# AMORE: CNN-BASED MOVING OBJECT DETECTION AND REMOVAL TOWARDS SLAM IN DYNAMIC ENVIRONMENTS

**A. Pancham[1*#], D. Withey[2] & G. Bright[1]**

## ARTICLE INFO

***Article details***

| | |
|---|---|
| Submitted by authors | 19 May 2019 |
| Accepted for publication | 30 Sep 2020 |
| Available online | 14 Dec 2020 |

***Contact details***
* Corresponding author
ap.slamide@gmail.com

***Author affiliations***
1 Faculty of Mechanical Engineering, University of KwaZulu-Natal, South Africa

2 Centre for Robotics and Future Production, Manufacturing, Council for Scientific and Industrial Research, South Africa

# The author was enrolled for a Doctor of Philosophy in Engineering degree at the University of KwaZulu-Natal, South Africa

***ORCID® identifiers***
A. Pancham
https://orcid.org/0000-0002-1255-8074

D. Withey
https://orcid.org/0000-0003-1515-7839

G. Bright
https://orcid.org/0000-0003-4386-0329

***DOI***
http://dx.doi.org/10.7166/31-4-2180

## ABSTRACT

Simultaneous Localisation And Mapping (SLAM) In Dynamic Environments (IDE) may be improved by detecting and removing moving objects that may otherwise lead to localisation errors. This work combines convolutional neural networks and feature clustering to serve as A Moving Object detection and REmoval method (AMORE) that removes moving objects from the SLAM process and improves the performance of SLAMIDE. Experiments show that a visual SLAM algorithm and AMORE combined are more robust with high-dynamic objects than the SLAM algorithm alone, and performance is comparable to state-of-the-art visual SLAMIDE approaches. AMORE has the advantage of simplicity, requiring minimal implementation effort.

## OPSOMMING

Gelyktydige lokalisering en kartering in dinamiese omgewings kan verbeter word deur voorwerpe te bespeur en te verwyder wat andersins mag bydra tot lokaliseringsfoute. Hierdie artikel kombineer konvolusie neurale netwerke en kenmerk groepering as 'n voorwerp bespeuring en verwyderings metode wat bewegende voorwerpe van die lokalisering en kartering proses verwyder. Eksperimente toon dat 'n visuele gelyktydige lokalisering en kartering algoritme beter resultate lewer in hoogs-dinamiese omgewings wanneer dit met hierdie verwyderingsmetode gekombineer word. Die vertoning is soortgelyk aan bestaande benaderings, maar dit is eenvoudig en benodig minimale inspanning om te implementeer.

## 1 INTRODUCTION

Simultaneous Localisation And Mapping (SLAM) enables a mobile robot to construct a map of an unknown static environment and localise itself simultaneously [20]. The SLAM problem in static environments has been researched extensively. Applications have evolved for different environments such as indoor-to-outdoor, aerial, underwater, and mining robotics [21]. Most of these applications are undertaken in static environments.

However, real-world environments are dynamic and contain moving objects, such as people, pets, cars, and robots, which may lead to localisation errors and so reduce the map quality of SLAM. The performance of SLAM In Dynamic Environments (SLAMIDE) may be improved by detecting moving objects and removing or tracking them [22].

There are several works on SLAMIDE [24-26], each with its own assumptions, advantages, and disadvantages. However, there is no prevailing solution, and questions remain about sensor types, methods for differentiating stationary and moving objects [27], and how best to remove or track moving objects.

The advent of affordable Red Green Blue — Depth (RGB-D) cameras makes both colour and depth data available from a single sensor [15]. Convolutional Neural Networks (CNN) have achieved superior results for object detection in images [1-3], and recently for SLAM and moving object detection and removal [6, 10, 11].

In this work, CNN and 3D feature clustering are combined to serve as A Moving Object detection and REmoval method (AMORE), which removes moving objects from the SLAM process. AMORE is integrated with an RGB-D SLAM algorithm to improve performance in dynamic environments, using only sparse feature information.

For the experiments in this study, AMORE combines the CNN object detector, You Only Look Once Version 3 (YOLOv3) [1], mean shift clustering [7], and the visual SLAM (vSLAM) algorithm ORB-SLAM [9]. ORB-SLAM AMORE refers to the combination of these methods. The SLAM method used here is ORB-SLAM, but AMORE may be coupled to any vSLAM algorithm. ORB-SLAM is a state-of-the-art SLAM algorithm that has shown good performance in static and low-dynamic environments [8]. YOLOv3 is a state-of-the-art object detector with outstanding results [1]. Mean shift showed best overall performance for clustering features from RGB-D images in a recent study [14].

The performance of ORB-SLAM AMORE is validated in experiments with the TUM RGB-D dataset [15], which is widely used as the benchmark to evaluate vSLAM algorithms. Results show that it is more robust with high-dynamic objects than ORB-SLAM alone. The accuracy of ORB-SLAM AMORE is comparable to state-of-the-art, low-cost RGB-D SLAMIDE algorithms in these experiments, and it has the advantage of simplicity, requiring minimal implementation effort.

The rest of the paper is organised as follows: Section 2 discusses related work, Section 3 provides technical background to the algorithms used in AMORE in this study, Section 4 describes AMORE, Section 5 presents the experimental methods, Section 6 contains the experimental results, Section 7 provides a thorough discussion of the proposed approach, highlighting its strengths and shortcomings, and Section 8 concludes and projects future work.

## 2 RELATED WORK

Detailed reviews [24, 28] on existing SLAMIDE approaches describe their novelty and highlight their advantages and disadvantages.

ORB-SLAM [8, 9] is the first open-source SLAM algorithm that can be applied to monocular, stereo, and RGB-D cameras. It is explained in detail in the research of Mur-Artal, Montiel and Tardos [8]. It builds on parallel tracking and mapping [29] and other algorithms. ORB features are used, as they are computationally efficient and rotation invariant [30]. ORB-SLAM consists of three parallel threads: tracking, local mapping, and loop closing. The tracking thread performs camera localisation and new keyframe decisions. Keyframes contain camera and feature information, and are selected based on specific criteria [8]. The local mapping thread carries out new keyframe processing, local bundle adjustment, and redundant keyframe removal. The loop closing thread performs loop detection and closure [8]. Extensive evaluations of ORB-SLAM have demonstrated its excellent accuracy. It is robust with low-dynamic changes [8], is not affected by brightness variations, and offers computational efficiency. However, it is unsuitable for environments without features; similar features may cause incorrect loop closures; and drift arises without loop closures [31].

