing=90.91%
K=50 max_iter=700 epsilon=0.01	C=4 $\gamma = 1$ max_iter=2000	training=91.91% testing=89.83%
K=50 max_iter=700 epsilon=0.01	C=10 $\gamma = 1$ max_iter=2000	training=92.19% testing=92.01%
K=50 max_iter=700 epsilon=0.01	C=20 $\gamma = 1$ max_iter=2000	training=92.67% testing=92.54%
K=50 max_iter=700 epsilon=0.01	C=30 $\gamma = 1$ max_iter=2000	training=94.03% testing=90.93%

TABLE V
AFTER DATA AUGMENTATION (WHILE CHANGING C)

VGG and Inception to extract features from images and then apply some traditional classifier such as SVM, they were able to achieve state of the art results in both classification and object detection. This approach seemed very promising and especially for our case because we don't have a lot of data for training a convolutional neural network from scratch even with data augmentation. The working model of our approach is demonstrated in the figure 8.

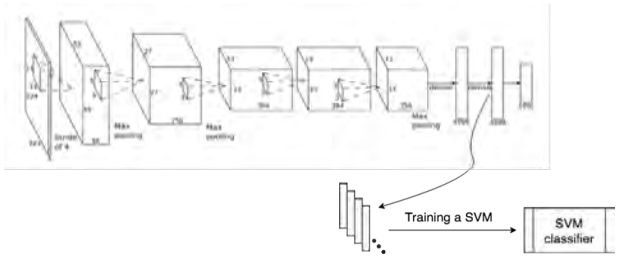


Fig. 8. Our scheme at training time

The figure 8 can be summarized in the following points:

- We pass the images through a pre-trained network such as Inception or VGG.
- We extract the output of the bottleneck layer, also called the transfer layer (referring to transfer learning). This layer is different from one network to another depending on the network architecture. For example it has a size of 4096 in VGG network and 2048 in Inception network.
- We collect features vectors from all the images and use them in training a SVM classifier.

At test time we do as the figure 9 shows. We first need to pass the test image through the network, extract the transfer layer output and then pass it through the trained SVM classifier and get the corresponding prediction.

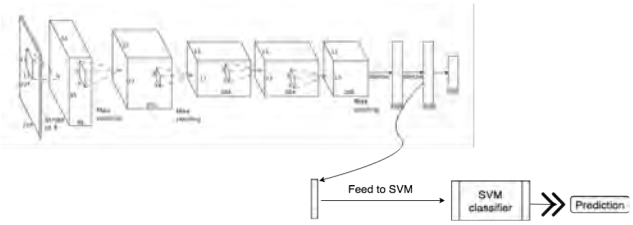


Fig. 9. Our scheme at test time

Experimentation

We used a **linear SVM** with the extracted features and we obtained the results shown in table VI.

SVM	Accuracies
C=1 γ=1	training=100% testing=99.73%
C=5 γ=1	training=100% testing=100%

TABLE VI
INCEPTION V1 AND LINEAR SVM

As we can see from the table VI, the accuracies are very high and the classification is excellent. This is the best classifier that we used for our pipeline and the efficiency of this classifier comes first and foremost from the pre-trained convolutional neural network *Inception v1*. The reason that the extracted features are very discriminative is because the network has been trained on millions of images (ImageNet data set [8]) and it has learned to extract the most relevant parts from the image. Most relevant in the sense of distinguishing between different classes.

B. Object detection

Introduction

In this section we will show how we used deep learning to solve a difficult object detection task. The problem that we tried to solve is the following : We need to detect 7 different types of objects in an image. The objects that we tried to detect are shown in the figure 10.

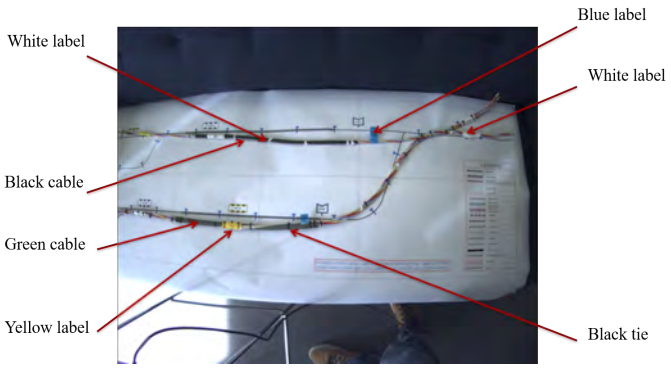


Fig. 10. Different objects that we need to detect

We need to detect 7 different classes, in the figure 10 we have :

- Blue labels : there are 3.
- White labels : there are 4.
- Yellow labels : there are 2.
- White ties : there are 11.
- Black ties : there are 19.
- Black cable : there is only one.
- Green cable : there is only one.

Data

To tackle this object detection problem we first needed to collect data i.e : images of the cable that contains labels and ties and create a data set. We first started by making acquisitions which looked like the previous figure 10. Other examples are shown in figures 11 and 12.

We acquired around 130 images and then we did some data augmentation (same as the previous section) to get a total of 310 images. For the acquisition part, we took different images at different times of the day to have different lighting conditions.

SSD with Inception v2

When we started looking into state of the art object detection using deep learning, we found that there are a lot of possibilities because a tremendous amount of work has been done in the field. In industrial setups there is always a tradeoff between accuracy and speed. In this paper [4], they

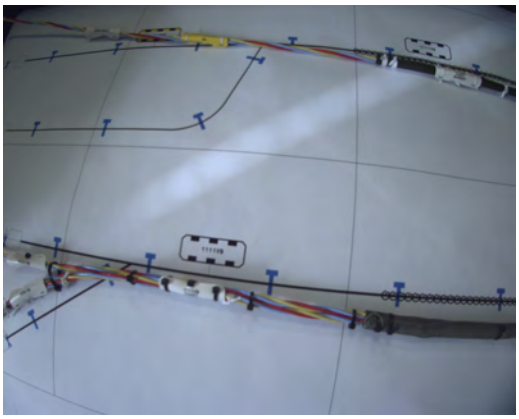


Fig. 11. Acquisition in low light conditions

Fig. 12. Acquisition in good light conditions

discussed many experiments that used deep learning techniques for object detection and they reported the results of their experiments and what are their recommendations with regards to which networks to use and what speed/accuracy tradeoff is expected. They also reported in this paper that among all the fastest models, the most accurate one is the one using SSD (single shot multibox detector) with Inception v2 feature extractor. So we chose this model for our tests.

The SSD [6] approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections. The CNN looks like figure 13.

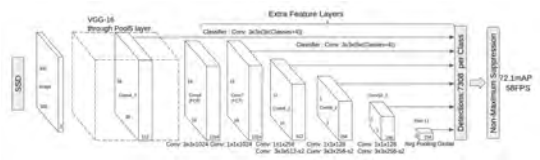


Fig. 13. SSD network with VGG as a features extractor

For training the network we used tensorflow object detection API. Before we trained the network, we had to annotate our images i.e : we needed to give a reference to ground truth bounding boxes. The way we do this is by specifying for every image the coordinates of the rectangles that bound our objects, we also need to specify the class that corresponds to that bounding box. To achieve this, we used an annotation tool which looks like figure 14.

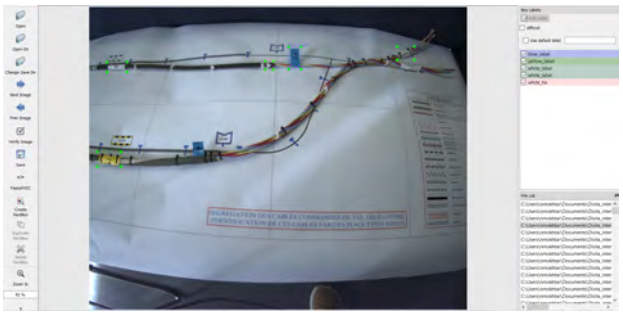


Fig. 14. Annotating images

In the middle of figure 14, we see the image that we are currently annotating. We can notice those green dots around some of the objects in the image, those dots highlight the corners of the bounding box that we consider as ground truth. On the right, we see the names of the classes associated with every bounding box. The output of our annotation process is a *xml* file that contains the coordinates of the bounding boxes and the classes, it looks like figure 15.

```
<size>
    <width>1280</width>
    <height>1024</height>
    <depth>3</depth>
</size>
<segmented>0</segmented>
<object>
    <name>blue_label</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
        <xmin>675</xmin>
        <ymin>134</ymin>
        <xmax>717</xmax>
        <ymax>210</ymax>
    </bndbox>
</object>
```

Fig. 15. XML file containing coordinates and classes

So our data set is comprised of 2 things : Images and files containing the coordinates of the bounding boxes and their corresponding classes. The bounding box coordinates in the *xml* file represent *the top left corner* of the box and *the down right corner*. For training our model we used tensorflow object detection API which takes care of most of the complexities that come with training the SSD network. To train the network we needed to provide parameters that correspond to our specific task. The parameters we used are :

- Training images sizes : 600×600.
- Classification loss function : softmax loss.

$$L_c(x,c) = -\sum_{i\geq 0}^{N} x_{ij}^p \log(\tilde{c}_i^p) - \sum_{i\geq 0}^{N} \log(\tilde{c}_i^0)$$

$$where \qquad \tilde{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)}$$

- Localization loss function : $smooth_{L1}$ based loss.

$$|d|_{smooth_{L1}} = \begin{cases} 0.5d^2 & if \quad |d| \leq 1 \\ |d| - 0.5 & otherwise \end{cases}$$

Then we use it as such:

$$L_{loc}(x, l, g) \sum_{i \geq 0}^{N} \sum_{m \in \{cx, cy, w, y\}} x_{ij}^{k}.smooth_{L1}(l_i^m - \tilde{g}_j^m)$$

$$\tilde{g}_j^{cx} = \frac{(g_j^{cx} - d_i^{cx})}{d_i^w} \quad ; \quad \tilde{g}_j^{cy} = \frac{(g_j^{cy} - d_i^{cy})}{d_i^h}$$

$$\tilde{g}_j^{w} = \log \frac{g_j^w}{d_j^w} \quad ; \quad \tilde{g}_j^{h} = \log \frac{g_j^h}{d_j^h}$$

And the total loss :

$$L(x, c, l, g) = \frac{1}{N}(L_c(x, c) + \alpha L_{loc}(x, l, g))$$

$x_{ij}^p = \{0, 1\}$: an indicator for matching the i-th default box to the j-th ground truth box of category p.
g : the ground truth box.
l : the predicted box.
d : the default box.
cx, cy : the coordinates of the center of the box.
w, h : respectively the width and the height of the box.
N : the number of matched default boxes. If $N = 0$ we set the loss to 0.

- For non-maximum suppression we used : iou = 0.6.
- For optimization we used RMSprop (a derivative of gradient descent).
- Batch size is : 2.
- Training images : 90.
- Testing images : 10.

