*The EU Framework Programme for Research and Innovation H2020*
*Research and Innovation Action*

CENTAURO

# *Deliverable D6.2 Object and Workspace Perception*

**Dissemination Level: Restricted until all accompanying papers have been published**

| | |
|---|---|
| Project acronym: | CENTAURO |
| Project full title: | Robust Mobility and Dexterous Manipulation in Disaster Response by Fullbody Telepresence in a Centaur-like Robot |
| Grant agreement no.: | 644839 |
| Lead beneficiary: | LiU – Linköpings Universitet |
| Authors: | A. Robinson, F. Järemo Lawin, A. Milan, M. Schwarz, K. Nordberg, and M. Felsberg |
| Work package: | WP6 Manipulation |
| Date of preparation: | 2016-08-30 |
| Type: | Report |
| Version number: | 1.0 |

**Document History**

| Version | Date | Author | Description |
|---|---|---|---|
| 0.1 | 2016-08-09 | AR and FJL | First draft |
| 0.2 | 2016-08-12 | KN and MF | Second draft |
| 0.3 | 2016-08-21 | AM and MS | Third draft |
| 0.4 | 2016-08-22 | AR, KN, and MF | Beta version |
| 0.6 | 2016-08-26 | AM, MS | Modified beta |
| 0.7 | 2016-08-30 | AR, FJL, and MF | Version for submission |
| 1.0 | 2016-08-31 | MF | Submitted version |

**Executive Summary**

This deliverable describes our approach to object and workspace perception, one essential functionality in the CENTAURO system, consisting of several core components. It covers the following aspects:

- sensor preprocessing,
- detection of objects,
- semantic segmentation,
- tracking of objects,
- object classification,
- object pose estimation, and
- ego-pose in the workspace.

The workspace perception builds the finest scale map of the three-level mapping system in CENTAURO and extends the intermediate scale map from the navigation module. The object perception extracts data about the objects in the workspace. Both perception modules together provide the required data to the modules for visualization and simulation of the world model as well as for manipulation planning and arm motion generation.

# Contents

# 1 Introduction

This CENTAURO Deliverable D6.2: Object and workspace perception is a report summarizing the realized methods for perception of the manipulation workspace of the CENTAURO robot, including object detection and tracking.

It covers activities from Task T6.1: Object and workspace perception, which aims to develop robot perception for bottom-up scene segmentation into objects, for learning models of specific objects, for detecting them, and estimating their pose.

## 1.1 Robot Platform

The CENTAURO robot will carry a set of image and range sensors for perceiving its environment, both for the purpose of navigation, while moving, and for carrying out various assigned manipulation tasks. The detailed specification of the robot platform is still in progress in WP2, but it is envisioned to be similar to the illustration in Figure 1, left.

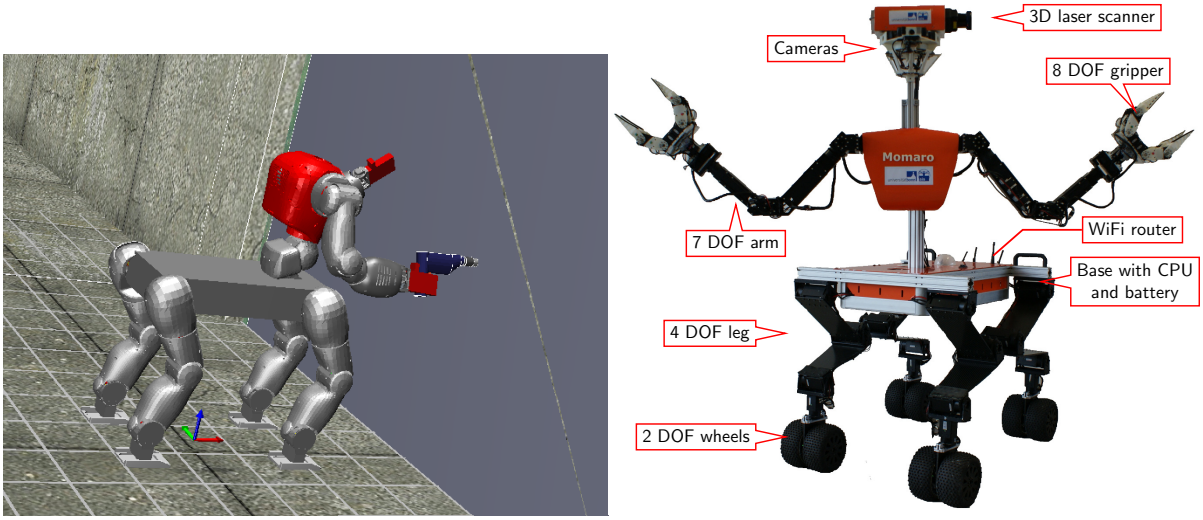



Figure 1: Left: rendering of the CENTAURO robot concept using power tools. Right: The mobile manipulation robot *Momaro* [24], serving as a test platform for navigation and manipulation tasks.

The current design (described in more detail in Deliverables D2.1 and D2.2) can be seen as adaptation of existing robots from other projects to the requirements of the CENTAURO project. Already available at partner UBO is the mobile manipulation robot Momaro [24], serving as a test platform for navigation and manipulation tasks, see Figure 1, right. It has a hybrid mobile base consisting of four pairs of steerable wheels that are connected to the main body by 3DoF legs. Its anthropomorphic upper body has two 7DoF arms that end in 8DoF four finger grippers. The Momaro robot is equipped with a 3D laser scanner, seven color cameras, an RGB-D camera, and a fast CPU.

## 1.2 Sensor Platform

For manipulation tasks the robot is expected to be stationary, but sensors as well as objects in the environment may still be moving relative to the robot platform. Currently, the sensors planned for the CENTAURO robot head include

- One Kinect, version 2. This is an RGB-D sensor, which means that it provides an image that contains both a color measurement and a depth measurement for each pixel. Equipped with pan-tilt mechanism.
- Three Point Grey Blackfly 23S6C-C wide-angle color cameras, mounted vertically with minimal overlap and total field of view of about 100×200 degrees.
- One Velodyne VLP-16, a lidar used mainly for navigation. By rotating the sensor around the vertical axis, a spherical field of view can be achieved.

In addition to the sensors on the head, two Intel SR300 RGB-D cameras are planned to be mounted on each manipulator arm. A potential source of error in the sensor setup could be interference between the Kinect and the Velodyne VLP-16, which both measure time-of-flight using infrared light pulses. However, the Kinect uses 860nm wavelength with a optical bandwidth of 45nm [2] while the Velodyne uses 903nm wavelength [28]. The potential interference should thus be negligible. An intermediate stage of development of the sensor platform for the CENTAURO head, manufactured by partner LiU, is shown in Figure 2.

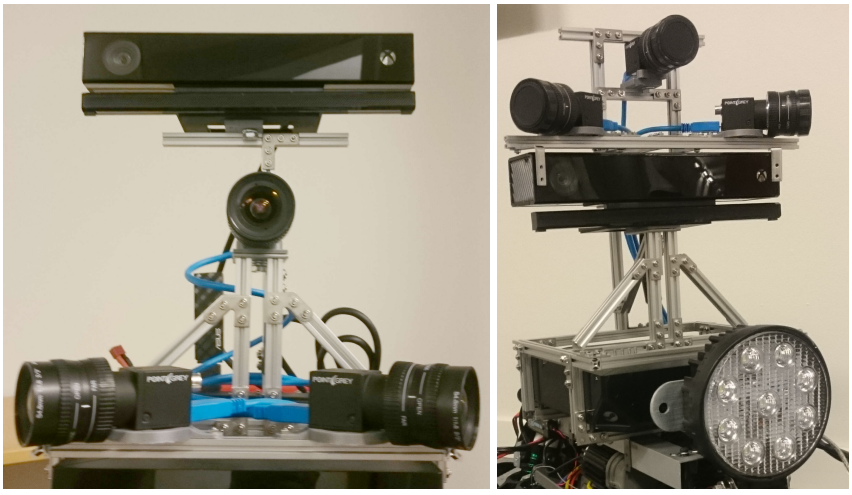


Figure 2: A prototype of the sensor head to be used in the CENTAURO project. Left: the lidar will later be located in place of the Kinect 2 RGB-D camera (top), the Kinect 2 in place of the center Point Grey camera (below), and the center camera further down, nearly between the two side cameras. The Point Grey cameras will be pointing more downwards. Right: optional setup with Kinect 2 moved down and LED lamp for illumination in dark areas.

These sensors have been chosen since they provide the visual information necessary for solving the tasks described in project deliverable D8.1 CENTAURO Evaluation Concept. With respect to object and workspace perception, the Kinect v2 and the three color cameras are primarily relevant. From the raw measurements captured by these sensors, objects and workspace information need to be extracted in order for the different tasks of the robot to be carried out. For workspace perception, also navigation data from the local navigation map (cf. D5.1) may be used for initialization.

Several modules for processing of visual data will be needed and, below, those which relate to perception of object and workspace are outlined in Figure 3. The output from the visual processing comprises colored 3D data registered in a joint coordinate system and detected objects including their pose, to allow visualization and simulation of world models as intended in WP4 as well as manipulation planning and arm motion generation, cf. D6.1.

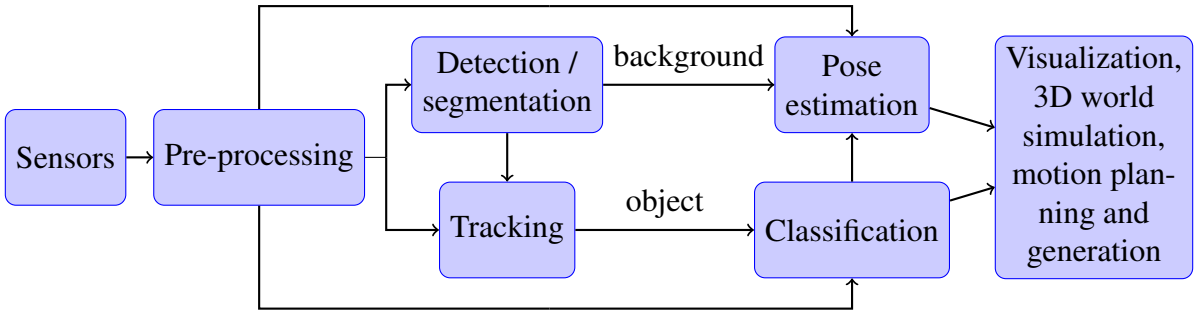



Figure 3: Block diagram of the suggested processing pipeline that includes the functionalities mentioned here. The modules (drawn as boxes) establish core components and auxiliary functionalities of the system.

## 1.3 Visual Processing Pipeline

The visual processing pipeline is illustrated in Figure 3. In this figure, the modules (drawn as boxes) establish core components and auxiliary functionalities of the system.

The sensors listed in the previous section cannot be expected to produce perfect data, for example in terms of accuracy, certainty or range. In particular, this is the case for range sensors, where a pixel can have a missing or unreliable range measurement due to specularities, or a point cloud may be missing parts of the scene due to occlusion. Parts of the performed work addressed also this issue, for example by **extending the measurement range and increasing the accuracy** of the Kinect sensor (cf. Sec. 2.2).

Each of the visual sensors outputs a high bandwidth data stream. Therefore, a primary step to achieve the objectives of object and workspace perception is to reduce the data, which will be done along two processing streams that both are facilitated by **detecting the various objects** of interest for the task. The objects can be specific object categories predefined for the task, for example tools or other objects that should be manipulated, or they could be general objects that are moving, have a particular color or shape, or somehow deviate from the background. The first processing stream involves further object processing for the purpose of manipulation, which can be focused only on these objects once they are detected. The second stream subtracts the objects from the sensor data, leaving the background that instead **forms the workspace** in which the object manipulation takes place.

Once an object of interest has been detected, its own motion in combination with the motion of the sensors makes it likely that the object will appear at different positions relative to the sensors over time. This means that the perspective may vary over time, for example in terms of position in an image, the attention has to be shifted from where the object was first detected. Therefore, it is essential to **track the position** of the object relative to the robot image sensors over time.

Not only the image- and 3D-position of the object may change over time, also its 3D-orientation relative to the robot platform must be known to allow manipulation of the object. Therefore, processing of sensor data to **produce a full 3D-pose estimate** of an object is part of the system. In most cases, classification of objects into different types or categories is a further relevant step to be included into the processing pipeline in order to facilitate planning of manipulation.

Mathematically, object pose estimation and pose estimation in the workspace are the same problems. Thus techniques similar to the object case are also applied to the workspace part of the sensor stream. By registering the **pose of the current workspace** to a local map of the

robot's closest surrounding, this map can be extended, refined, and updated. This map will be visualized to the human operator and can be used in 3D world simulation of planned actions and their effects, cf. WP4 Modeling and Simulation.

The following sections present an overview of the functionalities mentioned above. The technical and scientific details of the documented contributions can be found in the papers in the Appendix.

# 2 Processing of RGB-D Data

Although modern RGB-D sensors provide data in a very convenient way, they require appropriate calibration and additional improvements to achieve high densities of point clouds.

## 2.1 Calibration

Since there are multiple sensors and measurements from two or more of them may be combined in the subsequent processing, it is important that they are calibrated. The calibration can be either relative to a common reference coordinate system of the robot platform, or between a pair of sensors if their data streams are to be fused.

We have developed a method to calibrate the 6D transformations of color cameras relative to the measurements of the 3D laser scanner, based on matching of image brightness to laser reflectance. We define an objective based on mutual information (MI) between the intensity images from the camera and laser intensity values. To find the optimal transformation, the objective is then optimized using hyperopt [3]. Using the calibrated transformation, we can produce colored point clouds, similar to RGB-D sensors, but suitable for outdoor use (Figure 4). Coloring points makes teleoperation based on a head mounted display (HMD) interface more intuitive.

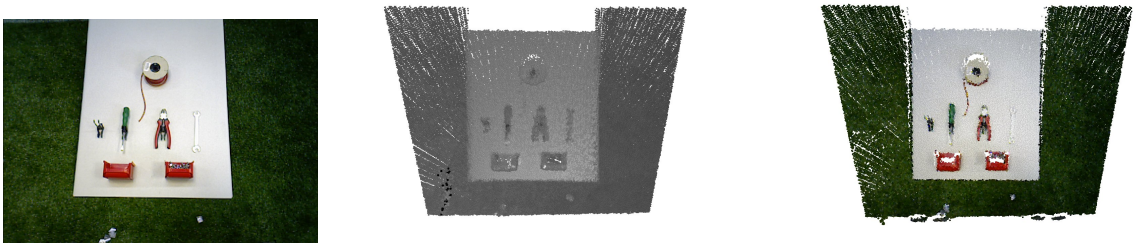


Figure 4: Result of our camera-to-laser calibration. The intensity laser measurements (middle) are correctly colorized with the corresponding RGB image (left) to obtain a colorized point cloud (right).

## 2.2 Improving Time-of-flight Ranging

We use a *Microsoft* Kinect v2 as the primary sensor on the robot platform for object and workspace perception. The Kinect v2 is a RGB-D sensor, providing both color and depth images, where the depth is measured using time-of-flight technology. Points from these images can be projected out into 3D space, constructing colored point clouds, which could be used to interpret the geometry of the environment and its objects.

However, noise in the depth measurements causes the point clouds to contain outliers and reduced point density. The reduced point density is due to outlier rejection performed by the depth decoding algorithm. These flaws occur more commonly in large depth scenes, causing the existing Kinect v2 drivers to limit the range of the sensor to 8 meters.

Hence, by making the depth decoding algorithm less sensitive to noise, not only would we acquire denser point clouds but also be able to use the sensor for larger depth scenes. In our ECCV 2016 paper [18] we introduce a new depth decoding algorithm that has an improvement of about 52% more valid points in large depth scenes compared to the existing methods in the *Microsoft Kinect SDK* and the open source *libfreenect2*. A brief description of our algorithm is presented below, while more details can be found in the full paper included in the appendix.

Time-of-flight sensors, such as the one in the Kinect v2, measure the phase shift between a received and an emitted amplitude modulated light signal. The phase shift depends on the distance to the 3D point were the emitted signal is reflected before returning to the camera. From the phase shift $\phi$ the distance $d$ can be calculated as

$$d = \frac{c\phi}{4\pi f_m}, \tag{1}$$

where $c$ is the speed of light, and $f_m$ is the used modulation frequency of the light signal.

However, since the signal has a periodic modulation the obtained depth is ambiguous for environments with depths larger than $c/(2f_m)$. The procedure for extending the range were the phase measurements are unambiguous is called phase unwrapping. Phase unwrapping is desirable since it enables the sensor to perceive larger depth scenes. One way of performing phase unwrapping is by combining phase measurements from multiple signals with different modulation frequencies. In the Kinect v2 case three different frequencies are used, setting the unambiguous range to 18.75 meters. This is illustrated in Figure 5.

Our main contribution to the improvement of the depth decoding procedure of the Kinect v2 is how the phase unwrapping is performed. The open source driver *libfreenect2*, which is based on a disassembly of the *Microsoft Kinect SDK*, uses a greedy approach to unwrap the phase measurements. This is described in Section 2.2 in [18]. Our method instead considers a set of multiple hypotheses for the phase in each pixel. We then select the hypothesis with the largest kernel density estimate (KDE) over all hypotheses in a spatial region, see Section 3 in [18].

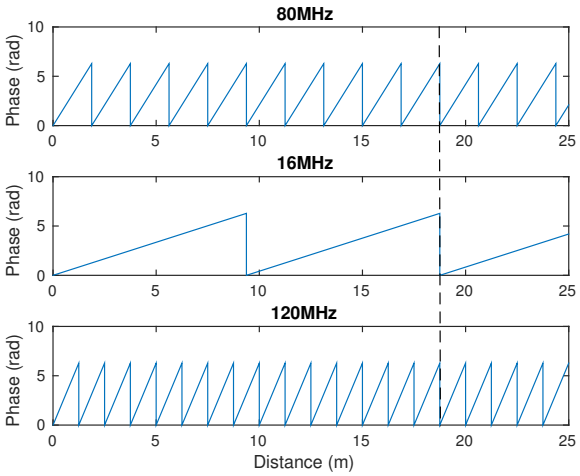



Figure 5: Wrapped phases for Kinect v2, in the range 0 to 25 meters. The dashed line at 18.75 meters indicates the common wrap-around point for all three phases.

Additionally, we utilize the KDE as a confidence measure for detecting and rejecting bad pixels. While *libfreenect2* performs outlier rejection using several steps, where each step has its own threshold and set of parameters to be tuned, our outlier rejections is performed by a single threshold to the KDE.

Qualitative comparisons on a large depth scene, an outdoor scene, and a basement scene from CENTAURO are shown in Figure 6. We see a clear improvement for our method compared with *libfreenect2* for all types of scenes.

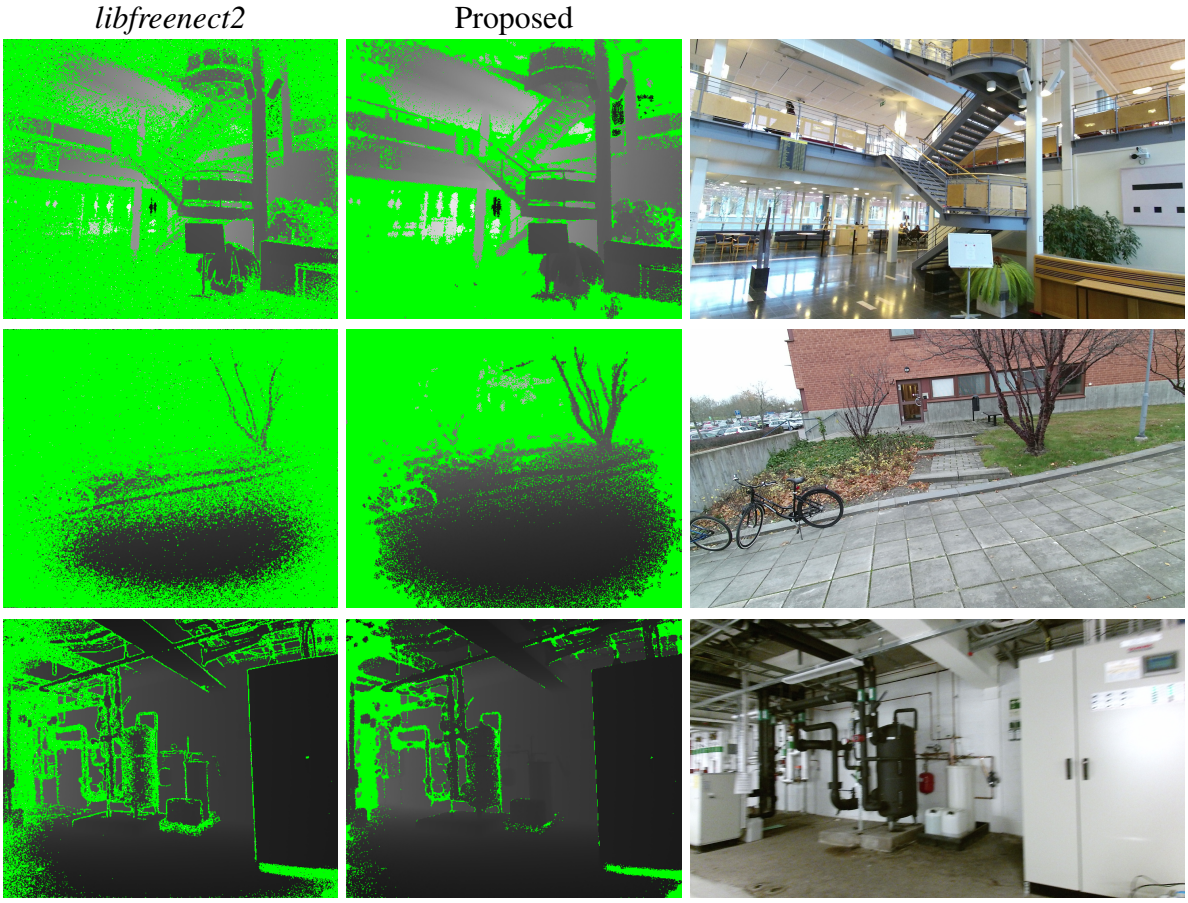



Figure 6: Single frame output comparisons. Top: scene with greater than 18.75m depth range, middle: **outdoor** scene, bottom **basement**. Left: *libfreenect2*, Center: proposed method. Right: corresponding RGB image. Depth measurements are shown in gray scale were bright means far and dark means near. Pixels suppressed by outlier rejection are shown in green. The proposed method has more valid depth points than *libfreenect2* resulting in a denser and more well defined depth scene. While the suppressed areas are clean from outliers for the proposed method, the *libfreenect2* image is covered in salt and pepper noise. Note also that the increase of density is more significant under sunlight conditions.

For quantitative assessment we evaluated the correctness of the phase unwrappings to ground truth depth data. The ground truth data was constructed by fusing multiple views of the same scene, seen in Figure 7. We then compared the output depth from the different algorithms from the same view to the fused depth image to count the number of inliers. Since the depth was limited in the *Microsoft Kinect SDK* we also included a comparison on a scene were all depths were within the limit.

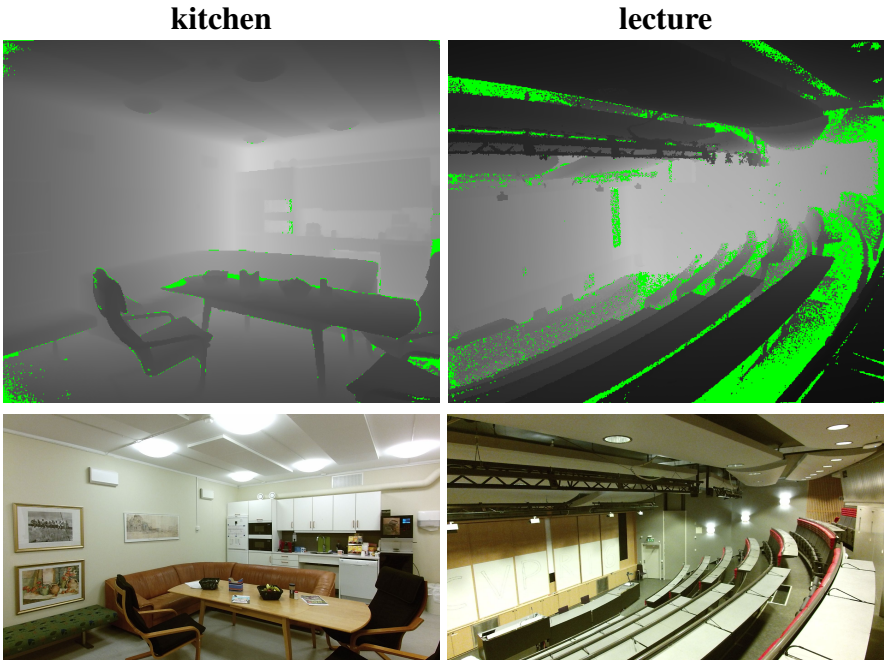



Figure 7: Unwrapping ground truth for the evaluation datasets. Top row: ground truth depth maps. Green pixels are suppressed, and not used in the evaluation. Bottom row: corresponding images from the RGB camera. Taken from [18].

The results shown in Figure 8 demonstrate that our method outperforms *libfreenect2* and *Microsoft*. In the large depth **lecture** scene the proposed method has a 73% inlier rate at a 1% outlier rate, which is an relative improvement of 52% over *libfreenect2*.

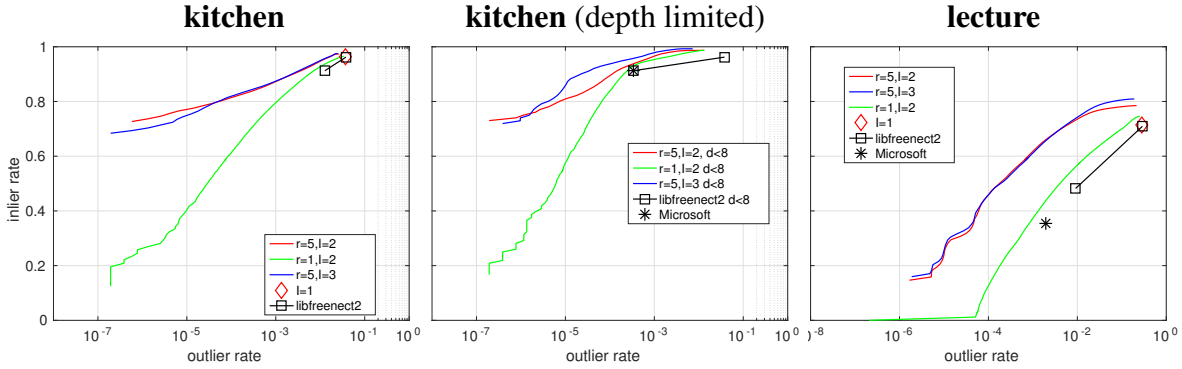



Figure 8: Inlier and outlier rate plots. Each point or curve is the average over 25 frames. The read, green and blue curves shows the performance of our method, depending on the threshold applied the KDE, with different parameter settings. The parameter $r$ defines the maximum distance from the center pixel in a spatial region, $I$ is the number of hypotheses considered for each pixel in the KDE and $d$ is the depth threshold. The center plot **kitchen** (depth limited) shows the result from were the depth was limited for all three methods.

We also applied the algorithm to the Kinect Fusion implementation KinFu in the Point Cloud Library [23]. As can be seen in Figure 9, the scans using our method have a higher quality, generating models with fewer outlier points and consistently more complete scene details.

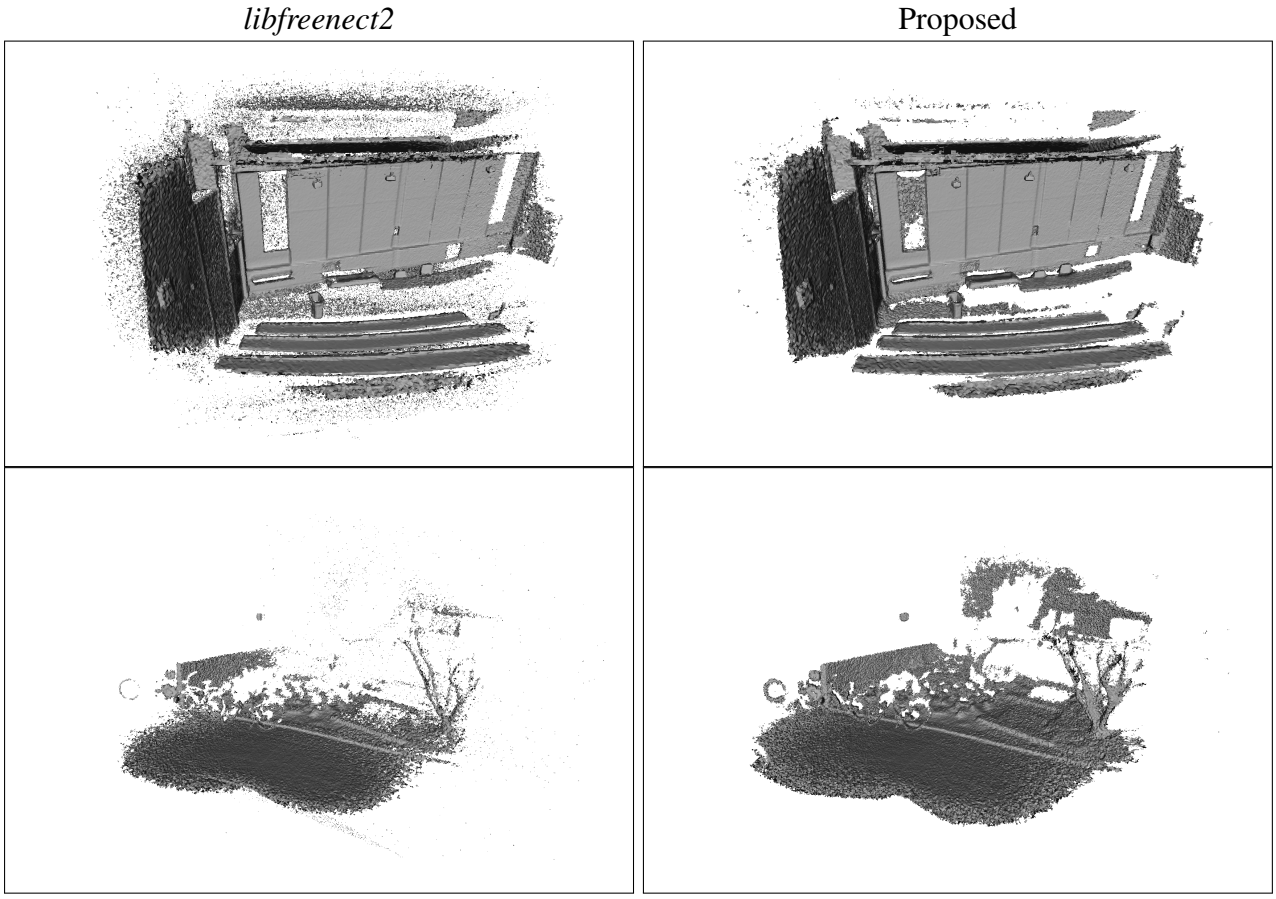



Figure 9: Meshes of **lecture** (top) and the **outdoor** (bottom) scenes from KinFu. See Figures 7 and 6 for corresponding RGB images of the **lecture** and **outdoor** scene respectively. Left: unwrapped with *libfreenect2*. Right: unwrapped with the proposed method.