Sun, Liu and Meng's [5] Motion Removal (SMR) approach serves as a front end to RGB-D SLAM, and filters out dynamic object data. It uses ego-motion compensated image differencing, a particle filter, and a maximum a posteriori estimator. However, it can only detect a single foreground moving object; therefore, if there are many moving objects at different depths, motion removal might be difficult.

StaticFusion (SF) [4] jointly estimates camera pose and scene segmentation, and filters foreground dynamic objects. The segmentation is used for weighted dense RGB-D fusion to build a 3D surfel model only of stationary objects. SF has a quick runtime, although, for initialisation, at least 70 per cent of the environment needs to be static.

Detect-SLAM (DS) [6] is the first work to combine CNN and RGB-D SLAM for mutual benefit. It uses Single Shot multibox object Detector (SSD) [3] as the object detector and ORB-SLAM [8, 9] as the RGB-D SLAM method. SSD is not fast enough to be applied to each frame of the tracking process, and therefore is only applied to ORB-SLAM keyframes. This allows DS to operate in real time. Moving objects are regarded as objects that have a tendency to move — e.g., a person, dog, cat, or car — regardless of their action, such as walking or standing. DS has two versions: DS1 removes moving features from the bounding boxes that SSD detects; and DS2 uses moving feature probability propagation to remove moving features. The method is integrated into ORB-SLAM, and requires modifications to both the tracking and the local mapping threads of ORB-SLAM. DS requires SSD to be fine-tuned to improve object detection under partial observation, motion blur, and occlusion. DS constructs an instance-level, dense, semantic map of static objects, which is used as prior knowledge for better detection in demanding environments.

DS-SLAM (DSS) [10] builds on ORB-SLAM [9]. It consists of five parallel threads: tracking, semantic segmentation, local mapping, loop closing, and dense semantic mapping. The semantic segmentation network SegNet [13] and a moving consistency check method are combined to remove moving objects. It is assumed that features that belong to people are likely to be outliers or moving. DSS operates in real time, although the object recognition in the semantic segmentation network is constrained to certain classes, which limits its application. The moving consistency check method is integrated into the tracking thread of ORB-SLAM [10].

DynaSLAM (DyS) [11] augments ORB-SLAM [9] with front-end moving object detection and background inpainting. Mask R-CNN [12] and multi-view geometry models are combined to detect moving objects. For the TUM dynamic objects dataset [15], DyS performs better than the other methods compared. DyS has the advantage that it can detect any object's movement, although it has yet to be optimised for real-time performance.

## 3 TECHNICAL BACKGROUND

In addition to ORB-SLAM, which is described in Section 2, the algorithms below are used in this study.

### 3.1 Mean shift clustering

The mean shift algorithm [35] iteratively moves or shifts each data point to the mean of the data points in the kernel. Mean shift is non-parametric, the number of clusters is not required, and it can find arbitrarily shaped clusters [36, 37]. The mean shift algorithm is explained in detail in Comaniciu and Meer [37] and Derpanis [38]. For the set of $n$ independent identically distributed data points $X = \{x_{i,}, i = 1, \ldots, n\}$, in the $d$-dimensional space $R^d$, the multivariate kernel density estimator for the point $x$ is given by

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{1}$$

The bandwidth $h > 0$, and $K(x)$ is the kernel for radially symmetric kernels. The modes are found at the zeros of the gradient $\nabla f(x) = 0$. The gradient of the kernel density estimator is given by

$$\hat{\nabla} f_{h,K}(x) = \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)\right] \left[\frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x\right] \tag{2}$$

where the function $g(x) = -k'(x)$. The function $k(x)$ is the kernel profile for $x \geq 0$. The normalisation constant $c_{k,d} > 0$, ensures that $\int_{R^d} K(x)dx = 1$. The first term reflects the density estimate at $x$ and the second term is the mean shift vector $m$, which indicates the direction of increasing density. The main steps of mean shift are:

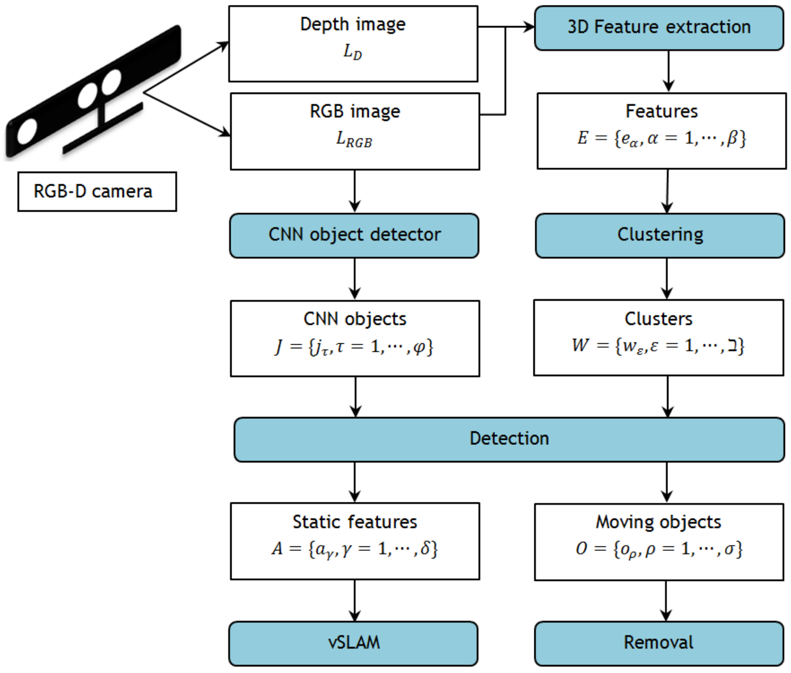
1. Calculate the mean shift vector $m_{h,G(x)}$.
2. Translate the kernel $G(x)$ by $m_{h,G(x)}$.
3. Repeat steps 1 and 2 until $\nabla f(x) = 0$.