The training process took around *6 hours* on a machine that contains *NVIDIA Quadro M620 2Gb GPU*. We obtained some very promising results with only 90 images for training. The reason why we didn't use more images is that the annotation process takes a long time, we spent an average of $1h$ for every 7 images. So based on this, we decided to work in an incremental way : We train the model using 100 images (90 for training and 10 for testing) and based on the results, we decide if we need to annotate more images and then run the training again. The plot in figure **??** shows how the total loss was decreasing during the training:

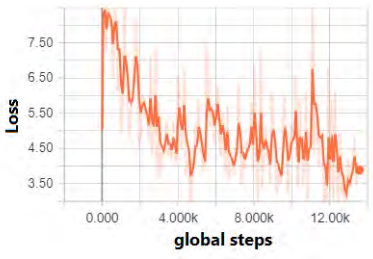


Fig. 16. Total loss

The figure 16 shows the progress of the total loss with respect to global steps (one global step is equivalent to finishing one batch). We notice that it's decreasing over time which shows that our network is learning to localize and classify better our data. We obtained some very good results from our first test so we decided to move forward with the trained model. The figures 17 and 18 show some of our detection results.

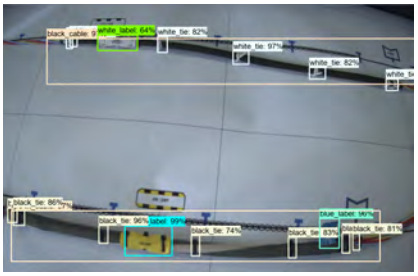


Fig. 17. Detecting objects in low lighting conditions

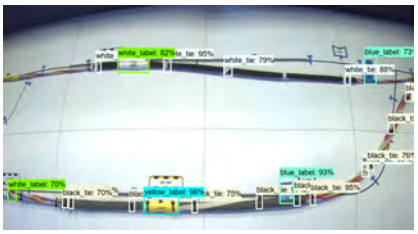


Fig. 18. Detecting objects in good lighting conditions

As we can see, we have some very good results but there is still room for improvements. To quantify the detection results we used a metric called : *mean average precision* (*mAP*). To compute *mAP*, we first need to compute the *average precision (AP)* for each class.

The values of *AP* for each one of the 7 classes are:

- White labels : 0.64
- Yellow labels : 0.83
- Blue labels : 0.78
- White tie : 0.26
- Black tie : 0.36
- Black cable : 0.97
- Green cable : 0.79

Which yeilds ***mAP=0.66***

To give an intuitive sense of how good this result is, we can compare with object detection tasks done on famous data sets such as COCO [5]. In the paper [4] they showed the best results their trained models achieved on the COCO data set have $mAP \leq 0.4$. In our case $mAP = 0.66$. Of course, this comparison is not fair, because for our case, we only trained the model on 90 images while *COCO* data set has around $330K$ images and also, we are only detecting 7 different types of objects while they are detecting thousands. Still, this gives us an idea about how well our object detection is. We also notice that there are classes with low AP such as *white tie* and *black tie*. This is because these objects are

the hardest to detect. If we look at figure 10 for example, we can see that these objects are very small. Moreover, in many instances, they are very close to each other. This leads the model to detect only one object where there are actually two.

III. CONCLUSION

In this work, we presented some machine learning techniques to tackle industrial visual inspection problems. We mainly worked on 2 problems : classification and object detection. For classification, we developed a pipeline. Part of this pipeline includes a classifier to distinguish between different classes. We experimented with some techniques. We first started by using *SVM with HOG descriptor*. This method was not very successful. In all the tests we conducted, the accuracies were low. We then experimented with another technique called *bag of visual words*. We used this technique with SVM and we noticed some significant improvements. Although this method was successful, it was still making false predictions 6% to 10% of the time. To push the accuracy higher, we used a mixed of deep learning and SVM. We used a CNN called *Inception v1* as a feature extractor. We then used the extracted features to train a SVM. This approach gave impressive results that are very close to perfect. Another problem we worked on is : object detection in an industrial setup. We used *SSD* with *Inception v2* as a feature extractor to detect 7 different types of objects in an image and we obtained some very good results. This work has shown how deep learning can dramatically improve the performance of industrial visual inspection systems.

REFERENCES

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[2] O. Deniz, G. Bueno, J. Salido, and F. De la Torre. Face recognition using histograms of oriented gradients. *Pattern Recognition Letters*, Sept. 2011.

[3] AG Faheema and Subrata Rakshit. Feature selection using bag-of-visual-words representation. *International Advance Computing Conference (IACC), IEEE*, 2010.

[4] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016.

[5] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[7] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[9] J. Xiao, J. Hays, and K. Ehinger. Sun database: Largescale scene recognition from abbey to zoo. *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[10] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. *Computer Vision and Pattern Recognition (CVPR)*, 2011.

Real-time perception for motion planning of Jaco2 Kinova with Multiple Kinects

Avinash Narayana

***Abstract*— In future industries and factories we will have robotic arms working autonomously,by using visual feedback to recognize or to localize. My thesis analyzes the capability of Microsoft kinect V2 sensor to act as real time perception for robotic arm path planning. Kinect V2 sensor contains a rich array of sensors for grabbing 3D scene information.we will present our work in situations in which we have multiple kinect sensors system ,multiple kinect sensors may serve as low cost and affordable means to track ArUco markers postion information across a large space applications.We focused on the calibration of the system,Interference and the network requirements to link and synchronize the multiple data-streams together to a common reference point.**

I. INTRODUCTION

In this work, we mainly focus on the Jaco2 kinova perception with multiple kinects. From inside the industrial revolution to now robotics evolution has grown rapidly . In the last few years we are seeing the evolution of robots in different applications and many fields of the human life. Human-Robot collaboration, which is unpredicted until few years ago, now a days trending topic of research field in many areas such as industrial, medical and assistive and service robotics. An explanatory evidence is the industrial field. During 70's and 80's robots were mainly used for difficult situations where humans take more time and repetitive jobs but they were used just to work in safety surroundings, avoiding the cooperation between human operators and robots for safety issues. During the years robots have gained more and more capabilities and features, due to several technologies and the falling of electrical and sensor goods prices, the increase in development and the spread of the open-source resources. All this resources allowed the spread of a new paradigm for the manufacturing industry based on the cooperation between human and robots [1]. In this perspective acquires a major importance the problem of the safety of the operations [2], [3]. Robotic manipulators are often required to perform tasks in the operational space, such as to move the end-effector at a certain position and/or orientation. However, a number of additional tasks have to be taken into account.All these robots are working blind with hard coded trajectories.

The use of vision systems is the most common and intelligent approach for this problem.In industries and factories of the future, most of the tasks will be done autonomously by the robotic arms that need the visual source to move around and to complete the required task in work space by avoiding obstacles [5] and to work collaboratively with humans to identify and localize the working parts and objects, to complete the information provided by other sensors to improve their precise positioning accuracy, etc. Depending on the objective, the vision system can be scene-related or object-related. In scene-related tasks the camera is usually mounted on a mobile robot[6] and applied for mapping, localization and obstacle detection. In object-related tasks, the camera is usually attached to the end-effector of the robot manipulator (eye-in-hand configuration)[4] or fixed somewhere in the workspace, so that new images can be acquired by changing the point of view of the camera. Vision and perception, mobility, grasping, ease of use will ensure that robotics will meet the promise of easing the burden of labor in terms of physical activity or decision making.

Vision systems can also be used to determine the location of the object that needs to be grabbed with the exact location of the object does not need to be programmed manually. This gives less constraints to surroundings, as it is then not necessary to design a visual system to put a product precisely in a spot so that the robot can grab it. By using vision systems the object can be in any different location and in any different orientations. This makes the system a lot more flexible also more accurate. However for both of these objectives the location of the camera needs to be known to get a precise location of the scene or objects. Therefore it is necessary to calibrate the location of the camera. Manual calibration is a tedious procedure. So it should also be automatic so that this part can also increase the flexibility of the robot system.In this paper you will see an approach for automatic calibration and you will also see the vision technique for using multiple kinect with multiple sensor system. The out comes of these works are used in may applications such as obstacle detection.

The paper is organised as follows:

- Section II Describes the method of sensor calibration with the available ROS packages with Linux operating system and Accuracy analysis.
- Section III Describes the Multiple kinect sensor systems and problems of interference with multiple kinects.
- section IV Describes Muliple ArUco markers detection and the common reference frame Method when we use multiple kinects.

*This work was done at University of Cassino,Lai Robotics Lab.

[1]Avinash Narayana research masters student in Computer vision and robotics from Universtiy of Bourgogne, Le Creusot,France. avi.avinash015@gmail.com

[2]This thesis work is supervised under Dr. Gianluca Antonelli Associate Professor in University of Cassino, Italy. antonelli@unicas.it

II. Acquisition and Calibration

A. Kinect V2

In our research we use low cost kinect V2 sensor [7], kinect V2 is a RGB-D sensor used for acquisition designed by Microsoft. In addition to the color camera kinect v2 is also composed of depth camera(including infrared camera(IR) and IR projector) which allows to find out both color and spatial information about filmed scene. Kinect can be connected to computer using USB 3 connection and used as input for 3D modeling(Reconstruction) task. It is also used to interact with out physical contact (contact free) but just by using hand gestures and voice commands.

B. Principal of Kinect V2

1) Time of Flight: The main components of time of flight[8] are namely lens, integrated light source, sensor and a signal processor. This kind of system is able to capture both depth an intensity information simultaneously for every pixel in the image. These cameras depends on pulsed light sources have high potential in machine vision applications. The working of ToF[16] is as follows, a continuous emitting laser source sends out a pulse, and a sensor detects that pulses reflection from the target object(scene) to record its time of flight. Knowing that and the constant speed of light, the system can determine how far away the target object(scene) is placed. Illustration is shown in the figure 1.

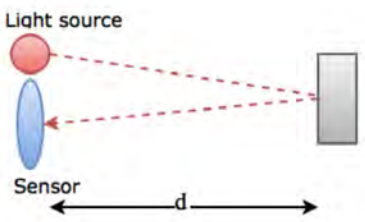


Fig. 1: Illustration of Time of flight.

C. Setup and calibration

Calibration means making sure that our kinect V2 is giving high level of accuracy and reliability. The system needs a kinect v2 sensor, tripod and a plane surface to paste the checker board pattern.The setup is shown in the figure 2

D. Calibration Checkerboard

The checkerboard contains 7 by 9 squares, each one of 0.025m by 0.025m size, printed on a A4 sheet and glued it to a plane object in our case it is white board, or for a good method you can take another tripod for the checker board. The intrinsic and extrinsic calibration procedure works by using a calibration pattern. One of the side squares of the pattern are modified to be hallow [20] is shown in the figure 3, is used to detect the correct orientation.

Fig. 2: Setup of kinect V2 during calibration.

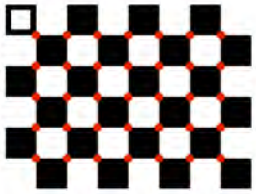


Fig. 3: Hallow square in topleft for orientation detection.

E. Software

In the course of work, we have used open-source kinect v2 driver by Blake et al and its associated with ROS package contributed by Thiemo Wiedemeyer [9], [19] which uses OpenCV calibration [18]. This package is built specifically for the kinect V2 that we will be using in the project, kinect V2 integrating with ROS [17] and other binaries has been initiated for ubuntu 16.04 LTS.