# 3 Detecting Objects in the Workspace

The identification of objects in the scene, thus separating workspace and objects of interest, can be achieved in different ways, depending on the assumptions that are made. If the object classes and appearances are known before hand, or given from annotation, supervised learning can be used.

## 3.1 Dataset Collection and Annotation

Almost all current methods for object detection and semantic segmentation are based on supervised learning techniques which require labeled training data. Even though there is a large number of public datasets, the data does not represent scenes or objects that are expected to be found in a CENTAURO-like environment. To nevertheless enable high-quality object and workspace perception, we have collected and annotated our own dataset. At the time of this deliverable, there are 129 annotated RGB-D images, six of which contain no objects and serve as background. The data were captured using the Kinect v2 sensor by UBO in its lab. As a starting point, we chose 6 relevant object categories, including wrench, drill, clamp, stapler, extension box and door handle. The annotation is performed manually by outlining each object with a polygon and assigning it the corresponding class. Our annotation tool and three example close-ups are shown in Figure 10.

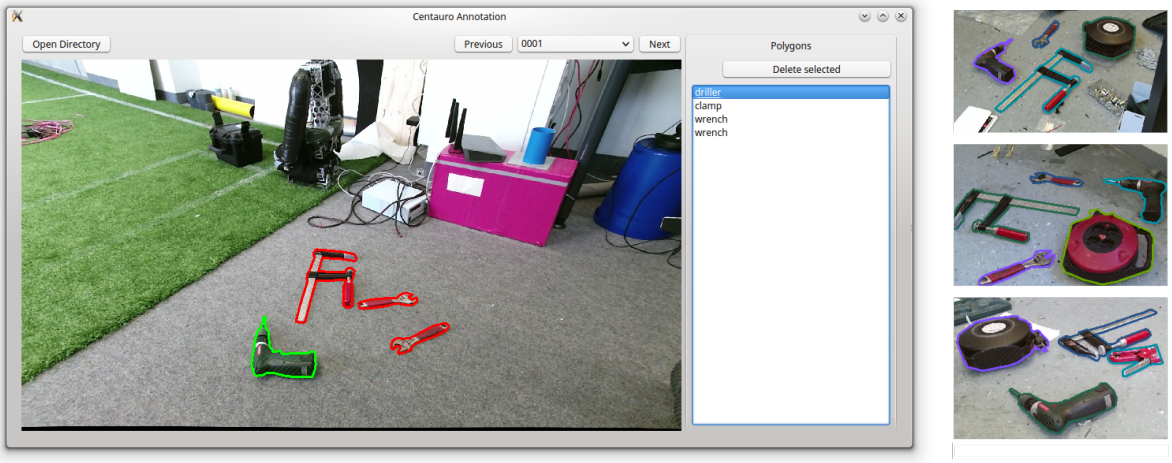



Figure 10: Our annotation tool (left) and three exemplar cropped frames from the captured dataset (right).

We will collect and annotate more data when needed. Moreover, we will explore the option of generating synthetic data both using image based rendering and CAD models.

## 3.2 Object Detection

The sensors on the robot platform provide the system with a large amount of data in the form of image and point cloud representations of the surrounding environment (more than 0.5 gigabyte per second). In order for the robot to perform tasks, such as bimanual manipulation using human tools, we need to detect and extract relevant objects and structures from the sensor data.

### 3.2.1 YOLO

There is a rapid development in image object detection and the advances are generating faster and more accurate methods. At this time, the state of the art methods are machine learning algorithms based on Faster Regional CNN [20]. In the Faster R-CNN methods, a R-CNN is merged with a regional proposal network, while in previous work these two modules where separated. This results in improved performance as well as increased speed, running at 5 frames per second [20]. Although the improvements in the Faster R-CNN methods looks promising, it is still unsuitable for real-time processing that is required in robotic systems such as the CENTAURO robot platform.

One relatively recent method for object detection is YOLO (You Only Look Once) [19]. It accomplishes multiple object detection and classification, not far from the state of the art methods, in 45 frames per second on a Titan X GPU [19]. Examples of the output produced by the YOLO system is shown in Figure 11, presenting a bounding box for each detected object in an RGB image.

The YOLO system consists of a Convolutional Neural Network that divides the input image into a grid. Each grid cell predicts two bounding boxes with conditional probabilities for all pre-trained classes along with a confidence for each bounding box. The bounding boxes with the highest combined confidence and class probabilities are then selected as the output predictions. For more details, see [19].

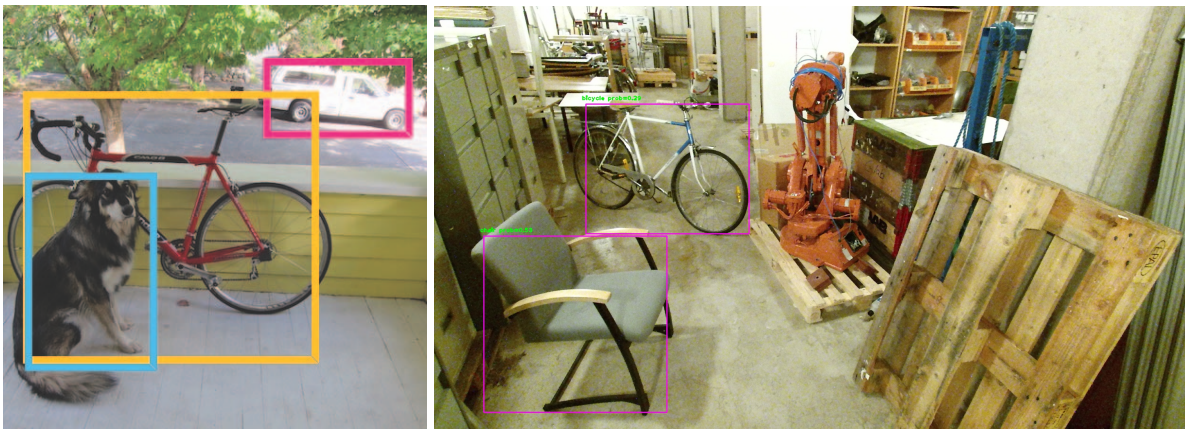


Figure 11: Output images from the YOLO system. Left: A dog, a bike and a car is detected simultaneously and at the same time marking them with bounding boxes. The image is taken from [19]. Right: Output image from a disastrously disorganized basement. The YOLO system detects a chair and a bicycle.

A C implementation of YOLO can be found at `http://pjreddie.com/yolo/`. The net is trained on the PASCAL 2012 dataset [8] with the ability to detect the following 20 Pascal object classes:

- person
- bird, cat, cow, dog, horse, sheep
- airplane, bicycle, boat, bus, car, motorbike, train
- bottle, chair, dining table, potted plant, sofa, TV/monitor.

For CENTAURO, further object classes such as hoses, power-cords, tools, and valves are relevant. To use these classes, the network needs to be retrained on sufficiently large data sets yet to be acquired. A first quantitative evaluation of the retrained network is planned to be part of Deliverable D8.2.

### 3.2.2 DenseCap

As an alternative, we investigate an object detection approach based on the DenseCap network [14]. DenseCap approaches the problem of dense captioning, i.e. providing detailed textual descriptions of interesting regions (bounding boxes) in the input image. Figure 12 shows the general architecture of the DenseCap network. The underlying CNN was pretrained on the ImageNet [22] dataset. Afterwards, the entire pipeline was trained end-to-end on the Visual Genome dataset [16].

While the textual descriptions are not interesting for CENTAURO, the network provides high-quality ranked object proposals with corresponding descriptive features. In order to apply the network to object detection, we add an additional soft-max classification layer and train in on the collected dataset, described in Sec. 3.1 The 123 frames that contain objects are split into 109 training frames, and two times seven frames for validation/testing. The pretrained network is then finetuned end-to-end on the training split. Figure 13 shows preliminary results on the test split. In our experiments, the method copes well with cluttered backgrounds. We achieve a mean average precision (mAP) of 80.92% over five different intersection over union (IoU) thresholds (see Figure 14). The method is currently based on RGB data. Ways to include depth measurements—which will be available in the CENTAURO system—are being investigated.

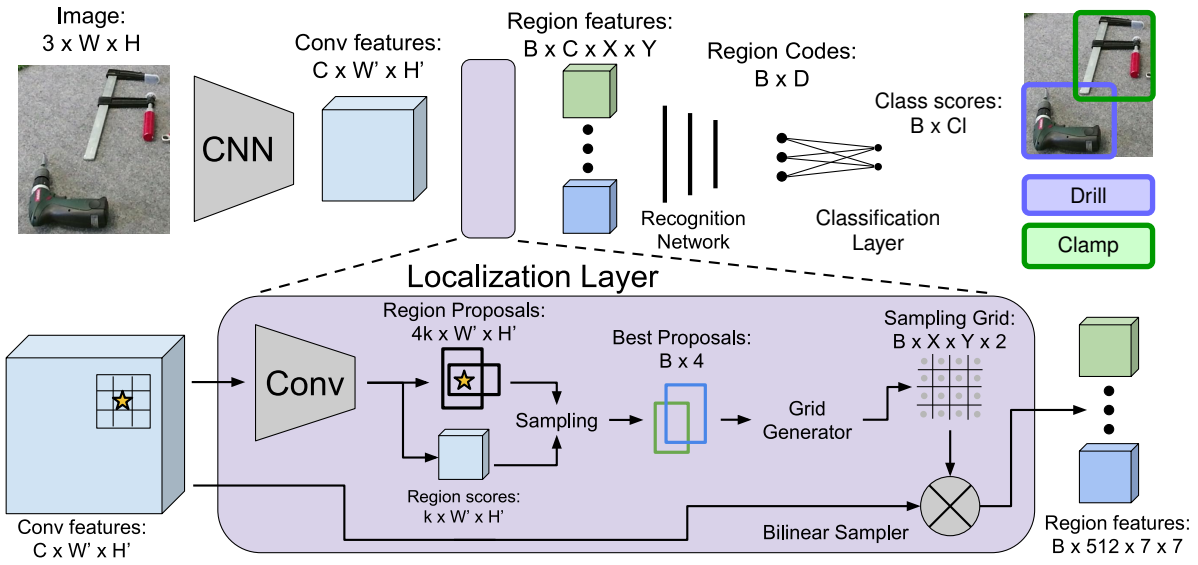



Figure 12: Architecture of the DenseCap detection system. Adapted from [14].


Figure 13: Object detection based on the DenseCap network. The top detections up to the first object hypothesis classified as non-object are shown with corresponding labels and soft-max confidences.

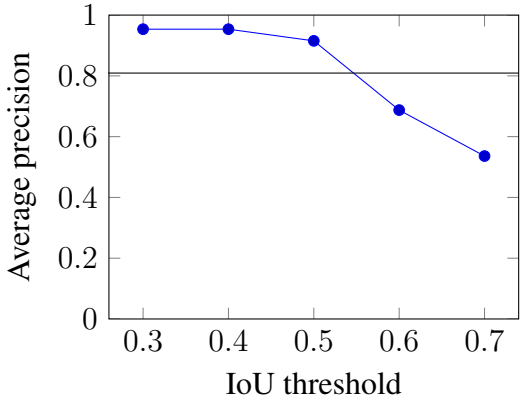



Figure 14: Average precision of the DenseCap object detector over different intersection over union (IoU) thresholds. The mAP is marked with a horizontal line.

## 3.3 Semantic Segmentation

The output of an object detector is typically a bounding box that encloses the object of interest. While this representation is compact and localizes the object sufficiently well for visual recognition, it is not precise enough for manipulation tasks, where the robot has to interact with the physical world. Therefore, in addition to detecting objects in the scene, we also perform pixel-wise segmentation, providing for each pixel the most likely object class.

To that end, we adapt our previous work [12] to the scenario at hand. The method employs a 6-layer fully convolutional neural network (CNN) similar to the OverFeat architecture [25]. The full network architecture is illustrated in Figure 15. As a first step, low-level features are extracted from the captured RGB-D images using a set of filters that was pretrained on ImageNet [21]. We then finetune the network to CENTAURO-related scenes by training the last three layers of the network. To achieve that, we use the same dataset as in Section 3.2.2. Figure 16 shows such a typical scene along with our segmentation result. A quantitative evaluation on the test set yields $98.8\%$ global pixelwise accuracy, which is the ratio of correctly classified object pixels, and $96.9\%$ average per-class accuracy.

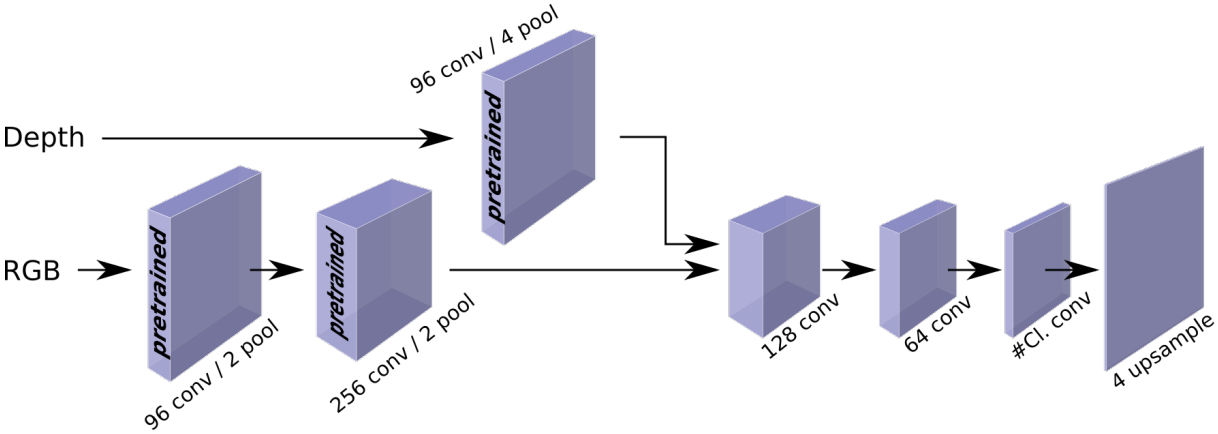
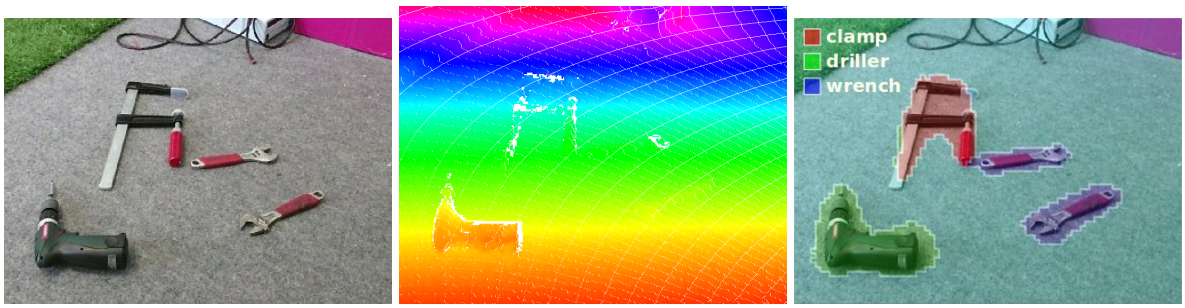



Figure 15: Our network architecture for semantic object segmentation.




(a) RGB (b) Depth (c) Segmentation

Figure 16: Example of our pixel-wise semantic segmentation in a typical setting.

In future, we will also integrate additional geometric features such as height above ground or local normal orientation [10] to further improve the performance. Moreover, we will develop a method generating synthetic training data from both real and rendered images to reduce the burden of manual annotation.

# 4 Object Tracking

If the CENTAURO robot or its manipulator-mounted cameras are moving, it can be computationally less expensive to track an already detected object of interest than to repeatedly re-detect it. Given a bounding box provided by the detector, a visual tracker follows the object as it moves around in the camera view.

Recently, object trackers based on discriminative correlation filters (DCF) have shown very good performance in standard benchmarks and competitions [17, 29]. This kind of tracker employs regression to create an optimal correlation filter and is robust against changes in object appearance. The appearance can be represented by any type of features, like pixel gray values or deep convolutional neural network activations, alone or in combination.

These methods require that all features are sampled in a common coordinate system. This complicates the fusing of features, whether from neural network activations (also known as deep features), from pixels in multiple-sensor cameras such as the Kinect or indeed even from different cameras altogether. The traditional approach is to resample all image features to one common resolution, but this strategy also adds to the computational and memory cost and can introduce artifacts. To counter this, we have developed a DCF-based visual object tracker formulated with continuous rather than discrete filters, where samples are implicitly, rather than explicitly, interpolated [6].

We evaluated our tracker, named the Continuous Convolution Operator Tracker (C-COT), on the Object Tracking Benchmark of 2015 (OTB-2015) [29]. One suitable performance measure is the bounding box overlap, defined as

$$b = \frac{|B_{\mathrm{GT}} \cap B_{\mathrm{T}}|}{|B_{\mathrm{GT}} \cup B_{\mathrm{T}}|}, \tag{2}$$

where $B_{\mathrm{GT}}$ and $B_{\mathrm{T}}$ represent the ground-truth and tracker-estimated bounding-boxes, $|B_{\mathrm{GT}} \cap B_{\mathrm{T}}|$ is the area of their overlap and $|B_{\mathrm{GT}} \cup B_{\mathrm{T}}|$ is the area of their union. Consequently, $b = 1$ when the overlap is perfect and $b = 0$ when the bounding-boxes do not overlap at all. $b$ is aggregated over multiple video frames to form the overlap precision (OP) measure. This is the fraction of images in a sequence where $b$ exceeds a threshold $t \in [0, 1]$. Plotting the OP as a function of $t$, yields the tracker success-rate.

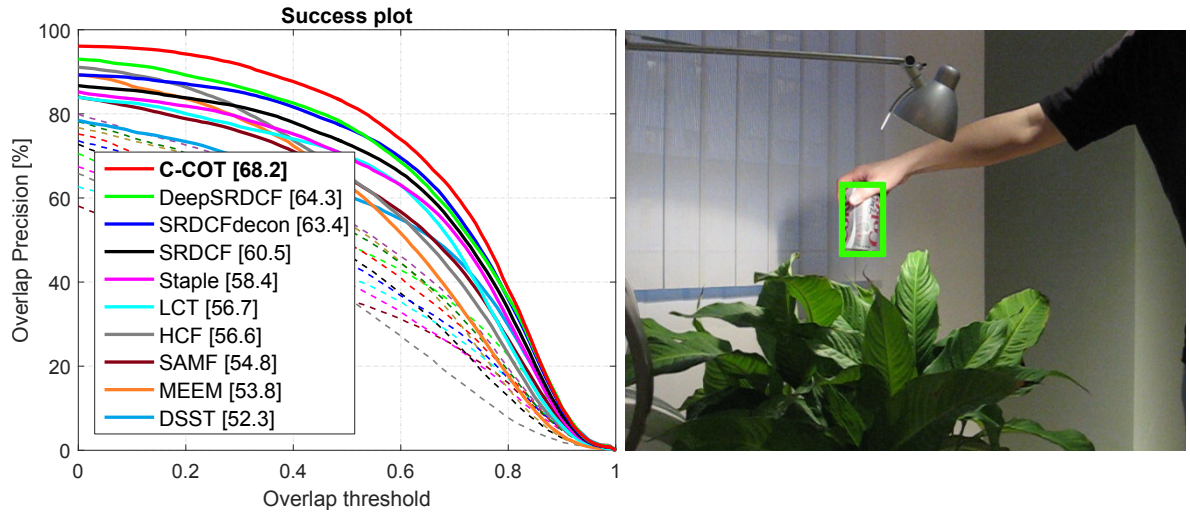



Figure 17: Left: The success rate on the OTB-2015 of our current tracker and nine other state-of-the-art contenders. Right: A frame from a video sequence in OTB-2015, tracking a small hand-held object (a soda can). The tracker bounding box is drawn in green.

The success-rates of our C-COT tracker and nine other methods are shown in Figure 17 where it comes out on top, as ranked by the area under the curve (AUC). The AUC is reported in the legend. In this evaluation we compared the best trackers currently available. For example, the DeepSRDCF tracker which was developed by LiU last year, won the OpenCV Vision Challenge of 2015 in the tracking category and placed second in the 2015 Visual Object Tracking Challenge, being the best DCF-based method. Tracking in CENTAURO scenarios is illustrated in Figure 18, showing frames where the method tracks an industrial robot and a valve.

In addition to tracking whole objects, we have adapted C-COT to track patches of grayscale image pixels with sub-pixel precision, in much the same way as the classic and widely used Kanade-Lucas-Tomasi (KLT) feature tracker. In the CENTAURO project, this may be employed in both structure-from-motion 3D reconstruction of a workspace as well as estimating vehicle motion during navigation.

We tested the tracker and compared it with the KLT and the MOSSE object tracker, on the MPI Sintel dataset. This is a suite of 23 short sequences from the 3D-animated movie ”Sintel”, featuring dynamic scenes, lighting, motion blur and most importantly ground-truth optical flow. In addition, we tested a variant of our tracker (named Ours-FF in the legend) that only remembers the target appearance from the previous frame, rather than continuously adapting to the appearance changes from the first frame in each video sequence.

The comparison is based on the estimated endpoint-error (EPE) of each tracked point in every frame in the dataset. The EPE is the Euclidean distance between the tracked point and its corresponding ground-truth location. Tracked points with an EPE of 3 pixels or less are regarded as inliers. The left plot in Figure 19 shows the EPE distribution over all sequences and tracked points, with the average reported in the legend.

Here, our tracker shows superior accuracy with an average inlier EPE of 0.449 pixels vs the KLT average EPE of 0.773 pixels. The center plot in Figure 19 shows a measure of the method’s robustness, plotting the fraction of endpoint errors below a varying threshold. The fraction at the threshold of 3 pixels, i.e. the inlier ratio, is reported in the legend. Again, our tracker is significantly more robust than KLT with an inlier ratio 0.886 vs 0.773. The lower endpoint error of our tracker relative to the KLT should enable more precise sparse 3D reconstructions. In the CENTAURO context, this could for example be useful in bright daylight, where the RGB-D sensor on the robot is expected to function poorly.

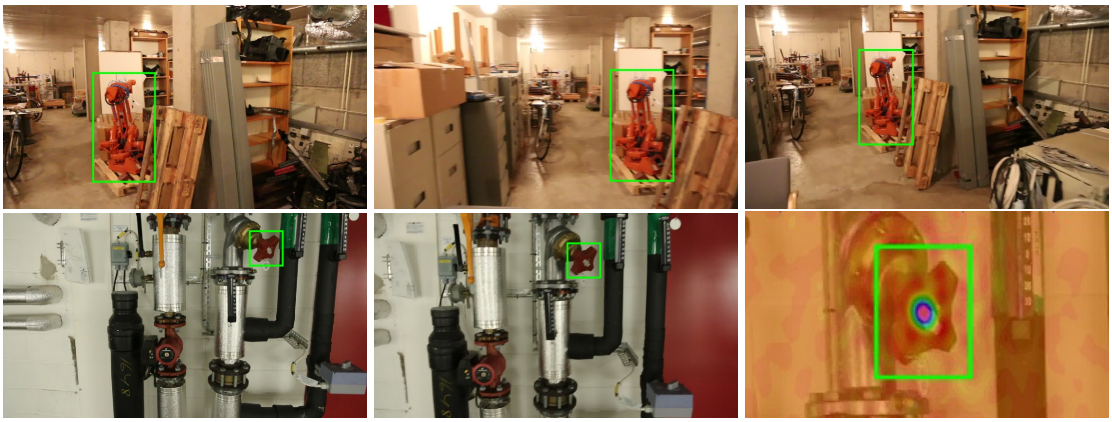


Figure 18: Top: Frames from a sequence of tracking a large object - the original and archetypical ASEA industrial robot. Bottom: Frames from a sequence tracking a small object of interest - a valve in this case. The tracker bounding box is drawn in green. The rightmost bottom frame shows the tracker’s confidence function of seeing the target object.

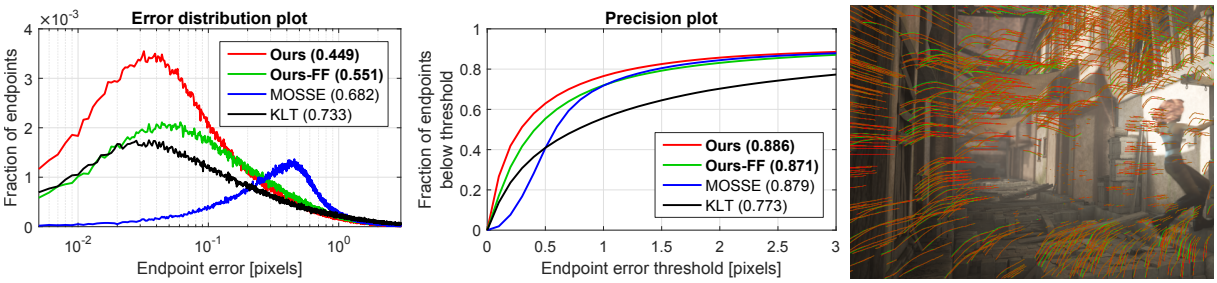



Figure 19: Left: distribution of endpoint errors, Center: robustness, Right: one frame from the MPI Sintel dataset, with estimated point tracks and ground-truth point tracks overlaid in green and red, respectively.

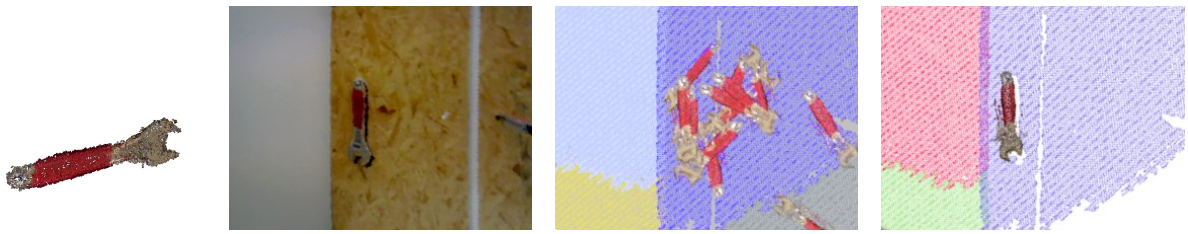
Detailed descriptions of our method and the experiments are available in our paper [6]. Future research directions include incorporating motion-based features that are derived from optical flow [9].

# 5 Object Pose- and Workspace Estimation

Once an object of interest is successfully classified and tracked, its 6D pose (3D position and 3D orientation) must be determined before it can be grasped in a task-specific way. With the objects removed from the sensor data, the remaining points are considered as the background which forms the workspace of the robot. By registering each new set of workspace data relative to a local map of the robot's close environment, this map can be extended, refined or updated.

## 5.1 Grasping Pose Estimation

In order to determine the optimal grasping pose, it is essential to not only localize the object we want to manipulate, but also to estimate its 6D pose with respect to the robot. Our method for pose estimation is based on the approach by Aldoma *et al.* [1] and is illustrated in Figure 20. It learns a 3D colored model of the object of interest by registering RGB-D views of it that are recorded on a turn table. Multiple detection hypotheses are tested in the current view of the scene and the 3D model is registered to the best detection. This yields a 6D pose of the detected object.



(a) 3D model (b) RGB-D input (c) Proposals (d) Registered

Figure 20: Object pose estimation of an adjustable wrench. A 3D model *(a)* is sampled in different locations and orientations *(c)* and the best fit is registered to the current view *(d)*.

## 5.2 3D-Registration of Object and Workspace Data

A generic alternative approach is to convert the output of the RGB-D sensor into a point cloud, register it against a prototype of the detected object with a known pose (also stored as a point cloud) and finally compute the pose from the estimated rigid transformation. In the case of workspace registration, the pose is instead computed relative to the robot's local workspace map.

The typical approach to point-cloud registration is the Iterative Closest Point (ICP) method. However, probabilistic methods have recently produced state-of-the-art results [7, 11, 27, 26]. These methods model the point clouds as density distributions and use either correlation to maximize a similarity measure between the two point clouds or expectation-maximization to jointly estimate the point distributions and registration transform.

However, despite the existence of both RGB-D and color-LIDAR sensors, color is largely ignored in these probabilistic approaches (with exception [26]) and in order to exploit their full functionality, we developed a new expectation-maximization based method [5] taking color into account.