## 3.2 YOLOv3

YOLOv3 [1] is a real-time object detection algorithm. It uses dimension clusters as anchor boxes to predict bounding boxes. Logistic regression is used to predict an objectness score for each bounding box. Multilabel classification is used to predict the classes that the bounding box may have. In experiments with the COCO dataset [33], YOLOv3 can predict 80 classes (including people, vehicles, animals, and inanimate objects). Boxes are predicted at three different scales, from which features are extracted. YOLOv3 uses Darknet-53, a 53-layer convolutional network for feature extraction, which combines the Darknet-19 network from YOLOv2 and residual network concepts. At 320×320, YOLOv3 executes in 22 $ms$ with 28.2 $mAP$ [1].

## 4 MOVING OBJECT DETECTION AND REMOVAL WITH AMORE

In real-world environments people have the potential to move, and their motion will tend to decrease the performance of SLAM. AMORE only regards people as moving objects, but it can be modified to include other moving objects to increase its versatility. A person is classified as moving, regardless of whether they are stationary (e.g., standing) or moving (e.g., walking). DS [6] and DSS [10] also regard people as moving objects, and the former method also does not differentiate between stationary and moving people. The AMORE process combined with vSLAM is shown in Figure 1.



Depth image
$L_D$
3D Feature extraction
RGB image
$L_{RGB}$
Features
$E = \{e_\alpha, \alpha = 1, \cdots, \beta\}$
RGB-D camera
CNN object detector
Clustering
CNN objects
$J = \{j_\tau, \tau = 1, \cdots, \varphi\}$
Clusters
$W = \{w_\varepsilon, \varepsilon = 1, \cdots, ℶ\}$
Detection
Static features
$A = \{a_\gamma, \gamma = 1, \cdots, \delta\}$
Moving objects
$O = \{o_\rho, \rho = 1, \cdots, \sigma\}$
vSLAM
Removal

**Figure 1: Processes are shown in rounded blocks, and data inputs and outputs are shown in sharp cornered blocks. AMORE extracts 3D features from the RGB and depth image. These features are clustered. The RGB image is used for CNN object detection. The CNN objects and feature clusters are processed in the detection step, where moving and static objects are classified. Static features are provided to the vSLAM algorithm, and moving objects are removed.**

The RGB-D camera produces an image sequence of RGB images and depth images. AMORE is applied to each RGB-D image pair in the sequence. The CNN object detector detects a set of $\varphi$ CNN objects $J = \{j_\tau, \tau = 1, \cdots, \varphi\}$ in the RGB image $L_{RGB}$. Each CNN object has a corresponding 2D bounding box $b_\tau$ defined by $uv$ image pixels. A set of $\beta$ image features $E = \{e_\alpha, \alpha = 1, \cdots, \beta\}$ is extracted and converted to 3D spatial coordinates $e_\alpha = (X_\alpha, Y_\alpha, Z_\alpha)$, with the corresponding depth image $L_D$ [18]. The clustering algorithm clusters the 3D features into a set of ℶ clusters $W = \{w_\varepsilon, \varepsilon = 1, \cdots, ℶ\}$. The clusters help to manage features that occur on the bounding box edges.

The CNN objects and the clusters are processed in the detection step. Clusters are classified as potentially moving if at least one point in the cluster is within the bounding box of a CNN person object. If a potentially moving cluster $l_\omega$ has more than 50 per cent of its points in the person bounding box, $l_\omega \cap b_\tau \geq 50\%$, it is classified as a moving object $o_\rho$, or else the cluster is classified as a static object. This threshold was chosen empirically.

Features are classified according to the object they belong to. If a feature belongs to a static object, it is classified as a static feature $a_\gamma$, or else it is classified as a moving feature. A set of $\delta$ static features $A = \{a_\gamma, \gamma = 1, \cdots, \delta\}$ is provided to the vSLAM algorithm, and a set of $\sigma$ moving objects $O = \{o_\rho, \rho = 1, \cdots, \sigma\}$ is removed.

## 5 EXPERIMENTAL METHODOLOGY

### 5.1 RGB-D data

The performance of ORB-SLAM AMORE is evaluated with the benchmark TUM RGB-D dataset [15, 18]. Ground truth, including the true camera trajectory, measured from an accurate motion capture system, is provided with this dataset. To evaluate ORB-SLAM AMORE fully, static, low-dynamic, and high-dynamic sequences are selected for the experiments. The sequences are abbreviated as Freiburg-fr, halfsphere-half, walking-w, sitting-s, validation-v, and desk-d in the names of the sequences [6]. In the dynamic object sequences, at certain times a large part of the scene is dynamic, making SLAM challenging [6, 18].

### 5.2 Evaluation measures

#### *5.2.1 Standard deviation*

The sample Standard Deviation (SD) [34] with the mean $\bar{x}$ is given by

$$SD = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3}$$

#### *5.2.2 Euclidean distance*

The Euclidean Distance (ED) [23] between the Cartesian points $\boldsymbol{p} = (p_1, p_2, \cdots, p_s)$ and $\boldsymbol{q} = (q_1, q_2, \cdots, q_s)$ in Euclidean $s$-space is given by

$$SED = \sqrt{\sum_{i=1}^{s}(p_i - q_i)^2} \tag{4}$$

#### *5.2.3 Absolute trajectory error*

The global consistency of the camera pose estimates is evaluated using Absolute Trajectory Error (ATE), which is the benchmark evaluation measure for vSLAM algorithms [15]. For a sequence of camera pose estimates $P_1, \cdots, P_N \,\epsilon (SE)^3$ and the corresponding ground truth $Q_1, \cdots, Q_N \,\epsilon (SE)^3$, the Root Mean Square Error (RMSE) of the translational components of ATE over the number $N$ of time indices $I$ is given by

$$RMSE(ATE_{1:N}) = \sqrt{\frac{1}{N}\sum_{I=1}^{N}\|Q_I^{-1}TP_I\|^2} \tag{5}$$

The camera pose estimates are transformed to the ground truth frame with a rigid-body transformation $T$ [15].