The main feature and advantage of using ROS is its modular design allowing the algorithm to be shared into separate smaller nodes performing individual tasks and sharing the results over the network. Since the Kinect v2 camera need a separate USB3 controller for connecting to PC, here we are using two PC's connected in ROS network.

F. Calibration Method

The proposed automatic calibration method consists of different modules working together to get a good calibration accuracy. In our case calibration for RGB-D cameras are calibrated for intrinsic and extrinsic parameters in order to compensate for the following systematic errors:

1) Color camera lens distortion.
2) Infrared (IR) camera lens distortion.
3) Reprojection error, or color to depth image offset.
4) Depth distortion.

The presented calibration process was successfully performed provided that the checkerboard was detected by

the 3D camera to be calibrated. The calibration pattern is overlaid on the detected checkerboard is shown in the below figure 4.

Fig. 4: Calibration pattern overlay-ed on detected checker board pattern.

1) Method: The pattern has to match exactly the dimensions that we set. Before calibrating any sensor(RGB,IR,Sync) firstly we have to create directory for respective sensor and save images in the respective sensor folder while you start recording the images. The kinect fixed with the tripod is moved and can be locked easily while you take an image. It is very important to stabilize the kinect before taking image to avoid blur in the image.

Placing of the pattern is very important, in which less images, or more number of images of a similar view to the pattern you can get a good re-projection error, but a bad calibration. I took around 0 - 200 images by following these steps:

1) Place the position of the camera in the top left corner of the image.
2) Move the camera along that row and take the images.
3) After the first row go to the next row and repeat step 2 until the last corner of the image.
4) place the pattern further away from the camera and go to step 2. Do this for different distances, near ($<=$ 1m), middle range ($>$ 1m, $<$ 2.5m) and far ($>=$ 2.5m). (The possible distances depend on the chess board size.)
5) When recording images for all the sensors indicated above (RGB, IR, SYNC), start the recording node, then press space bar to record each image. The calibration pattern should be detected indicated by color lines overlay-ed on the calibration pattern, and the image should be clear and stable as shown in figure 4
6) once you calibrated the respective sensor(RGB,IR,SYNC) you will find the re-projection error on the terminal. Higher the re-projection error lower the calibration accuracy. Make sure that you have less than 1cm re-projection error for good calibration accuracy.

For every Kinect there will be a serial number by looking at the first lines printed out by the kinect2_bridge node. We can find the serial number of respective kinect the line looks like this: **device serial: 012526541941**

Create the calibration results directory in kinect2_bridge/data/serial: The directory should be in the name of the serial number. we should copy all the calibrated files from each sensor directory and place it in the kinect serial number directory.

Restart the kinect2_bridge and run the image_viewer for better view of the scene, the kinect view before calibration and after calibration can be seen in the figure 5.

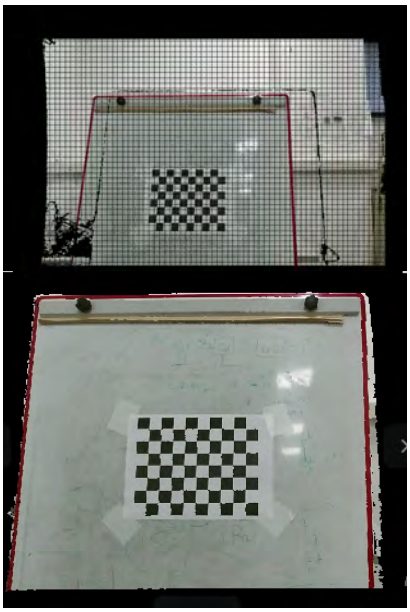


Fig. 5: Before and after calibration.

G. Calibration Error

Calibration is not completely robust operation, i.e that the calibration performed above we will have an error between estimated and actual values this is called as calibration error or re-projection error. An estimate of any calibration is very important since we may get good accuracy and precision of our applications.

H. Experiments and Analysis of Calibration Accuracy

We have undergone some investigation to find out minimum number of images for a good calibration. For this we did few experiments by taking different number of images in each sensor and calibrated with less no of images and took those parameters for accuracy testing therefore the change in re-projection does not show much difference between more number of images or less number of images can be seen in the plots figure

TABLE I: Experiments with Color Frames

Experiment	color_frames	reprojection_error	time in milli seconds
Exp1	5	0.134	454.7
Exp2	10	0.140	1393.81
Exp3	15	0.138	3928.83
Exp4	20	0.1404	9655.78
Exp5	25	0.143	16300.2
Exp6	30	0.145	27552.9

TABLE II: Experiments with IR frames

Experiment	ir_frames	reprojection_error	time in milli seconds
Exp1	5	0.104	429.781
Exp2	10	0.101	1224.79
Exp3	15	0.1028	3393.64
Exp4	20	0.103	10301.8
Exp5	25	0.1029	14823.1
Exp6	30	0.1016	26808.3

TABLE III: Experiments with Synchronized frames

Experiment	sync_frames	reprojection_error	time in milli seconds
Exp1	5	0.120	270.794
Exp2	10	0.130	270.794
Exp3	15	0.1333	614.283
Exp4	20	0.1333	1006.24
Exp5	25	0.1435	1755.25
Exp6	30	0.1420	2692.47

I. Accuracy Plots with respect to number of images and time in Milliseconds

The tests has been plotted between increasing number of images from 1 to 30 with respective to time and re-projection error is shown in figure 6 and 7

III. Multiple Kinect V2 Sensor System

Multiple kinects systems is main To increase the field of view, quality of measurements, and improving the system for high level applications. In addition, it also solves occlusion problems. Object occlusion is the big research field in the computer vision field. Most of the scenes in which an object is cover fully or partially of another object. By using another camera or number of cameras in the workspace, it is possible to capture the blind spots that other camera cant see. The

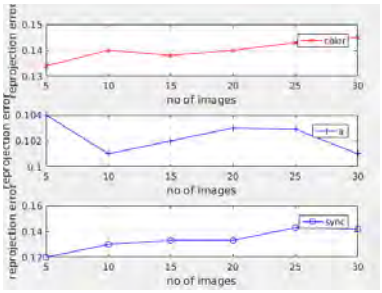


Fig. 6: Plot between number of images and re-projection error.

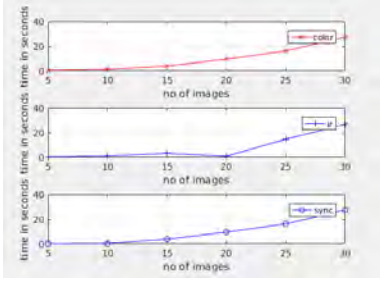


Fig. 7: Plot between number of images and time.

range of vision of the system is the limitation of kinect that multi-sensor system can solve this problem. The normal range of the Kinect v2 is from 0.7m to 3.2m, if this coverage have to be increased its accuracy have to be increased, we can place a second sensor and combine both streams, obtaining a single point cloud with larger coverage area.

The main objective is to achieve a multi-sensor system[10], which should be robust and resistant to changes in the scene conditions such as light, irregular shapes, occlusions and so on. Adaptability is also a desired characteristic, since the location of the sensors de-pends on the scene, thus an optimum coverage can be achieved. They evaluate different angular settings for the multiple sensors but interestingly conclude that the orientation significantly improve the resolution quality. Multiple kinects are evenly placed in a virtual circle around the scene.

A. Interference

In kinect v2 point of view interference comes when there are multiple cameras[12], cameras can see each other when we place one in front of the other but laser cannot see they collide each other. The problem here is more a matter of sensor setup. In my research we found that for two Kinects v2 is [0, 45) and (45, 60] degrees. At that range you can have the minimal interference.

B. Investigation of Kinect v2 Interference

To characterize the interference between multiple kinect v2, we have placed both the kinects facing each other.The infrared pattern emitting from both the kinects can interfere and hit each other,causing holes noise or artifacts which is loss of information from the scene to acquired point clouds. Figure 8 shows the image with and with out interference shows that a lot of information is lost when interference occur. Finally with the test of the kinect interference data revealed that the angle between kinect sensors has significant influences [11] on the kinect data resolution. A good placement of cameras

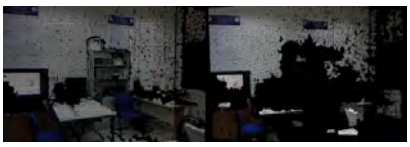


Fig. 8: Image with and without interference.

has to take into account to reduce camera interference's, one way to avoid the interference is not making the kinects field of view touching each other. so we can connect as shown in the figure 9 and figure 10.

IV. Common Reference Frame

The main goal is however, is to use the output from multiple cameras. By combining the information from multiple cameras to scan the surroundings of the robotic arm for obstacles in a safe manner and to prevent occlusions for robotic arm Jaco2 kinova. To be able to detect an object from different directions and to reconstruct a model from point cloud we need multiple kinects.

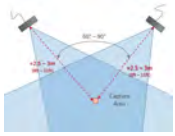


Fig. 9: setup-1.

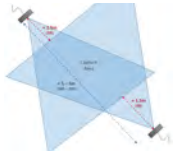


Fig. 10: setup-2.

In order to have common point(world) of view or coordinate system for multiple kinects we have done some research and found a solution with multiple marker cube. Cube with 6 faces, each face is sticked with the ArUco marker generated from ArUco marker generator.

A. *Pipe Line of The method*

One of the most popular approach is the use of binary square fiducial markers[13] as shown in the figure below. To determine the position of the known or unknown environment or to localize something we use fiducial markers. An ArUco marker is a synthetic square marker[15] composed by a wide black border and a inner binary matrix which determines its identifier (id). The main benefit of these markers is that a single marker provides enough correspondences (its four corners) to obtain the camera pose. The black border make it easy for its fast detection in the image and the binary codification allows its identification and the application of error detection and correction techniques. The marker size determines the size of the internal matrix. For instance a marker size of 4x4 is composed by 16 bits. We have generated five markers from Aruco marker generator where we can choose specific marker id, marker size(edge size of the black box containing the marker) and marker padding(thickness of the white boarder surrounding the marker). We generated 5 markers of id: 250,300,350,400,450, glued each marker to each face of the cube which we prepared can be seen in the below figure 12.

A reference frame provides a relationship in pose between one coordinate system and another. Every reference frame can relate back to a universal coordinate system. All positions and orientations are referenced with respect to the universal coordinate system or with respect to another Cartesian coordinate system.

Initially we set the position of the marker cube in the work space as seen in the figure 12. The pose of the marker cube is coded manually with respective to the robotic arm as world frame(reference frame)[14]. Now when the kinect v2 seeing the work space and when the markers cube is in the field of view of kinect, since the cube is glued with markers, kinect detects multiple markers as shown in the 12, so we cant get the transformation of the kinect since it sees multiple markers at a time, the pipeline of algorithm shows in figure 11.

For this we have calculated the distance from each marker. When ever the kinect dectects multiple markers it calculates the distance from each marker to kinect and marker with the minimum distance is used for the transformation. The goal of this method is to obtain a transformation matrix able to unify successfully the measurements of two Kinect cameras under a common coordinate system. A common reference frame usually placed on the work space called world arm frame. This is usually done when we are working with multiple frames(stereo).