In this work the cloud point coordinates are clustered and modeled as a mixture of Gaussians, following [7], while jointly representing the color distribution within each cluster with a secondary Gaussian mixture model (GMM). An example of this setup is shown in Figure 21. With this model, both the coordinate clusters and the relative rigid transform between two (or more) point clouds can then be found through expectation maximization. An example with CENTAURO data is shown in Figure 22, where four point clouds of the same scene are jointly registered.

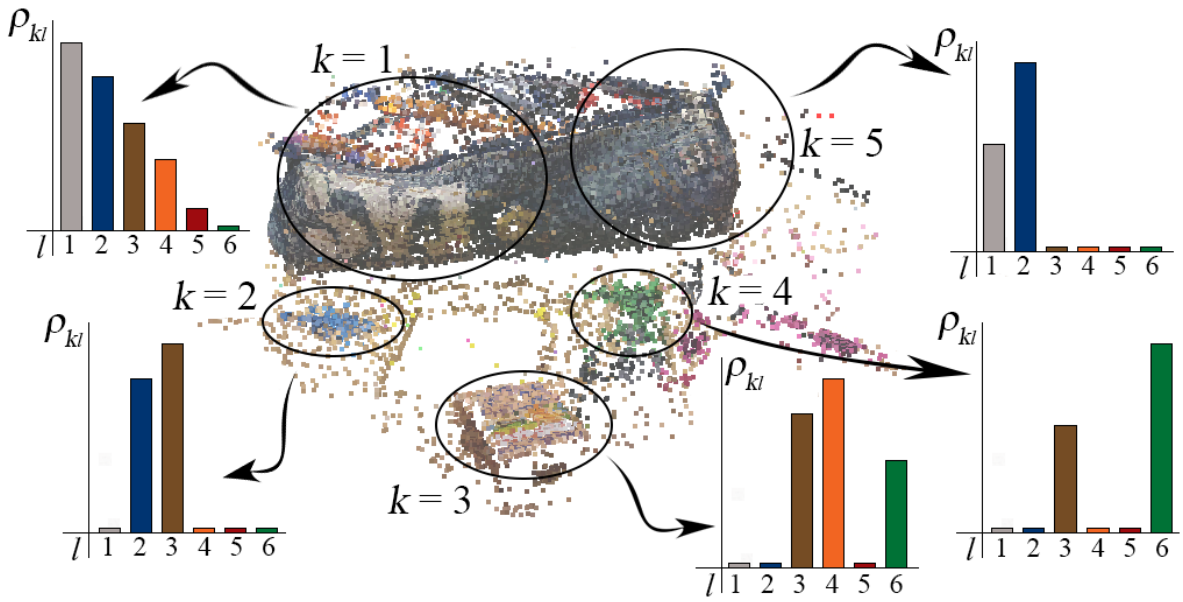



Figure 21: An illustration of our mixture model of the joint point-color space (from [5]). Each spatial component k is associated with a mixture model in the color space, here visualized as histograms. The point cloud is depicting a soft pencil pouch (top) and a few other items on a desk.

We perform quantitative experiments on the Stanford Lounge dataset [30], consisting of 3000 RGB-D images. A sample image is shown in Figure 23.

Figure 22: Registration of point clouds in a CENTAURO scenario. Left: some industrial machinery and shelves. Right: A point cloud of the same scene after registration. The camera frustums are outlined in red.

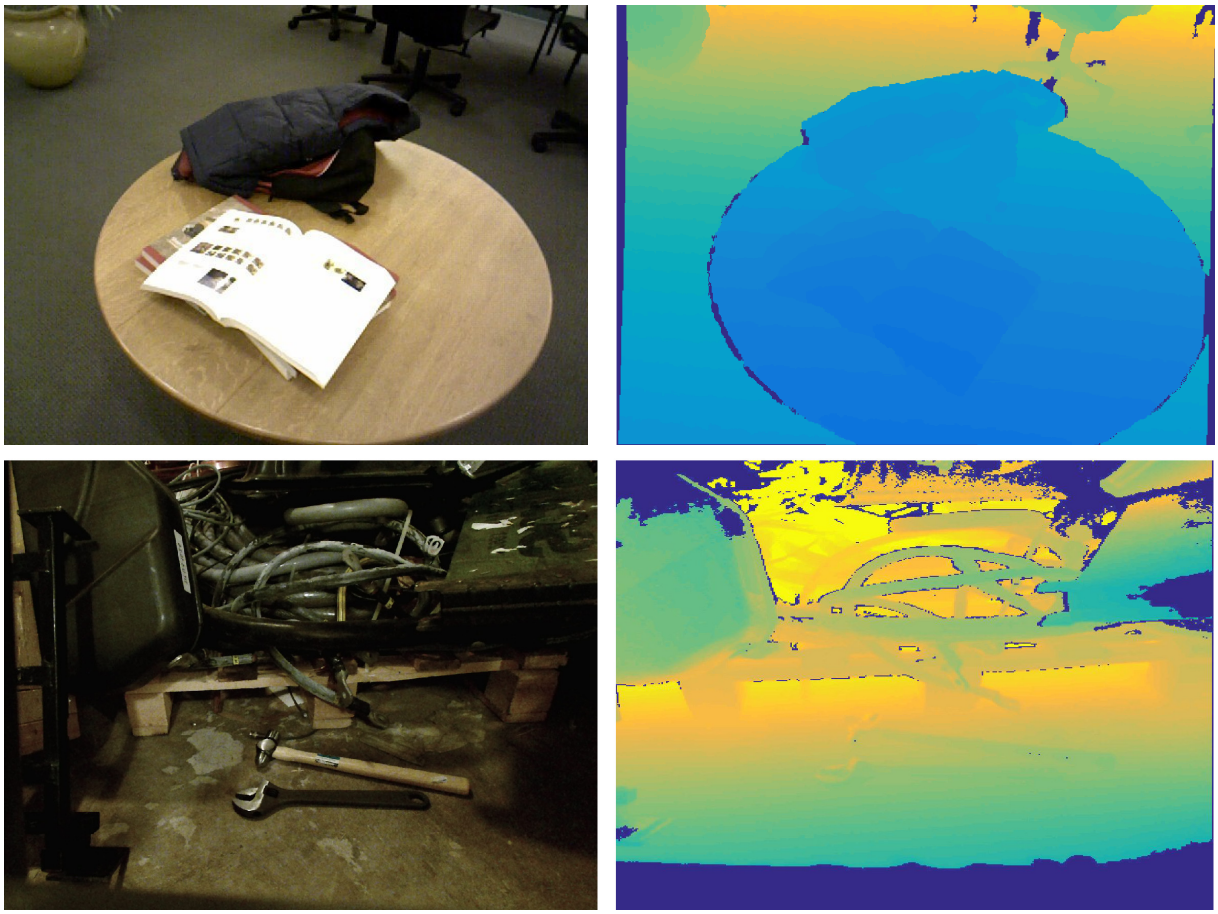


Figure 23: Example RGB and depth image pair from the Stanford Lounge dataset (top) and a CENTAURO workspace (bottom). Once the pipeline with detection and registration is integrated, objects such as the hammer and the wrench will be masked out from the workspace (background).

We compare our method to standard ICP, color-supported generalized ICP (GICP) [15], two probabilistic registration methods, GMMReg [13] and JRMPS [7] as well as two alternatives to merging color and spatial information: the "direct" method of concatenate the point coordinates and colors in a single vector, and the "independent" method where colors and coordinates are treated as stochastically independent. The results in Table 1 show that our method achieves significantly lower rotation error and failure rate. The rotation error is defined as the Frobenius distance between two rotation matrices, i.e. $||\hat{R} - R||_F$ while failure rate is the percentage of rotation errors larger than 0.1 (approximately 4 degrees), with a good margin sufficient to reliably grasp objects with the CENTAURO robot.

A more detailed account and more experiments, including tests on color LIDAR point clouds, are found in [5]. In future research we intend to extend our model with more sophisticated, structural features such as local shape information [4].

| | Avg. error | Std. dev. | Failure rate (%) |
|---|---|---|---|
| ICP | $4.32 \cdot 10^{-2}$ | $2.53 \cdot 10^{-2}$ | 15.70 |
| GMMReg [13] | $6.09 \cdot 10^{-2}$ | $2.31 \cdot 10^{-2}$ | 59.04 |
| Color GICP [15] | $1.72 \cdot 10^{-2}$ | $1.75 \cdot 10^{-2}$ | 1.27 |
| JRMPS [7] | $1.68 \cdot 10^{-2}$ | $1.24 \cdot 10^{-2}$ | 3.41 |
| Direct Approach | $1.91 \cdot 10^{-2}$ | $1.30 \cdot 10^{-2}$ | 2.14 |
| Independent Approach | $1.68 \cdot 10^{-2}$ | $1.24 \cdot 10^{-2}$ | 3.41 |
| **Our Approach** | $\mathbf{1.47} \cdot 10^{-2}$ | $\mathbf{1.01} \cdot 10^{-2}$ | **0.74** |

Table 1: A comparison with other registration methods on the Stanford Lounge dataset. Quantitative results on CENTAURO data will be presented in the forthcoming Deliverable D8.2.

# 6 Future Work

The next steps in the project work related to Task 6.1 can be summarized as

- Evaluate the core components according to the evaluation plan.
- Complete the pipeline that contains the described functionalities.
- Evaluate the pipeline.
- Integrate the pipeline into the robot platform.
- Connect to WP4 and motion modules from WP6.
- Evaluate the system.

Regarding the first item, at least one method for each core component has been suggested. To which extent the chosen methods are sufficient will be tested in the component evaluations (D8.2). As in any research project, it is expected that after the evaluation steps some of the current functionality may have to be modified, extended, or even replaced. For the pipeline integration, the chosen methods need to be adapted to or reimplemented for the CENTAURO system architecture.

# Appendix

The following papers are enclosed below in the order of their appearance in the report:
[4, 5, 6, 9, 18]

# References


[1] A. Aldoma Buchaca, F. Tombari, J. Prankl, A. Richtsfeld, L. di Stefano, and Markus Vincze. Multimodal cue integration through hypotheses verification for rgb-d object recognition and 6dof pose estimation. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA 2013)*, 2013.

[2] Cyrus S. Bamji et al. A 0.13 $\mu$m CMOS system-on-chip for a $512 \times 424$ time-of-flight image sensor with multi-frequency photo-demodulation up to 130 MHz and 2 GS/s ADC. *IEEE Journal on Solid-State Circuits*, 50(1), January 2015.

[3] James Bergstra, Daniel Yamins, and David D Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *ICML (1)*, 28:115–123, 2013.

[4] Martin Danelljan, Giulia Meneghetti, Fahad Shahbaz Khan, and Michael Felsberg. Aligning the dissimilar: A probabilistic method for feature-based point set registration. In *23rd International Conference on Pattern Recognition*, 2016. Accepted.

[5] Martin Danelljan, Giulia Meneghetti, Fahad Shahbaz Khan, and Michael Felsberg. A probabilistic framework for color-based point set registration. In *CVPR*, 2016.

[6] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*. Springer, 2016. To appear.

[7] Georgios D Evangelidis, Dionyssos Kounades-Bastian, Radu Horaud, and Emmanouil Z Psarakis. A generative model for the joint registration of multiple point sets. In *European Conference on Computer Vision*, pages 109–122. Springer, 2014.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[9] Susanna Gladh, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Deep motion features for visual tracking. In *23rd International Conference on Pattern Recognition*, 2016. Accepted.

[10] Saurabh Gupta, Ross Girshick, Pablo Arbelaez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*. 2014.

[11] Radu Horaud, Florence Forbes, Manuel Yguel, Guillaume Dewaele, and Jian Zhang. Rigid and articulated point registration with expectation conditional maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):587–602, 2011.

[12] F. Husain, H. Schulz, B. Dellen, C. Torras, and S. Behnke. Combining semantic and geometric features for object class segmentation of indoor scenes. *IEEE Robotics and Automation Letters*, 2(1):49–55, May 2016.

[13] Bing Jian and Baba C Vemuri. Robust point set registration using gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1633–1645, 2011.

[14] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. DenseCap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] Michael Korn, Martin Holzkothen, and Josef Pauli. Color supported generalized-icp. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, volume 3, pages 592–599. IEEE, 2014.

[16] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*, 2016.

[17] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–23, 2015.

[18] Felix Järemo Lawin, Per-Erik Forssén, and Hannes Ovrén. Efficient multi-frequency phase unwrapping using kernel density estimation. In *European Conference on Computer Vision (ECCV)*, Amsterdam, October 2016. Springer International Publishing AG. To appear.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. 2014.

[22] 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.

[23] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE ICRA*, May 9-13 2011.

[24] Max Schwarz, Tobias Rodehutskors, David Droeschel, Marius Beul, Michael Schreiber, Nikita Araslanov, Ivan Ivanov, Christian Lenz, Jan Razlaw, Sebastian Schüller, David

Schwarz, Angeliki Topalidou-Kyniazopoulou, and Sven Behnke. Nimbro rescue: Solving disaster-response tasks through mobile manipulation robot momaro. *Journal of Field Robotics*, 2016.

[25] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[26] Jörg Stückler and Sven Behnke. Multi-resolution surfel maps for efficient dense 3d modeling and tracking. *J. Vis. Comun. Image Represent.*, 25(1):137–147, January 2014.

[27] Yanghai Tsin and Takeo Kanade. A correlation-based approach to robust point set registration. In *European conference on computer vision*, pages 558–569. Springer, 2004.

[28] Velodyne LiDAR. *Velodyne LiDAR Puck: Real time 3D LiDAR Sensor*, 2016.

[29] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[30] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics (TOG)*, 32(4):112, 2013.

# Aligning the Dissimilar: A Probabilistic Method for Feature-Based Point Set Registration

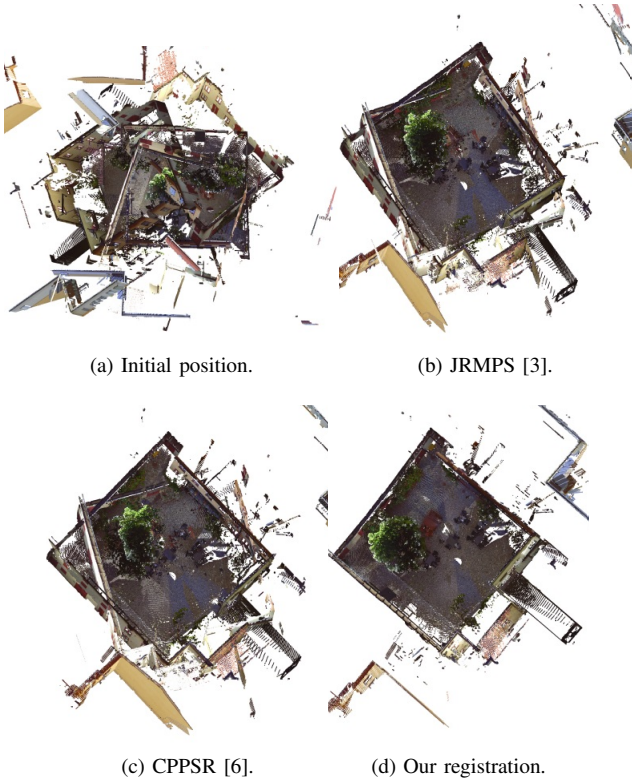
Martin Danelljan[1], Giulia Meneghetti[1], Fahad Shahbaz Khan, Michael Felsberg
Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, Sweden
Email: {martin.danelljan, giulia.meneghetti, fahad.khan, michael.felsberg} @ liu.se

***Abstract*—3D-point set registration is an active area of research in computer vision. In recent years, probabilistic registration approaches have demonstrated superior performance for many challenging applications. Generally, these probabilistic approaches rely on the spatial distribution of the 3D-points, and only recently color information has been integrated into such a framework, significantly improving registration accuracy. Other than local color information, high-dimensional 3D shape features have been successfully employed in many applications such as action recognition and 3D object recognition. In this paper, we propose a probabilistic framework to integrate high-dimensional 3D shape features with color information for point set registration. The 3D shape features are distinctive and provide complementary information beneficial for robust registration. We validate our proposed framework by performing comprehensive experiments on the challenging Stanford Lounge dataset, acquired by a RGB-D sensor, and an outdoor dataset captured by a Lidar sensor. The results clearly demonstrate that our approach provides superior results both in terms of robustness and accuracy compared to state-of-the-art probabilistic methods.**

## I. Introduction

Registration of 3D-point sets is a challenging problem in computer vision, with many potential applications, such as scene reconstruction, robotics, and 3D object localization. The registration problem involves estimating the relative rigid transformations between two or more point sets. In recent years, the probabilistic registration methods have demonstrated promising results on challenging datasets. These approaches model the distribution of 3D-points as a density function. The registration is then performed by either maximizing a similarity measure between the density models [1], [2], or applying the Expectation Maximization (EM) algorithm to iteratively find the registration parameters [3], [4], [5], [6].

Initially, most existing probabilistic registration approaches only utilized the spatial distribution of the 3D-points. Recently, feature information, such as color, has been integrated in such a probabilistic framework [6]. The integration of local color information is performed by constructing a density model of the joint spatial-color space. Other than local color information, high-dimensional 3D shape features, e.g. 3D-SIFT [7] and PFH [8], have been successfully employed in many applications such as action recognition [9] and 3D object recognition [10]. The 3D features extract information from a local neighborhood in the point set, and are therefore distinctive and posses high discriminative power. Additionally, when color is insufficient, 3D shape information provides complementary information beneficial for such registration tasks. However, the problem of *integrating* high-dimensional shape features and *fusing* them with color information are yet to be investigated for probabilistic point set registration.



(a) Initial position. (b) JRMPS [3]. (c) CPPSR [6]. (d) Our registration.

Fig. 1. Registration of four different Lidar scans (a) of an outdoor scene. The two state-of-the-art GMM-based methods JRMPS [3] (b) and CPPSR [6] (c) fail to register the point sets due to severe occlusions and non-uniform point density. Our method (d) accurately registers the sets by exploiting high dimensional shape features in the registration.

The recently introduced color-based probabilistic framework [6] efficiently models the distribution of feature observations with a mixture model in the color space. However, this strategy cannot be directly employed for 3D shape features due to their high dimensionality. In the context of object recognition, high-dimensional features are quantized by learning a task-specific codebook, to obtain a bag-of-words (BOW) based probabilistic representation. In this work, we propose a probabilistic

[1]Both authors contributed equally to this work.

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 22, 2016.

representation of the feature space, that is reminiscent to the BOW methodology, for integrating high-dimensional shape features. In our approach, the feature space is first quantized by clustering based on the feature descriptors extracted from the point sets. This leads to a compact feature representation that is data adaptive. The clustered shape representation then serves as a basis for estimating the local feature distributions in the point set.

As discussed above, the 3D shape features are expected to provide improved registration performance in terms of robustness due to their distinctiveness and high discriminative power. On the other hand, local information, such as color observations, have a high spatial resolution, leading to improved accuracy [6]. A careful strategy when integrating 3D shape information is necessary for improved robustness, while preserving the accuracy of the registration. In this work, we tackle this problem by introducing an adaptive fusion strategy for robust point set registration.

**Contributions:** In this paper, we propose to integrate high-dimensional 3D-shape features in a probabilistic framework for point set registration. To construct a compact probabilistic representation of the feature space, we introduce an efficient model that is reminiscent to the popular bag-of-words paradigm. To aid the registration process, our approach jointly estimates the local feature distribution parameters in an EM-framework. We further introduce an adaptive strategy for integrating the shape features with color information to obtain improved robustness, while maintaining the accuracy.

We evaluate our approach on two challenging datasets: one indoor scene captured by an RGB-D sensor and one outdoors scene captured by a Lidar. The results demonstrate that integration of shape feature significantly improves the robustness of the registration, with a significant reduction in failure rate. Further, our adaptive fusion strategy ensures the improved robustness is obtained without any significant degradation in accuracy. Figure 1 shows a visualization of the registration results on the Lidar Outdoor dataset. Our method accurately registers the four, highly dissimilar views, while state-of-the-art probabilistic approaches [3], [6] struggle in these challenging situations.

## II. Related Work

In recent years, the problem of point sets registration has received much attention. Most of the earlier approaches were based on the classical Iterative Closest Points (ICP) algorithm [11]. The ICP approach computes the rigid transformation that minimizes the distance between the assumed point correspondences. The assumed correspondences are then iteratively updated by finding the nearest neighbors. The standard ICP algorithm requires proper initialization for correct convergence. Several approaches exist in literature that extend the standard ICP algorithm to improve robustness to large initialization errors [12], [13], [14].

Other than the ICP based approaches, probabilistic registration methods have shown promising results in recent years. The probabilistic methods employ a density model of the distribution of the points. The GMMReg [1] approach optimizes a distance measure between Gaussian mixture models (GMM) of the two point sets, to find the transformation. A correlation based statistical approach for point set registration has been proposed by Tsin and Kanade [2]. In this approach, the KL divergence is maximized with respect to the transformation parameters. Evangelidis et al. [3] propose a generative approach for jointly registering multiple point sets (JRMPS). Different to previous methods [4], [5], this approach assumes the point sets to be generated from the same GMM. The rigid transformation and mixture parameters are jointly estimated using a EM-based Maximum Likelihood (ML) optimization.

Despite the success of the above mentioned probabilistic approaches, feature information such as color and shape have been largely ignored. In a recent work, Danelljan et al. [6] propose a probabilistic framework to integrate color information for point set registration. The approach extends the probabilistic model of [3] by modeling the density of the joint point-color space. In [6] only color information was investigated. However, the model is generic and can be used with any invariant feature.

In the context of 3D object recognition, 3D shape features have demonstrated promising results [10], [15]. These features typically integrate information from a spatial neighborhood in the point set into a high-dimensional descriptor. Feature-based registration approaches are typically based on matching to obtain correspondences. Poreba et al. [16] developed a method based on features consisting of two steps: an initial estimation based on robust feature matching using RANSAC and a second step that refines the initial transformation. Basdogan et al. [17] propose a registration framework based on a geometric descriptor and an efficient k-NN search for finding correspondences. Different from these methods, we do not employ explicit matching of feature descriptors. Instead, we investigate the integration of 3D shape features in a probabilistic registration framework.

## III. Probabilistic Point Set Registration Framework

Our registration framework is based on the recent Color-based Probabilistic Point Set Registration (CPPSR) method [6]. This framework extends the probabilistic registration model [3] with feature observations for increased robustness and accuracy. In [6], only the incorporation of color measurements obtained from the sensor, e.g. an RGB-D camera or a Lidar, was investigated. The model can however be extended to any feature information that is invariant to rigid transformations. Unlike most registration methods, the approaches [6], [3] allows joint registration of multiple point sets.

In the CPPSR, a point cloud $\mathcal{X}_i$ is modeled as a set of observed 3D-points $\mathbf{x}_{ij} \in \mathbb{R}^3$ and corresponding feature values $y_{ij} \in \Omega$. Here, $\Omega$ denotes the feature space and $(\mathbf{x}_{ij}, y_{ij}) \in \mathcal{X}_i$ is the $j$th observation in the $i$th point set. We denote the random variables associated with the corresponding observations with capital letters $\mathbf{X}_{ij}, Y_{ij}$. All observations

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 22, 2016.

are assumed to originate from a common joint distribution $p_{\mathbf{V},Y}$ that represents the scene in a reference coordinate system (reference frame). The points $\mathbf{X}_{ij}$ in set $i$ are related to the reference frame by an unknown rigid transformation $\phi_i(\mathbf{x}) = R_i\mathbf{x} + \mathbf{t}_i$, that maps the points in $\mathcal{X}_i$ to the reference frame. The transformed observations are thus distributed as $(\phi_i(\mathbf{X}_{ij}), Y_{ij}) \sim p_{\mathbf{V},Y}$. The registration problem is then formulated as finding the transformation parameters $R_i, \mathbf{t}_i$ along with the model parameters for the density $p_{\mathbf{V},Y}$, given the observations $(\mathbf{x}_{ij}, y_{ij})$.

The density of the joint point-feature space is described as a mixture model. Gaussian components are used in the spatial dimensions to represent the spatial distribution of 3D-points. The feature distribution at each spatial component is modeled by a mixture of non-parametric components. For an observation $(\mathbf{X}, Y)$, a pair of discrete latent random variables $(Z, C)$ are introduced. These assign the observation to the spatial mixture component $Z \in \{0, \dots, K\}$ and the feature component $C \in \{1, \dots, L\}$. Here, $K$ and $L$ denote the number of spatial and feature components respectively. The mixture model of the joint point-feature space is based on the conditional independence assumption $\mathbf{X} \perp\!\!\!\perp Y, C \,|\, Z$, which enables the following factorization of the complete-data likelihood, $p_{\mathbf{X},Y,C,Z} = p_{\mathbf{X}|Z} p_{Y|C,Z} p_{C|Z} p_Z$. The factor $p_Z$ is defined by the mixture weight $\pi_k = p_Z(k)$ of component $k$. The first factor is given by a Gaussian function $p_{\mathbf{X}|Z}(\mathbf{x}|k) = \mathcal{N}(\phi_i(\mathbf{x}); \boldsymbol{\mu}_k, \Sigma_k)$, where $i$ is the set from which the observation originates. In addition, a uniform component is used for $k = 0$ to model outliers.

The feature distribution is modeled by the factors $p_{Y|C,Z}$ and $p_{C|Z}$. In the CPPSR, a general non-parametric component density function $p_{Y|C,Z}(y|l,k) = B_l(y)$ is used for $k \geq 0$. As for the spatial case, a uniform component is used for $k = 0$. Since the components $B_l$ are non-parametric, the feature distribution for each spatial component $k$ is completely determined by the parameters $\rho_{kl} = p_{C|Z}(l|k)$. They represent the feature component weights for each $k$. By marginalizing over the latent variables $Z, C$, the mixture model of the observation $(X, Y)$ is computed as,

$$p_{\mathbf{X},Y}(\mathbf{x}, y) = \sum_{k=1}^{K} \sum_{l=1}^{L} \pi_k \rho_{kl} B_l(y) \mathcal{N}(\phi_i(\mathbf{x}); \boldsymbol{\mu}_k, \Sigma_k) + \pi_0 \mathcal{U}_U(\phi_i(\mathbf{x})) \mathcal{U}_\Omega(y). \quad (1)$$

Here, $\mathcal{U}_U$ and $\mathcal{U}_\Omega$ denote uniform distributions over the scene and the feature space respectively.

The registration is performed by finding a finding a Maximum Likelihood (ML) estimate of the mixture model parameters $\Theta = \big(\{\pi_k, \boldsymbol{\mu}_k, \Sigma_k, \rho_{k1}, \dots, \rho_{kL}\}_{k=1}^{K}, \{R_i, \mathbf{t}_i\}_{i=1}^{M}\big)$. This is obtained using Expectation Conditional Maximization (ECM) [18] as described in [6], [3]. In the Expectation step, the posterior distributions of the latent variables are updated. The two consecutive Conditional Maximization steps updates the transformation and mixture parameters respectively. The ECM process thus jointly estimates the rigid transformations and the density model of the scene.

## IV. Our Approach

Here, we present our feature-based probabilistic registration approach. Our framework integrates high-dimensional 3D shape features in a probabilistic manner, for increased robustness of the registration.

### A. Feature Description

In this work, we investigate the use of descriptive high-dimensional features in a probabilistic registration framework. Different from point-wise color observations, such high-dimensional features capture the geometrical properties of the local neighborhood and are typically based on histograms. In our experiments, we employ the Point Feature Histograms (PFH) [8] due to its discriminative power and invariance to rigid transformations. However, other types of invariant features can also be employed in our framework. The PFH uses both the locations and normals of points in a fix sized neighborhood of $N$ points. For each pair of points in the neighborhood, three angular features are extracted using an invariant reference frame. The descriptor is then constructed as a 3-dimensional histogram of the extracted angles for all pairs, resulting in a $5^3$-dimensional feature vector. We refer to [8] for more details.

The incorporation of high-dimensional features into the probabilistic framework presented in section III, requires a set of mixture components $B_l$ to be defined in the feature space. Danelljan et al. [6] used products of B-spline functions to construct a probabilistic model of the point-wise color feature observations. The components were placed in a regular grid in the 3-dimensional HSV space. However, this strategy implies an exponential increase of the number of feature components $L$ with the dimensionality of the feature space. It is therefore not suitable for high-dimensional 3D shape features. Instead, we cluster the feature space, using a methodology reminiscent to the Bag of Words (BoW) for image classification. This enables a compact and data adaptive representation of the feature space.

As a first step in our registration pipeline, 3D shape features are extracted from all point sets. The feature space is then clustered using K-means, based on all extracted feature vectors. The observed feature value of a point $\mathbf{x}_{ij}$ is represented by the index $y_{ij} \in \{1, \dots, L\}$ of the closest cluster centroid. Here, $L$ is the number of K-means clusters. This effectively transforms the features to the discrete space $\Omega = \{1, \dots, L\}$. The feature components are set to the indicator functions $B_l = \delta_l$. Thus, $B_l(y_{ij}) = 1$ whenever $y_{ij}$ belongs to the $l$th cluster and $B_l(y_{ij}) = 0$ otherwise. In our model, the feature component weights $\rho_{kl}$ specify the categorical distribution of a feature vector from cluster $l$ appearing at the spatial component $k$. This resembles a normalized BoW histogram computed in a spatial neighborhood.

### B. Feature Distribution Initialization

Since our Maximum Likelihood estimation problem is non-convex, the initialization of the parameters $\Theta$ is an important step in EM-based frameworks. Here, we propose a robust

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 22, 2016.