### 5.3 ORB-SLAM AMORE implementation

For the experiments in this study, AMORE combines the CNN object detector, YOLOv3 [1], mean shift clustering [7], and the vSLAM algorithm, ORB-SLAM [9]. ORB-SLAM AMORE refers to the combination of these methods.

The ROS Kinetic implementation of YOLOv3 [16] is used with default parameters and with no re-training. The mean shift clustering implementation uses the squared ED measure and a bandwidth of 0.3 [14].

The open-source release of ORB-SLAM [19] in C++ is modified to accommodate AMORE. The core functionality of ORB-SLAM is not changed. It merely calls functions for AMORE, allowing simple interfacing. The number of features detected per frame in ORB-SLAM is increased from 1 000 to 3 000 empirically. This ensures that there are enough static features for ORB-SLAM to initialise, because AMORE removes moving features and only static features are given to ORB-SLAM.

## 6 RESULTS

The experiments were conducted on a computer with an Intel Core i7 — 3970X CPU at 3.5 GHz, with 32 GB of RAM and a GeForce RTX 2080 GPU, using ROS Kinetic on Ubuntu 16.04. The GPU was only used for YOLOv3.

Two sets of experiments were performed. In the first, ORB-SLAM AMORE was compared with the open-source implementation of ORB-SLAM [19]. In the second, ORB-SLAM AMORE was compared with several state-of-the-art, low-cost RGB-D SLAMIDE approaches.

In the first set of experiments, ORB-SLAM and ORB-SLAM AMORE were executed five times on each selected video in the benchmark TUM RGB-D dataset. The video file was played at a full rate for ORB-SLAM and at a slow rate for ORB-SLAM AMORE. This allowed processing time, as AMORE does not presently operate in real time.

Table 1 compares the median of the ATE RMSE, mean, and SD, from the five runs for ORB-SLAM AMORE and ORB-SLAM. In the static and high-dynamic scenes, ORB-SLAM AMORE performed better than ORB-SLAM. ORB-SLAM AMORE had lower ATE than ORB-SLAM, because the moving objects were removed from the SLAM process and performance was improved for the tested datasets. In the low-dynamic scenes ORB-SLAM performed best, although the errors for both were low.

Figures 2—4 show the camera trajectories for one execution of ORB-SLAM, ORB-SLAM AMORE, and the corresponding ground truth for some of the static, low, and high-dynamic sequences respectively. ORB-SLAM AMORE’s trajectories were much closer to the ground truth than those of ORB-SLAM, for the static (Figure 2) and high dynamic environments (Figure 4). However, for the low-dynamic environments (Figure 3), ORB-SLAM was closer.

In the second set of experiments, the performance of ORB-SLAM AMORE was compared with the state-of-the-art approaches: DS [6], SMR [5], DyS(N+G) variant [11], DSS [10], and ORB-SLAM [9]. DS and DSS operate in real-time, unlike ORB-SLAM AMORE, SMR, and DyS. SMR and DyS can detect movement of any object, whereas DS, DSS, and ORB-SLAM AMORE detect people as moving objects.

Table 2 compares the median of the ATE RMSE of ORB-SLAM AMORE with the state-of-the-art approaches. The results for DS, SMR, DyS(N+G), and DSS are given in [6], [5], [11], and [10] respectively, where available. The results for ORB-SLAM and ORB-SLAM AMORE in Table 2 are taken directly from Table 1 for comparison purposes.

In the static scenes and the high-dynamic scenes, ORB-SLAM AMORE performed well. In the high-dynamic scenes DyS(N+G) performed best. In the low-dynamic scenes, ORB-SLAM, DyS(N+G), and SMR outperformed the other methods in terms of localisation accuracy in certain scenes. For the sequences tested, DyS(N+G) had the lowest average error, followed by ORB-SLAM AMORE.

Table 3 shows the average time and SD to run ORB-SLAM AMORE over all images in the fr3/w/xyz sequence for one run. The AMORE time included YOLOv3, mean shift clustering, static and moving object classification, and removal of moving objects. The code for AMORE has not been optimised for real-time performance. Execution speed is mainly limited by the time for mean shift clustering, but may be improved with a GPU implementation [32].

**Table 1: Comparison of ORB-SLAM (OS) and ORB-SLAM AMORE (OA). The video file was executed five times, and the median results are shown.**

| *ATE (m)* | | | | | | |
|---|---|---|---|---|---|---|
| *TUM RGB-D dataset* | *RMSE* | | *Mean* | | *SD* | |
| | *OS* | *OA* | *OS* | *OA* | *OS* | *OA* |
| *High-dynamic* | | | | | | |
| **fr3/w/xyz** | 0.6185 | **0.0209** | 0.4982 | **0.0180** | 0.3664 | **0.0106** |
| **fr3/w/xyz/v** | 1.5703 | **0.0153** | 1.5310 | **0.0135** | 0.3491 | **0.0074** |
| **fr3/w/half** | 0.4410 | **0.0291** | 0.3459 | **0.0250** | 0.2735 | **0.0149** |
| **fr3/w/half/v** | 0.3516 | **0.0294** | 0.3031 | **0.0258** | 0.1781 | **0.0141** |
| **fr3/w/rpy** | 0.8874 | **0.0831** | 0.7884 | **0.0631** | 0.4073 | **0.0541** |
| **fr3/w/rpy/v** | 0.5647 | **0.1180** | 0.4543 | **0.0770** | 0.3355 | **0.0895** |
| **fr3/w/static** | 0.2487 | **0.0236** | 0.2105 | **0.0121** | 0.1325 | **0.0203** |
| *Low-dynamic* | | | | | | |
| **fr3/s/xyz** | **0.0122** | 0.0234 | **0.0108** | 0.0193 | **0.0058** | 0.0132 |
| **fr3/s/half** | **0.0295** | 0.1129 | **0.0246** | 0.0786 | **0.0162** | 0.0811 |
| **fr2/d/person** | **0.0646** | 0.0708 | **0.0615** | 0.0681 | 0.0196 | **0.0195** |
| *Static* | | | | | | |
| **fr2/flowerbouquet** | 0.5529 | **0.0324** | 0.4642 | **0.0322** | 0.3004 | **0.0037** |
| *Average* | 0.4856 | **0.0508** | 0.4266 | **0.0393** | 0.2168 | **0.0299** |