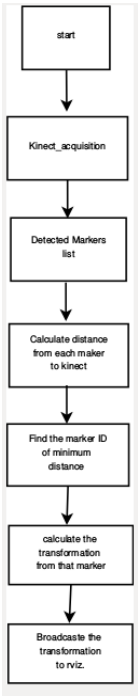


Fig. 11: Pipe Line for common reference frame.

Fig. 12: Transformations.

V. Results

To test the implemented work we connected multiple kinects in a ROS network with the help of multiple PC's the node structure of camera network is shown in figure 13.

Figure 14 shows the tf topic visualized in RVIZ. Figure **??** shows the point clouds from kinect2_1 and kinect2_2. Figure 17 shows the aligned point clouds with respect to common reference point.

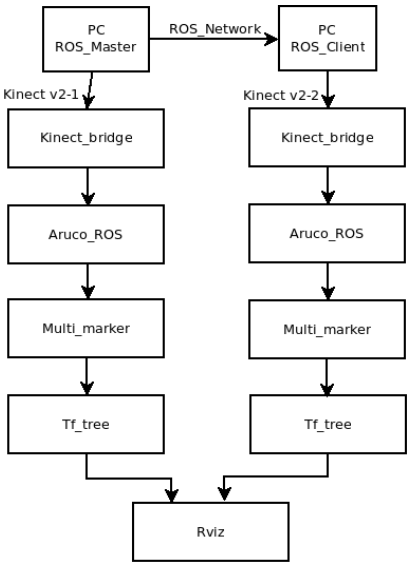


Fig. 13: Node structure with respect to camera network.

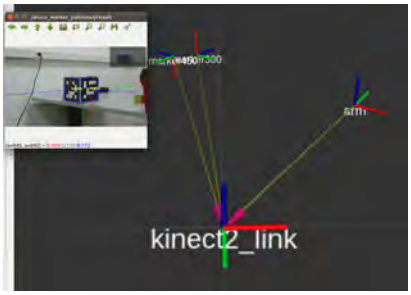


Fig. 14: All the transformations with respect to common reference frame..

Fig. 15: Point cloud from Kinect2_1.

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this thesis a vision technique for robotic arm path planning have been implemented. The results we see are, it is possible to overcome occlusion problems, increase the area, the field of view of coverage and improve the quality of the data.

Fig. 16: Point cloud from Kinect2_2.

Fig. 17: Aligned point clouds with respect to common reference frame.

Tests were made with the of setup of multiple kinect V2 sensors in which object is illuminated from different directions which shows some blind spots that other kinect cannot see. Multiple sensor setup method shows the potential of high resolution, compared by using one kinect. In addition to this, industrial environments conditions also need to be considered. Each application and each type of robotic arm need a specific vision solution. There is no universal vision technique to perform several tasks. .

B. Future Works

Future woks may focus on multi-tasking or multi-purpose vision systems and their integration with other sensor types and systems.

VII. ACKNOWLEDGMENTS

This project was developed in LAI robotics lab University of Cassino.

REFERENCES

[1] Y. Lu, Industry 4.0: a survey on technologies, applications and open research issues, Journal of Industrial Information Integration, vol. 6, pp. 110, 2017.

[2] G. Avanzini, N. M. Ceriani., A. M. Zanchettin, P. Rocco, and L. Bascetta, Safety control of industrial robots based on a distributed distance sensor, IEEE Transactions on Control Systems Technology, vol. 22, no. 6, pp. 21272140, 2014.

[3] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, Safety in human-robot collaborative manufacturing environments: Metrics and control, IEEE Transactions on Automation Science and Engineering, vol. 13, no. 2, pp. 882893, 2016.

[4] Justinas Miseikis and Kyrre Glette and Ole Jakob Elle and Jim Torresen, Automatic Calibration of a Robot Manipulator and Multi 3D Camera System.

[5] Fabrizio FlaccoTorsten and KroegerAlessandro and De Luca and Oussama Khatib, A Depth Space Approach for Evaluating Distance to Objects with Application to Human-Robot Collision Avoidance. October, 2014.

[6] G Alenya and S Foix and C Torras, Using ToF and RGBD cameras for 3D robot perception and manipulation in human environments.

[7] P. Fankhauser, M. Bloesch, D. Rodriguez, , R. Kaestner, M. Hutter, and R. Siegwart, Kinect v2 for mobile robot navigation: Evaluation and modeling, in IEEE International Conference on Advanced Robotics (ICAR) (submitted), 2015.
[8] Tim Beyl and Philip NicolaiMirko and D. ComparettiJrg RaczkowskyElena and De MomiHeinz Wrn, Time-of-flight-assisted Kinect camera-based people detection for intuitive human robot cooperation in the surgical operating room.submitted 2015,November.
[9] T. Wiedemeyer, IAI Kinect2: Tools for using the Kinect One (Kinect v2) in ROS, January, 2015."Efficient range estimation and material quantification from multispectral Lidar waveforms." In Sensor Signal Processing for Defence (SSPD), 2016, pp. 1-5. IEEE, 2016.
[10] Kai Berger, A State Of the Art Report on Research in Multiple RGB-D sensor Setups, October, 2013.
[11] Nima RAFIBAKHSH and Jie GONG and K.Mohsin and SIDDIQUI and Chris GORDON and H. Felix LEE. "Analysis of XBOX Kinect Sensor Data for Use on Construction Sites: Depth Accuracy and Sensor Interference Assessment",October.
[12] T. Mallick and P. P. Das and A. K. Majumdar. "Study of Interference Noise in multi-Kinect set-up.", January, 2014.
[13] "S. Garrido-Jurado and R. Muoz-Salinas and F. J. Madrid-Cuevas and M. J. Marn-Jimnez", "Automatic generation and detection of highly reliable fiducial markers under occlusion", October,2014.
[14] M.Orsag and C.Korpela and S.Bogdan,"Aerial Manipulation", 2018. http://www.springer.com/978-3-319-61020-7.
[15] Andrej Babinec and Ladislav Juriica and Peter Hubinsk and Frantiek Ducho,"Visual localization of mobile robot using artificial markers."march, 2000.
[16] Sergi Foix and Guillem Aleny and Carme Torras,"Lock-in Time-of-Flight (ToF) Cameras: A Survey." 2011.
[17] title="ROS,http://www.ros.org/,
[18] "OpenCV Calibration method",https://docs.opencv.org,
[19] J.Blake and F.Echtler and C. Kerl, Open Source Drivers for the Kinect for Linux v2 Device. Available,published=https://github.com/OpenKinect/libfreenect2,
[20] Justinas Miseikis and Kyrre Glette and Ole Jakob Elle and Jim Torresen,"Automatic Calibration of a Robot Manipulator and Multi 3D Camera System."

DBT Density: Comparison of Local Breast Density Maps from Mammography and Breast Tomosynthesis

Anissa Lintang Ramadhani[a], Robert Martí[b], Oliver Diaz[b]

[a]Computer Vision and Robotics, University of Burgundy, France, [b]Computer Vision and Robotics, University of Girona, Spain

Abstract

Objectives: **The aim of this paper is to compare the spatial glandular tissue distribution of DBT against FFDM as well as the density grade estimation provided by commercial software namely Volpara and Quantra 3D, and BI-RADS density grade from Radiologist.**

Materials and Methods: **The dataset composed of 241 pairs of FFDM and DBT from 61 different women (aged 38-81 years) with moderate/high of risk of developing breast cancer. All participants underwent a two-view (craniocaudal and mediolateral oblique) FFDM and DBT of both breasts using Hologic Selenia Dimensions which took the projections in a single breast compression.**

Results: **Global assessment obtained from Volpara show high correlation between FFDM and DBT data. Pearson's correlation coefficients of 1, 0.92 and 0.93 were obtained for breast volume, glandular tissue volume, and volumetric breast density, respectively. Breast density assessment obtained substantial agreement between VDG estimation and BI-RADS density grade (κ = 0:61) and moderate agreement between QDC estimation and BI-RADS density grade (κ = 0:55), and substantial agreement between VDG estimation and QDC estimation (κ = 0:68). Local assessment regarding glandular tissue distribution also show high similarity between FFDM and DBT density maps computed by various similarity metrics.**

Conclusions: **There is a high similarity of density maps provided by Volpara between FFDM and DBT acquisition. Although divergences arise due to different breast size of both acquisition technique.**

Keywords: Breast cancer, Breast Density, Image analysis, Mammography, Tomosynthesis, Quantra, Volpara.

I. INTRODUCTION

Breast cancer is one of the leading causes of death among the women worldwide. It has become the second leading cause of death from cancer in women [1]. At the same time, it is also among the most curable cancer types if diagnosed in an early stage [2]. In order to detect breast cancer in an early stage, and increasing the survival rates, breast screening programmes are performed in many countries.

The gold standard technique used in breast cancer screening procedures is full field digital mammography (FFDM) [3]. However, in FFDM, breast tissue 3D structure is projected into a 2D image resulting in tissue superposition which may lead to reduce lesion's visibility and loss the third dimension information where it can be overcome by digital breast tomosynthesis (DBT). DBT is basically a modified FFDM technique, which acquires projection images within a limited angular range [4], producing a set of reconstructed images that can be viewed individually or sequentially in a cine loop and provide 3D information for the reader, overcoming many of the interpretation difficulties associated with FFDM [5], [4]. These reconstructed images are projected at different breast's depths, for example, if a 60 cm compressed breast is reconstructed at 1 mm slice thickness, there will be 60 slices to review.

Volumetric breast density (VBD) has consistently been one of the strongest risk factors for breast cancer [7] and represents the amount of glandular tissues in a breast. Automated breast assessment software represents promising method of evaluating VBD such as Volpara (Volpara Solutions; Wellington, New Zealand)[1] and Quantra (Hologic; Danbury, Connecticut, USA)[2]. Since two breast of the same VBD can have different spatial distribution of the dense tissue, a deeper examination of information of the dense tissue distribution called density maps is needed.

The main objective of this work is to evaluate the distribution of the glandular tissue in FFDM and DBT. This comparison is aiming to see the similarity of the density maps measures provided by the commercial software named Volpara (v.1.5.4). Volpara uses the physics-based model to extract pixel-wise information from the FFDM and DBT [3]. Various studies reported promising results using Volpara for the assessment of breast density. Garcia et al. [3] concluded that Volpara is reliable in estimating the local glandular tissue distribution and can be used for its assessment and follow-up. Complemented by Gubern-Merida et al. [6] evaluated automatic volumetric breast density assessment in FFDM with Volpara which resulted to be very accurate with the potential to be used in objective breast cancer risk models and personalized screening. In addition, Quantra 3D (v.2.1), run at the colaborated hospital, will also be used to compare global measurement results, i.e. breast volume, glandular tissues volume and VBD with Volpara global measurement results. Furthermore, the density grade between radiologist, Quantra 3D and Volpara will also be investigated. The aim of this comparison is to try to correlate the qualitative assessment of the radiologists with the quantitative measure of the automated sofware tools (i.e. Volpara, Quantra 3D).

In order to simplify these objectives achievement, the comparison is divided into two, ie: global assessment and local assessment. Whereas global assessment covers the com-

[1]http://volparasolutions.com/

[2]http://www.hologic.com/

parison of global measurement results and local assessment covers the comparison of glandular tissue distribution.