Fig. 2. A visualization of the Lidar Outdoor Dataset, consisting of 4 Lidar scans of the same scene. The dataset is extremely challenging due to severe occlusions and varying point density.

initialization procedure for the feature component parameters $\rho_{kl}$. The proposed approach is more suitable to the feature representation introduced in section IV-A. Compared to the low-level color observations employed in [6], high-dimensional 3D shape features are more descriptive since they integrate information from a spatial neighborhood. Such features are therefore more discriminative as a spatial neighborhood typically contains features from a few different components. To fully exploit the descriptiveness of 3D shape features, we enhance the initialization procedure of the mixture weights $\rho_{kl}$. In [6], these weights were initialized by uniform sampling on the $L-1$ simplex for each component $k$. Instead, we draw independent samples from a Dirichlet distribution $(\rho_{k1}^{(0)}, \ldots, \rho_{kL}^{(0)}) \sim \text{Dir}(\alpha \mathbf{m})$ to obtain the initial feature weights for each spatial component $k$. Here, $\alpha$ is a concentration parameter and $\mathbf{m} = (m_1, \ldots, m_L)$ is the frequency of feature values $m_l = \frac{1}{N} \sum_{ij} \delta_l(y_{ij})$ normalized with the number of observations $N = \sum_{ij} 1$.

The normalized frequency $\mathbf{m}$ specifies the expectation of the Dirichlet distribution, the concentration parameter $\alpha$ specifies its shape. Increasing $\alpha$ concentrates the probability mass around $\mathbf{m}$. Setting $\alpha = L$ leads to approximately uniform sampling, provided that $\mathbf{m}$ is almost uniform. On the other hand, a decreased value of $\alpha$ moves the probability mass towards the boundaries of the simplex. This provides samples of more distinctive categorical distributions $(\rho_{kl}^{(0)})_{l=1}^{L}$, where most values are close to zero. A small concentration parameter value proved important for the convergence speed and robustness of our registration method. Throughout our experiments, we use $\alpha = 1$.

### C. Adaptive Feature Fusion

3D shape features, such as PFH, integrate information from a spatial neighborhood. This leads to a larger discriminative power but also to a reduced spatial resolution in the feature representation. An incorporation of such features can therefore lead to increased robustness at the cost of a reduced accuracy. To avoid this issue, we employ an approach which corresponds to multi-resolution search strategy. The 3D shape features are used in the first half of the EM iterations in the registration process. This alleviates the problem of converging to the correct local Maximum Likelihood mode. The estimate is then refined by performing the second half of the EM iterations without the 3D features, for preserved accuracy.

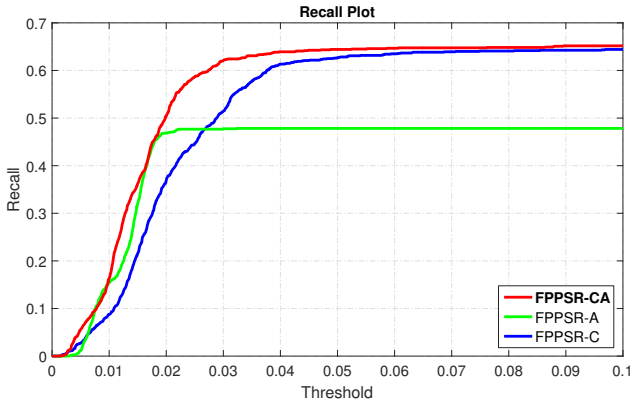



Fig. 3. A baseline comparison of three different versions of our approach. The recall plot is computed from 200 multi-view registrations on the Lidar outdoor dataset. We compare our final approach (FPPSR-CA), employing both the clustering-based representation and the adaptive fusion strategy, with a straightforward feature representation (FPPSR-A) and no adaptive fusion strategy (FPPSR-C). Our final version (FPPSR-CA) achieves significantly improved robustness and accuracy compared to the two baseline versions.

## V. Experiments

We evaluate our approach on two challenging datasets: an outdoor dataset [19] acquired by a FARO Focus 3D Lidar and the Stanford Lounge [20] acquired by a Kinect RGB-D sensor.

### A. Details and parameters

In our experiments, we fix the number of spatial components $K = 500$, the outlier ratio parameter $\pi_0 = 0.005$ and the number of ECM iterations (100) for the JRMPS, CPPSR and our approach. The number of feature clusters $L$ in our method, controls the trade-off between distinctiveness and invariance of the feature representation. We found $L = 10$ to provide a good balance on all datasets.

We use the Frobenius norm between the rotation matrices [3], [6] to quantitative evaluation. The rotation error is defined as $\| \hat{R} - R \|_F$, where $\hat{R}$ represents the estimated rotation and $R$ denotes the ground-truth rotation. Our approach is implemented in MATLAB.

### B. Baseline Comparison

We first perform a baseline comparison on the Lidar Outdoor Dataset [19]. The dataset consists of four scans, contain-

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 22, 2016.

TABLE I
THE RESULTS FROM THE MULTI-VIEW REGISTRATION EXPERIMENTS ON THE LIDAR OUTDOOR DATASET, IN TERMS OF AVERAGE ERROR AND STANDARD DEVIATION FOR INLIERS AND FAILURE RATE. OUR APPROACH ACHIEVES A SIGNIFICANT REDUCTION IN FAILURE RATE COMPARED TO THE STATE-OF-THE-ART APPROACHES.

| | Avg. err | Std. dev. | Failure rate (%) |
|---|---|---|---|
| JRMPS [3] | $1.26 \cdot 10^{-2}$ | $3.18 \cdot 10^{-3}$ | 52.9 |
| CPPSR [6] | $1.20 \cdot 10^{-2}$ | $4.40 \cdot 10^{-3}$ | 52.6 |
| **Our** | $1.53 \cdot 10^{-2}$ | $9.22 \cdot 10^{-3}$ | 31.2 |

ing more than one million points each. Figure 2 shows all the four views. The dataset is challenging due to severe occlusions and variations in the point density. This dissimilarity between the individual point sets significantly complicates the task of registering the different scans. We perform a multi-view registration experiment by jointly aligning all four views. We perform 200 registrations by initializing each point set with a uniformly sampled random rotation. In each view, about 5000 points are sampled using a keypoint detector[2] in order to partially alleviate the uneven distribution of the points. The PFH descriptor for each keypoint is computed using the full point set to ensure a sufficient amount of neighboring points.

Three different feature-based versions of our method are evaluated. To verify the feature model proposed in section IV-A, we compare with a version, called FPPSR-A, that instead employs a straightforward model of the feature space. For this purpose, we normalize the PFH descriptors and let the feature components be the coordinate projections $B_l(y) = y^{(l)}$, where $y^{(l)}$ denotes the $l$th dimension in the normalized histogram $y$. To validate the adaptive feature fusion, presented in section IV-C, we compare with FPPSR-C, that does not employ this component. Note that FPPSR-A includes the adaptive fusion (section IV-C) and that FPPSR-C employs the clustering-based feature representation (section IV-A). Lastly, we evaluate the version FPPSR-CA that employs both the proposed feature representation and the adaptive fusion. For a fair comparison, we use the initialization strategy described in section IV-B for all three versions.

Figure 3 shows the recall plot of the comparison between the three versions. The recall is obtained by computing the fraction (vertical axis) of pairwise registration errors that are smaller than a rotation error threshold (horizontal axis). Compared to the straightforward feature representation (FPPSR-A), our approach (FPPSR-CA) achieves superior robustness, shown by the increased recall for larger thresholds. Further, our adaptive fusion strategy significantly improves the accuracy of the registration compared to FPPSR-C, while obtaining similar robustness. For the remaining experiments, we use FPPSR-CA as our final approach.

### C. Lidar Dataset

Here, we perform a comprehensive comparison of our feature-based approach with the two state-of-the-art probabilistic joint registration approaches: the JRMPS [3], employing no feature information, and CPPSR [6] that utilizes the color observations. We use an experimental setup similar to the baseline experiment (section V-B), but perform over 500 registrations for a more extensive evaluation. Table I reports the results in terms of average inlier error, standard deviation and failure rate. The failure rate measures the robustness and is defined as the percentage of pair-wise rotation errors that are greater than $0.1$ (approximately 4 degrees). A registration is regarded an inlier if the error is smaller than $0.1$. To evaluate the accuracy of the registrations, we report the average and standard deviation of the pair-wise inlier rotation errors.

[2]We use a PCL implementation of the Scale Invariant Features Transform (SIFT) 3D detector. http://www.pointclouds.org/

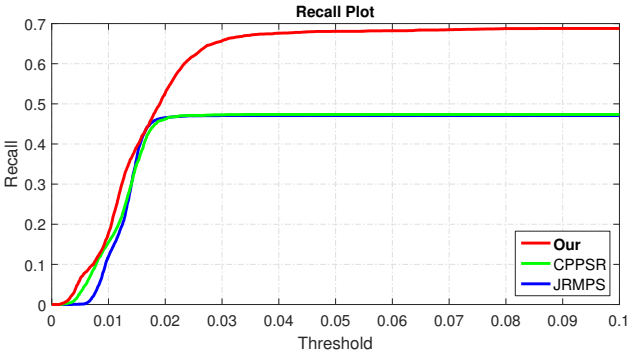



Fig. 4. A comparison of our method with the JRMPS [3] and CPPSR [6] for the multi-view registration on the Lidar outdoor dataset. Our approach achieves significantly improved robustness while maintaining the accuracy.

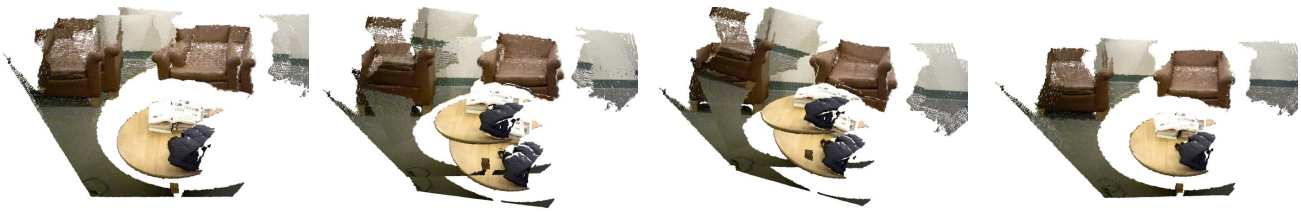
Both the JRMPS and the CPPSR struggle in registering the four views, with a failure rate of $52.9\%$ and $52.6\%$ respectively. Our method significantly improves the state-of-the-art with a failure rate of $31.3\%$, while maintaining a competitive accuracy of $1.50 \cdot 10^{-2}$ with respect to both JRMPS ($1.26 \cdot 10^{-2}$) and CPPSR ($1.20 \cdot 10^{-2}$). Note that the error measures are computed on the inlier registrations. Therefore, the vastly increased number of successful registrations of our method leads to a slightly lower average accuracy. On the other hand, in the recall plot (figure 4) our approach provides consistently better results for all error thresholds. This indicates that our method in fact maintains the accuracy of JRMPS and CPPRS, while significant improving the robustness.

In summary, the results clearly demonstrate that a proper integration of high-dimensional 3D shape features leads to superior registration performance in challenging scenarios. Further, our approach efficiently registers multiple views, despite partial overlap, occlusions and varying point density. A visualization of an example registration is shown figure 1.

### D. Stanford Lounge Dataset

Finally, we present results on the Stanford Lounge Dataset [20], consisting of 3000 RGB-D frames acquired by Kinect sensor. As ground-truth, we employ the poses provided by the authors [20]. For computational efficiency, we randomly downsample the frames to 10k points. In our experiments, we do not observe any improvements when using keypoint sampling techniques. This is partially attributed to the fact that the variations of the point density is less significant in this dataset. For every subsampled frame, the PFH descriptors

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 22, 2016.



(a) Initial Position. (b) Baseline JRMPS. (c) CPPSR. (d) Our.

Fig. 5. A qualitative comparison on the Stanford Lounge dataset. Registration results are shown for an example RGB-D frame. Our approach (d) more accurately registers the point sets (a), compared to both JRMPS (b) and CPPSR (c).

TABLE II
A COMPARISON OF OUR APPROACH WITH STATE-OF-THE-ART REGISTRATION METHODS ON THE STANFORD LOUNGE DATASET. THE RESULTS ARE REPORTED IN TERMS OF FAILURE RATE, AVERAGE, AND STANDARD DEVIATION OF THE INLIER ROTATION ERRORS. OUR APPROACH SIGNIFICANTLY REDUCES THE RELATIVE FAILURE RATE BY 40% ON THIS DATASET.

| | Avg. err | Std. dev. | Failure rate (%) |
|---|---|---|---|
| ICP [11] | $4.32 \cdot 10^{-2}$ | $2.53 \cdot 10^{-2}$ | 15.70 |
| Color GICP [21] | $1.72 \cdot 10^{-2}$ | $1.75 \cdot 10^{-2}$ | 1.27 |
| JRMPS [3] | $1.78 \cdot 10^{-2}$ | $1.35 \cdot 10^{-2}$ | 3.67 |
| CPPSR [6] | $1.54 \cdot 10^{-2}$ | $1.08 \cdot 10^{-2}$ | 1.00 |
| **Our** | $1.51 \cdot 10^{-2}$ | $1.08 \cdot 10^{-2}$ | 0.60 |

are computed for each sampled point, by utilizing the complete point set. As in [6], we perform pairwise registration between frame number $n$ and $n+5$ for all the frames in the dataset.

Table II reports the average error, the standard deviation, and the failure rate for the compared methods. In addition to the probabilistic methods, we also compare with the standard ICP and the Color-GICP. The JRMPS approach provides a failure rate of $3.67\%$. The recently introduced CPPSR, employing the color features, obtains competitive results with a failure rate of $1.00\%$. Our approach significantly improves the state-of-the-art on this dataset, with a failure rate of $0.60\%$. It is worth to mention that our approach provides this significant reduction of failure rate without any degradation in accuracy. Figure 5 shows a qualitative comparison of our approach with both the JRMPS and CPPSR on the Stanford dataset.

## VI. CONCLUSION

We propose a probabilistic framework to integrate high dimensional 3D features for point set registration. Our approach constructs a compact probabilistic representation by clustering the high-dimensional feature space. The local feature distribution parameters are jointly estimated in an EM-framework. Moreover, we introduce an adaptive fusion strategy to integrate high-dimensional 3D shape features with local color information. Experiments on two challenging datasets clearly demonstrate that our approach leads to significant improvement in robustness without any degradation in accuracy. Future work involves further investigations on the impact of the clustered shape representation employed in our framework. Another research direction is to perform a comprehensive evaluation of the 3D shape descriptors for probabilistic point set registration.

## REFERENCES


[1] B. Jian and B. C. Vemuri, "Robust point set registration using gaussian mixture models," *PAMI*, vol. 33, no. 8, pp. 1633–1645, 2011.

[2] Y. Tsin and T. Kanade, "A correlation-based approach to robust point set registration," in *ECCV*, 2004, pp. 558–569.

[3] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, and E. Z. Psarakis, "A generative model for the joint registration of multiple point sets," in *ECCV*, 2014, pp. 109–122.

[4] R. Horaud, F. Forbes, M. Yguel, G. Dewaele, and J. Zhang, "Rigid and articulated point registration with expectation conditional maximization," *PAMI*, vol. 33, no. 3, pp. 587–602, 2011.

[5] A. Myronenko and X. B. Song, "Point set registration: Coherent point drift," *PAMI*, vol. 32, no. 12, pp. 2262–2275, 2010.

[6] M. Danelljan, G. Meneghetti, F. Shahbaz Khan, and M. Felsberg, "A probabilistic framework for color-based point set registration," in *CVPR*, 2016.

[7] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *ACM MM*, 2007.

[8] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Learning informative point classes for the acquisition of object model maps," in *ICARCV*, 2008, pp. 643–650.

[9] A. Kläser, M. Marszalek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *BMVC*, 2008.

[10] Y. Guo, F. A. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3d local surface description and object recognition," *IJCV*, vol. 105, no. 1, pp. 63–86, 2013.

[11] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *PAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[12] A. Rangarajan, H. Chui, and F. L. Bookstein, "The softassign procrustes matching algorithm," in *IPMI*, 1997.

[13] D. Chetverikov, D. Stepanov, and P. Krsek, "Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm," *IMAVIS*, vol. 23, no. 3, pp. 299–309, 2005.

[14] A. Segal, D. Hähnel, and S. Thrun, "Generalized-icp," in *RSS*, 2009.

[15] Y. Guo, M. Bennamoun, F. A. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3d local feature descriptors," *IJCV*, vol. 116, no. 1, pp. 66–89, 2016.

[16] M. Poreba and F. Goulette, "A robust linear feature-based procedure for automated registration of point clouds," *Sensors*, vol. 15, no. 1, pp. 1435–1457, 2015.

[17] C. Basdogan and A. C. ztireli, "A new feature-based method for robust and efficient rigid-body registration of overlapping point clouds," *The Visual Computer*, vol. 24, no. 7-9, pp. 679–688, 2008.

[18] X. L. Meng and D. B. Rubin, "Maximum Likelihood Estimation via the ECM Algorithm: A General Framework," *Biometrika*, vol. 80, no. 2, pp. 267–278, 1993.

[19] J. Unger, A. Gardner, P. Larsson, and F. Banterle, "Capturing reality for computer graphics applications," in *Siggraph Asia Course*, 2015.

[20] Q.-Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 112:1–112:8, 2013.

[21] M. Korn, M. Holzkothen, and J. Pauli, "Color supported generalized-icp," in *VISAPP*, 2014.


Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 22, 2016.

# A Probabilistic Framework for Color-Based Point Set Registration

Martin Danelljan, Giulia Meneghetti, Fahad Shahbaz Khan, Michael Felsberg
Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, Sweden
{martin.danelljan, giulia.meneghetti, fahad.khan, michael.felsberg}@liu.se

## Abstract

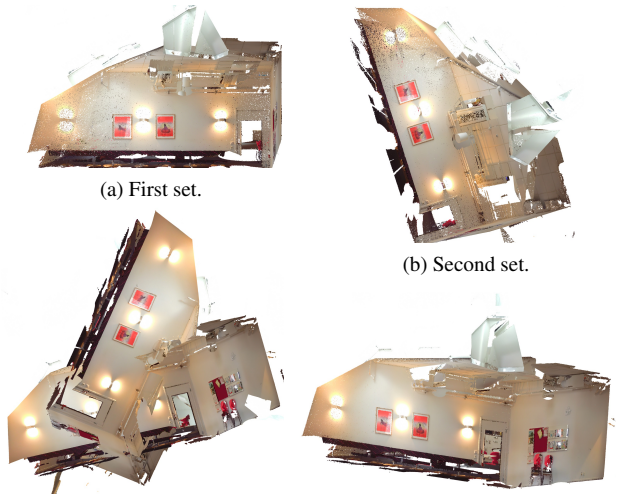
*In recent years, sensors capable of measuring both color and depth information have become increasingly popular. Despite the abundance of colored point set data, state-of-the-art probabilistic registration techniques ignore the available color information. In this paper, we propose a probabilistic point set registration framework that exploits available color information associated with the points. Our method is based on a model of the joint distribution of 3D-point observations and their color information. The proposed model captures discriminative color information, while being computationally efficient. We derive an EM algorithm for jointly estimating the model parameters and the relative transformations.*

*Comprehensive experiments are performed on the Stanford Lounge dataset, captured by an RGB-D camera, and two point sets captured by a Lidar sensor. Our results demonstrate a significant gain in robustness and accuracy when incorporating color information. On the Stanford Lounge dataset, our approach achieves a relative reduction of the failure rate by 78% compared to the baseline. Furthermore, our proposed model outperforms standard strategies for combining color and 3D-point information, leading to state-of-the-art results.*

## 1. Introduction

3D-point set registration is a classical computer vision problem with important applications. Generally, the points originate from measurements of sensors, such as time-of-flight cameras and laser range scanners. The problem is to register observed point sets from the same scene by finding their relative geometric transformations. One class of approaches [2, 16], based on the Iterative Closest Point (ICP) [1], iteratively assumes pairwise correspondences and then finds the transformation by distance minimization. Alternatively, probabilistic methods [5, 7, 9, 14] model the distribution of points using *e.g.* Gaussian Mixture Models (GMMs).

Recently, probabilistic approaches demonstrated promising results for point set registration [5, 7]. The improved performance in probabilistic methods is achieved by modeling the distribution of points as a density function. The probabilistic approaches can be further categorized into correlation-based and Expectation Maximization (EM) based methods. The correlation-based approaches [9, 17] estimate the transformation parameters by maximizing a similarity measure between the density models of the two point sets. Instead, the EM-based methods simultaneously estimate the density model and the transformation parameters [5, 7, 14]. In this paper, we explore probabilistic models for EM-based *colored* point set registration.



(a) First set. (b) Second set. (c) Baseline registration [5]. (d) Our color-based registration.

Figure 1. Registration of the two colored point sets (a) and (b), of an indoor scene captured by a Lidar. The baseline GMM-based method (c) fails to register the two point sets due to the large initial rotation error of 90 degrees. Our method accurately registers the two sets (d), by exploiting the available color information.

State-of-the-art probabilistic techniques [5, 7, 14] rely on the distribution of points in 3D-space, while ignoring additional information, such as color, for point set registration. On the other hand, the increased availability of cheap RGB-D cameras has triggered the use of colored 3D-point sets in many computer vision applications, including 3D object recognition [4], scene reconstruction [3] and robotics [6]. Besides RGB-D cameras, many laser range scanners also capture RGB or intensity information. Additionally, col-

ored point sets are produced by stereo cameras and ordinary cameras by using structure from motion. In this paper, we investigate the problem of incorporating color information for probabilistic point set registration, regardless of the sensor used for capturing the data.

When incorporating color information in probabilistic point set registration, the main objective is to find a suitable probability density model of the joint observation space. The joint space consists of the 3D-point observations and their associated color information. Color information can be incorporated into a probabilistic point set model in two standard ways. (i) A first approach is to directly introduce joint mixture components in the complete observation space. This model requires large amounts of data due to the high dimensionality of the joint space, leading to a high computational cost. (ii) A second approach is to assume stochastic independence between points and color, which enables separable modeling of both spaces. However, this assumption ignores the crucial information about the spatial dependence of color. The aforementioned shortcomings of both fusion approaches motivate us to investigate alternative probabilistic models for incorporating color information.

**Contributions:** In this paper, we propose a color-based probabilistic framework for point set registration. Our model combines the advantages of (i) and (ii), by assuming *conditional* independence between the location of a point and its color value, given the spatial mixture component. In our model, each spatial component also contains a non-parametric density estimator of the local color distribution. We derive an efficient EM algorithm for joint estimation of the mixture and the transformation parameters. Our approach is generic and can be used to integrate other invariant features, such as curvature and local shape.

Comprehensive experiments are performed on the Stanford Lounge dataset [19] containing 3000 RGB-D frames with ground-truth poses. We also perform experiments on two colored point sets captured by a Lidar: one indoor scene and one outdoor scene [18]. The results clearly demonstrate that our color-based registration significantly improves the baseline method. We further show that the proposed color-based registration method outperforms standard color extensions, leading to state-of-the-art performance. Figure 1 shows registration results on the indoor Lidar dataset, using the baseline [5] and our color-based registration model.

## 2. Related Work

Initially, most point set registration methods [2, 16] were based on the classical ICP [1] algorithm. The ICP-based approaches alternate between assuming point-to-point correspondences between the two sets and finding the optimal transformation parameters. The standard ICP [1] is known to require a good initialization, since it is prone to get stuck in local minima. Several methods [2, 15, 16] have been proposed to tackle this robustness issue.

Probabilistic registration techniques employ, *e.g.*, Gaussian mixtures to model the distribution of points. In correlation based probabilistic approaches [9, 17], the two point sets are modeled separately in a first step. A similarity measure between the density models, *e.g.* the KL divergence, is then maximized with respect to the transformation parameters. However, these methods lead to nonlinear optimization problems with non-convex constraints. To avoid complex optimization problems, several recent methods [5, 7, 14] simultaneously estimate the density model and the registration parameters in an EM-based framework. Among these methods, the recent Joint Registration of Multiple Point Sets (JRMPS) [5] models all involved point sets as transformed realizations of a single common GMM. Compared to previous EM-based methods [7, 14], JRMPS does not constrain the GMM centroids to the points in a particular set. This further enables a joint registration of multiple point sets.

The use of color information for point set registration has been investigated in previous works [8, 11, 10, 12]. Huhle *et al*. [8] propose a kernel-based extension to the normal distributions transform, for aligning colored point sets. Most approaches [10, 11, 12] aim at augmenting ICP-based methods [1, 16] with color. In these approaches, a metric is introduced in a joint point-color space, to find correspondences in each iteration. A drawback of these ICP variants is that the metric relies on a data dependent parameter that controls the trade-off between spatial distance and color difference. Different to these methods, we incorporate color information in a probabilistic registration framework. The registration is performed using an EM-based maximum likelihood estimation. Next, we describe the baseline probabilistic registration framework.

## 3. Joint Registration of Point Sets

We base our registration framework on the JRMPS [5] method, since it has shown to provide improved performance compared to previous GMM based approaches [7, 14]. Contrary to these methods, JRMPS assumes both sets to be transformed realizations of one reference GMM. This avoids the underlying asymmetric assumption of using one of the sets as a reference model in the registration [7, 14]. Further, the JRMPS has the advantage of naturally generalizing to joint registration of multiple sets.

### 3.1. Point Set Observation Model

In the problem of joint registration of multiple point sets, the observations consist of 3D-points in $M$ different views of the same scene. The aim is then to find the transformation of each set to a common reference coordinate system, called the reference frame. All observations of 3D-points are assumed to originate from the same spatial distribution $\mathbf{V} \sim p_{\mathbf{V}}$, representing the entire scene. Here, $\mathbf{V} \in \mathbb{R}^3$ is a

random variable (r.v.) of a point in the reference frame, and $p_{\mathbf{V}}$ is the probability density function (p.d.f.) of $\mathbf{V}$.

Let $\mathbf{X}_{ij} \in \mathbb{R}^3$ be the r.v. of the $j$:th observed point in view $i \in \{1, \dots, M\}$ and let $\mathbf{x}_{ij}$ be its observed value. Observations in view $i$ are related to the reference frame by the unknown rigid transformation $\phi_i(\mathbf{x}) = R_i\mathbf{x} + \mathbf{t}_i$, such that $\phi_i(\mathbf{X}_{ij}) \sim p_{\mathbf{V}}$. The transformed observations $\phi_i(\mathbf{X}_{ij})$ thus have the distribution $p_{\mathbf{V}}$ in the reference frame. Consequently, the p.d.f. of the observation $\mathbf{X}_{ij}$ is given by $p_{\mathbf{X}_{ij}}(\mathbf{x}_{ij}) = p_{\mathbf{V}}(\phi_i(\mathbf{x}_{ij}))$. To simplify notation, we often write $p_{\mathbf{X}_{ij}}(\mathbf{x}_{ij}) = p(\mathbf{x}_{ij})$.

As described above, the observed points are assumed to be transformed samples of the distribution $p_{\mathbf{V}}$. The point distribution $p_{\mathbf{V}}$ is modeled as a mixture of Gaussian distributions. Let $K$ be the number of Gaussian components. We then introduce the discrete latent r.v. $Z \in \{0, \dots, K\}$ that assigns the point $\mathbf{V}$ to the mixture component $Z = k$. The extra 0th component is a uniform distribution that models the occurrence of outlier points. The joint p.d.f. of $\mathbf{V}$ and $Z$ factorizes as $p(\mathbf{v}, z) = p(\mathbf{v}|z)p(z)$. For discrete variables, we use the notation $p(Z = k) = p_Z(k)$. The mixture component weights $\pi_k$ are defined as the prior probabilities $\pi_k = p(Z = k)$ of the latent variable $Z$. The conditional distribution of $\mathbf{V}$ given $Z = k$ is then defined as,

$$p(\mathbf{v}|Z = k) = \begin{cases} \mathcal{U}_U(\mathbf{v}), & k = 0 \\ \mathcal{N}(\mathbf{v}; \boldsymbol{\mu}_k, \Sigma_k), & k \neq 0. \end{cases} \quad (1)$$

Here, $\mathcal{U}_U$ denotes a uniform distribution in the convex hull $U \subset \mathbb{R}^3$ of the observations [7]. The multivariate normal distribution with expectation $\boldsymbol{\mu}$ and covariance $\Sigma$ is denoted by $\mathcal{N}(\cdot\,; \boldsymbol{\mu}, \Sigma)$. The point density function $p_{\mathbf{V}}$ is obtained by marginalizing over the latent variable $Z$,

$$p_{\mathbf{V}}(\mathbf{v}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{v}; \boldsymbol{\mu}_k, \Sigma_k) + \pi_0 \mathcal{U}_U(\mathbf{v}). \quad (2)$$

Next, we describe how the above described observation model is used for point set registration.

### 3.2. Point Set Registration

The registration is performed by jointly estimating the transformation and the GMM parameters, in (2), using the EM algorithm. We denote the set of all observations by $\mathcal{X} = \{\mathbf{x}_{ij}\}_{j=1,i=1}^{N_i,M}$ and the collection of corresponding latent variables by $\mathcal{Z} = \{Z_{ij}\}_{j=1,i=1}^{N_i,M}$. Here, $N_i$ denotes the number of observations in point set $i$. All observations are assumed to be independent. As in [5], a fix outlier weight $\pi_0$ is assumed. The model parameters are summarized as,

$$\Theta = \left(\{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1}^{K}, \{R_i, \mathbf{t}_i\}_{i=1}^{M}\right). \quad (3)$$

The point registration is performed by jointly estimating the parameters $\Theta$ from the observed data $\mathcal{X}$. In [5], a Maximum Likelihood (ML) estimate of $\Theta$ is obtained using the Expectation Maximization (EM) framework. The E-step evaluates the conditional expectation of the complete data log-likelihood $\log p(\mathcal{X}, \mathcal{Z}|\Theta)$. The expectation is taken with respect to the latent variables $\mathcal{Z}$ given the observed data $\mathcal{X}$ and the current estimate of the parameters $\Theta^{(n)}$,

$$\begin{aligned} Q(\Theta; \Theta^{(n)}) &= \mathrm{E}_{\mathcal{Z}|\mathcal{X},\Theta^{(n)}}\left[\log p(\mathcal{X}, \mathcal{Z}|\Theta)\right] \\ &= \sum_{\mathcal{Z}} p(\mathcal{Z}|\mathcal{X}, \Theta^{(n)}) \log p(\mathcal{X}, \mathcal{Z}|\Theta) \end{aligned} \quad (4)$$

In the M-step, the aim is to find the optimizer of (4) as $\Theta^{(n+1)} = \arg\max_{\Theta} Q(\Theta; \Theta^{(n)})$. To obtain a closed form solution, the M-step is divided into two conditional maximization (CM) steps [13], where the transformation and GMM parameters are updated separately [7].