**Table 2: Comparison of ORB-SLAM AMORE (OA) with ORB-SLAM (OS) and other state-of-the art RGB-D SLAMIDE approaches. A dash (—) indicates cases where results were not available.**

| *ATE RMSE (m)* | | | | | | | |
|---|---|---|---|---|---|---|---|
| *TUM RGB-D dataset* | *Method* | | | | | | |
| | *SMR [5]* | *OS* | *DS1 [6]* | *DS2 [6]* | *OA* | *DyS(N+G) [11]* | *DSS [10]* |
| **fr3/w/xyz** | 0.0932 | 0.6185 | 0.0254 | 0.0241 | 0.0209 | **0.015** | 0.0247 |
| **fr3/w/xyz/v** | 0.0655 | 1.5703 | 0.0268 | 0.0218 | **0.0153** | **—** | — |
| **fr3/w/half** | 0.1252 | 0.4410 | 0.2021 | 0.0514 | 0.0291 | **0.025** | 0.0303 |
| **fr3/w/half/v** | 0.0811 | 0.3516 | 0.0697 | 0.0522 | **0.0294** | **—** | — |
| **fr3/w/rpy** | 0.1333 | 0.8874 | 0.4559 | 0.2959 | 0.0831 | **0.035** | 0.4442 |
| **fr3/w/rpy/v** | 0.2333 | 0.5647 | 0.1050 | **0.0767** | 0.1180 | **—** | — |
| **fr3/w/static** | 0.0656 | 0.2487 | — | — | 0.0236 | **0.006** | 0.0081 |
| **fr3/s/xyz** | 0.0470 | **0.0122** | 0.0210 | 0.0201 | 0.0234 | 0.015 | — |
| **fr3/s/half** | 0.0482 | 0.0295 | 0.0273 | 0.0231 | 0.1129 | **0.017** | — |
| **fr2/d/person** | **0.0596** | 0.0646 | 0.0825 | 0.0813 | 0.0708 | **—** | — |
| **fr2/flowerbouquet** | - | 0.5529 | 0.1489 | 0.0569 | **0.0324** | **—** | — |
| *Average* | 0.0952 | 0.4856 | 0.1165 | 0.0703 | **0.0508** | **0.0188** | 0.1268 |

**Table 3: Average time and SD to run each process in ORB-SLAM AMORE. *The times for YOLOv3 and ORB-SLAM were not measured explicitly.**

| *Process* | *AMORE* | | | *ORB-SLAM [8]* | *Total* |
|---|---|---|---|---|---|
| | *YOLOv3 [16]* | *Mean shift clustering* | *Object classification and removal* | | |
| *Time (ms)* | 25.00 | 1146.13 | 0.74 | 33.33 | 1.24 |
| *SD* | * | 0.5113 | 0.0005 | * | * |

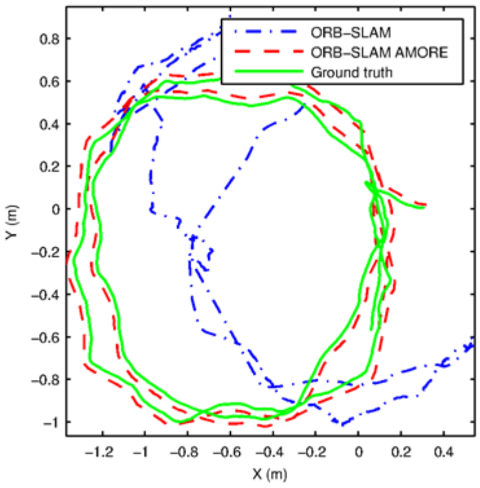
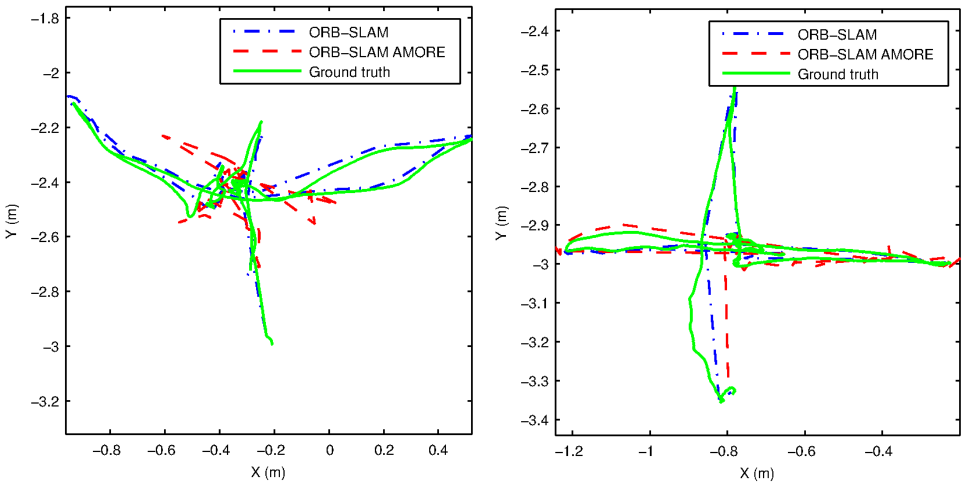
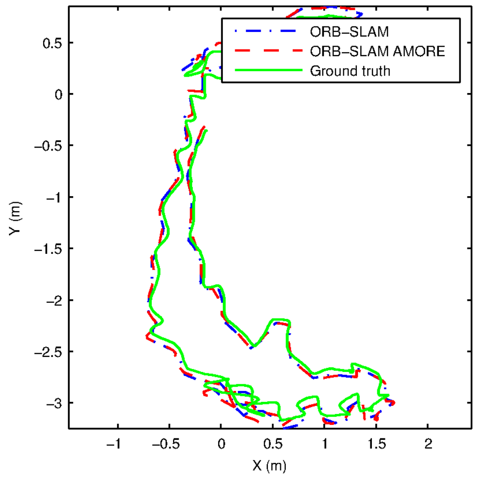