II. Materials and methodology

The dataset consists of 61 patients data out of 68 patients data, whereas 7 of them are excluded due to corrupted data, mastectomy and prosthesis. The age of the screened women ranged from 38 to 81 years old with most of the patient are in range of 45-49 years old. Each patients underwent mammography and tomosynthesis image acquisition that consists of 4 views, ie. right breast CC, left breast CC, right breast MLO, and left breast MLO, which produce a total of 482 mammograms (241 FFDMs and 241 DBTs). Some views were not available and were not computed in some cases.

The dataset was acquired between November 2017 and December 2017 at the Hospital Universitari Parc Tauli (Sabadell, Spain) where it was acquired for screening purposes of high-risk women, i.e. high familiar or genetic risk in standard clinical settings or some with symptoms regarding their breast. The devices used for acquiring the dataset is Hologic Selenia Dimensions system (Hologic; Massachusetts, USA) which took both of FFDM and DBT image projection in a single breast compression for each breast/patient. Therefore, the two images were in perfect conditions to compute, and compare, the glandularity of the breast obtained using Volpara.

A. Data Validation

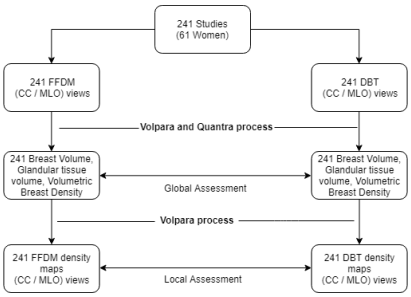


Fig. 1. Schematic overview of the validation process.

Fig. shows an overview of validation process from the dataset. 241 FFDM and 241 DBT images were validated for being processed using Volpara and Quantra 3D. Both processes calculated the dataset which provided a value breast volume (cm^3), glandular tissue volume (cm^3), and volumetric breast density (%) which are being compared in global assessment. Moreover, Volpara also processed the so-called density maps, which is being compared in local assessment regarding the glandular tissues distribution.

B. Image Registration

FFDM and DBT density maps have different size being DBT larger than FFDM. Hence, a direct subtraction of both images is not a feasible way to compute the similarity between them. These misalignments can be minimize by performing image registration. The registration is done using intensity-based image registration in Matlab called imregister.

The function transforms the 2-D or 3-D image, placed in variable moving, so that it is registered with the reference image, in variable fixed. The metric used in the registration is Mean Square which is computed by squaring the difference of corresponding pixels in each image and taking the mean of the squared differences. The mean squares metric is an element-wise difference between two input images with ideal value of zero. Affine transformation is applied in this registration since it consist of translation, rotation, scale, and shear, and is considered enough for registering FFDM and DBT images' from different sizes. Affine transformation is also the most popular used transformation in registration applications due to the rigid body constraint found in many common medical images [8].

C. Evaluation

In order to evaluate density maps between FFDM and DBT, global and local assessment are analyzed. Global assessment is performed after obtaining global measures from Volpara, complementing by Quantra 3D which run in hospital, comparing the breast volume (cm^3), glandular tissue volume (cm^3), and volumetric breast density (%) between FFDM and DBT images. In addition, local assessment is performed after obtaining density maps $DM(x, y)$ generated by Volpara with following equation:

$$\mathrm{DM}(x, y) = \frac{\ln(P(x, y))/P_{fat}}{\mu_{fat} - \mu_{dense}}$$

where $P(x, y)$ corresponds to the grey level intensity at pixel (x, y) in the raw mammogram, which is proportional to the X-ray energy absorbed at the image receptor and (P_{fat}) as mean intensity value of this area.

Key acquisition parameters from the meta-data of the image (e.g. kVp, X-ray tube anode material, filter material, compressed breast thickness) are read from the DICOM header to use the appropriate X-ray linear attenuation coefficients (μ_{fat} and μ_{dense}). h_{fat} is directly computed while h_{dense} is computed as the difference between the breast thickness (H) and the adipose tissue thickness.

Local assessment is done by calculating various similarity metrics. One approach is by direct comparison based on histogram, in particular the mutual information (MI) to measure of the mutual dependence between the two variables, Kullback-Leibler to measure of how one probability distribution diverges from another, Euclidean distance to computes the similarity between the intensity distributions of the two images, and histogram intersection to calculates the similarity of two discretized probability distributions, where various bins range from 8-64 are proposed. A unitary bin size, where each bin represents 1 mm glandular tissue thickness also calculated for a fair comparison.

Another metrics proposed in this work are statistic of difference image to determine changes between images by finding the difference between each pixel in each image, mean absolute error (MAE) to measure of absolute difference

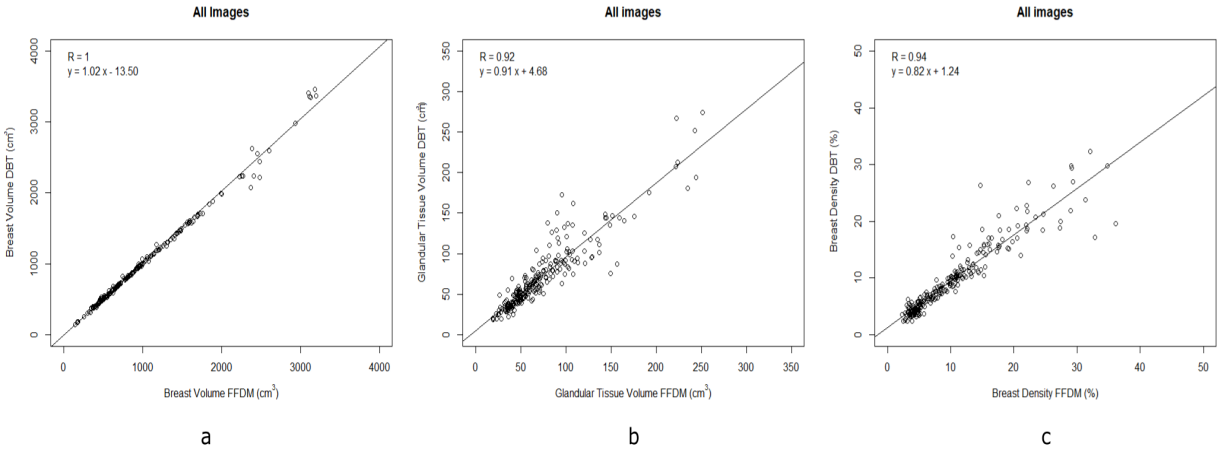


Fig. 2. Comparison of the global measures obtained by Volpara when analysing FFDM and DBT: (a) overall breast volume (R= 1), slope m =1.02), (b) glandular tissue volume (R=0.92), slope m= 0.91), and (c) volumetric breast density (R=0.94), slope m= 0.82).

between two continuous variables, mean square error (MSE) to measures the average of the squares of the errors or deviations, intensity correlation to compute the average of maximum value from normalised cross-correlation of both image, structural similarity (SSIM) to measure the similarity between two images based on an initial uncompressed or distortion-free image as reference, and dice similarity coefficient (DSC) to compare the similarity of two samples by calculated overlapped pixel.

D. Implementation Details

Affine transformation registration and the functions to estimate global and local features were implemented in MATLAB v.2017b (The MathWorks Inc., Natick, MA, USA). Meanwhile, data analysis and statistical tests were carried out using the statistical software R (v.3.0.3).

III. Results

IV. Global Assessment

Firstly we evaluate global measurement obtained from Volpara. Fig. 2 shows the dispersion plots of the obtained breast volume, glandular tissue volume, and volumetric breast density, where each point represents the values obtained in the two image acquisition techniques (x-axis represents value obtained in the FFDM while the y-axis the value obtained in DBT). In all the cases, the Pearson's correlation coefficients are R $\geq$ 0.92, while the slope of the linear models are m $\geq$ 0.82. In particular, the glandular tissue volume (Fig. 2(b)) obtained correlation coefficient R = 0.92 and slope m = 0.91.

In addition, global measurement from both software, Volpara and Quantra also being evaluate. Since the data we have is coming from Quantra 3D, we compare it only with DBT images for fair comparison. Fig. 3 shows the dispersion plots of the obtained glandular tissue volume between DBT images computed in both software. Overall, the Pearson's correlation coefficients, respectively for breast volume, glandular tissue volume, and volumetric breast density, are R $\geq$ 0.9, while the slope of the linear models are m $\geq$ 0.51.

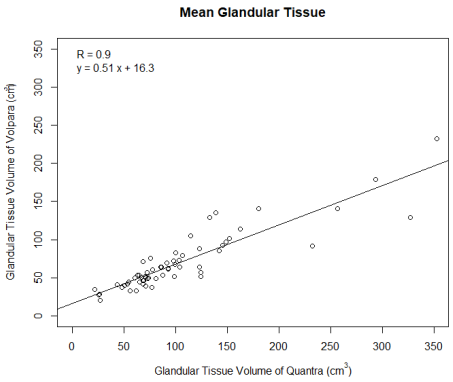


Fig. 3. Comparison of the glandular tissue volume obtained between DBT images by Volpara and Quantra 3D (R=0.9), slope m= 0.51).

Knowing the global measurement value, it is also interesting to compare breast density grade estimation between Volpara, Quantra 3D and radiologist. Kappa value obtained for each comparison, Quantra 3D and radiologist ($\kappa = 0.62$) shown in Table I, Volpara and radiologist ($\kappa = 0.58$) shown in Table II, and Quantra and Volpara ($\kappa = 0.64$) shown in Table III.

TABLE I
DENSITY GRADE COMPARISON (RADIOLOGIST - QUANTRA)

		Radiologist				
		A	B	C	D	Total
Quantra	A	2	2	0	0	4
	B	1	23	3	1	28
	C	0	5	18	3	26
	D	0	1	1	1	3
	Total	3	31	22	5	61

TABLE II

DENSITY GRADE COMPARISON (RADIOLOGIST - VOLPARA)

Volpara \ Radiologist	A	B	C	D	Total
A	0	1	0	0	1
B	3	21	2	1	27
C	0	8	12	1	21
D	0	1	8	3	12
Total	3	31	22	5	61

TABLE III

DENSITY GRADE COMPARISON (QUANTRA - VOLPARA)

Volpara \ Quantra	A	B	C	D	Total
A	0	1	0	0	1
B	4	21	1	1	27
C	0	6	15	0	21
D	0	0	10	2	12
Total	4	28	26	3	61

Cohen's kappa measures the agreement between two raters who each classify N items into C mutually exclusive categories. The definition of κ is defined as follows:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}$$

where p_0 is the relative observed agreement among raters (identical to accuracy), and p_e is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly seeing each category.

V. LOCAL ASSESSMENT

The local comparison between density maps is performed by using various similarity metrics. One approach depicted in Fig. 4 are a direct comparison based on histogram, i.e mutual information (MI) given similarity value of 0.20 $\pm$ 0.07 [0.06, 0.39], Kullback-Leibler given similarity value of 0.09 $\pm$ 0.15 [0, 1.26], Euclidean distance given similarity value of 0.66 $\pm$ 0.44 [0.13, 3.03], and histogram intersection given similarity value of 0.77 $\pm$ 0.12 [0.28, 0.95].