Using the definitions in section 3.1 and the independent observations assumption, the complete data likelihood is expressed as $p(\mathcal{X}, \mathcal{Z}|\Theta) = \prod_{ij} p(\mathbf{x}_{ij}, z_{ij}|\Theta)$, where

$$p(\mathbf{x}_{ij}, Z_{ij} = k|\Theta) = \pi_k \mathcal{N}(\phi_i(\mathbf{x}_{ij}); \boldsymbol{\mu}_k, \Sigma_k)\,,\ k \neq 0. \quad (5)$$

The posterior density of the latent variables factorizes as $p(\mathcal{Z}|\mathcal{X}, \Theta^{(n)}) = \prod_{ij} p(z_{ij}|\mathbf{x}_{ij}, \Theta^{(n)})$. The E-step then reduces to computing the posterior probabilities of the latent variables $\alpha_{ijk}^{(n)} := p(Z_{ij} = k|\mathbf{x}_{ij}, \Theta^{(n)})$ [5]. Eq. 4 now simplifies to,

$$Q(\Theta; \Theta^{(n)}) = \sum_{ijk} \alpha_{ijk}^{(n)} \log p(\mathbf{x}_{ij}, Z_{ij} = k|\Theta). \quad (6)$$

By applying (5) and ignoring constant terms, (6) can be rewritten to the equivalent minimization problem,

$$\begin{aligned} f(\Theta; \Theta^{(n)}) = \sum_{ij} \sum_{k=1}^{K} \alpha_{ijk}^{(n)} \Bigg( &\frac{1}{2} \log |\Sigma_k| \\ &+ \frac{1}{2} \|R_i\mathbf{x}_{ij} + \mathbf{t}_i - \boldsymbol{\mu}_k\|_{\Sigma_k^{-1}}^2 - \log \pi_k \Bigg). \end{aligned} \quad (7)$$

Here, $|\Sigma_k|$ denotes the determinant of $\Sigma_k$ and we have defined $\|\mathbf{x}\|_{\Sigma_k^{-1}}^2 = \mathbf{x}^{\mathrm{T}}\Sigma_k^{-1}\mathbf{x}$. For simplicity, isotropic covariances are assumed $\Sigma_k = \sigma_k^2 I$, as in [5].

The parameters $\Theta$ are updated in the two CM-steps of the algorithm. The first CM-step minimizes (7) with respect to the transformation parameters $\{R_i, \mathbf{t}_i\}_{i=1}^{M}$, given the current GMM parameters $\{\pi_k^{(n-1)}, \boldsymbol{\mu}_k^{(n-1)}, \Sigma_k^{(n-1)}\}_{k=1}^{K}$. The second CM-step minimizes (7) with respect to the GMM parameters given the new $\{R_i^{(n)}, \mathbf{t}_i^{(n)}\}_{i=1}^{M}$. We refer to [5] for the closed form solutions of the two CM-steps. Next we introduce our color based registration technique.

## 4. Feature Based Point Set Registration

We reformulate the registration problem from section 3 to incorporate feature information associated with each 3D-point. In this work, we investigate the incorporation of *color*

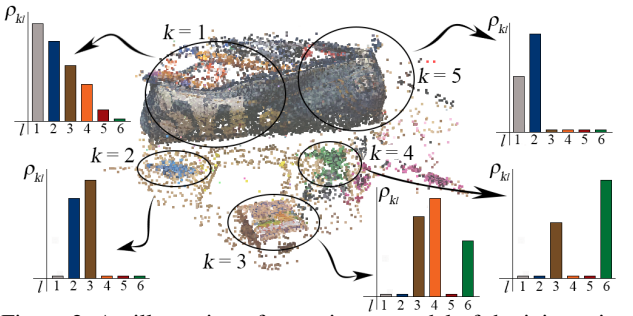


Figure 2. An illustration of our mixture model of the joint point-color space. The ellipses represent spatial mixture components $p(\mathbf{v}|Z=k)$ in our model. Each spatial component $k$ is associated with a mixture model in the color space, given by the weights $\rho_{kl}$ (visualized as histograms). This mixture model encodes the color distribution of points associated with the spatial component $k$.

information for point set registration. However, our framework is not restricted to color features. It also enables the use of, *e.g.*, structural features that describe the local shape or curvature of the point set.

## 4.1. Feature Based Observation Model

Our framework assumes the observations to consist of a 3D-point and its associated feature value, *e.g.* color. Let $Y \in \Omega$ denote the r.v. of the feature value associated with the 3D-point $\mathbf{V}$. Here, $\Omega$ is the set of all possible feature values, called the feature space. For example, if $Y$ is the color of the 3D-point in normalized HSV coordinates, then the feature space is the unit cube $\Omega = [0,1]^3$. We assume observations of points and features to originate from a common joint distribution $(\mathbf{V}, Y) \sim p_{\mathbf{V},Y}$. The aim of this paper is to propose an efficient yet distinctive mixture model of the joint point-feature density $p_{\mathbf{V},Y}$. Next, we investigate three different strategies to construct a mixture model of the joint point-feature space.

### 4.1.1 The Direct Approach

A direct generalization of the GMM based registration technique (section 3), is to introduce joint mixture components in the point-feature space $\mathbb{R}^3 \times \Omega$. In general, let $F(\mathbf{v}, y; \theta_k)$ denote the density function of a mixture component in the joint space $(\mathbf{v}, y) \in \mathbb{R}^3 \times \Omega$. Here, $\theta_k$ denote the parameters of the $k$:th component. A mixture model in the joint point-feature space is expressed as

$$p_{\mathbf{V},Y}(\mathbf{v}, y) = \sum_{k=1}^{K} \pi_k F(\mathbf{v}, y; \theta_k). \quad (8)$$

However, this strategy of directly introducing joint components $F(\mathbf{v}, y; \theta_k)$ requires a large amount of data, due to the exponential growth of volume with the number of dimensions (*i.e.* the curse of dimensionality). This leads to a higher computational cost.

### 4.1.2 The Independent Approach

To alleviate the problems induced by the direct strategy (8), a simple approach is to assume stochastic independence between 3D-points and feature values. The joint distribution $p_{\mathbf{V},Y}$ then factorizes as the product of the marginal distributions for the 3D-points $p_{\mathbf{V}}$ and feature values $p_Y$, such that $p_{\mathbf{V},Y} = p_{\mathbf{V}} p_Y$. This assumption enables the spatial distribution of points $p_{\mathbf{V}}$ and the distribution of features $p_Y$ to be modeled separately. Let $\tilde{F}$, $\tilde{\theta}_l$ and $\tilde{\pi}_l$ denote the components, parameters and weights respectively for the mixture model of the feature density $p_Y$. We denote the number of feature components by $L$. The joint distribution can then be expressed as

$$p_{\mathbf{V},Y}(\mathbf{v}, y) = \sum_{k=1}^{K} \sum_{l=1}^{L} \pi_k \tilde{\pi}_l \mathcal{N}(\mathbf{v}; \boldsymbol{\mu}_k, \Sigma_k) \tilde{F}(y; \tilde{\theta}_l). \quad (9)$$

Here, we have used the GMM presented in section 3.1 for the spatial distribution $p_{\mathbf{V}}$ and ignore the uniform component for simplicity. While the independence assumption allows for a separation of the mixture models, it completely removes information regarding the spatial dependence of feature values. Such information is crucial for aiding the registration process.

The aforementioned approaches have major limitations when incorporating feature information for point set registration. Next, we describe an approach that combines the discriminative power of the direct approach with the efficiency of the independent approach.

### 4.1.3 Our Approach

We propose a mixture model of the joint point-feature space $\mathbb{R}^3 \times \Omega$ that tackles the drawbacks of the aforementioned approaches. Contrary to the direct strategy (section 4.1.1), our method does not require an increased amount of points to infer the model parameters. We thereby avoid the problems induced by the higher dimensionality of the observation space. Additionally, our model accurately captures the local characteristics in the distribution of features, *e.g.*, how colors are distributed in the scene. This enables our framework to exploit the underlying discriminative feature information associated with each 3D-point.

The proposed mixture model contains a separate feature distribution for each spatial mixture component (illustrated in figure 2). In addition to the spatial latent variable $Z$, we introduce a second latent r.v. $C \in \{1, \dots, L\}$. This variable assigns a point-feature pair $(\mathbf{V}, Y)$ to one of the $L$ mixture components in the feature space $\Omega$. Our model is based on the conditional independence assumption between the point $\mathbf{V}$ and the feature variables $Y, C$ given the spatial mixture component $Z$. This is symbolically expressed as

$\mathbf{V} \perp Y, C \mid Z$. Our model assumption enables the following factorization of the joint p.d.f. of $(\mathbf{V}, Y, C, Z)$,

$$\begin{aligned} p(\mathbf{v}, y, c, z) &= p(\mathbf{v}, y, c|z)p(z) = p(\mathbf{v}|z)p(y, c|z)p(z) \\ &= p(\mathbf{v}|z)p(y|c, z)p(c|z)p(z). \end{aligned} \tag{10}$$

The first and fourth factor of (10) do not depend on the feature information, and are defined in section 3.1 (see (1)).

Each spatial component is given a separate feature distribution that characterizes the occurrences of feature values in the vicinity of the component. These distributions are defined by the feature component weights, determined by the conditional probability of a feature component $C = l$ given a spatial component $Z = k$,

$$p(C = l|Z = k) = \rho_{kl}\,, \quad k \neq 0. \tag{11}$$

This expression defines the third factor in (10). The feature mixture weights must satisfy $\rho_{kl} \geq 0$ and $\sum_l \rho_{kl} = 1$ for each spatial component $k$. For the outlier component $k = 0$, we assume uniform weights $p(C = l|Z = 0) = {}^1\!/_L$.

The second factor $p(y|c, z)$ in (10) is determined by the mixture components in the feature space. Since the feature space $\Omega$ can be compact or discrete, we do not restrict our choice to Gaussian distributions. Instead, we consider arbitrary non-negative functions $B_l : \Omega \to \mathbb{R}$ satisfying $\int_\Omega B_l = 1$. We define,

$$p(y|C = l, Z = k) = \begin{cases} \mathcal{U}_\Omega(y), & k = 0 \\ B_l(y), & k \neq 0. \end{cases} \tag{12}$$

As for the spatial mixture components (1), we also use a uniform component in the feature space for $Z = 0$ to model outliers. The integration feature information into the registration process comes at an increased computational cost. In order to minimize this cost, we use non-parametric feature components $B_l$ in our model. This allows the probabilities $B_l(y_{ij})$ to be precomputed and avoids additional costly maximizations of in the M-step.

The proposed mixture model of the joint space is computed by marginalizing over the latent variables $Z, C$ in (10) and using the definitions (1), (11) and (12),

$$\begin{aligned} p_{\mathbf{V},Y}(\mathbf{v}, y) = &\sum_{k=1}^{K}\sum_{l=1}^{L} \pi_k \rho_{kl} B_l(y) \mathcal{N}(\mathbf{v}; \boldsymbol{\mu}_k, \Sigma_k) \\ &+ \pi_0 \mathcal{U}_U(\mathbf{v}) \mathcal{U}_\Omega(y). \end{aligned} \tag{13}$$

Our model (13) differs from the direct approach (8) in that it enables a separation between the point and feature components. It also differs from the independent approach (9) in that the feature component weights $\rho_{kl}$ depend on the spatial component $k$. Our model thus shares distinctiveness with the direct approach (8) and efficiency with the independent approach (9).

## 4.2. Registration

Different from the standard GMM based registration (section 3), our model includes the feature observations $y_{ij}$ and the corresponding latent feature variables $C_{ij}$. In our framework, the set of all observations is $\mathcal{X} = \{(\mathbf{x}_{ij}, y_{ij})\}_{j=1,i=1}^{N_i,M}$ and the collection of corresponding latent variables is $\mathcal{Z} = \{(Z_{ij}, C_{ij})\}_{j=1,i=1}^{N_i,M}$. The model parameters have been extended with the feature distribution weights $\rho_{kl}$ in (11), and are given as

$$\Theta = \left(\{\pi_k, \boldsymbol{\mu}_k, \Sigma_k, \rho_{k1}, \ldots, \rho_{kL}\}_{k=1}^{K}, \{R_i, \mathbf{t}_i\}_{i=1}^{M}\right). \tag{14}$$

We apply an EM procedure, as described in section 3.2, to estimate the parameters (14) of our model. The model assumptions in section 4.1.3 imply the complete data likelihood $p(\mathcal{X}, \mathcal{Z}|\Theta) = \prod_{ij} p(\mathbf{x}_{ij}, y_{ij}, c_{ij}, z_{ij}|\Theta)$, where the joint probability of an observation and its latent variables is

$$\begin{aligned} &p(\mathbf{x}_{ij}, y_{ij}, C_{ij} = l, Z_{ij} = k|\Theta) = \\ &\quad = \pi_k \rho_{kl} B_l(y_{ij}) \mathcal{N}(\phi_i(\mathbf{x}_{ij}); \boldsymbol{\mu}_k, \Sigma_k)\,,\ k \neq 0. \end{aligned} \tag{15}$$

The independence of observations imply the factorization $p(\mathcal{Z}|\mathcal{X}, \Theta^{(n)}) = \prod_{ij} p(z_{ij}, c_{ij}|\mathbf{x}_{ij}, y_{ij}, \Theta^{(n)})$. By applying (15), the latent posteriors are expressed as,[1]

$$\alpha_{ijkl}^{(n)} := p(Z_{ij} = k, C_{ij} = l|\mathbf{x}_{ij}, y_{ij}, \Theta^{(n)}) = \tag{16}$$
$$\frac{\pi_k^{(n)} \rho_{kl}^{(n)} B_l(y_{ij}) \mathcal{N}\left(\phi_i^{(n)}(\mathbf{x}_{ij}); \boldsymbol{\mu}_k^{(n)}, \Sigma_k^{(n)}\right)}{\sum_{q=1}^{K}\sum_{r=1}^{L} \pi_q^{(n)} \rho_{qr}^{(n)} B_r(y_{ij}) \mathcal{N}\left(\phi_i^{(n)}(\mathbf{x}_{ij}); \boldsymbol{\mu}_q^{(n)}, \Sigma_q^{(n)}\right) + \lambda}.$$

Here, the constant in the denominator, originating from the outlier component is given by $\lambda = \frac{\pi_0}{m(U)m(\Omega)}$, where $m$ denotes the reference measure of the space.

For our mixture model, the expected complete data log-likelihood (4) reduces to,

$$Q(\Theta; \Theta^{(n)}) = \sum_{ijkl} \alpha_{ijkl}^{(n)} \log p(\mathbf{x}_{ij}, y_{ij}, C_{ij} = l, Z_{ij} = k|\Theta). \tag{17}$$

As in section 3.2, maximization of the expected complete data log-likelihood (17) can be reformulated as an equivalent minimization problem by applying (15),[1]

$$\begin{aligned} g(\Theta; \Theta^{(n)}) = \sum_{ij}\sum_{k=1}^{K}\sum_{l=1}^{L} \alpha_{ijkl}^{(n)} \Bigg( &\frac{1}{2}\log|\Sigma_k| \\ &+ \frac{1}{2}\|R_i\mathbf{x}_{ij} + \mathbf{t}_i - \boldsymbol{\mu}_k\|_{\Sigma_k^{-1}}^2 - \log\pi_k - \log\rho_{kl}\Bigg). \end{aligned} \tag{18}$$

To simplify the expression (17), we first define the marginal latent posteriors by summing over the latent feature variable $\alpha_{ijk}^{(n)} = \sum_l \alpha_{ijkl}^{(n)}$. This enables our loss (18) to be rewritten as,

[1] See the supplementary material for a detailed derivation.

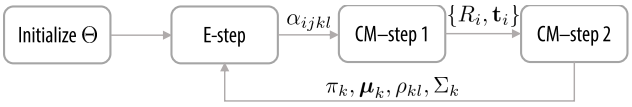



Figure 3. Overview of our EM-based registration. The parameters updated after each step are indicated on the arrow.

$$g(\Theta;\Theta^{(n)}) = f(\Theta;\Theta^{(n)}) - \sum_{ij}\sum_{k=1}^{K}\sum_{l=1}^{L}\alpha_{ijkl}^{(n)}\log\rho_{kl}. \quad (19)$$

Here, $f(\Theta;\Theta^{(n)})$ is the corresponding loss (7) in the standard GMM-based registration. This implies that the transformation parameters $(R_i, \mathbf{t}_i)$ and the spatial mixture parameters $(\pi_k, \boldsymbol{\mu}_k, \Sigma_k)$ can be obtained as in section 3.2. However, in our method, the latent posteriors given by (16) are used in the M-step. Different from section 3, our marginal latent posteriors $\alpha_{ijk}^{(n)}$ thus also integrate feature information into the EM-procedure. Finally, the feature distribution weights are obtained by minimizing the second term in (19) using Lagrangian multipliers,[1]

$$\rho_{kl}^{(n)} = \frac{\sum_{ij}\alpha_{ijkl}^{(n)}}{\sum_{ij}\alpha_{ijk}^{(n)}}, \quad k = 1,\dots,K. \quad (20)$$

We incorporate the estimation of the feature distribution parameters (20) in the second CM-step (see section 3.2), along with the estimation of the other mixture parameters. Figure 3 shows an overview of our approach.

### 4.3. Feature Description

Here, we provide a detailed description of how the distribution of features is modeled, by the selection of feature mixture components $B_l$. We restrict our discussion to color features. In our model, the feature observations are represented by an HSV triplet $y = (y^H, y^S, y^V) \in \Omega = [0,1]^3$. In this work, we use second order B-splines to construct the feature components $B_l$. However, other functions with similar characteristics can also be used. Each component $B_l$ is a separable function $B_l(y) = a_l B_l^1(y^H) B_l^2(y^S) B_l^3(y^V)$. In each dimension, the component is given by a scaled and shifted second order B-spline function $B_l^i$. The constant $a_l$ is a normalization factor given by the condition $\int_\Omega B_l = 1$. The components $B_l$ are placed in a regular grid inside the unit cube $\Omega = [0,1]^3$. The spacing between the components is set to $1/L_d$ along feature dimension $d$, where $L_d$ denotes the number of components in dimension $d$. The total number of components is hence $L = \prod_d L_d$.

Similar to GMMs, our method is able to model multimodal color distributions. However, our choice of nonparametric mixture components $B_l$ is computationally beneficial. In contrast, employing a standard GMM in the color space requires computation of the color means and covariances in the EM-procedure. Our approach further allows the probabilities $B_l(y_{ij})$ to be precomputed for all points.

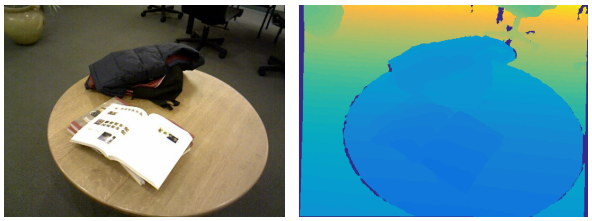


Figure 4. An RGB-D frame from the Stanford Lounge dataset, containing the RGB image (left) and the depth (right).

## 5. Experiments

We perform a comprehensive quantitative and qualitative evaluations on one RGB-D and two Lidar datasets.

### 5.1. Details and Parameters

We use the same number of spatial components $K = 500$, the same outlier ratio $\pi_0 = 0.005$ and 100 EM-iterations for both the standard JRMPS and our color-based versions. We also initialize all methods with the same parameters for the spatial GMM. The initial means $\boldsymbol{\mu}_k^{(0)}$ are uniformly sampled on a sphere with the radius equal to the standard deviation of the point distribution. As in [5], we fix the spatial component weights $\pi_k$ to uniform, since we did not observe any improvement in updating them. The feature component weights $\rho_{kl}$ are initialized by uniformly sampling the $L-1$ simplex for each $k$. Our approach is implemented in Matlab. Compared to the baseline JRMPS, our approach marginally increases the computation time (25 to 27 sec. on a single core), for 2000 points per set.

For the direct approach, presented in section 4.1.1, the joint components are constructed as products of a spatial Gaussian and a feature component $F(\mathbf{v}, y; \theta_k) = \mathcal{N}(\mathbf{v}; \boldsymbol{\mu}_k, \Sigma_k) B_{l_k}(y)$. Here, $B_{l_k}$ is constructed as in section 4.3, and the index $l_k \in \{1,\dots,L\}$ is selected randomly for each component $k$. For the independent approach (section 4.1.2), we also set the feature components based on the B-splines presented in section 4.3. That is, we set $\tilde{F}(y;\tilde{\theta}_l) = B_l(y)$ in (9). For all methods, we use $L_d = 4$ feature components in each dimension of the HSV space, which gives $L = 64$ feature components in total. For both the direct and independent approaches, we also employ the additional uniform outlier component (see section 3.1).

**Evaluation Criteria:** We compute the rotation errors compared to the ground truth by measuring the Frobenius distance between rotation matrices [5]. The rotation error is defined as $\|\hat{R} - R\|_F$, where $\hat{R}$ and $R$ are the estimated and ground-truth relative rotations between two point sets.

### 5.2. Stanford Lounge Dataset

We perform experiments on the Stanford Lounge Dataset [19], consisting of 3000 RGB-D frames taken by a Kinect. Figure 4 contains an example frame. We use the estimated poses, provided by the authors, as ground truth.

| | Avg. error | Std. dev. | Failure rate (%) |
|---|---|---|---|
| ICP [1] | $4.32 \cdot 10^{-2}$ | $2.53 \cdot 10^{-2}$ | 15.70 |
| GMMReg [9] | $6.09 \cdot 10^{-2}$ | $2.31 \cdot 10^{-2}$ | 59.04 |
| Color GICP [11] | $1.72 \cdot 10^{-2}$ | $1.75 \cdot 10^{-2}$ | 1.27 |
| JRMPS [5] | $1.68 \cdot 10^{-2}$ | $1.24 \cdot 10^{-2}$ | 3.41 |
| Direct Approach | $1.91 \cdot 10^{-2}$ | $1.30 \cdot 10^{-2}$ | 2.14 |
| Independent Approach | $1.68 \cdot 10^{-2}$ | $1.24 \cdot 10^{-2}$ | 3.41 |
| **Our Approach** | $\mathbf{1.47} \cdot 10^{-2}$ | $\mathbf{1.01} \cdot 10^{-2}$ | **0.74** |

Table 1. A comparison with other registration methods on the Stanford Lounge dataset. We report the failure rate along with the average and standard deviation of the inlier rotation errors. Compared to the baseline JRMPS [5], our approach achieves significantly better robustness with a relative reduction in the failure rate by 78%. Further, our approach outperforms other color based methods, including Color GICP [11].

### 5.2.1 Pairwise Registration

We compare our approach with several state-of-the-art methods with publicly available code, namely ICP[2] [1], GMMReg [9], Color GICP[3] [11], and the baseline JRMPS [5]. To ensure a significant initial transformation, we perform registration between frame number $n$ and $n+5$, for all frames $n$ in the dataset. We randomly downsample the frames to 10000 points. As a measure of robustness, we report the failure rate defined as the percentage of rotation errors larger than $0.1$ (approximately 4 degrees). We further define a registration to be an inlier if the error is smaller than $0.1$. We compute the average and standard deviation of the inlier rotation errors, as measures of accuracy.

The results are reported in Table 1. The standard ICP obtains inferior performance with a failure rate of $15.7\%$. The baseline JRMPS achieves a failure rate of $3.41\%$. The Color GICP provides competitive results with a failure rate of $1.27\%$. The two standard color extensions, using the independent and direct approaches, provides the failure rates $3.41\%$ and $2.14\%$ respectively. Our approach achieves the best results on this dataset, with a failure rate of $0.74\%$. Additionally, our method obtains a significant reduction of the average rotation error by $12.5\%$ compared to JRMPS.

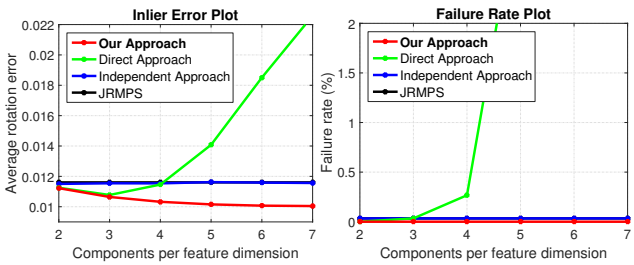



Figure 5. An analysis of the number of feature mixture components $L$, on the Stanford Lounge dataset. We compare our approach with the baseline JRMPS and the two standard color-extensions. We show the average inlier rotation error (left) and failure rate (right) for different numbers of components per feature dimension $L_d$ in the HSV space. Our approach provides consistent improvements compared to the other probabilistic approaches.

| | Avg. error | Std. dev. | Failure rate (%) |
|---|---|---|---|
| JRMPS [5] | $0.913 \cdot 10^{-2}$ | $0.636 \cdot 10^{-2}$ | 0.467 |
| **Ours** | $\mathbf{0.768} \cdot 10^{-2}$ | $\mathbf{0.539} \cdot 10^{-2}$ | **0.067** |

Table 2. A comparison of joint multi-view registration on the Stanford Lounge dataset, in terms of average inlier error, standard deviation and failure rate. Our approach significantly reduces the relative failure rate with 86% compared to JRMPS.

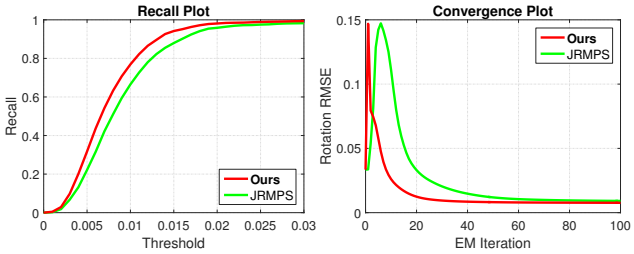


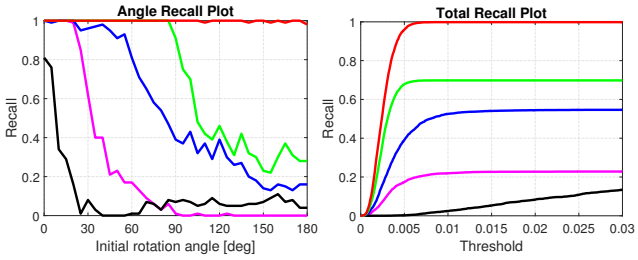

Figure 6. A joint multi-view registration comparison of our method with JRMPS [5] on the Stanford Lounge dataset. The recall plot (left) shows the fraction of correct registrations over a range of rotation-error thresholds. The convergence plot (right) shows the average frame-to-frame inlier rotation error after each EM iteration. Our method demonstrates superior accuracy and robustness, while achieving faster convergence.

In figure 5 we investigate the impact of varying the number of feature components $L$ on the Stanford Lounge dataset, when using 2000 points per set.[4] The left plot shows the average frame-to-frame rotation error for inliers, when increasing the number of components per HSV-dimension from 2 to 7. As a reference, we also include the baseline JRMPS. The independent approach (section 4.1.2) provides similar results to JRMPS. The direct approach (section 4.1.1), requires a larger amount of data points when increasing the number of feature components. The performance therefore rapidly degrades as the number of feature components is increased. Contrary to this, our model benefits from increasing the number of feature components, leading to improved results.

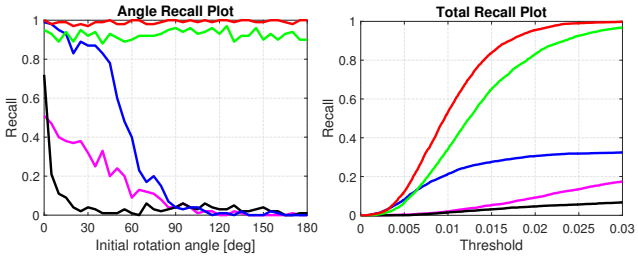
### 5.2.2 Joint Multi-view Registration

Here, we investigate the performance of our approach for joint registration of multiple point sets. Alignment of multiple point sets is important in many applications. Most registration methods [1, 7, 11] are however limited to pairwise registration. In these cases, multi-view registration must be performed either by sequential pair-wise alignment or by performing a one-versus-all strategy, leading to drift or biased solutions. Similar to JRMPS [5], our method is able to jointly register an arbitrary number of point sets. We perform joint registration of every 10 consecutive frames, with

[2] We use the built-in MATLAB implementation of ICP.
[3] We use the Color GICP implemented in Point Cloud Library.
[4] Analysis of $K$ and $\pi_0$ is provided in the supplementary material.




(a) Lidar Indoor dataset.




(b) Lidar Outdoor dataset.

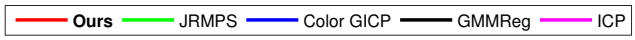


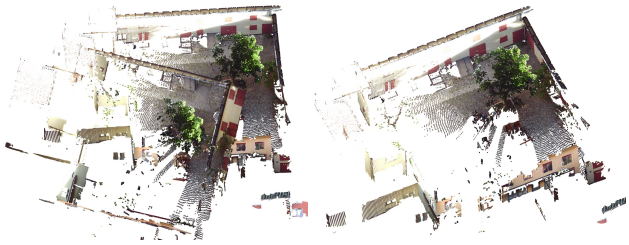

Figure 7. Initialization robustness comparison on the Lidar Indoor (a) and Outdoor (b) datasets. The left plots show the recall at a threshold of $0.025$. The recall is computed over 100 randomly sampled rotation axes for each angle. The right plots contain the total recall over all registrations, plotted with respect to the error threshold. Compared to previous methods, our approach provides superior robustness, while maintaining the accuracy.

an interval of 9 frames, on the Stanford Lounge dataset. This implies that joint multi-view registration is performed on frame 1-10, 10-19, etc. Table 2 contains the results, by measuring the frame-to-frame rotation errors. Our color based model reduces the relative failure rate by $86\%$ compared to the baseline JRMPS. In case of average rotation error, our approach provides a significant reduction of $15.9\%$.