**Figure 2: Static environment — fr2/flowerbouquet**

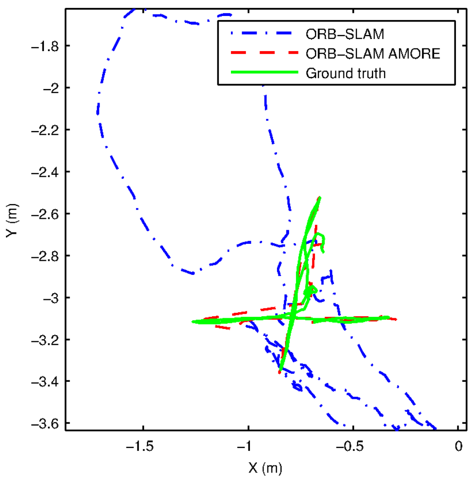


**(a) fr3/s/half** **(b) fr3/s/xyz**

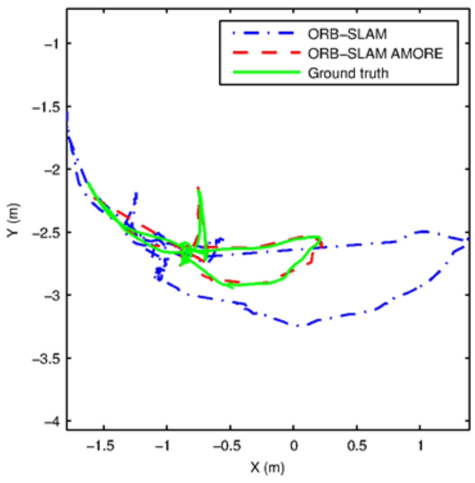


**(c) fr2/d/person**

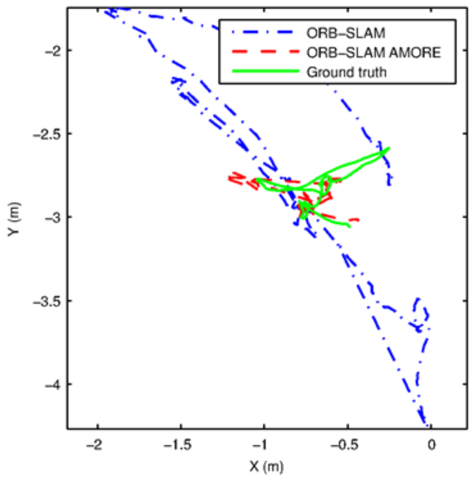
**Figure 3: Low-dynamic environments**



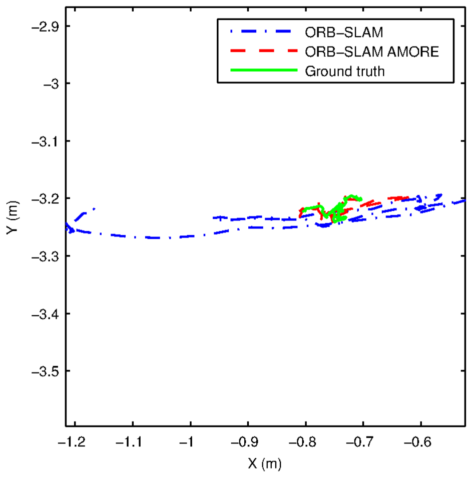
**(a) fr3/w/xyz**



**(b) fr3/w/half**



**(c) fr3/w/rpy**



**(d) fr3/w/static**

**Figure 4: High-dynamic environments**

## 7 DISCUSSION

The accuracy of ORB-SLAM AMORE was comparable with state-of-the-art, low-cost RGB-D SLAMIDE approaches in these experiments, as shown in Table 1 and Table 2. In Table 1, the RMSE, mean, and SD of ORB-SLAM AMORE is high in some scenes owing to the rapid movement and complex trajectory of the camera, although the error was lower than that of ORB-SLAM in most scenes. Overall, ORB-SLAM AMORE performed well in static and high-dynamic scenes, and not as well in low-dynamic scenes. In the low-dynamic scenes, person/s occupied most of the space in the frames of the sequence, and there were fewer true static features for SLAM, making localisation difficult. This is shown by ORB-SLAM AMORE's incomplete trajectory for the fr3/s/half and fr3/s/xyz sequences in Figures 3(a) and 3(b). Figure 3(c) shows that, for the fr2/d/person sequence, ORB-SLAM AMORE's trajectory was close to that of ORB-SLAM and the ground truth. ORB-SLAM performed well in the low-dynamic sequences; therefore there is little room for improvement [5], as seen in the similar errors shown in Table 1 and Table 2 for some of the low-dynamic sequences.

In the fr2/d/person sequence, a person sat at a desk and moved objects on the desk. AMORE was programmed only to recognise persons as moving; and because the other objects moved were not programmed as moving, the performance diminished. Performance might improve if these objects were recognised by AMORE as moving objects.

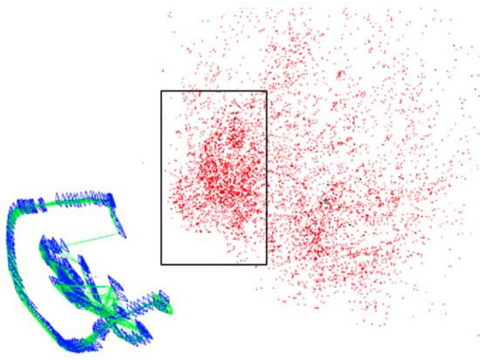
AMORE depends on the CNN to perform object detection. Where the CNN object detector fails to detect a moving object in the image, possibly owing to an unusual camera angle, motion blur, or poor illumination, then features on the moving object will be passed through to the vSLAM algorithm as static features. This can reduce localisation accuracy.