Another metrics proposed in this work, such as statistic of difference image, mean absoluter error (MAE), mean squared error (MSE), intensity correlation, structural similarity (SSIM), and dice similarity coefficient (DSC) is compared between before registration and after registration. Mean of difference image has a statistically significant value correspondence to MAE (Fig. 5), before and after registration respectively, 3.77 $\pm$ 2.10 [1.04, 14.05] and 2.22 $\pm$ 1.08 [0.81, 9:80]. This result shows that registration methods produced an improvement with respect the original comparison of the density maps.

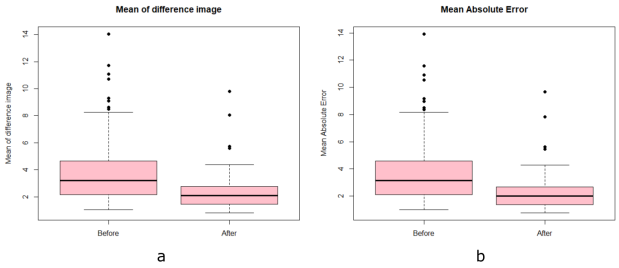


Fig. 5. Plot showing mean of difference image and MAE obtained by Volpara density maps when analyzing FFDM and DBT.

MSE mean value also benefits image registration which, resulting MSE a statistically significant value of 11.04 $\pm$ 12.02 [0.99, 118.90] from 31.72 $\pm$ 37.39 [1.86, 284.67]. Yet, MSE calculation is too disperse and is not showing great value even after registration, where value near 0 is showing better similarity. Moreover, DSC analyzed overlapped area between both density maps with various threshold ranging from 1 until 10 (measures on mm, based on glandular tissue thickness). Fig. 6 shows the trend from DSC. As the threshold increases, the area of the breast with respective glandular tissues thickness decreases, reducing the likelihood of overlap, and therefore the DSC value. Average DSC value for each threshold after registration respectively from 1 until 10 mm, are defined as follow: 0.94, 0.91, 0.87, 0.83, 0.80, 0.79, 0.79, 0.78, 0.78, and 0.77. It should be noted that in general a value of 0.7 or higher is often regarded as a satisfactory similarity. Meanwhile, SSIM metric only show non statistically significant value of 0.52 $\pm$ 0.17 [0.14, 0.88].

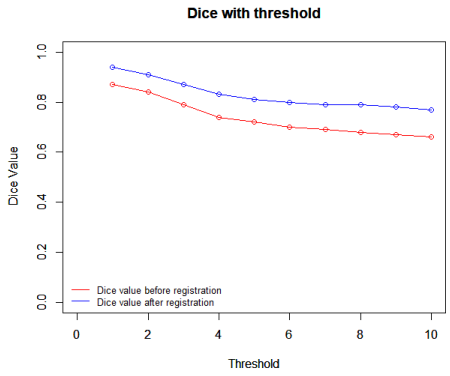


Fig. 6. Plot showing DSC trend from various threshold value ranging from 1 until 10 (measures on mm, based on glandular tissue thickness).

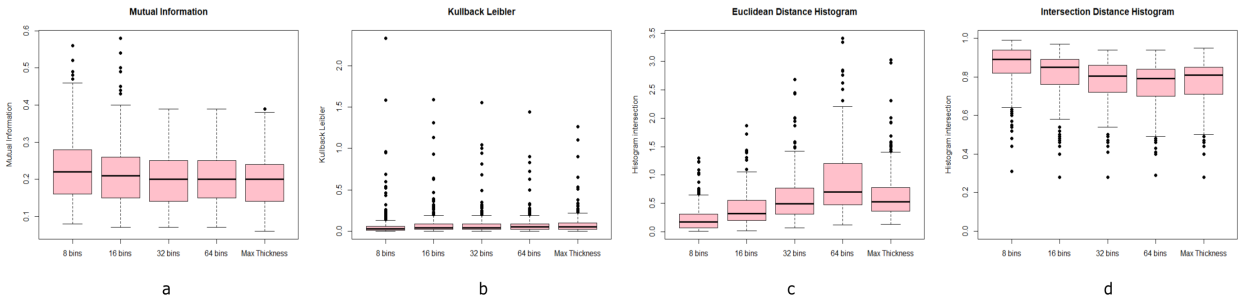


Fig. 4. Direct comparison of the local measures obtained by Volpara density maps when analyzing FFDM and DBT: (a) average of mutual information (0.21 (0.20, 0.23)), (b) average of Kullback-Leiber (0.09 (0.08 - 0.09), average of Euclidean distance (0.57 (0.24 - 0.92), and average of histogram intersection (0.79 (0.75 - 0.86).

VI. Discussion

In this work, we evaluated the volumetric density measures and local glandular tissue density map provided by Volpara Density Maps between FFDM and DBT images acquired using the same device and without decompression between both acquisition techniques.This date ensures that the glandular information in both modalities will be the same. Hence, we can evaluate how similar is the information from both images acquisition.

Global assessment is performed after obtaining global measures from Volpara, complementing by Quantra 3D which run in a colaborating hospital, comparing the breast volume (cm^3), volume of glandular tissue (cm^3) and volumetric breast density (VBD) (%) between FFDM and DBT images. The results show a high correlation. Observed from VBD measurement, the correlation and Bland-Altman plots show that Volpara tends to underestimate breast density in dense breasts compared to Quantra 3D. Volpara is based on a physics-based image model and it use a set of pixels of the breast that belong to fatty tissue as an internal reference to predict glandular tissue thickness. Gubern-Merida et al. [17] stated that the selection of the internal reference is more complex in dense breasts than in fatty breasts, which affects the calibration of fatty tissue attenuation and leads to breast density underestimation.

The breast density grade is compared between Volpara density grade (VDG), Quantra density category (QDC) and BI-RADS density grade provided by radiologist. The results show a clear association and substantial agreement calculated using weighted kappa with quadratic weights coefficient from the confusion matrix. VDG estimation and BI-RADS density grade (κ = 0:61), QDC estimation and BI-RADS density grade (κ = 0:55), and VDG estimation and QDC estimation (κ = 0:68) was found. In general, VDG estimation tends to be higher than the BI-RADS density estimation and QDC estimation, specifically in higher density grade. For instance, 9 studies that were estimated with BI-RADS density grade C obtained a VDG estimation of D and 9 studies that were estimated with QDC estimation C obtained a VDG estimation of D. These results are in contrast with results of volumetric breast density underestimation in Volpara. However, the volumetric breast density underestimation in Volpara does not seem to affect the final VDG categorization as observed both automated software has different algorithm and thresholds, used to estimate the density grade.

In order to do local comparison to evaluate the similarity of density maps between FFDM and DBT, a direct comparison based on histogram, in particular the mutual information (MI), Kullback-Leibler, Euclidean distance and histogram intersection are performed. MI resulted a low value of 0.20 comparing to MI result between left and right breast from FFDM and DBT with average value of 0.615. Where other metrics computed for Kullback-Leibler, Euclidean distance and histogram intersection has a value of 0.09, 0.66 and 0.77, respectively. Specifically, lower Kullback-Leibler and Euclidean distance and higher histogram intersection imply a better similarity in both density maps. However, many outliers are seen, showing a big range in the results and decreasing the significance level.

On the other hand, another metrics proposed in this work, such as statistic of difference image, mean absoluter error (MAE), mean squared error (MSE), intensity correlation, structural similarity (SSIM), and dice similarity coefficient (DSC) showed a high divergence when comparing FFDM and DBT density images directly. Presumably, this is caused by different size of both density maps where DBT size is

bigger than FFDM density maps size. However, a registration algorithms allowed us to overcome this issue. The affine registration transformation with mean square metric used in this work, have shown a statistically significant improvement compared to no registration. This is proved by the mean of difference images result which coincides with MAE, showing an improvement before and after registration, respectively 3.77 and 2.22. MSE results is improve from 31.72 to 11.04 after registration. Complemented by intensity correlation and DSC results, respectively 0.94 and 0.83, where value near 1 shows better similarity. Meanwhile, SSIM results shows a moderate similarity with an insignificance value of 0.52 after registration.

Differences in both image acquisition techniques have been studied considering Pearson's correlation of the global measurement results and results from various similarity metrics between corresponding density maps. To conclude, there is a high similarity of density maps provided by Volpara between FFDM and DBT acquisition. The main divergences between the density maps computed from both acquisitions are mainly due to the different size and it can be minimized by doing an image registration.

REFERENCES

[1] Sigurdur Ingvarsson. Breast cancer: introduction. Seminar in Cancer Biology, 11:323-326, 2011.

[2] Lee R. Dice. Measures of the amount of ecologic association between species. Ecology, 26 (3):297-302, 1945.

[3] Eloy Garcia, Oliver Diaz, Robert Marti, Yago Diez, Albert Gubern-Merida, Melcior Sentis, Joan Marti, and Arnau Oliver. Local breast density assessment using reacquired mammographic images. European Journal of Radiology, 93:121-127, 2017.

[4] M.J. Michell, A. Iqbal, R.K. Wasan, and et al. A comparison of the accuracy of film-screen mammography, full-field digital mammography and digital breast tomosynthesis. Clinical Radiology, 67:976-981, 2012.

[5] M.A. Helvie. Digital mammography imaging: Breast tomosynthesis and advanced applications. Radiologic Clinics of North America, 48:917-929, 2010.

[6] Albert Gubern-Merida, Michiel Kallenberg, Bram Platel, Ritse M. Mann, Robert Marti, and Nico Karssemeijer. Volumetric breast density estimation from full-field digital mammograms: A validation study. PLoS ONE, 9:e85952, 2014.

[7] Danielle van der Waal, Gerard J. den Heeten, Ruud M. Pijnappel, Klaas H. Schuur, Johanna M. H. Timmers, Andre L. M. Verbeek, and Mireille J. M. Broeders. Comparing visually assessed bi-rads breast density and automated volumetric breast density software: A cross-sectional study in a breast cancer screening setting. PLoS ONE, 10:e0136667, 2015.

[8] JB Antoine and MAV Maintz. An overview of medical image registration methods. Imaging Science Department, 1996.

CNN for SfT - 3D Reconstruction from a Single Image without Ground-Truth

Anneke Annassia Putri Siswadi[1] and Adrien Bartoli[2]

Abstract— **The Reconstruction of 3D models of deformable objects from a monocular data (Single 2D image) is still being a fundamental problem in computer vision. SfT is a promising method with which 3D shape of a deformable object can be recovered by comparing the object in the input image and the objects template before deformation. However, SfT has its limitations; it fails when the texture is homogeneous. To overcome this issue, SfT can be integrated with deep learning approaches. Yet deep learning technique mostly needs Ground-truth data which cannot be provided all the time. An alternative way is to implement CNN without ground-truth which was also successful in few other approaches. In this research, we propose a CNN Network that can reconstruct a 3D model of a deformable object in an input image using SfT without providing any ground-truth data.**

I. INTRODUCTION

The Reconstruction of 3D models from a monocular data (Single 2D image) has been an interesting research area over decades [1]. This is a quite challenging task when it comes to deformable objects which are still being a fundamental problem in computer vision. The 2D image lacks essential data to reconstruct a 3D model as it can not provide the depth information from its pixel correspondence. A solution to this problem is to provide additional information of the object present in the 2D image such as its volume of weighted minimal surfaces [14], contours topology [11] or its template (Shape from Template - SfT) [1].