Figure 6 shows the recall and convergence rate plots. Recall is computed as the fraction of frame-to-frame rotation errors smaller than a threshold. In figure 6, the recall is plotted over a range of error thresholds. To compare the convergence rate of our method with the baseline JRMPS, we plot the average frame-to-frame inlier rotation error after each EM iteration. Our method converges in significantly fewer iterations, enabling a more efficient registration.

### 5.3. Lidar Datasets

We experimented with two Lidar datasets, acquired by a FARO Focus3D. Both consist of more than a million colored 3D points in a single 360 degree view. The Indoor dataset is visualized in figure 1 and the Outdoor dataset is visualized in figure 8. We compare with state-of-the-art methods by evaluating the robustness to initial rotation errors. Registration is performed using initial rotation errors between 0 and 180 degrees with an interval of 5 degrees. For every angle, we uniformly sample 100 random rotation



(a) Color GICP. (b) Ours.

Figure 8. Registration of an outdoor scene captured by a Lidar. Color GICP (a) fails to register the point sets due to a large initial transformation. Our approach (b) accurately register the point sets.

axes. The point sets are constructed by randomly sampling points from the single Lidar scan. For each transformation, we sample two sets with 2000 points each. One of the sets is then transformed with the rotation defined by its corresponding axis and angle. We plot the recall at a rotation error threshold of $0.025$ (approximately 1 degree) with respect to the initial angle. We also compare the total recall over all registrations.

**Lidar Indoor Dataset:** Figure 7a shows the angle robustness comparison in terms of angle recall and total recall. ICP, GMMreg and Color GICP struggle for initial angles larger than 60 degrees. The robustness of JRMSP starts to degrade at an initial angle of 90 degrees. Our approach provides consistent registrations for angles up to 180 degrees.

**Lidar Outdoor Dataset:** Figure 7b shows the initial angle robustness comparison on the Lidar Outdoor dataset. As in the Indoor dataset, the ICP and Color GICP provides inferior results due to large initial transformations. Our approach provides consistent improvements compared to the JRMPS. Figure 8 shows a qualitative comparison between Color GICP and our approach on this dataset.

## 6. Conclusions

In this work, we propose a novel probabilistic approach to incorporate color information for point set registration. Our method is based on constructing an efficient mixture model for the joint point-color observation space. An EM algorithm is then derived to estimate the parameters of the mixture model and the relative transformations.

Experiments are performed on three challenging datasets. Our results clearly demonstrate that color information improves accuracy and robustness for point set registration. We show that a careful integration of spatial and color information is crucial to obtain optimal performance. Our approach exploits the discriminative color information associated with each point, while preserving efficiency.

**Acknowledgments**: This work has been supported by SSF (VPS), VR (EMC$^2$), Vinnova (iQMatic), EU's Horizon 2020 R&I program grant No 644839, the Wallenberg Autonomous Systems Program, the NSC and Nvidia.

## References


[1] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *PAMI*, 14(2):239–256, 1992.

[2] D. Chetverikov, D. Stepanov, and P. Krsek. Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. *IMAVIS*, 23(3):299–309, 2005.

[3] S. Choi, Q. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015.

[4] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *CVPR*, 2010.

[5] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, and E. Z. Psarakis. A generative model for the joint registration of multiple point sets. In *ECCV*, 2014.

[6] Z. Fang and S. Scherer. Real-time onboard 6dof localization of an indoor mav in degraded visual environments using a rgb-d camera. In *ICRA*, 2015.

[7] R. Horaud, F. Forbes, M. Yguel, G. Dewaele, and J. Zhang. Rigid and articulated point registration with expectation conditional maximization. *PAMI*, 33(3):587–602, 2011.

[8] B. Huhle, M. Magnusson, W. Straßer, and A. J. Lilienthal. Registration of colored 3d point clouds with a kernel-based extension to the normal distributions transform. In *ICRA*, 2008.

[9] B. Jian and B. C. Vemuri. Robust point set registration using gaussian mixture models. *PAMI*, 33(8):1633–1645, 2011.

[10] A. E. Johnson and S. B. Kang. Registration and integration of textured 3d data. *IMAVIS*, 17(2):135–147, 1999.

[11] M. Korn, M. Holzkothen, and J. Pauli. Color supported generalized-icp. In *VISAPP*, 2014.

[12] H. Men, B. Gebre, and K. Pochiraju. Color point cloud registration with 4d ICP algorithm. In *ICRA*, 2011.

[13] X. L. Meng and D. B. Rubin. Maximum Likelihood Estimation via the ECM Algorithm: A General Framework. *Biometrika*, 80(2):267–278, 1993.

[14] A. Myronenko and X. B. Song. Point set registration: Coherent point drift. *PAMI*, 32(12):2262–2275, 2010.

[15] A. Rangarajan, H. Chui, and F. L. Bookstein. The softassign procrustes matching algorithm. In *IPMI*, 1997.

[16] A. Segal, D. Hähnel, and S. Thrun. Generalized-icp. In *RSS*, 2009.

[17] Y. Tsin and T. Kanade. A correlation-based approach to robust point set registration. In *ECCV*, 2004.

[18] J. Unger, A. Gardner, P. Larsson, and F. Banterle. Capturing reality for computer graphics applications. In *Siggraph Asia Course*, 2015.

[19] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Trans. Graph.*, 32(4):112:1–112:8, 2013.

# Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking

Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, Michael Felsberg

CVL, Department of Electrical Engineering, Linköping University, Sweden
{martin.danelljan, andreas.robinson, fahad.khan, michael.felsberg}@liu.se

**Abstract.** Discriminative Correlation Filters (DCF) have demonstrated excellent performance for visual object tracking. The key to their success is the ability to efficiently exploit available negative data by including all shifted versions of a training sample. However, the underlying DCF formulation is restricted to single-resolution feature maps, significantly limiting its potential. In this paper, we go beyond the conventional DCF framework and introduce a novel formulation for training *continuous* convolution filters. We employ an implicit interpolation model to pose the learning problem in the continuous spatial domain. Our proposed formulation enables efficient integration of multi-resolution deep feature maps, leading to superior results on three object tracking benchmarks: OTB-2015 (+5.1% in mean OP), Temple-Color (+4.6% in mean OP), and VOT2015 (20% relative reduction in failure rate). Additionally, our approach is capable of sub-pixel localization, crucial for the task of accurate feature point tracking. We also demonstrate the effectiveness of our learning formulation in extensive feature point tracking experiments.

## 1 Introduction

Visual tracking is the task of estimating the trajectory of a target in a video. It is one of the fundamental problems in computer vision. Tracking of objects or feature points has numerous applications in robotics, structure-from-motion, and visual surveillance. In recent years, Discriminative Correlation Filter (DCF) based approaches have shown outstanding results on object tracking benchmarks [30, 46]. DCF methods train a correlation filter for the task of predicting the target classification scores. Unlike other methods, the DCF efficiently utilize all spatial shifts of the training samples by exploiting the discrete Fourier transform.

Deep convolutional neural networks (CNNs) have shown impressive performance for many tasks, and are therefore of interest for DCF-based tracking. A CNN consists of several layers of convolution, normalization and pooling operations. Recently, activations from the last convolutional layers have been successfully employed for image classification. Features from these deep convolutional layers are discriminative while preserving spatial and structural information. Surprisingly, in the context of tracking, recent DCF-based methods [10, 35] have demonstrated the importance of shallow convolutional layers. These layers provide higher spatial resolution, which is crucial for accurate target localization. However, fusing multiple layers in a DCF framework is still an open problem.

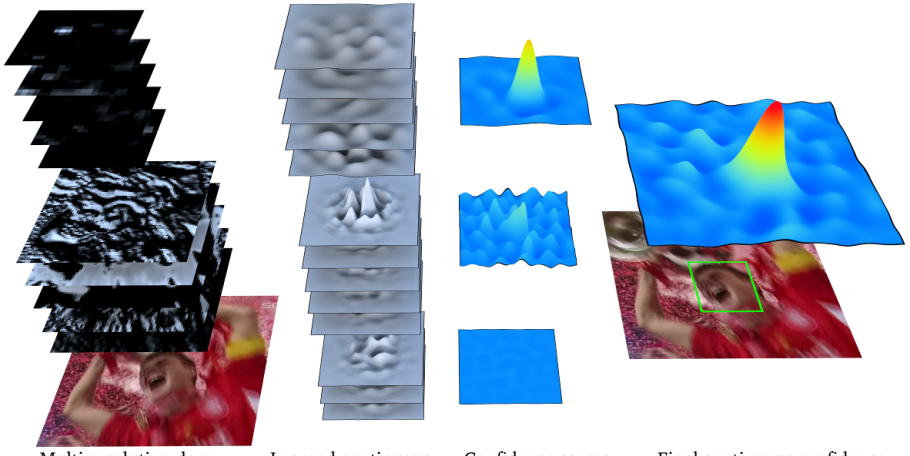



**Fig. 1.** Visualization of our continuous convolution operator, applied to a multi-resolution deep feature map. The feature map (*left*) consists of the input RGB patch along with the first and last convolutional layer of a pre-trained deep network. The second column visualizes the continuous convolution filters learned by our framework. The resulting continuous convolution outputs for each layer (third column) are combined into the final continuous confidence function (*right*) of the target (green box).

The conventional DCF formulation is limited to a single-resolution feature map. Therefore, all feature channels must have the same spatial resolution, as in e.g. the HOG descriptor. This limitation prohibits joint fusion of multiple convolutional layers with different spatial resolutions. A straightforward strategy to counter this restriction is to explicitly resample all feature channels to the same common resolution. However, such a resampling strategy is both cumbersome, adds redundant data and introduces artifacts. Instead, a principled approach for integrating multi-resolution feature maps in the learning formulation is preferred.

In this work, we propose a novel formulation for learning a convolution operator in the *continuous* spatial domain. The proposed learning formulation employs an implicit interpolation model of the training samples. Our approach learns a set of convolution filters to produce a *continuous-domain* confidence map of the target. This enables an elegant fusion of multi-resolution feature maps in a joint learning formulation. Figure 1 shows a visualization of our continuous convolution operator, when integrating multi-resolution deep feature maps. We validate the effectiveness of our approach on three object tracking benchmarks: OTB-2015 [46], Temple-Color [32] and VOT2015 [29]. On the challenging OTB-2015 with 100 videos, our object tracking framework improves the state-of-the-art from 77.3% to 82.4% in mean overlap precision.

In addition to multi-resolution fusion, our continuous domain learning formulation enables accurate sub-pixel localization. This is achieved by labeling the training samples with sub-pixel precise continuous confidence maps. Our formu-

lation is therefore also suitable for accurate feature point tracking. Further, our learning-based approach is discriminative and does not require explicit interpolation of the image to achieve sub-pixel accuracy. We demonstrate the accuracy and robustness of our approach by performing extensive feature point tracking experiments on the popular MPI Sintel dataset [7].

## 2 Related Work

Discriminative Correlation Filters (DCF) [5, 11, 24] have shown promising results for object tracking. These methods exploit the properties of circular correlation for training a regressor in a sliding-window fashion. Initially, the DCF approaches [5, 23] were restricted to a single feature channel. The DCF framework was later extended to multi-channel feature maps [4, 13, 17]. The multi-channel DCF allows high-dimensional features, such as HOG and Color Names, to be incorporated for improved tracking. In addition to the incorporation of multi-channel features, the DCF framework has been significantly improved lately by, e.g., including scale estimation [9, 31], non-linear kernels [23, 24], a long-term memory [36], and by alleviating the periodic effects of circular convolution [11, 15, 18].

With the advent of deep CNNs, fully connected layers of the network have been commonly employed for image representation [38, 43]. Recently, the last (deep) convolutional layers were shown to be more beneficial for image classification [8, 33]. On the other hand, the first (shallow) convolutional layer was shown to be more suitable for visual tracking, compared to the deeper layers [10]. The deep convolutional layers are discriminative and possess high-level visual information. In contrast, the shallow layers contain low-level features at high spatial resolution, beneficial for localization. Ma et al. [35] employed multiple convolutional layers in a hierarchical ensemble of independent DCF trackers. Instead, we propose a novel continuous formulation to fuse multiple convolutional layers with different spatial resolutions in a *joint* learning framework.

Unlike object tracking, feature point tracking is the task of accurately estimating the motion of distinctive key-points. It is a core component in many vision systems [1, 27, 39, 48]. Most feature point tracking methods are derived from the classic Kanade-Lucas-Tomasi (KLT) tracker [34, 44]. The KLT tracker is a generative method, that is based on minimizing the squared sum of differences between two image patches. In the last decades, significant effort has been spent on improving the KLT tracker [2, 16]. In contrast, we propose a discriminative learning based approach for feature point tracking.

**Our approach:** Our main contribution is a theoretical framework for learning discriminative convolution operators in the continuous spatial domain. Our formulation has two major advantages compared to the conventional DCF framework. Firstly, it allows a natural integration of multi-resolution feature maps, e.g. combinations of convolutional layers or multi-resolution HOG and color features. This property is especially desirable for object tracking, detection and action recognition applications. Secondly, our continuous formulation enables accurate sub-pixel localization, crucial in many feature point tracking problems.

## 3 Learning Continuous Convolution Operators

In this section, we present a theoretical framework for learning continuous convolution operators. Our formulation is generic and can be applied for supervised learning tasks, such as visual tracking and detection.

### 3.1 Preliminaries and Notation

In this paper, we utilize basic concepts and results in continuous Fourier analysis. For clarity, we first formulate our learning method for data defined in a one-dimensional domain, i.e. for functions of a single spatial variable. We then describe the generalization to higher dimensions, including images, in section 3.5.

We consider the space $L^2(T)$ of complex-valued functions $g : \mathbb{R} \to \mathbb{C}$ that are periodic with period $T > 0$ and square Lebesgue integrable. The space $L^2(T)$ is a Hilbert space equipped with an inner product $\langle \cdot, \cdot \rangle$. For functions $g, h \in L^2(T)$,

$$\langle g, h \rangle = \frac{1}{T} \int_0^T g(t)\overline{h(t)}\,\mathrm{d}t\,, \qquad g * h(t) = \frac{1}{T} \int_0^T g(t-s)h(s)\,\mathrm{d}s\,. \tag{1}$$

Here, the bar denotes complex conjugation. In (1) we have also defined the circular convolution operation $* : L^2(T) \times L^2(T) \to L^2(T)$.

In our derivations, we use the complex exponential functions $e_k(t) = e^{i\frac{2\pi}{T}kt}$ since they are eigenfunctions of the convolution operation (1). The set $\{e_k\}_{-\infty}^{\infty}$ further forms an orthonormal basis for $L^2(T)$. We define the Fourier coefficients of $g \in L^2(T)$ as $\hat{g}[k] = \langle g, e_k \rangle$. For clarity, we use square brackets for functions with discrete domains. Any $g \in L^2(T)$ can be expressed in terms of its Fourier series $g = \sum_{-\infty}^{\infty} \hat{g}[k]e_k$. The Fourier coefficients satisfy Parseval's formula $\|g\|^2 = \|\hat{g}\|_{\ell^2}^2$, where $\|g\|^2 = \langle g, g \rangle$ and $\|\hat{g}\|_{\ell^2}^2 = \sum_{-\infty}^{\infty} |\hat{g}[k]|^2$ is the squared $\ell^2$-norm. Further, the Fourier coefficients satisfy the two convolution properties $\widehat{g * h} = \hat{g}\hat{h}$ and $\widehat{gh} = \hat{g} * \hat{h}$, where $\hat{g} * \hat{h}[k] := \sum_{l=-\infty}^{\infty} \hat{g}[k-l]\hat{h}[l]$.

### 3.2 Our Continuous Learning Formulation

Here we formulate our novel learning approach. The aim is to train a continuous convolution operator based on training samples $x_j$. The samples consist of feature maps extracted from image patches. Each sample $x_j$ contains $D$ feature channels $x_j^1, \ldots, x_j^D$, extracted from the same image patch. Conventional DCF formulations [11, 17, 24] assume the feature channels to have the same spatial resolution, i.e. have the same number of spatial sample points. Unlike previous works, we eliminate this restriction in our formulation and let $N_d$ denote the number of spatial samples in $x_j^d$. In our formulation, the feature channel $x_j^d \in \mathbb{R}^{N_d}$ is viewed as a function $x_j^d[n]$ indexed by the discrete spatial variable $n \in \{0, \ldots, N_d - 1\}$. The sample space is expressed as $\mathcal{X} = \mathbb{R}^{N_1} \times \ldots \times \mathbb{R}^{N_D}$.

To pose the learning problem in the continuous spatial domain, we introduce an implicit interpolation model of the training samples. We regard the continuous

interval $[0, T) \subset \mathbb{R}$ to be the spatial support of the feature map. Here, the scalar $T$ represents the size of the support region. In practice, however, $T$ is arbitrary since it represents the scaling of the coordinate system. For each feature channel $d$, we define the interpolation operator $J_d : \mathbb{R}^{N_d} \to L^2(T)$ of the form,

$$J_d\{x^d\}(t) = \sum_{n=0}^{N_d-1} x^d[n] b_d \left( t - \frac{T}{N_d} n \right). \tag{2}$$

The interpolated sample $J_d\{x^d\}(t)$ is constructed as a superposition of shifted versions of an interpolation function $b_d \in L^2(T)$. In (2), the feature values $x^d[n]$ act as weights for each shifted function. Similar to the periodic assumption in the conventional discrete DCF formulation, a periodic extension of the feature map is also performed here in (2).

As discussed earlier, our objective is to learn a linear convolution operator $S_f : \mathcal{X} \to L^2(T)$. This operator maps a sample $x \in \mathcal{X}$ to a target confidence function $s(t) = S_f\{x\}(t)$, defined on the continuous interval $[0, T)$. Here, $s(t) \in \mathbb{R}$ is the confidence score of the target at the location $t \in [0, T)$ in the image. Similar to other discriminative methods, the target is localized by maximizing the confidence scores in an image region. The key difference in our formulation is that the confidences are defined on a continuous spatial domain. Therefore, our formulation can be used to localize the target with higher accuracy.

In our continuous formulation, the operator $S_f$ is parametrized by a set of convolution filters $f = (f^1, \dots, f^D) \in L^2(T)^D$. Here, $f^d \in L^2(T)$ is the continuous filter for feature channel $d$. We define the convolution operator as,

$$S_f\{x\} = \sum_{d=1}^{D} f^d * J_d\{x^d\}, \quad x \in \mathcal{X}. \tag{3}$$

Here, each feature channel is first interpolated using (2) and then convolved with its corresponding filter. Note that the convolutions are performed in the continuous domain, as defined in (1). In the last step, the convolution responses from all filters are summed to produce the final confidence function.

In the standard DCF, each training sample is labeled by a discrete function that represents the desired convolution output. In contrast, our samples $x_j \in \mathcal{X}$ are labeled by confidence functions $y_j \in L^2(T)$, defined in the continuous spatial domain. Here, $y_j$ is the desired output of the convolution operator $S_f\{x_j\}$ applied to the training sample $x_j$. This enables sub-pixel accurate information to be incorporated in the learning. The filter $f$ is trained, given a set of $m$ training sample pairs $\{(x_j, y_j)\}_1^m \subset \mathcal{X} \times L^2(T)$, by minimizing the functional,

$$E(f) = \sum_{j=1}^{m} \alpha_j \left\| S_f\{x_j\} - y_j \right\|^2 + \sum_{d=1}^{D} \left\| w f^d \right\|^2. \tag{4}$$

Here, the weights $\alpha_j \geq 0$ control the impact of each training sample. We additionally include a spatial regularization term, similar to [11], determined by

the penalty function $w$. This regularization enables the filter to be learned on arbitrarily large image regions by controlling the spatial extent of the filter $f$. Spatial regions typically corresponding to background features are assigned a large penalty in $w$, while the target region has small penalty values. Thus, $w$ encodes the prior reliability of features depending on their spatial location. Unlike [11], the penalty function $w$ is defined on the whole continuous interval $[0, T)$ and periodically extended to $w \in L^2(T)$. Hence, $\|wf^d\| < \infty$ is required in (4). This is implied by our later assumption of $w$ having finitely many non-zero Fourier coefficients $\hat{w}[k]$. Next, we derive the procedure to train the continuous filter $f$, using the proposed formulation (4).

### 3.3 Training the Continuous Filter

To train the filter $f$, we minimize the functional (4) in the Fourier domain. By using results from Fourier analysis it can be shown[1] that the Fourier coefficients of the interpolated feature map are given by $\widehat{J_d\{x^d\}}[k] = X^d[k]\hat{b}_d[k]$. Here, $X^d[k] := \sum_{n=0}^{N_d-1} x^d[n] e^{-i\frac{2\pi}{N_d}nk}$, $k \in \mathbb{Z}$ is the discrete Fourier transform (DFT) of $x^d$. By using linearity and the convolution property in section 3.1, the Fourier coefficients of the output confidence function (3) are derived as

$$\widehat{S_f\{x\}}[k] = \sum_{d=1}^{D} \hat{f}^d[k] X^d[k] \hat{b}_d[k]\,, \quad k \in \mathbb{Z}\,. \tag{5}$$

By applying Parseval's formula to (4) and using (5), we obtain

$$E(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} \hat{f}^d X_j^d \hat{b}_d - \hat{y}_j \right\|_{\ell^2}^2 + \sum_{d=1}^{D} \left\| \hat{w} * \hat{f^d} \right\|_{\ell^2}^2 . \tag{6}$$

Hence, the functional $E(f)$ can equivalently be minimized with respect to the Fourier coefficients $\hat{f}^d[k]$ for each filter $f^d$. We exploit the Fourier domain formulation (6) to minimize the original loss (4).

For practical purposes, the filter $f$ needs to be represented by a finite set of parameters. One approach is to employ a parametric model to represent an infinite number of coefficients. In this work, we instead obtain a finite representation by minimizing (6) over the finite–dimensional subspace $V = \operatorname{span}\{e_k\}_{-K_1}^{K_1} \times \ldots \times \operatorname{span}\{e_k\}_{-K_D}^{K_D} \subset L^2(T)^D$. That is, we minimize (6) with respect to the coefficients $\{\hat{f}^d[k]\}_{-K_d}^{K_d}$, while assuming $\hat{f}^d[k] = 0$ for $|k| > K_d$. In practice, $K_d$ determines the number of filter coefficients $\hat{f}^d[k]$ to be computed for feature channel $d$ during learning. Increasing $K_d$ leads to a better estimate of the filter $f^d$ at the cost of increased computations and memory consumption. In our experiments, we set $K_d = \left\lfloor \frac{N_d}{2} \right\rfloor$ such that the number of stored filter coefficients for channel $d$ equals the spatial resolution $N_d$ of the training sample $x^d$.

[1] See the supplementary material for a detailed derivation.

To derive the solution to the minimization problem (6) subject to $f \in V$, we introduce the vector of non-zero Fourier coefficients $\hat{\mathbf{f}}^d = (\hat{f}^d[-K_d] \cdots \hat{f}^d[K_d])^{\mathrm{T}} \in \mathbb{C}^{2K_d+1}$ and define the coefficient vector $\hat{\mathbf{f}} = \left[(\hat{\mathbf{f}}^1)^{\mathrm{T}} \cdots (\hat{\mathbf{f}}^D)^{\mathrm{T}}\right]^{\mathrm{T}}$. Further, we define $\hat{\mathbf{y}}_j = (\hat{y}_j[-K] \cdots \hat{y}_j[K])^{\mathrm{T}}$ be the vectorization of the $K := \max_d K_d$ first Fourier coefficients of $y_j$. To simplify the regularization term in (6), we let $L$ be the number of non-zero coefficients $\hat{w}[k]$, such that $\hat{w}[k] = 0$ for all $|k| > L$. We further define $W_d$ to be the $(2K_d + 2L + 1) \times (2K_d + 1)$ Toeplitz matrix corresponding to the convolution operator $W_d \hat{\mathbf{f}}^d = \operatorname{vec} \hat{w} * \hat{f}^d$. Finally, let $W$ be the block-diagonal matrix $W = W_1 \oplus \cdots \oplus W_D$. The minimization of the functional (6) subject to $f \in V$ is equivalent to the following least squares problem,

$$E_V(\hat{\mathbf{f}}) = \sum_{j=1}^{m} \alpha_j \left\| A_j \hat{\mathbf{f}} - \hat{\mathbf{y}}_j \right\|_2^2 + \left\| W \hat{\mathbf{f}} \right\|_2^2 . \tag{7}$$

Here, the matrix $A_j = [A_j^1 \cdots A_j^D]$ has $2K + 1$ rows and contains one diagonal block $A_j^d$ per feature channel $d$ with $2K_d + 1$ columns containing the elements $\{X_j^d[k]\hat{b}_d[k]\}_{-K_d}^{K_d}$. In (7), $\|\cdot\|_2$ denotes the standard Euclidian norm in $\mathbb{C}^M$.

To obtain a simple expression of the normal equations, we define the sample matrix $A = [A_1^{\mathrm{T}} \cdots A_D^{\mathrm{T}}]^{\mathrm{T}}$, the diagonal weight matrix $\Gamma = \alpha_1 I \oplus \cdots \oplus \alpha_D I$ and the label vector $\hat{\mathbf{y}} = [\hat{\mathbf{y}}_1^{\mathrm{T}} \cdots \hat{\mathbf{y}}_D^{\mathrm{T}}]^{\mathrm{T}}$. The minimizer of (7) is found by solving the normal equations,

$$\left(A^{\mathrm{H}} \Gamma A + W^{\mathrm{H}} W\right) \hat{\mathbf{f}} = A^{\mathrm{H}} \Gamma \hat{\mathbf{y}} . \tag{8}$$

Here, $^{\mathrm{H}}$ denotes the conjugate-transpose of a matrix. Note that (8) forms a sparse linear equation system if $w$ has a small number of non-zero Fourier coefficients $\hat{w}[k]$. In our object tracking framework, presented in section 4.2, we employ the Conjugate Gradient method to iteratively solve (8). For our feature point tracking approach, presented in section 4.3, we use a single-channel feature map and a constant penalty function $w$ for improved efficiency. This results in a diagonal system (8), which can be efficiently solved by a direct computation.

### 3.4 Desired Confidence and Interpolation Function

Here, we describe the choice of the desired convolution output $y_j$ and the interpolation function $b_d$. We construct both $y_j$ and $b_d$ by periodically repeating functions defined on the real line. In general, the $T$-periodic repetition of a function $g$ is defined as $g_T(t) = \sum_{-\infty}^{\infty} g(t - nT)$. In the derived Fourier domain formulation (6), the functions $y_j$ and $b_d$ are represented by their respective Fourier coefficients. The Fourier coefficients of a periodic repetition $g_T$ can be retrieved from the continuous Fourier transform $\hat{g}(\xi)$ of $g(t)$ as $\hat{g}_T[k] = \frac{1}{T}\hat{g}(\frac{k}{T})$.[2] We use this property to compute the Fourier coefficients of $y_j$ and $b_d$.

To construct the desired convolution output $y_j$, we let $u_j \in [0, T)$ denote the estimated location of the target object or feature point in sample $x_j$. We define $y_j$ as the periodic repetition of the Gaussian function $\exp\left(-\frac{(t-u_j)^2}{2\sigma^2}\right)$ centered

at $u_j$. This provides the following expression for the Fourier coefficients,

$$\hat{y}_j[k] = \frac{\sqrt{2\pi\sigma^2}}{T} \exp\left(-2\sigma^2 \left(\frac{\pi k}{T}\right)^2 - i\frac{2\pi}{T} u_j k\right) . \tag{9}$$

The variance $\sigma^2$ is set to a small value to obtain a sharp peak. Further, this ensures a negligible spatial aliasing. In our work, the functions $b_d$ are constructed based on the cubic spline kernel $b(t)$. The interpolation function $b_d$ is set to the periodic repetition of a scaled and shifted version of the kernel $b\big(\frac{N_d}{T}\big(t - \frac{T}{2N_d}\big)\big)$, to preserve the spatial arrangement of the feature pyramid. The Fourier coefficients of $b_d$ are then obtained as $\hat{b}_d[k] = \frac{1}{N_d} \exp\big(-i\frac{\pi}{N_d}k\big)\hat{b}\big(\frac{k}{N_d}\big)$.[2]

### 3.5 Generalization to Higher Dimensions

The proposed formulation can be extended to domains of arbitrary number of dimensions. For our tracking applications we specifically consider the two-dimensional case, but higher-dimensional spaces can be treated similarly. For images, we use the space $L^2(T_1, T_2)$ of square-integrable periodic functions of two variables $g(t_1, t_2)$. The complex exponentials are then given by $e_{k_1,k_2}(t_1, t_2) = e^{i\frac{2\pi}{T_1}k_1 t_1} e^{i\frac{2\pi}{T_2}k_2 t_2}$. For the desired convolution output $y_j$, we employ a 2-dimensional Gaussian function. Further, the interpolation functions are obtained as a separable combination of the cubic spline kernel, i.e. $b(t_1, t_2) = b(t_1)b(t_2)$. The derivations presented in section 3.3 also hold for the higher dimensional cases.

## 4 Our Tracking Frameworks

We apply our continuous learning formulation for two problems: visual object tracking and feature point tracking. We first present the localization procedure, which is based on maximizing the continuous confidence function. This is shared for both the object and feature point tracking frameworks.