AMORE improved the map quality of ORB-SLAM by removing spurious measurements from moving objects, as shown in Figure 5. AMORE reduced the high-dynamic object motions to low-dynamic changes that ORB-SLAM was able to handle. Without AMORE, more spurious measurements from moving objects would have been included in the map. A better map allows for better localisation.

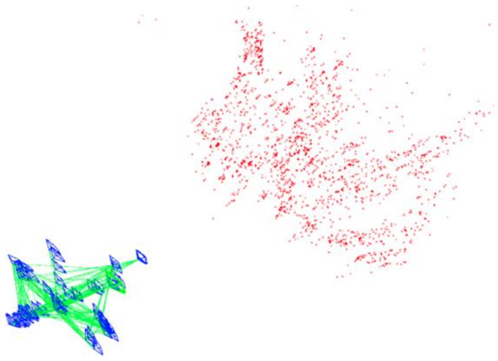
AMORE occasionally misclassified the edges of moving objects as static, as shown in Figure 6(a). However, ORB-SLAM's robustness with moderate dynamic changes allowed it to exclude some of the misclassifications, as shown in Figure 6(b). This means that AMORE may work better with ORB-SLAM, and may perform less well if coupled with another SLAM method that lacks this capability.

At present AMORE only detects people as moving objects, which limits its application to environments where people are the moving objects, such as shopping malls, airports, and laboratories. AMORE's moving object class set could be expanded (by CNN retraining, if needed) to include other moving object classes that may be in a given robot's environment — for example, electric or manual carts, wheeled luggage, in-mall trains, and guide dogs. This would increase AMORE's robustness and versatility in a wider range of applications.

**(a) Image from fr3/w/xyz sequence [15]**

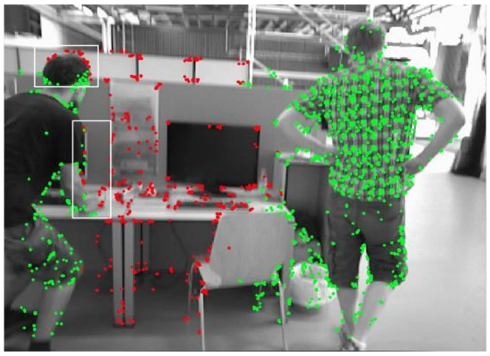


**(b) Map from ORB-SLAM showing spurious measurements from moving objects indicated by box.**

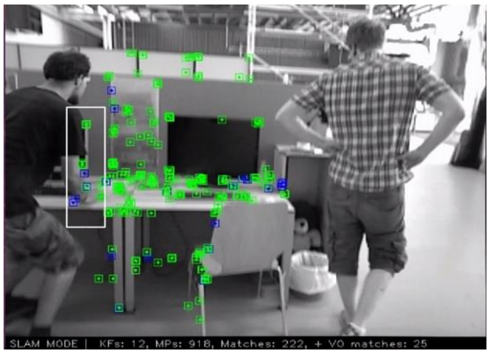


**(c) Map from ORB-SLAM AMORE with fewer spurious measurements from moving objects.**

**Figure 5: Maps from ORB-SLAM and ORB-SLAM AMORE showing map quality for fr3/w/xyz sequence [15]. In (b) and (c), red points indicate features in the map, blue polygons indicate keyframes, and the green line indicates the camera trajectory.**



**(a) Image from fr3/w/xyz [15] sequence showing moving (green) and static (red) features that are detected by AMORE. Some features on the head and hand of the person on the left are misclassified as static, as indicated by white boxes.**


SLAM MODE | KFs: 12, MPs: 918, Matches: 222, + VO matches: 25

**(b) ORB-SLAM's current frame window showing features that are included in the map. The features on the hand of the person on the left are included in the map while the features on the head are not.**

**Figure 6: ORB-SLAM AMORE outputs, with AMORE'S feature classification and ORB-SLAM's current frame, showing how ORB-SLAM's robustness with low dynamic changes does not include all of the moving features.**

## 8 CONCLUSION

AMORE combines a CNN object detector and a clustering algorithm to create a method to remove moving objects from the SLAM process to improve SLAMIDE. ORB-SLAM AMORE is an implementation of AMORE using ORB-SLAM, YOLOv3, and mean shift clustering.

ORB-SLAM AMORE's performance was validated in experiments with the benchmark TUM RGB-D datasets [15]. The results showed that ORB-SLAM AMORE is more robust with high-dynamic objects than ORB-SLAM alone. For the sequences tested, DyS(N+G) had the lowest average error, followed by ORB-SLAM AMORE. The accuracy of ORB-SLAM AMORE in these experiments was comparable with state-of-the-art, low-cost RGB-D SLAMIDE approaches, and it had the advantage of simplicity, requiring minimal implementation effort.

Additional work can be done to improve the performance of AMORE, extend AMORE's moving object class set to allow more applications, and achieve real-time performance — for example, using a GPU for clustering.

## REFERENCES

[1] **Redmon, J. & Farhadi, A.** 2018. YOLOv3: An incremental improvement. *arXiv preprint, arXiv1804.02767*.

[2] **Fu, C-Y., Liu, W., Ranga, A., Tyagi, A. & Berg, A.C.** 2017. DSSD: Deconvolutional single shot detector. *arXiv preprint, arXiv:1701.06659*.

[3] **Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. & Berg, A.C.** 2016. SSD: Single shot multibox detector. *European Conference on Computer Vision (ECCV)*, Amsterdam, Netherlands, 8-16 October 2016, pp. 21-37.

[4] **Scona, R., Jaimez, M., Petillot, Y.R., Fallon, M. & Cremers, D.** 2018. StaticFusion: Background reconstruction for dense RGB-D slam in dynamic environments. *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 20–25 May 2018, pp. 1-9.

[5] **Sun, Y., Liu, M. & Meng, M.Q.H.** 2017. Improving RGB-D SLAM in dynamic environments: A motion removal approach. *Robotics and Autonomous Systems*, 89, pp. 110-122.