The solution proposed by *Toeppe et al.* [14] focuses on predicting the depth values of the input image. It can be done by providing the minimal surfaces of the object volume with considering its silhouette consistency but this method requires the user interaction because the volume should be increased or decreased manually to get the better result. In contrast, in *Mukta et al.* [11] approach, the solution is to look at the object contour's topology. The 3D reconstruction of a deformed object can be build by knowing the information about its contour's topology map. However, this method fails provided incorrect edge detection of the object. The other solution is SfT approach *Adrien et al.* [1]. Instead of using the edge information, SfT method uses the points information for 3D reconstruction which is more flexible to be used in case of 3D modeling of deformable object [5].

Among these solutions, SfT is better one which can provide the fundamental result in 3D reconstruction as mentioned in [1] so that 3D shape of a deformable object can be recovered by comparing the object in the input image and the object's template before deformation. Most of the SfT technique use feature matching algorithm which is not so efficient and hence deep learning techniques can be applied which is proposed in [16].

There are many researchers who implement the CNN for reconstructing the 3D model of an object from monocular image by considering some other object information, such as object's volume with 3D geometry of input image [5], depth image with 3D voxel [7], depth image with 3D geometry [17][12], object's appearance and object's silhouette of the input data [19] [10] [15] [18]. Most of these implementations require the ground-truth data to acquire the better result for reconstruction.

In the real-life implementation, ground-truth cannot be provided all the time, since it is difficult to generate it. However, it is possible to build a CNN that doesn't require ground-truth (unsupervised CNN). There are a few researchers who succeeded in implementing this concept for other computer vision problems like depth estimation [8] and optical flow estimation [9].

The aim of our research is to build a CNN for reconstructing the 3D model of a deformed object from a 2D image without any ground-truth data. In SfT concept, as the object's template is sufficient to reconstruct the 3D model, no further information about the deformed object is necessary. The idea is to train the CNN with the input images of some deformed objects and corresponding template. Our research focuses on using the 3D geometry of the object to avoid ground-truth. Some constraints are provided for simplification of our research:

- **The object is flat:** The object used in our research is a rectangular flat object with known texture.
- **The object is isometric:** The deformation of the object is predictable since the object in the template is the same as the object in the input image.
- **The input is a real image:** The input data for the CNN is the 2-Dimensional real image without any supervision.
- **The object exists in the image:** The object always exists in every input image.
- **object is in same conditions in input and template:** The input images are taken in the same time with the same illumination conditions as of the template.
- **The template data are provided:** The template is the object information before deformation.

[1]University of Burgundy
[2]Institut Pascal, EnCoV - UMR 6602 CNRS (ex-ISIT)

II. Background

A. Shape from Template (SfT)

The idea of SfT is to fit the object in the input image with its template (before deformation) by computing the deformation parameters of the 3D surface template in a way that the appearance of the projective deformation (warped with the template's texture) will match the appearance of the object in the input image [1][2][6]. Most of SfT techniques implement the features matching algorithm for selecting the pixel correspondence between the object in the template and input image [1]. These algorithms fail when the texture template is homogeneous or has too many repetitions. To overcome this issue researchers began to adopt the deep learning technique in 3D reconstruction [16].

B. CNN with Ground-Truth

Convolution Neural Network (CNN) is a kind of ANN which consists of some convolution layers before entering the interconnected neurons. Like other ANNs, CNN also adopts the forward propagation and backward propagation algorithms for training. CNN uses the advantages of image pixels to extract its feature information for learning the pattern [13].

In some research approaches like 3D-Recurrent Reconstruction Neural Network (3D-R2N2) [3], Appearance Flow Network (AFN) [19] and Transformation-Grounded Image Generation Network (TVSN) [10], 3D reconstruction is mainly focused on the object's appearance. The targets for these approaches are rigid objects. The main idea behind the CNN implementation in these approaches is to learn the structure of the 3D object by reconstructing its appearance from multiple views. This approach requires a 3D CAD model of the object as the additional information for reconstruction.

Besides the object's appearance, some research approaches like 3PointOutNet [5], T-Network [7] and SurfNet (Surface Network) [12] makes use of 3D geometry of the object in the scene for reconstruction. The targets for these approaches are the deformable objects. The main idea of the CNN implementation in this approach is producing the 3D geometry of the input object and compute the difference error between the predicted 3D geometry and 3D geometry provided in ground-truth.

In our research, we adopt the SurfNet architecture for our network. SurfNet predicts the 3D surface of an object based on its 3D geometry. This network implements the CNN for producing the geometry of the object in three different axes (x, y and z). The main concept in SurfNet is to combine all the three coordinates of the object and merging it with the basic shape which is already computed analytically [7][4].

C. CNN without Ground-Truth

Implementing CNN without any ground-truth has been successful in some methods like depth estimation [8] and optical flow estimation [9]. In [8], Godard et al. proposed the CNN implementation for estimating the depth of the image by generating its disparity map. They provides an idea to train the network by computing its loss appearance instead of computing the distance between the network output and the ground-truth data. The similar strategy has been used in UnFlow to train the network without ground-truth [9]. UnFlow network implements CNN for estimating the optical flow from two input images and warps its output with the next input image, considering the pixel consistency. It is possible to reach our goal by computing the loss cost of object's appearance to make it trained without ground-truth and to generate its appearance by processing the 3D geometry from CNN layers based on SfT.

III. Research Methodology

Our research is based on implementing the CNN without ground-truth for reconstructing the 3D model of a deformable object from a 2D image by employing the 3D geometry data. The main goal is to generate 3D geometry with the strategy of CNN without ground-truth. Like other CNN process flow, our research methodology has been classified into three stages: data preparation, training and testing.

A. Data Preparation

1) Input Image: The input of our network is RGB images with the deformed object overlay in. The input images are captured with a normal camera. In contrast to the existing networks in 3D reconstruction which uses the image without background, we use input images containing both the object and its background. The size of the input image is 128x128.

2) Template: The template data in our case is the object data before deformation. The object data holds in it the texture images and 3D elements of the object in the scene. Texture images are the captured images with actual texture of the object. The texture images and the input images are captured at the same time in order to keep the environment stable. These images are RGB images with the size of 195x130 pixels, approximately. The actual size of the texture image is 20.3x14.2 cm. The 3D object elements are the 3D geometry data of the object in the template (See Fig. 1). It consists of 3D vertices topology and points with face correspondences.

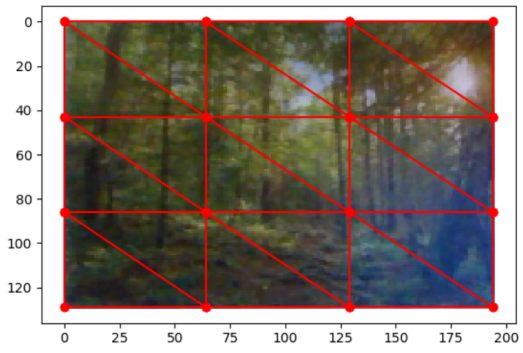


Fig. 1. Object Vertices Topology

B. Training

The strategy of our network is to train without ground-truth for estimation of 3D geometry (Fig. 2). The training phase implements two algorithms in the sequence, forward propagation and backward propagation. We adopted forward propagation in which the input image is processed through the CNN layers for producing the 3D geometry images of the deformed object. These geometry images contain three different vertices in three axes (x, y and z). The SfT concept is applied in the next process for calculating the loss costs of the network.

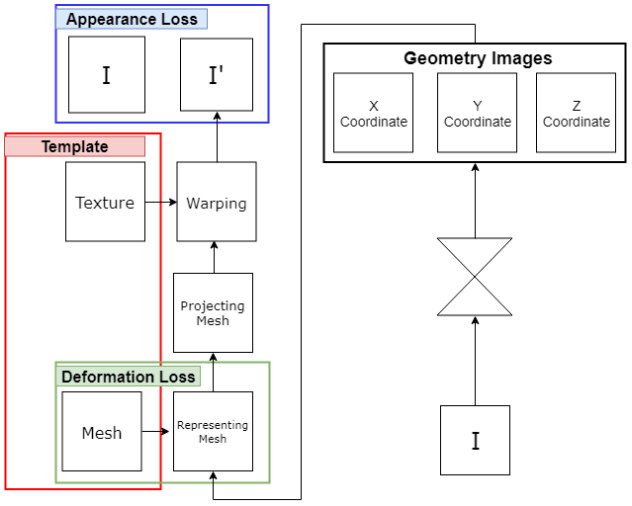


Fig. 2. Training Strategy

1) CNN Architecture: In our network, we adopt the CNN architecture for the non-rigid object in Surfnet in order to estimate the 3D geometry images of the deformed object. Our network architecture is shown in Fig. 3.

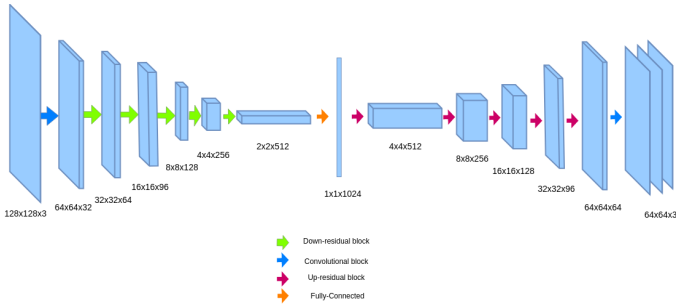


Fig. 3. CNN-based SfT Architecture

To simplify the network process, we convert the input RGB image as grayscale image. Like the Surfnet, Our network architecture also implements the resnet model. In encoder architecture, the first layer in our network is a pure convolution layer. Instead of using the multiple convolution layers, it continues with five down-residual block which consists of convolution layers, batch normalization layers and activation layers (ReLu).

The down-residual block is a combination of downsampling block and residual block. The projection shortcut is applied in the downsampling block. In this block, the 1 x 1 convolution with 2 strides are performed to match these dimension. On the other hand, the identity shortcut is applied performed in residual block. The 3 x 3 convolution with 1 stride is performed for this block. The last layer in encoder architecture is a fully-connected layer. Our network uses 1024 connected-neurons for learning the pattern of the input object.

The decoder architecture in our network is responsible for the reconstruction process. This decoder implements five up-residual blocks which is the combination of the upsampling block and residual block. In contrast to most of the CNN architecture, this decoder uses the convolution transpose layer (transpose the gradient) for 2 x 2 filter and 2 strides instead of deconvolution layer. The residual upsampling block consists of convolution transpose layer, convolution layers, batch normalization layers and activation layers (ReLu). Same as the residual downsampling block, this block also applies the projection shortcut. A convolution layer is added at the end of the network for producing the three geometry images in the different axes (x, y and z). The size of these output images are 64 x 64. These three geometry images contain the vertices in 3D system coordinate. In other words, this network produces 4096 vertices for each axis.

2) SfT for CNN: In the training phase, calculation of loss function is important. CNN (with ground-truth) calculates the loss function between the ground-truth and the output from the network, whereas in our network loss function is calculated between the generated appearance of the deformed object with its appearance in the input image (appearance loss function) or the difference of the geodesic distance between deformed object mesh and template mesh (deformation loss function).