### 4.1 Localization Step

Here, the aim is to localize the tracked target or feature point using the learned filter $f$. This is performed by first extracting a feature map $x \in \mathcal{X}$ from the region of interest in an image. The Fourier coefficients of the confidence score function $s = S_f\{x\}$ are then calculated using (5). We employ a two-step approach for maximizing the score $s(t)$ on the interval $t \in [0, T)$. To find a rough initial estimate, we first perform a *grid search*, where the score function is evaluated at the discrete locations $s\big(\frac{Tn}{2K+1}\big)$ for $n = 0, \ldots, 2K$. This is efficiently implemented as a scaled inverse DFT of the non-zero Fourier coefficients $\hat{s}[k], k = -K, \ldots, K$. The maximizer obtained in the grid search is then used as the initialization for an iterative optimization of the Fourier series expansion $s(t) = \sum_{-K}^{K} \hat{s}[k] e_k(t)$. We employ the standard Newton's method for this purpose. The gradient and Hessian are computed by analytic differentiation of $s(t)$.

[2] Further details are given in the supplementary material.

### 4.2 Object Tracking Framework

We first present the object tracking framework based on our continuous learning formulation introduced in section 3.2. We employ multi-resolution feature maps $x_j$ extracted from a pre-trained deep network.[3] Similar to DCF based trackers [11, 13, 24], we extract a single training sample $x_j$ in each frame. The sample is extracted from an image region centered at the target location and the region size is set to $5^2$ times the area of the target box. Its corresponding importance weight is set to $\alpha_j = \frac{\alpha_{j-1}}{1-\lambda}$ using a learning rate parameter $\lambda = 0.0075$. The weights are then normalized such that $\sum_j \alpha_j = 1$. We store a maximum of $m = 400$ samples by replacing the sample with the smallest weight. The Fourier coefficients $\hat{w}$ of the penalty function $w$ are computed as described in [11]. To detect the target, we perform a multi-scale search strategy [11, 31] with 5 scales and a relative scale factor 1.02. The extracted confidences are maximized using the grid search followed by five Newton iterations, as described in section 4.1.

The training of our continuous convolution filter $f$ is performed by iteratively solving the normal equations (8). The work of [11] employed the Gauss-Seidel method for this purpose. However, this approach suffers from a quadratic complexity $\mathcal{O}(D^2)$ in the number of feature channels $D$. Instead, we employ the Conjugate Gradient (CG) [37] method due to its computational efficiency. Our numerical optimization scales linearly $\mathcal{O}(D)$ and is therefore especially suitable for high-dimensional deep features. In the first frame, we use 100 iterations to find an initial estimate of the filter coefficients $\hat{\mathbf{f}}$. Subsequently, 5 iterations per frame are sufficient by initializing CG with the current filter.[2]

### 4.3 Feature Point Tracking Framework

Here, we describe the feature point tracking framework based on our learning formulation. For computational efficiency, we assume a single-channel feature map ($D = 1$), e.g. a grayscale image, and a constant penalty function $w(t) = \beta$. Under these assumptions, the normal equations (8) form a diagonal system of equations. The filter coefficients are directly obtained as,

$$\hat{f}[k] = \frac{\sum_{j=1}^{M} \alpha_j \overline{X_j[k]\hat{b}[k]}\hat{y}_j[k]}{\sum_{j=1}^{M} \alpha_j \left|X_j[k]\hat{b}[k]\right|^2 + \beta^2} \,, \quad k = -K, \dots, K \,. \tag{10}$$

Here, we have dropped the feature dimension index for the sake of clarity. In this case (single feature channel and constant penalty function), the training equation (10) resembles the original MOSSE filter [5]. However, our continuous formulation has several advantages compared to the original MOSSE. Firstly, our formulation employs an implicit interpolation model, given by $\hat{b}$. Secondly, each sample is labeled by a continuous-domain confidence $y_j$, that enables sub-pixel information to be incorporated in the learning. Thirdly, our convolution operator outputs continuous confidence functions, allowing accurate sub-pixel localization of the feature point. In our experiments, we show that the advantages of our continuous formulation are crucial for accurate feature point tracking.

[3] We use imagenet-vgg-m-2048, available at: `http://www.vlfeat.org/matconvnet/`.

**Table 1.** A baseline comparison when using different combinations of convolutional layers in our object tracking framework. We report the mean OP (%) and AUC (%) on the OTB-2015 dataset. The best results are obtained when combining all three layers in our framework. The results clearly show the importance of multi-resolution deep feature maps for improved object tracking performance.

| | Layer 0 | Layer 1 | Layer 5 | Layers 0, 1 | Layers 0, 5 | Layers 1, 5 | Layers 0, 1, 5 |
|---|---|---|---|---|---|---|---|
| Mean OP | 58.8 | 78.0 | 60.0 | 77.8 | 70.7 | *81.8* | **82.4** |
| AUC | 49.9 | 65.8 | 51.1 | 65.7 | 59.0 | *67.8* | **68.2** |

## 5 Experiments

We validate our learning framework for two applications: tracking of objects and feature points. For object tracking, we perform comprehensive experiments on three datasets: OTB-2015 [46], Temple-Color [32], and VOT2015 [29]. For feature point tracking, we perform extensive experiments on the MPI Sintel dataset [7].

### 5.1 Baseline Comparison

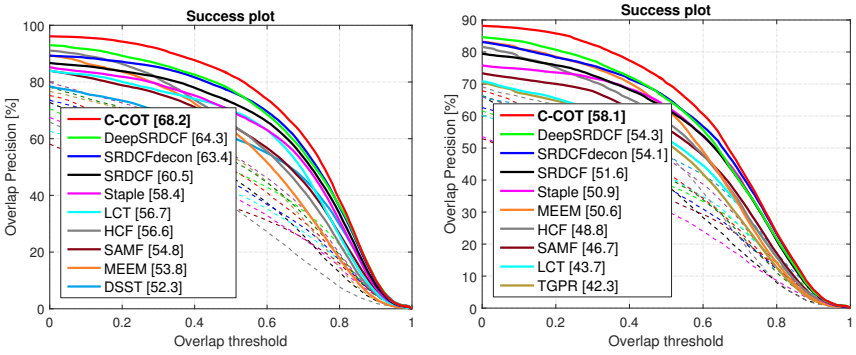
We first evaluate the impact of fusing multiple convolutional layers from the deep network in our object tracking framework. Table 1 shows the tracking results, in mean overlap precision (OP) and area-under-the-curve (AUC), on the OTB-2015 dataset. OP is defined as the percentage of frames in a video where the intersection-over-union overlap exceeds a threshold of 0.5. AUC is computed from the success plot, where the mean OP over all videos is plotted over the range of thresholds $[0, 1]$. For details about the OTB protocol, we refer to [45].

In our experiments, we investigate the impact of the input RGB image layer (layer 0), the first convolutional layer (layer 1) and the last convolutional layer (layer 5). No significant gain in performance was observed when adding intermediate layers. The shallow layer (layer 1) alone provides superior performance compared to using only the deep convolutional layer (layer 5). Fusing the shallow and deep layers provides a large improvement. The best results are obtained when combining all three convolutional layers in our learning framework. We employ this three-layer combination for all further object tracking experiments.

We also compare our continuous formulation with the discrete DCF formulation by performing explicit resampling of the feature layers to a common resolution. For a fair comparison, all shared parameters are left unchanged. The layers (0, 1 and 5) are resampled with bicubic interpolation such that the data size of the training samples is preserved. On OTB-2015, the discrete DCF with resampling obtains an AUC score of 47.7%, compared to 68.2% for our continuous formulation. This dramatic reduction in performance is largely attributed to the reduced resolution in layer 1. To mitigate this effect, we also compare with only resampling layers 0 and 5 to the resolution of layer 1. This improves the result of the discrete DCF to 60.8% in AUC, but at the cost of a 5-fold increase in data size. Our continuous formulation still outperforms the discrete DCF as it avoids artifacts introduced by explicit resampling.

**Table 2.** A Comparison with state-of-the-art methods on the OTB-2015 and Temple-Color datasets. We report the mean OP (%) for the top 10 methods on each dataset. Our approach outperforms DeepSRDCF by 5.1% and 5.0% respectively.

| | DSST | SAMF | TGPR | MEEM | LCT | HCF | Staple | SRDCF | SRDCFdecon | DeepSRDCF | **C-COT** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| OTB-2015 | 60.6 | 64.7 | 54.0 | 63.4 | 70.1 | 65.5 | 69.9 | 72.9 | 76.7 | *77.3* | **82.4** |
| Temple-Color | 47.5 | 56.1 | 51.6 | 62.2 | 52.8 | 58.2 | 63.0 | 62.2 | *65.8* | 65.4 | **70.4** |




(a) OTB-2015 (b) Temple-Color

**Fig. 2.** Success plots showing a comparison with state-of-the-art on the OTB-2015 (a) and Temple-Color (b) datasets. Only the top 10 trackers are shown for clarity. Our approach improves the state-of-the-art by a significant margin on both these datasets.

### 5.2 OTB-2015 Dataset

We validate our Continuous Convolution Operator Tracker (C-COT) in a comprehensive comparison with 20 state-of-the-art methods: ASLA [25], TLD [26], Struck [21], LSHT [22], EDFT [14], DFT [41], CFLB [18], ACT [13], TGPR [19], KCF [24], DSST [9], SAMF [31], MEEM [47], DAT [40], LCT [36], HCF [35], Staple [3] and SRDCF [11]. We also compare with SRDCFdecon, which integrates the adaptive decontamination of the training set [12] in SRDCF, and DeepSRDCF [10] employing activations from the first convolutional layer.

**State-of-the-art Comparison:** Table 2 (first row) shows a comparison with state-of-the-art methods on the OTB-2015 dataset.[4] The results are reported as mean OP over all the 100 videos. The HCF tracker, based on hierarchical convolutional features, obtains a mean OP of 65.5%. The DeepSRDCF employs the first convolutional layer, similar to our baseline "Layer 1" in table 1, and obtains a mean OP of 77.3%. Our approach achieves the best results with a mean OP of 82.4%, significantly outperforming DeepSRDCF by 5.1%.

Figure 2a shows the success plot on the OTB-2015 dataset. We report the AUC score for each tracker in the legend. The DCF-based trackers HCF and Staple obtain AUC scores of 56.6% and 58.4% respectively. Among the compared methods, the SRDCF and its variants SRDCFdecon and DeepSRDCF

[4] Detailed results are provided in the supplementary material.

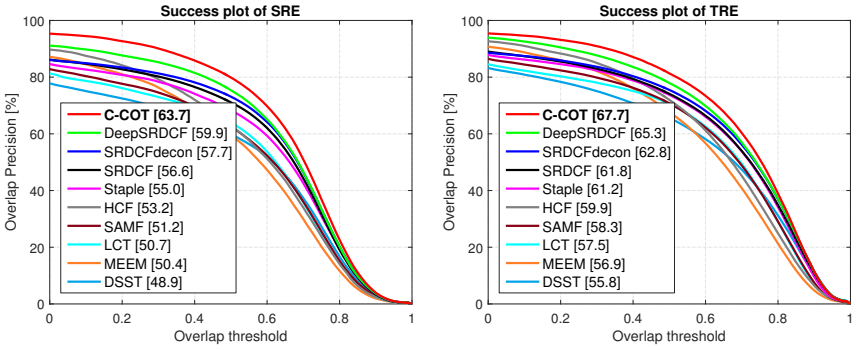



**Fig. 3.** An evaluation of the spatial (*left*) and temporal (*right*) robustness to initializations on the OTB-2015 dataset. We compare the top 10 trackers. Our approach demonstrates superior robustness compared to state-of-the-art methods.

provide the best results, all obtaining AUC scores above 60%. Overall, our tracker achieves the best results, outperforming the second best method by 3.9%.

**Robustness to Initialization:** We evaluate the robustness to initializations using the protocol provided by [46]. Each tracker is evaluated using two different initialization strategies: spatial robustness (SRE) and temporal robustness (TRE). The SRE criteria initializes the tracker with perturbed boxes, while the TRE criteria starts the tracker at 20 frames. Figure 3 provides the SRE and TRE success plots. Our approach obtains consistent improvements in both cases.

### 5.3 Temple-Color Dataset

Here, we evaluate our approach on the Temple-Color dataset [32] containing 128 videos. The second row of table 2 shows a comparison with state-of-the-art methods. The DeepSRDCF tracker provides a mean OP score of 65.4%. MEEM and SRDCFdecon obtain mean OP scores of 62.2% and 65.8% respectively. Different from these methods, our C-COT does not explicitly manage the training set to counter occlusions and drift. Our approach still improves the start-of-the-art by a significant margin, achieving a mean OP score of 70.4%. A further gain in performance is expected by incorporating the unified learning framework [12] to handle corrupted training samples. In the success plot in Figure 2b, our method obtains an absolute gain of 3.8% in AUC compared to the previous best method.

### 5.4 VOT2015 Dataset

The VOT2015 dataset [29] consists of 60 challenging videos compiled from a set of more than 300 videos. Here, the performance is measured both in terms of accuracy (overlap with the ground-truth) and robustness (failure rate). In VOT2015, a tracker is restarted in the case of a failure. We refer to [29] for details. Table 3 shows the comparison of our approach with the top 10 participants in the

**Table 3.** Comparison with state-of-the-art methods on the VOT2015 dataset. The results are presented in terms of robustness and accuracy. Our approach provides improved robustness with a significant reduction in failure rate.

| | S3Tracker | RAJSSC | Struck | NSAMF | SC-EBT | sPST | LDP | SRDCF | EBT | DeepSRDCF | **C-COT** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Robustness | 1.77 | 1.63 | 1.26 | 1.29 | 1.86 | 1.48 | 1.84 | 1.24 | 1.02 | 1.05 | 0.82 |
| Accuracy | 0.52 | 0.57 | 0.47 | 0.53 | 0.55 | 0.55 | 0.51 | 0.56 | 0.47 | 0.56 | 0.54 |

challenge according to the VOT2016 rules [28]. Among the compared methods, RAJSSC achieves favorable results in terms of accuracy, at the cost of a higher failure rate. EBT achieves the best robustness among the compared methods. Our approach improves the robustness with a 20% reduction in failure rate, without any significant degradation in accuracy.

### 5.5 Feature Point Tracking

We validate our approach for robust and accurate feature point tracking. Here, the task is to track distinctive local image regions. We perform experiments on the MPI Sintel dataset [7], based on the 3D-animated movie "Sintel". The dataset consists of 23 sequences, featuring naturalistic and dynamic scenes with realistic lighting and camera motion blur. The ground-truth dense optical flow and occlusion maps are available for each frame. Evaluation is performed by selecting approximately 2000 feature points in the first frame of each sequence. We use the Good Features to Track (GFTT) [42] feature selector, but discard points at motion boundaries due to their ambiguous motion. The ground-truth tracks are then generated by integrating flow vectors over the sequence. The flow vectors are obtained by a bilinear interpolation of the dense ground-truth flow. We terminate the ground-truth tracks using the provided occlusion maps.

We compare our approach to MOSSE [5] and KLT [34, 44]. The OpenCV implementation of KLT, used in our experiments, employs a pyramidal search [6] to accommodate for large translations. For a fair comparison, we adopt a similar pyramid approach for our method and MOSSE, by learning an independent filter for each pyramid level. Further, we use the window size of $31 \times 31$ pixels and 3 pyramid levels for all methods. For both our method and MOSSE we use a learning rate of $\lambda = 0.1$ and set the regularization parameter to $\beta = 10^{-4}$. For the KLT we use the default settings in OpenCV. Unlike ours and the MOSSE tracker, the KLT tracks feature points frame-to-frame without memorizing earlier appearances. In addition to our standard tracker, we also evaluate a frame-to-frame version (Ours-FF) of our method by setting the learning rate to $\lambda = 1$.

For quantitative comparisons, we use the endpoint error (EPE), defined as the Euclidian distance between the tracked point and its corresponding ground-truth location. Tracked points with an EPE smaller than 3 pixels are regarded as inliers. Figure 4 (left) shows the distribution of EPE computed over all sequences and tracked points. We also report the average inlier EPE for each method in the legend. Our approach achieves superior accuracy, with an inlier error of 0.449

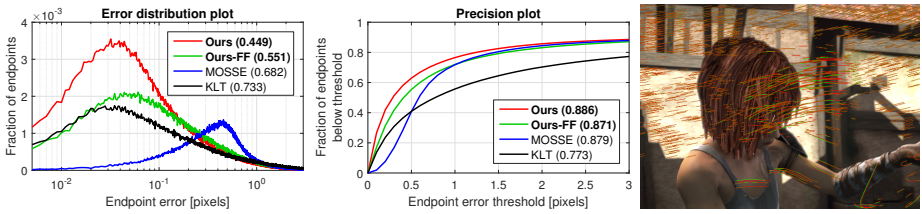



**Fig. 4.** Feature point tracking results on the MPI Sintel dataset. We report the endpoint error (EPE) distribution (*left*) and precision plot (*center*) over all sequences and points. In the legends, we display the average inlier EPE and the inlier ratio for the error distribution and precision plot respectively. Our approach provides consistent improvements, both in terms of accuracy and robustness, compared to existing methods. The example frame (*right*) from the Sintel dataset visualizes inlier trajectories obtained by our approach (red) along with the ground-truth (green).

pixels. We also provide the precision plot (Figure 4, center), where the fraction of points with an EPE smaller than a threshold is plotted. The legend shows the inlier ratio for each method. Our tracker achieves superior robustness in comparison to the KLT, with an inlier ratio of 0.886. Compared to MOSSE, our method obtains significantly improved precision at sub-pixel thresholds ($< 1$ pixel). This clearly demonstrates that our continuous formulation enables accurate sub-pixel feature point tracking, while being robust. Unlike the frame-to-frame KLT, our method provides a principled procedure for updating the tracking model, while memorizing old samples. The experiments show that already our frame-to-frame variant (Ours-FF) provides a spectacular improvement compared to the KLT. Hence, our gained performance is due to both the model update *and* the proposed continuous formulation. On a desktop machine, our Matlab code achieves real-time tracking of 300 points at a single scale, utilizing only a single CPU.

## 6 Conclusions

We propose a generic framework for learning discriminative convolution operators in the continuous spatial domain. We validate our framework for two problems: object tracking and feature point tracking. Our formulation enables the integration of multi-resolution feature maps. In addition, our approach is capable of accurate sub-pixel localization. Experiments on three object tracking benchmarks demonstrate that our approach achieves superior performance compared to the state-of-the-art. Further, our method obtains substantially improved accuracy and robustness for real-time feature point tracking.

Note that, in this work, we do not use any video data to learn an application specific deep feature representation. This is expected to further improve the performance of our object tracking framework. Another research direction is to incorporate motion-based deep features into our framework, similar to [20].

**Acknowledgments**: This work has been supported by SSF (CUAS), VR (EMC$^2$), CENTAURO, the Wallenberg Autonomous Systems Program, NSC and Nvidia.

## References


1. Badino, H., Yamamoto, A., Kanade, T.: Visual odometry by multi-frame feature integration. In: ICCV Workshop (2013)
2. Baker, S., Matthews, I.A.: Lucas-kanade 20 years on: A unifying framework. IJCV 56(3), 221–255 (2004)
3. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.S.: Staple: Complementary learners for real-time tracking. In: CVPR (2016)
4. Boddeti, V.N., Kanade, T., Kumar, B.V.K.V.: Correlation filters for object alignment. In: CVPR (2013)
5. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: CVPR (2010)
6. Bouguet, J.Y.: Pyramidal implementation of the lucas kanade feature tracker. Tech. rep., Microprocessor Research Labs, Intel Corporation (2000)
7. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: ECCV (2012)
8. Cimpoi, M., Maji, S., Vedaldi, A.: Deep filter banks for texture recognition and segmentation. In: CVPR (2015)
9. Danelljan, M., Häger, G., Shahbaz Khan, F., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: BMVC (2014)
10. Danelljan, M., Häger, G., Shahbaz Khan, F., Felsberg, M.: Convolutional features for correlation filter based visual tracking. In: ICCV Workshop (2015)
11. Danelljan, M., Häger, G., Shahbaz Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: ICCV (2015)
12. Danelljan, M., Häger, G., Shahbaz Khan, F., Felsberg, M.: Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In: CVPR (2016)
13. Danelljan, M., Shahbaz Khan, F., Felsberg, M., van de Weijer, J.: Adaptive color attributes for real-time visual tracking. In: CVPR (2014)
14. Felsberg, M.: Enhanced distribution field tracking using channel representations. In: ICCV Workshop (2013)
15. Fernandez, J.A., Boddeti, V.N., Rodriguez, A., Kumar, B.V.K.V.: Zero-aliasing correlation filters for object recognition. TPAMI 37(8), 1702–1715 (2015)
16. Fusiello, A., Trucco, E., Tommasini, T., Roberto, V.: Improving feature tracking with robust statistics. Pattern Anal. Appl. 2(4), 312–320 (1999)
17. Galoogahi, H.K., Sim, T., Lucey, S.: Multi-channel correlation filters. In: ICCV (2013)
18. Galoogahi, H.K., Sim, T., Lucey, S.: Correlation filters with limited boundaries. In: CVPR (2015)
19. Gao, J., Ling, H., Hu, W., Xing, J.: Transfer learning based visual tracking with gaussian process regression. In: ECCV (2014)
20. Gladh, S., Danelljan, M., Shahbaz Khan, F., Felsberg, M.: Deep motion features for visual tracking. In: ICPR (2016)
21. Hare, S., Saffari, A., Torr, P.: Struck: Structured output tracking with kernels. In: ICCV (2011)
22. He, S., Yang, Q., Lau, R., Wang, J., Yang, M.H.: Visual tracking via locality sensitive histograms. In: CVPR (2013)
23. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with kernels. In: ECCV (2012)

24. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. TPAMI 37(3), 583–596 (2015)
25. Jia, X., Lu, H., Yang, M.H.: Visual tracking via adaptive structural local sparse appearance model. In: CVPR (2012)
26. Kalal, Z., Matas, J., Mikolajczyk, K.: P-n learning: Bootstrapping binary classifiers by structural constraints. In: CVPR (2010)
27. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: ISMAR (2007)
28. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Čehovin, L., Vojír, T., Häger, G., Lukežič, A., Fernández, G.: The visual object tracking vot2016 challenge results. In: ECCV workshop (2016)
29. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Čehovin, L., Fernández, G., Vojír, T., Nebehay, G., Pflugfelder, R., Häger, G.: The visual object tracking vot2015 challenge results. In: ICCV workshop (2015)
30. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., et al.: The visual object tracking VOT 2014 challenge results. In: ECCV Workshop (2014)
31. Li, Y., Zhu, J.: A scale adaptive kernel correlation filter tracker with feature integration. In: ECCV Workshop (2014)
32. Liang, P., Blasch, E., Ling, H.: Encoding color information for visual tracking: Algorithms and benchmark. TIP 24(12), 5630–5644 (2015)
33. Liu, L., Shen, C., van den Hengel, A.: The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In: CVPR (2015)
34. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: IJCAI (1981)
35. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: ICCV (2015)
36. Ma, C., Yang, X., Zhang, C., Yang, M.H.: Long-term correlation tracking. In: CVPR (2015)
37. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, 2nd edn. (2006)
38. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: CVPR (2014)
39. Ovren, H., Forssén, P.: Gyroscope-based video stabilisation with auto-calibration. In: ICRA (2015)
40. Possegger, H., Mauthner, T., Bischof, H.: In defense of color-based model-free tracking. In: CVPR (2015)
41. Sevilla-Lara, L., Learned-Miller, E.G.: Distribution fields for tracking. In: CVPR (2012)
42. Shi, J., Tomasi, C.: Good features to track. In: CVPR (1994)
43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
44. Tomasi, C., Kanade, T.: Detection and Tracking of Point Features. Tech. rep. (1991)
45. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: CVPR (2013)
46. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. TPAMI 37(9), 1834–1848 (2015)
47. Zhang, J., Ma, S., Sclaroff, S.: MEEM: robust tracking via multiple experts using entropy minimization. In: ECCV (2014)
48. Zografos, V., Lenz, R., Ringaby, E., Felsberg, M., Nordberg, K.: Fast segmentation of sparse 3d point trajectories using group theoretical invariants. In: ACCV (2014)

# Deep Motion Features for Visual Tracking

Susanna Gladh, Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg
Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, Sweden

***Abstract*—Robust visual tracking is a challenging computer vision problem, with many real-world applications. Most existing state-of-the-art approaches employ hand-crafted appearance features, such as HOG or Color Names. Recently, deep RGB features extracted from convolutional neural networks have been successfully applied for tracking. Despite their success, these features only capture appearance information. On the other hand, motion cues provide discriminative and complementary information that can improve tracking performance. Contrary to visual tracking, deep motion features have been successfully applied for action recognition and video classification tasks. Typically, the motion features are learned by training a CNN on optical flow images extracted from large amounts of labeled videos.**

**This paper presents an investigation of the impact of deep motion features in a tracking-by-detection framework. We further show that hand-crafted, deep RGB, and deep motion features contain complementary information. To the best of our knowledge, we are the first to propose fusing appearance information with deep motion features for visual tracking. Comprehensive experiments clearly suggest that our fusion approach with deep motion features outperforms standard methods relying on appearance information alone.**

## I. Introduction

Generic visual tracking is the problem of estimating the trajectory of a target in a sequence of images. It is challenging since only the initial position of the target is known. The various applications of generic tracking range from surveillance to robotics. Most state-of-the-art approaches follow the tracking-by-detection paradigm, where a classifier or regressor is discriminatively trained to differentiate the target from the background. Recently, Discriminative Correlation Filter (DCF) based methods [1], [2], [3], [4], [5] have shown excellent performance for visual tracking. These approaches efficiently train a correlation filter to estimate the classification confidences of the target. This is performed by considering all cyclic shifts of the training samples and exploiting the fast Fourier transform (FFT) at the training and detection steps. In this work, we base our approach on the DCF framework.

DCF based trackers typically employ hand-crafted appearance features, such as HOG [2], [4], Color Names [6], or combinations of these features [7]. Recently, deep Convolutional Neural Networks (CNNs) has been successfully applied in DCF based trackers [5], [8]. A CNN applies a sequence of convolution, normalization, and pooling operations on the input RGB patch. The parameters of the network are trained using large amounts of labeled images, such as the ImageNet dataset. Deep convolutional features from pre-trained networks have been shown to be generic [9], and therefore also applicable for visual tracking.

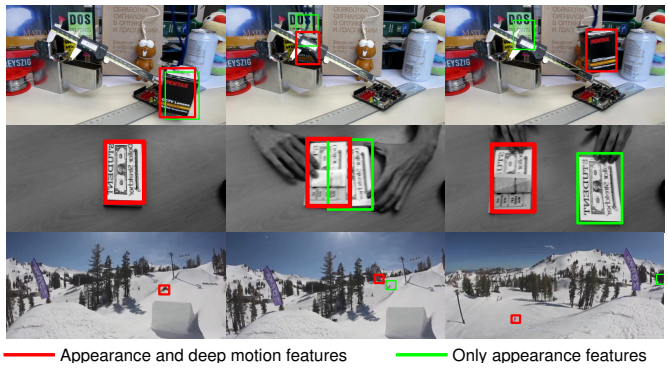



Fig. 1. A comparison of using combined appearance information (hand-crafted HOG and deep RGB features) (in green) and our fusion of appearance and deep motion features (in red). Tracking results are shown for three example sequences: *Box*, *Coupon* and *Skiing*. Our fusion approach (red), using deep motion features, achieves superior results in these scenarios where appearance alone is insufficient.

Besides deep RGB features, deep motion features have been successfully employed for action recognition [10], [11]. These motion features are constructed by learning a CNN that takes optical flow images as input. The network is trained using flow data extracted from large amounts of labeled videos, such as the UCF101 dataset [12]. Unlike deep RGB networks, these deep flow networks capture high-level information about the motion in the scene. To the best of our knowledge, deep motion features are yet to be investigated for the problem of visual tracking.

Tracking methods solely based on appearance information struggle in scenarios with, for example, out-of-plane rotations (figure 1 first row), background distractors with similar appearance (figure 1 second row), and distant or small objects (figure 1 third row). In these cases, high-level motion cues provide rich complementary information that can disambiguate the target. While appearance features only encode static information from a single frame, deep motion features integrate information from the consecutive *pair* of frames used for estimating the optical flow. Motion features can therefore capture the dynamic nature of a scene that is complementary to appearance features. This motivates us to investigate the fusion of standard appearance features with deep motion features for visual tracking.

**Contributions:** In this paper, we investigate the impact of deep motion features for visual tracking. We use a deep optical flow network that was pre-trained for action recognition. Our approach does not require any additional labeled data for training the network. We investigate fusing hand-crafted and deep appearance features with deep motion features in a state-

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 19, 2016.

of-the-art DCF-based tracking-by-detection framework [3]. To show the impact of motion features, we further evaluate the fusion of different feature combinations.

Extensive experiments are performed on the OTB-2015 [13], Temple-Color [14], and VOT-2015 [15] datasets. On OTB-2015, our fusion of appearance and deep motion features significantly improves the baseline method, employing only appearance features, by $3.4\%$ in mean overlap precision. Our fusion approach is further shown to advance the state-of-the-art performance with an absolute gain of $6.8\%$ and $5.8\%$ in mean overlap precision on OTB-2015 and Temple-Color respectively. Figure 1 shows a comparison of our fusion with deep motion features with the baseline method (employing only appearance features).

## II. Related Work

Discriminative tracking methods [3], [5], [16], [17] typically train a classifier or regressor for the task of differentiating the target from the background. These methods are often also termed tracking-by-detection approaches since they apply a discriminatively trained detector. This detector is trained online by extracting and labeling samples from the video frames. The training samples are often represented by e.g. raw image patches [1], [18], image histograms and Haar features [16], color [6], [17], or shape features [2], [4].

Among tracking-by-detection approaches, the Discriminative Correlation Filter (DCF) based trackers have recently shown excellent performance on standard tracking benchmarks [15], [19]. The key for their success is the ability to efficiently utilize limited data by including all shifts of local training samples in the learning. DCF-based methods train a least-squares regressor for predicting the target confidence scores by utilizing the properties of circular correlation and the fast Fourier transform (FFT). The MOSSE tracker [1] first considered training a single-channel correlation filter based on grayscale image samples of target and background appearance. A remarkable improvement is achieved by extending the MOSSE filter to multi-channel features. This can be performed by either optimizing the exact filter for offline learning applications [20] or using approximative update schemes for online tracking [2], [4], [6].