[6] **Zhong, F., Wang, S., Zhang, Z. & Wang, Y.** 2018. Detect-SLAM: Making object detection and SLAM mutually beneficial. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Tahoe, USA, 12-15 March 2018, pp. 1001-1010.

[7] **Fukunaga, K. & Hostetler, L.** 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1), pp. 32-40.

[8] **Mur-Artal, R., Montiel, J.M.M. & Tardos, J.D.** 2015. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5), pp. 1147-1163.

[9] **Mur-Artal, R. & Tardós, J.D.** 2017. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5), pp. 1255-1262.

[10] **Yu, C., Liu, Z., Liu, X.J., Xie, F., Yang, Y., Wei, Q. & Fei, Q.** 2018. DS-SLAM: A semantic visual SLAM towards dynamic environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 1-5 October 2018, pp. 1168-1174.

[11] **Bescos, B., Fácil, J.M., Civera, J. & Neira, J.** 2018. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4), pp. 4076-4083.

[12] **He, K., Gkioxari, G., Dollár, P. & Girshick, R.** 2017. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2961-2969.

[13] **Badrinarayanan, V., Kendall, A. & Cipolla, R.** 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI*), 39(12), pp. 2481-2495.

[14] **Pancham, A., Withey, D. & Bright, G.** 2016. Comparison of clustering methods for tracking features in RGB-D images. *IECON 42nd Annual Conference of the IEEE Industrial Electronics Society*, Florence, Italy, 23-26 October 2016, pp. 871–876.

[15] **Sturm, J., Engelhard, N., Endres, F., Burgard, W. & Cremers, D.** 2012. A benchmark for the evaluation of RGB-D SLAM systems. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, 7-12 October 2012, pp. 573-580.

[16] **Bjelonic, M. 2017.** YOLO ROS: *Real-time object detection for ROS*. Available at: https://github.com/leggedrobotics/darknet_ros [Accessed: 31 March 2019].

[17] **ROS.** 2007. *ROS.org*. Available at: http://wiki.ros.org/ [Accessed: 31 March 2019].

[18] **Sturm, J.** 2012. *RGB-D SLAM dataset and benchmark*. Available at: http://vision.in.tum.de/data/datasets/rgbd-dataset/ [Accessed: 31 March 2019].

[19] **Mur-Artal, R., Montiel, J.M.M. & Tardos, J.D.** 2015. *ORB-SLAM2*. Available at: https://github.com/raulmur/ORB_SLAM2 [Accessed: 27 November 2020].

[20] **Durrant-Whyte, H. & Bailey, T.** 2006. Simultaneous localisation and mapping (SLAM): Part I The essential algorithms. *Robotics and Automation Magazine*, 13(2), pp. 99-110.

[21] **Wang, C-C. & Thorpe, C.** 2002. Simultaneous localization and mapping with detection and tracking of moving objects. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Washington, USA, 11-15 May 2002, 3, pp. 2918–2924.

[22] **Pancham, A., Withey, D. & Bright, G**. 2018. Evaluation of a simultaneous localization and mapping algorithm in a dynamic environment using a red green blue—depth camera. *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, 668, pp. 717-724.

[23] **Howard, A.** 1994. *Elementary linear algebra,* 7th ed. New York, USA, John Wiley & Sons.

[24] **Saputra, M.R.U., Markham, A. & Trigoni, N.** 2018. Visual SLAM and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2), pp. 1-36.

[25] **Pancham, A., Tlale, N. & Bright, G.** 2011. Literature review of SLAM and DATMO, *4th Robotics and Mechatronics Conference,* Pretoria, South Africa, 23-25 November 2011, pp. 1-6.

[26] **Pancham, A., Tlale, N. & Bright, G.** 2012. Advancement of vision-based SLAM from static to dynamic environments. *19th International Conference on Mechatronics and Machine Vision in Practice*, Auckland, New Zealand, 28-30 November 2012, pp. 1-6.

[27] **Wolf, D.F. & Sukhatme, G.S.** 2005. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19(1), pp. 53–65.

[28] **Pancham, A. 2019.** Simultaneous localisation and mapping with moving object detection and removal: An RGB-D, CNN-based approach. Thesis, University of KwaZulu-Natal.

[29] **Klein, G. & Murray, D.** 2007. Parallel tracking and mapping for small AR workspaces. *6th International Symposium on Mixed and Augmented Reality*, Nara, Japan, 13-16 November 2007, pp. 225–234.

[30] **Rublee, E., Rabaud, V., Konolige, K. & Bradski, G.** 2011. ORB: An efficient alternative to SIFT or SURF. *International Conference on Computer Vision*, Barcelona, Spain, 6-13 November 2011, pp. 2564–2571.

[31] **Bove, C., Wald, A., Michalson, W., Donahue, M. & LaPenta, J.** 2016. Collaborative robotics heads-up display, Major qualifying project, Massachusetts Institute of Technology, Available at: https://web.wpi.edu/Pubs/E-project/Available/E-project-110416-101803/unrestricted/Bove_Wald_Collaborative_Robotics_Heads-Up_Display.pdf [Accessed: 28 November 2020].

[32] **Li, P. & Xiao, L.** 2009. Mean shift parallel tracking on GPU. *Iberian Conference on Pattern Recognition and Image Analysis*, Póvoa de Varzim, Portugal , 10-12 June 2009, pp. 120-127.

[33] **Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. & Zitnick, C.L.** 2014. Microsoft COCO: Common objects in context. *European Conference on Computer Vision,* Zurich, Switzerland, 6-12 September 2014, pp. 740-755.

[34] **Barber, D.** 2012. *Bayesian reasoning and machine learning*. Cambridge, UK, Cambridge University Press.

[35] **Fukunaga, K. & Hostetler, L.D.** 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory*, 21(1), pp. 32–40.

[36] **Cheng, Y.** 1995. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence,* 17(8), pp. 790–799.
[37] **Comaniciu, D. & Meer, P**. 2002. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence*, 24(5), pp. 603-619.
[38] **Derpanis, K.G**. 2005. Mean shift clustering. Lecture notes. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.217.3313&rep=rep1&type=pdf [Accessed: 28 November 2020].