The object's appearance is generated by implementing SfT concept to the 3D geometry output of the CNN layers (considering the geodesic distance consistency). There are on the whole of 4096 vertices, out of which 16 vertices are taken for processing since the number of vertices used in 3x3 template is 16. These vertices are already in synchronization with the template's vertices topology. The process for generating the appearance of the input object is is performed in three stages, representation of the vertices in mesh form, projection of the vertices and warping the image.

Mesh Representation: The deformation loss is computed as the difference of geodesic distance between the template vertices and object vertices. Since the object vertices are already in the same order with the template vertices topology, both vertices will also have the same mesh order. The mesh for object vertices is represented in the same manner with the template mesh representation.

Vertices Projection: Camera calibration is applied before projecting the object vertices in order to get the camera intrinsic parameters. However, this camera matrix can't be used for projecting the object vertices because of the different image size between the calibration image and input image. The equation (1) is computed in order to have the right projection matrix.

$$K'_{(3\times3)} = S_{(3\times3)} \times K_{(3\times3)} \quad (1)$$

where:

- K : Camera matrix from calibration camera
- K' : New camera matrix
- S : Scale vector

The vertices are projected by multiplying the 3D vertices with the parameters of the projection matrix (Equation (2))

$$(f_x \frac{p_x}{p_z} + c_x, f_y \frac{p_y}{p_z} + c_y) \quad (2)$$

where:

- (p_x, p_y, p_z) : Input coordinates
- (f_x, f_y) : Focal length
- (c_x, c_y) : Image center

By assuming that the object exists in the input image, we can define that the maximum area of the deformed object in the input image is 128x128. The synthesized data are used for checking and visualizing the process of generating appearance. The mesh representation of these synthesized data can be seen in Fig. 4.

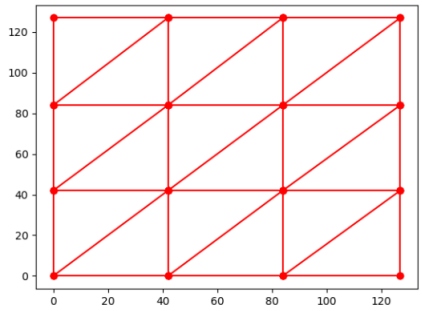


Fig. 4. Mesh Representation

Normalization is applied for projected vertices in order to handle the outlier. The aim of this normalization is to overlay the 2D object vertices into the image (128x128). In this process, the minimum and maximum threshold are initialized to be 0 and 127, respectively. If the projected object vertices have the coordinates below minimum threshold, by default it will become 0 whereas for those with maximum threshold by default it becomes 127.

Image Warping: To generate the object appearance one requires all points coordinate which is filled inside the object mesh. To obtain all these filled-points we need to find the pixel correspondences of the template's filled-points which is already provided.

The affine transformation is applied for defining the transformation of these different coordinates. The transformation matrix is computed in each face with three connected vertices between object vertices and template vertices. Once the transformation matrix of two faces is defined, all correspondence points in the object can be calculated. Finally, the object's appearance for a face can be generated by copying the pixel intensity from pixel correspondences to the texture image. The warping result of a single face is shown in Fig. 5.

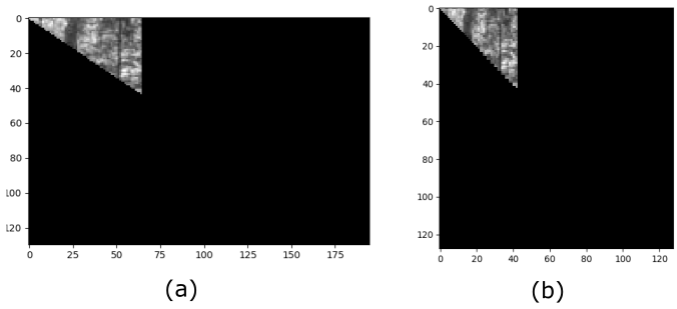


Fig. 5. Single Face Warping (a) Face Template, (b) Face Object

After warping gets completed in all the faces of the mesh, the object's appearance is generated as shown in Fig. 6. Both texture image and object's appearance are converted into gray image in order to get the appearance loss cost which is more accurate than using RGB image.

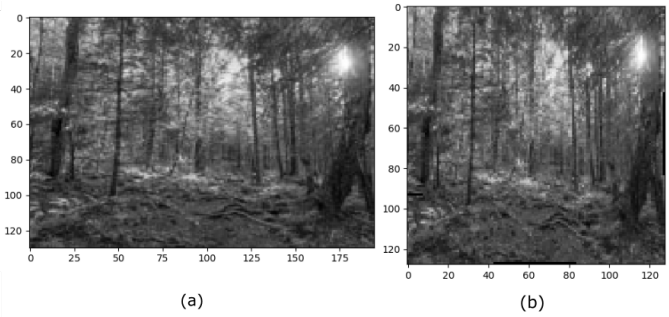


Fig. 6. Object Appearance (a) Template, (b) Object

Loss Function There are two kinds of loss functions applied in our network, appearance loss (La) and deformation loss (Ld). Like the other CNN implementation without ground-truth, we also apply the loss appearance. Since the target object is deformable and isometric, we also apply the deformation loss in our network.

The appearance loss is applied for computing the difference between the generated object appearance and the object's appearance in the input image. The generated object's appearance is the appearance retrieved by warping the filled-points object with the texture image while the object appearance in input image is the result of warping the filled-points object with the input image. The appearance loss is computed as sum square error by considering the Brightness Constancy Assumption (BCA) which is shown in equation (3).

$$L_a = min \sum_{i=0}^{n} \sum_{j=0}^{n} ||I_{(i,j)} - I'_{(i,j)}||^2 \quad (3)$$

In contrast, the deformation loss is applied for computing the difference of the distance between each vertices which is connected by the edges between 3D mesh of input object and 3D mesh of template. This loss function is computed as sum square error by considering the isometric consistency (see equation (4)).

$$L_d = min \sum_{i=1}^{n} (||p'_{i+1} - p'_i||^2 - ||p_{i+1} - p_i||^2)^2 \qquad (4)$$

C. Testing

The testing phase is to perform the evaluation of the networks. Since the goal of our network is to reconstruct the 3D shape from a single image, the testing phase in this network is to evaluate the result of the 3D reconstruction. Comparing to the training architecture, the network architecture in the testing phase consists of CNN layers without performing some SfT process. The input image of this testing phase is the RGB image which is fed into the testing network with the updated value of weight and bias parameters after training.

IV. Implementation

A. Training

We implemented both forward propagation process and backward propagation algorithms in training phase. The network is executed using the real images and shows how the training strategy handles the real environment. In training phase, forward propagation is executed once at the beginning of training. In forward propagation, the task is to allocate the memory of all commands and variables of training thereby executing the input images until the network computes the loss cost, whereas in the backward propagation, the task is to calculate the gradient of the loss cost and to minimize it while the network learns for the exact value of its weights.

In the SfT for CNN process, the loss functions are calculated through sequence of processes: Mesh Representation, Mesh Projection and Warping the Object's Appearance. The Fig. 7 below is one case of the SfT for CNN implemented in the real data.

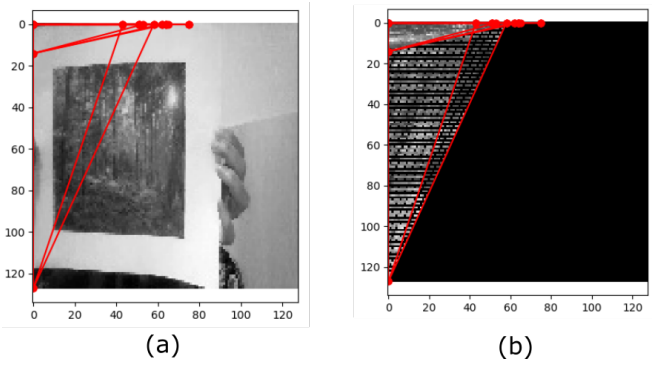


Fig. 7. Handling Real Data (a) Input Image, (b) Object

We can infer that with SfT, our network can generate the object's appearance if the triangular object face is less than or equal to the triangular template face. In contrast, our network fails to perform well when the triangular object face is greater than the triangular template face. This is a normal behaviour since the warping function creates a duplicate appearance from the specific image. However, both deformation loss and appearance loss are successfully calculated. The training is executed with 50 epochs and 8 batch size. The optimization algorithm used in our network is Adam optimizer with 0.001 learning rate. However, the training process is still not completely successful. The forward propagation does its job well but the back propagation fails to compute the gradient between the loss costs and the weight variables in CNN layers.

B. Testing

Due to the issues encountered in training phase, we couldn't test the network with real data. However, the works are continuing to reach the main goal of reconstructing a 3D model of a deformed object from a single 2D image without any ground-truth data.

V. Conclusion and Future Work

A. Conclusion

SfT based CNN without ground truth can be a boon in 3D reconstruction since it can aid in the 3D modeling of a deformable object. The main advantage of this technique is that it needs only one template for reconstruction and the same template can be used for more than one input images provided the object in the input image and template are same. The challenges we faced in this research is the back-propagation which fails to compute the gradient of loss cost. However, we are continuing the research to figure out this issue and optimize the process.

B. Future Work

In order to evaluate the network, all process need to be executed both training and testing. Since the failure in training process is in the backward propagation, the next task to do is to optimize the code by performing the custom layer for 'SfT for CNN' process for taking care of the gradient of the loss costs, and testing.

References

[1] Adrien Bartoli, Yan Gérard, François Chadebecq, Toby Collins, and Daniel Pizarro. Shape-from-template. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):2099–2118, 2015.

[2] Ajad Chhatkuli, Daniel Pizarro, and Adrien Bartoli. Stable template-based isometric 3d reconstruction in all imaging conditions by linear least-squares. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 708–715, 2014.

[3] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision*, pages 628–644. Springer, 2016.

[4] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1538–1546. IEEE, 2015.

[5] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 38, 2017.

[6] Mathias Gallardo, Toby Collins, and Adrien Bartoli. Can we jointly register and reconstruct creased surfaces by shape-from-template accurately? In *European Conference on Computer Vision*, pages 105–120. Springer, 2016.

[7] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016.

[8] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[9] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. *arXiv preprint arXiv:1711.07837*, 2017.

[10] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 702–711. IEEE, 2017.

[11] Mukta Prasad and Andrew Fitzgibbon. Single view reconstruction of curved surfaces. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1345–1354. IEEE, 2006.

[12] Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *Proc. CVPR*, 2017.

[13] UFLDL Stanford. Convolutional neural network. [Online]. Accessed on May 2018.

[14] E. Toeppe, M. R. Oswald, D. Cremers, and C. Rother. Image-based 3d modeling via cheeger sets. pages 53–64, Queenstown, New Zealand, November 2010.

[15] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, volume 1, page 3, 2017.

[16] Armine Vardazaryan and Adrien Bartoli. Sft-from-cnn — using deep learning to aid deformable 3d reconstruction.

[17] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 1696–1704, 2016.

[18] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 1696–1704, 2016.

[19] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European conference on computer vision*, pages 286–301. Springer, 2016.