Despite their success, the DCF approaches are affected by the periodic assumption of the training samples, leading to negative boundary effects and a restricted training and search region size. This problem has recently been addressed in [3] by adding a spatial regularization term. While the original SRDCF employs HOG features, the DeepSRDCF [8] investigates the use of convolutional features from a deep RGB network in the SRDCF tracker. In this work, we also base our tracking framework on the SRDCF.

Other than the deep RGB appearance features, recent works [11], [21], [22] have investigated the use of deep motion features for action recognition. Generally, optical flow is computed for each consecutive pair of frames. The resulting optical flow is aggregated in the x-, y- direction and the flow magnitude to construct a three channel image. A CNN is then trained using these flow patches as input. Simonyan and Zisserman [11] propose a two-stream ConvNet architecture to integrate spatial and temporal networks. The network is trained on multi-frame dense optical flow and multi-task learning is employed to increase the amount of training samples. Gkioxari and Malik [21] propose to use deep static and kinematic cues for action localization in videos. The work of [22] propose to combine pose-normalized deep appearance and motion features for action recognition. Unlike action recognition, existing tracking methods [5], [8] are limited to using only appearance based deep RGB features. In this work, we propose to combine appearance cue with deep motion information for visual tracking.

## III. Baseline Tracker

As a baseline tracker, we employ the SRDCF [3] framework, which has recently been successfully used for integrating single-layer deep features [8]. The standard DCF trackers exploit the periodic assumption of the local feature map to perform efficient training and detection using the FFT. However, this introduces unwanted boundary effects and restricts the size of the image region used for training the model and searching for the target. In the SRDCF, these shortcomings are addressed by introducing a spatial regularization term in the learning formulation. This enables training to be performed on larger image regions, leading to a more discriminative model.

In the SRDCF framework, a convolution filter is discriminatively learned based on training samples $\{(x_k, y_k)\}_{k=1}^{t}$. Here, $x_k$ is a $d$-dimensional feature map with a spatial size $M \times N$. We denote feature channel $l$ of $x_k$ by $x_k^l$. Typically, $x_k$ is extracted from an image region containing both the target and large amounts of background information. The label $y_k$ consists of the desired $M \times N$ confidence score function at the spatial region corresponding to the sample $x_k$. That is, $y_k(m, n) \in \mathbb{R}$ is the desired classification confidence at location $(m, n)$ in the feature map $x_k$. We use a Gaussian function centered at the target location in $x_k$ to determine the desired scores $y_k$. In the SRDCF formulation, the aim is to train a multi-channel convolution filter $f$ consisting of one $M \times N$ filter $f^l$ per feature dimension $l$. The target confidence scores for an $M \times N$ feature map $x$ are computed as $S_f(x) = \sum_{l=1}^{d} x^l * f^l$. Here, $*$ denotes circular convolution.

To learn the filter $f$, the SRDCF formulation minimizes the squared error between the confidence scores $S_f(x_k)$ and the corresponding desired scores $y_k$,

$$\varepsilon(f) = \sum_{k=1}^{t} \alpha_k \left\| S_f(x_k) - y_k \right\|^2 + \sum_{l=1}^{d} \left\| w \cdot f^l \right\|^2. \qquad (1)$$

The weights $\alpha_k$ determine the impact of each training sample and $\cdot$ denotes point-wise multiplication. The SRDCF employs a spatial regularization term determined by the penalty weight function $w$. The filter is trained by minimizing the least squares loss (1) in the Fourier domain using iterative sparse solvers. We refer to [3] for more details.

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 19, 2016.

## IV. Visual features

The visual feature representation is a core component of a tracking framework. In this work, we investigate the use of a combination of hand-crafted features, deep appearance features and deep motion features for tracking.

### A. Hand-crafted features

Hand-crafted features are typically used to capture low-level information, such as shape, color or texture. The Histograms of Oriented Gradients (HOG) is popularly employed for both visual tracking [2], [3], [4] and object detection [23]. It mainly captures shape information by calculating histograms of gradient directions in a spatial grid of cells. The histogram for each cell is then normalized with respect to neighboring cells to add invariance.

Other than shape features, various color-based feature representations have been commonly employed for tracking. For instance, the use of simple color transformations [17], [24] or color histograms. Recently, the Color Names (CN) descriptor have been popularly employed for tracking due to its discriminative power and compactness [6]. The CN descriptor applies a pre-learned mapping from RGB to the probabilities of 11 linguistic color names.

### B. Deep features

Features extracted by a trained Convolutional Neural Network (CNN) are known as deep features. The CNN consists of a number of computational layers that perform convolution, local normalization, and pooling operations on the input image patch. The final layers are usually fully connected (FC) and include an output classification layer. CNNs are typically trained in a supervised manner on large datasets of labeled images, such as ImageNet.

Feature representations learned by CNNs trained for a particular vision problem (e.g. image classification) have been shown to be generic and can be applied for a variety other vision tasks. For this purpose, most works apply the activations from the FC layer [25], [26]. Recently, activations form the convolutional layers have shown improved results for image classification [27], [28]. The deep convolutional features are discriminative and posses high-level visual information, while preserving spatial structure. Convolutional activations at a specific layer form a multi-channel feature map, which can be directly integrated into the SRDCF framework. Shallow-layer activations encode low-level features at a high spatial resolution, while deep layers contain high-level information at a coarse resolution.

*1) Deep RGB Features:* For the RGB images we use the imagenet-vgg-verydeep-16 network [25], with the MatConvNet library [29]. This network contains 13 convolutional layers. We investigate using both a shallow and a deep convolutional layer. For the shallow RGB layer, we use the activations from the fourth convolutional layer, after the Rectified Linear Unit (ReLU) operation. It consists of 128 feature channels and has a spatial stride of 2 pixels compared to the input image patch. For the deep layer of the RGB network, we use activations at the last convolutional layer, again after the ReLU-operation. This layer consists of a 512-dimensional feature map with a spatial stride of 16 pixels. Figure 2 shows example activations from the shallow layer (first row) and deep layer (second row) of the RGB network.

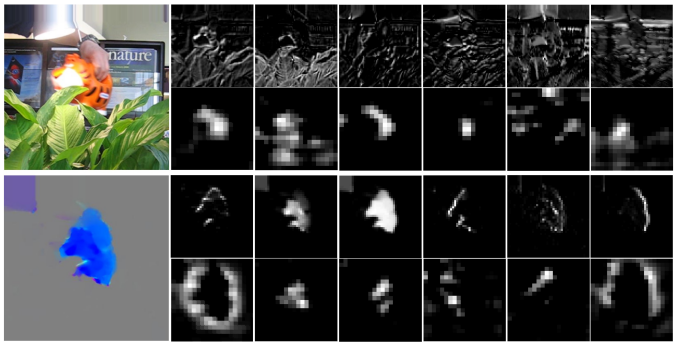


Fig. 2. Visualization of the features with highest energy from a shallow and deep convolutional layer in the appearance (top row) and motion network (bottom row). Appearance features are extracted from the raw RGB image (top left) from Tiger2, and motion features from the corresponding optical flow image (bottom left). In both cases, we show shallow and deep activations in the corresponding first and second sub-rows respectively.

*2) Deep Motion Features:* The motion features are extracted using the approach described by [22]. We start by calculating the optical flow for each frame, together with the previous frame, according to [30]. The motion in the x- and y-directions forms a 3-channel image together with the flow magnitude. The values are adjusted to the interval $[0, 255]$. For our experiments, we use the pre-trained optical flow network provided by [10]. It is pre-trained on the UCF101 dataset [12] for action recognition and contains five convolutional layers. For the motion network, we only use the activations from the deepest convolutional layer. Similar to the RGB network, we extract the activations after the ReLU-operation. The resulting feature map consists of 384 channels at a spatial stride of 16 pixel compared to the input. An example optical flow image is displayed in figure 2, along with corresponding shallow (third row) and deep (fourth row) activations from the motion network.

## V. Our Tracking Framework

Here, we describe our tracking framework where we investigate the fusion of hand-crafted and deep appearance features with deep motion features. Our framework is based on learning an independent SRDCF model for each feature map. That is, we learn one filter $f_j$ for each feature type $j$. In a frame $k$, we extract new training samples $x_{j,k}$ for each feature type $j$ from the same image region centered at the estimated target location. We use a quadratic training region with an area equal to $5^2$ times the area of the target box. For example, in our final tracking approach we combine three different feature maps: HOG $x_{1,k}$, the deep RGB layer $x_{2,k}$ and the deep motion layer $x_{3,k}$ (see section VI-A). The fused feature maps have different dimensionalities $d_j$ and spatial resolutions, leading to a different spatial sample size $M_j \times N_j$ for each feature $j$. The label function $y_{j,k}$ for feature $j$ is set to an $M_j \times N_j$

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 19, 2016.

TABLE I
COMPARISON OF DIFFERENT COMBINATIONS OF HAND-CRAFTED (HOG), DEEP RGB AND DEEP MOTION FEATURES ON THE OTB-2015 DATASET. RESULTS ARE REPORTED IN TERMS OF MEAN OVERLAP PRECISION (OP) AND AREA-UNDER-THE-CURVE (AUC) IN PERCENT. THE TWO BEST RESULTS ARE DISPLAYED IN RED AND BLUE FONT RESPECTIVELY. SHALLOW AND DEEP LAYERS OF THE RGB NETWORK ARE DENOTED RGB(S) AND RGB(D). FOR EACH COMBINATION OF APPEARANCE FEATURES, WE ALSO REPORT THE RESULTS OBTAINED WHEN INCLUDING DEEP MOTION FEATURES. THE FUSION WITH DEEP MOTION FEATURES SIGNIFICANTLY IMPROVES THE PERFORMANCE FOR ALL COMBINATIONS.

| | | HOG | RGB(s) | RGB(d) | RGB(s+d) | HOG+RGB(s) | HOG+RGB(d) | HOG+RGB(s+d) |
|---|---|---|---|---|---|---|---|---|
| **Mean OP (%)** | Without deep motion features | 74.5 | 74.1 | 56.3 | 78 | 74.9 | 80.7 | 79.1 |
| | With deep motion features | 81.3 | 81 | 58.9 | 81.1 | 79.5 | **84.1** | *82.2* |
| **Mean AUC (%)** | Without deep motion features | 61.1 | 62.8 | 48.5 | 65.1 | 62.6 | 65.2 | 65.3 |
| | With deep motion features | 65.7 | 66.4 | 49.7 | *66.7* | 65.6 | **67.4** | 66.4 |

sampled Gaussian function with its maximum centered at the estimated target location.

To train the filters, we minimize the SRDCF objective (1) for each feature type $j$ independently. This is performed similarly to [3] by first transforming (1) to the Fourier domain using Parseval's formula and then applying an iterative solver. We also use exponentially decreasing sample weights $\alpha_k$ [1], [2], [3] with a learning rate of $0.01$ and construct the penalty function $w$ as in [3].

To detect the target in a new frame, we first extract feature maps $z_j$ centered at the estimated target location in the previous frame. This is performed using the same procedure as for training samples. The learned filters $f_j$ from the previous frame can then be applied to each feature map $z_j$ individually. However, the target confidence scores $S_{f_j}(z_j)$ is of size $M_j \times N_j$ and therefore have a different spatial resolution for each feature type $j$. To fuse the confidence scores obtained from each filter $f_j$, we first interpolate the scores from each filter to a pixel-dense grid. We then fuse the scores by computing the average confidence value at each pixel location. For efficiency, we use the Fourier interpolation method employing complex exponential basis functions. Since the filters are optimized in the Fourier domain, we directly have the DFT coefficients $\hat{f}_j$ of each filter. Using the DFT convolution property, the DFT coefficients of the confidences are obtained as $\widehat{S_{f_j}(z_j)} = \sum_{l=1}^{d_j} \hat{z}_j^l \cdot \hat{f}_j^l$.

The Fourier interpolation is implemented by first zero-padding the DFT coefficients to the desired resolution and then performing inverse DFT. Formally, we define the padding operator $\mathcal{P}_{R\times S}$ that pads the DFT to the size $R \times S$ by adding zeros at the high frequencies. We denote the inverse DFT operator by $\mathscr{F}^{-1}$ and let $J$ denote the number of feature maps to be fused. The fused confidence scores $s$ are computed as,

$$s = \frac{1}{J} \sum_{j=1}^{J} \mathscr{F}^{-1} \left\{ \mathcal{P}_{R\times S} \left\{ \widehat{S_{f_j}(z_j)} \right\} \right\} \qquad (2)$$

We obtain pixel-dense confidence scores (2) of the target by setting $R \times S$ to be the size (in pixels) of the image region used for extracting the feature maps $x_{j,k}$. The new target location is then estimated by maximizing the scores $s(m, n)$ over the pixel locations $(m, n)$. To also estimate the target size, we apply the filters at five scales with a relative scale factor of $1.02$, similar to [3], [7].

## VI. EXPERIMENTS

We validate our tracking framework by performing comprehensive experiments on three challenging benchmark datasets: OTB-2015 [13] with 100 videos, Temple-Color [14] with 128 videos, and VOT2015 [15] with 60 videos.

### A. Baseline Comparison

We investigate the impact of deep motion features by evaluating different combinations of appearance and motion representations on the OTB-2015 dataset. Table I shows a comparison of different feature combinations using mean overlap precision (OP) and area-under-the-curve (AUC). OP is defined as the percentage of frames in a video where the intersection-over-union overlap exceeds a certain threshold. In the tables, we report the OP at the threshold of $0.5$, which corresponds to the PASCAL criterion. The AUC score is computed from the success plot, where OP is plotted over the range of thresholds $[0, 1]$. We refer to [13] for further details about the evaluation metrics.

The results show that using only HOG gives a mean OP score of $74.5\%$. Interestingly, adding a deep RGB feature layer (RGB(d)) improves the result by $6.2\%$, while adding a deep motion feature layer provides an improvement of $6.8\%$. The best result are obtained by combining all these three cues: HOG, RGB(d) and deep motion. This combination achieves an absolute gain of $9.6\%$ in mean OP over using only HOG and obtains the best AUC score of $67.4\%$. For the state-of-the-art comparisons described in section VI-B and VI-C, we employ this feature combination in our approach.

Another interesting comparison is that the result of using HOG or a shallow RGB layer (RGB(s)) alone both provide approximately the same mean OP and AUC score as their combination. This indicates that HOG and RGB(s) do not provide significant complementary information. On the other hand, adding deep motion features to either of these representations significantly improves the results. From our results, it is apparent that adding deep motion features consistently increases the tracking performance. Lastly, our results clearly show that deep appearance and motion features are complementary and that the best results are obtained when fusing these two cues.

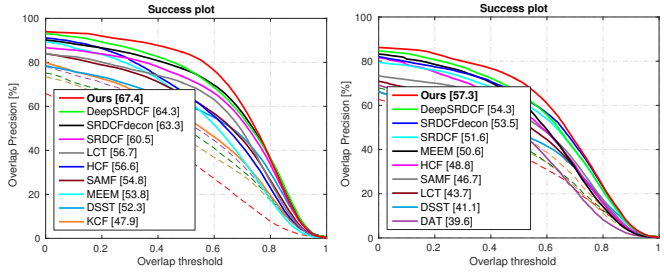
### B. OTB-2015 Dataset

We validate our approach by performing a comprehensive comparison with 12 state-of-the-art trackers: Struck [16],

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 19, 2016.

TABLE II
COMPARISON WITH STATE-OF-THE-ART TRACKERS USING MEAN OP (%) ON THE OTB-2015 AND TEMPLE-COLOR DATASETS. THE TWO BEST RESULTS ARE SHOW IN RED AND BLUE FONT RESPECTIVELY. OUR APPROACH SIGNIFICANTLY IMPROVES THE STATE-OF-THE-ART DEEPSRDCF TRACKER BY 6.8% AND 5.8% ON OTB-2015 AND TEMPLE-COLOR DATASETS RESPECTIVELY.

| | Struck | CFLB | ACT | KCF | DSST | SAMF | DAT | MEEM | LCT | HCF | SRDCF | SRDCFdecon | DeepSRDCF | **Ours** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OTB-2015 | 52.9 | 44.9 | 49.6 | 54.9 | 60.6 | 64.7 | 36.4 | 63.4 | 70.1 | 65.5 | 72.9 | 76.5 | *77.3* | **84.1** |
| Temple-Color | 40.9 | 37.8 | 42.1 | 46.5 | 47.5 | 56.1 | 48.2 | 62.2 | 52.8 | 58.2 | 62.2 | 65 | *65.4* | **71.2** |



(a) OTB-2015 (b) Temple-Color

Fig. 3. Success plots showing a comparison of our approach with state-of-the-art methods on the OTB-2015 (a) and Temple-Color (b) datasets. For clarity, only the top 10 trackers are shown. Our proposed method provides significant improvements on both these datasets.

CFLB [21], LCT [31], ACT [6], KCF [4], DSST [2], SAMF [7], DAT [32], MEEM [17], HCF [5], SRDCF [3] and SRDCFdecon [33]. We also compare with the DeepSRDCF [8], that employs the shallow layer of a deep RGB network in the SRDCF.

Table II (first row) presents a state-of-the-art comparison, in mean OP, on the OTB-2015 dataset. The HCF tracker employing an ensemble of deep RGB based appearance features obtains a mean OP score of 65.5%. The SRDCF tracker using hand-crafted appearance features achieves a mean OP score of 72.9%. The DeepSRDCF employs appearance based RGB features and obtains a mean OP score of 77.3%. Our approach that combines hand-crafted and deep appearance based features with deep motion features achieves state-of-the-art results on this datset with a mean OP of 84.1%. Figure 3a presents the success plot for top-10 trackers on the OTB-2015 dataset. The area-under-the-curve (AUC) for each method is shown in the legend. The SRDCF obtains an AUC score of 60.5%. Among the existing methods, the DeepSRDCF achieves an AUC score of 64.3%. Our approach significantly outperforms the DeepSRDCF tracker by obtaining an AUC score of 67.4%.

*1) Attribute-based Comparison:* We perform an attribute-based analysis on the OTB-2015 dataset. Each video in the dataset is annotated by 11 different attributes: illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane and out-of-plane rotation, out-of-view, background clutter and low resolution. Figure 4 shows success plots for 4 attributes. Our approach provides consistent improvements on all 11 attributes. A significant improvement is achieved in these scenarios: deformation (+6.4%), out of view (+6.1%), and out-of-plane rotation (+4.7%), compared to the best existing tracker. Figure 5 shows a qualitative comparison with three state-of-the-art trackers.

### C. Temple-Color Dataset

Next, we validate our proposed tracker on the challenging Temple-Color dataset [14]. The dataset consists of 128 videos. Table II (second row) presents a state-of-the-art comparison in mean OP. The HCF tracker obtains a mean OP score of 58.2%. The SRDCF tracker with hand-crafted appearance features provides a mean OP score of 62.2%. The DeepSRDCF further improves the results and obtains a mean OP score of 65.4%. Our approach obtains state-of-the-art results on this datset with a mean OP of 71.2%. A significant gain of 5.8% in mean OP is obtained over the DeepSRDCF tracker. Figure 3b presents the success plot for top-10 trackers on the Temple-Color dataset. The area-under-the-curve (AUC) for each tracker is shown in the legend of the plot. The SRDCF obtains an AUC score of 51.6%. Among the existing methods, the DeepSRDCF provides the best results and achieves an AUC score of 54.3%. Our approach obtains state-of-the-art results by significantly outperforming the DeepSRDCF tracker with a gain of 3%.

### D. VOT2015 Dataset

Table III presents a state-of-the-art comparison on the VOT2015 dataset [15] in comparison to the top 10 participants in the challenge according to the VOT2016 rules (see http://votchallenge.net). The dataset consists of 60 challenging videos compiled from a set of more than 300 videos. Here, the performance is measured in terms of accuracy (overlap with the ground-truth) and robustness (failure rate). The proposed method yields superior accuracy compared to the previously most accurate method (RAJSSC) and superior robustness compared to the previously most robust method (EBT). Compared to previous SRDCF-based methods, both accuracy and robustness are significantly improved.

## VII. CONCLUSIONS

We have investigated the impact of deep motion features in a DCF-based tracking framework. Existing approaches are limited to using either hand-crafted or deep appearance based features. We show that deep motion features provide complementary information to appearance cue and their combination leads to significantly improved tracking performance. Experiments are performed on three challenging benchmark tracking datasets: OTB-2015 with 100 videos, Temple-Color with 128 videos, and VOT2015 with 60 videos. Our results clearly demonstrate that fusion of hand-crafted appearance, deep appearance and deep motion features leads to state-of-the-art performance on both datasets.

Preprint submitted to 23rd International Conference on Pattern Recognition.
Received April 19, 2016.

TABLE III
COMPARISON WITH STATE-OF-THE-ART METHODS ON THE VOT2015 DATASET. THE PROPOSED METHOD PROVIDES STATE-OF-THE-ART ACCURACY AND ROBUSTNESS.

| | S3Tracker | RAJSSC | Struck | NSAMF | SC-EBT | sPST | LDP | SRDCF | EBT | DeepSRDCF | **Ours** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Robustness | 1.77 | 1.63 | 1.26 | 1.29 | 1.86 | 1.48 | 1.84 | 1.24 | *1.02* | 1.05 | **0.92** |
| Accuracy | 0.52 | *0.57* | 0.47 | 0.53 | 0.55 | 0.55 | 0.51 | 0.56 | 0.47 | 0.56 | **0.58** |

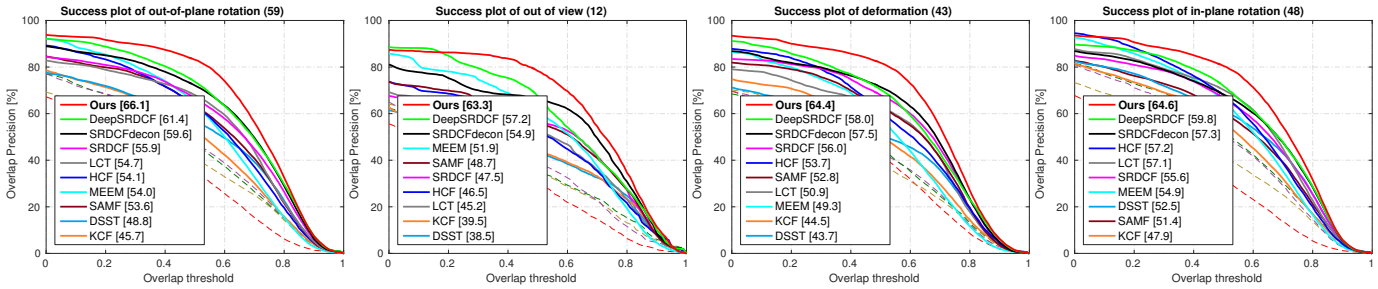



Fig. 4. Attribute-based analysis of our approach on the OTB-2015 dataset. Success plots are shown for 4 attributes. For clarity, we show the top 10 trackers in each plot. The title of each plot indicates the number of videos labeled with the respective attribute. Our approach provides consistent improvements compared to state-of-the-art methods on all 11 attributes.

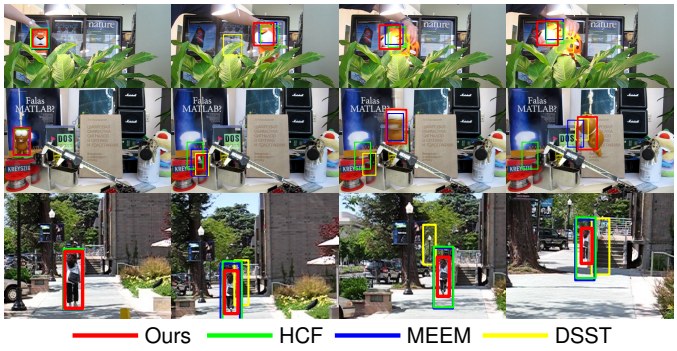



Fig. 5. Frame-by-frame comparison with state-of-the-art trackers on *Tiger2*, *Lemming*, *Freeman* and *Human7*. The deep motion features, employed in our tracker, add complementary information when appearance features are less distinctive, leading to favorable performance in scenarios with e.g. occlusions, out-of-plane rotations and blur.

## REFERENCES


[1] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *CVPR*, 2010.

[2] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *BMVC*, 2014.

[3] ——, "Learning spatially regularized correlation filters for visual tracking," in *ICCV*, 2015.

[4] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *PAMI*, 2015.

[5] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *ICCV*, 2015.

[6] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. van de Weijer, "Adaptive color attributes for real-time visual tracking," in *CVPR*, 2014.

[7] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *ECCV Workshop*, 2014.

[8] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *ICCV Workshop*, 2015.

[9] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *CVPR Workshop*, 2014.

[10] G. Gkioxari and J. Malik, "Finding action tubes," in *CVPR*, 2015.

[11] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014.

[12] K. Soomro, A. Roshan Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," in *CRCV-TR-12-01*, 2012.

[13] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *PAMI*, 2015.

[14] P. Liang, E. Blasch, and H. Ling, "Object tracking benchmark," *TIP*, 2015.

[15] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernández, T. Vojír, G. Nebehay, R. Pflugfelder, and G. Hger, "The visual object tracking vot2015 challenge results," in *ICCV workshop*, 2015.

[16] S. Hare, A. Saffari, and P. Torr, "Struck: Structured output tracking with kernels," in *ICCV*, 2011.

[17] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: robust tracking via multiple experts using entropy minimization," in *ECCV*, 2014.

[18] J. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *ECCV*, 2012.

[19] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *CVPR*, 2013.

[20] H. K. Galoogahi, T. Sim, and S. Lucey, "Multi-channel correlation filters," in *ICCV*, 2013.

[21] ——, "Correlation filters with limited boundaries," in *CVPR*, 2015.

[22] G. Chéron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *ICCV*, 2015.

[23] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[24] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," in *CVPR*, 2012.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[26] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *CVPR*, 2014.

[27] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *CVPR*, 2015.

[28] L. Liu, C. Shen, and A. van den Hengel, "The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification," in *CVPR*, 2015.

[29] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," *CoRR*, vol. abs/1412.4564, 2014.

[30] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *ECCV*, 2004.

[31] C. Ma, X. Yang, C. Zhang, and M. Yang, "Long-term correlation tracking," in *CVPR*, 2015.

[32] H. Possegger, T. Mauthner, and H. Bischof, "In defense of color-based model-free tracking," in *CVPR*, 2015.

[33] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, "Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking," in *CVPR*, 2016.

# Efficient Multi-Frequency Phase Unwrapping using Kernel Density Estimation

Felix Järemo Lawin, Per-Erik Forssén, and Hannes Ovrén

Computer Vision Laboratory, Linköping University, Sweden
{felix.jaremo-lawin, per-erik.forssen, hannes.ovren}@liu.se

**Abstract.** In this paper we introduce an efficient method to unwrap multi-frequency phase estimates for time-of-flight ranging. The algorithm generates multiple depth hypotheses and uses a spatial kernel density estimate (KDE) to rank them. The confidence produced by the KDE is also an effective means to detect outliers. We also introduce a new closed-form expression for phase noise prediction, that better fits real data. The method is applied to depth decoding for the Kinect v2 sensor, and compared to the *Microsoft Kinect SDK* and to the open source driver *libfreenect2*. The intended Kinect v2 use case is scenes with less than 8m range, and for such cases we observe consistent improvements, while maintaining real-time performance. When extending the depth range to the maximal value of 18.75m, we get about 52% more valid measurements than *libfreenect2*. The effect is that the sensor can now be used in large depth scenes, where it was previously not a good choice.

**Keywords:** Time-of-flight, Kinect v2, kernel-density-estimation

## 1 Introduction

Multi-frequency time-of-flight is a way to accurately estimate distance, that was originally invented for Doppler RADAR [1]. More recently it has also found an application in RGB-D sensors[1] that use time-of-flight ranging, such as the *Microsoft Kinect v2* [2].

Depth from time-of-flight requires very accurate time-of-arrival estimation. Amplitude modulation improves accuracy, by measuring phase shifts between the received and emitted signals, instead of time-of-arrival. However, a disadvantage with amplitude modulation is that it introduces a periodic depth ambiguity. By using multiple modulation frequencies in parallel, the ambiguity can be resolved in most cases, and the useful range can thus be extended.

We introduce an efficient method to unwrap multi-frequency phase estimates for time-of-flight ranging. The algorithm uses *kernel density estimation* (KDE) in a spatial neighbourhood to rank different depth hypotheses. The KDE also doubles as a confidence measure which can be used to detect and suppress bad pixels. We apply our method to depth decoding for the Kinect v2 sensor. For

[1] RGB-D sensors output both colour (RGB) and depth (D) images.

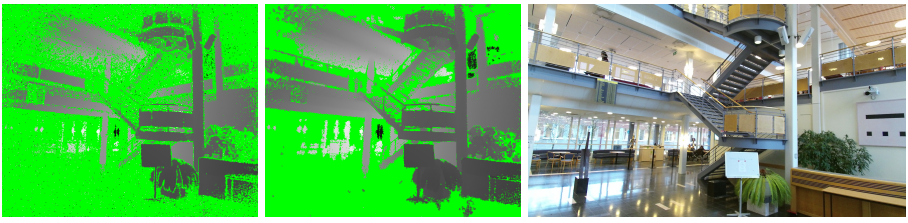


**Fig. 1.** Single frame output on scene with greater than 18.75m depth range. Left: *libfreenect2*, Center: proposed method. Right: corresponding RGB image. Pixels suppressed by outlier rejection are shown in green. The proposed method has more valid depth points than *libfreenect2* resulting in a denser and more well defined depth scene. While the suppressed areas are clean from outliers for the proposed method, the *libfreenect2* image is covered in salt and pepper noise.

large depth scenes we see a significant increase in coverage of about 52% more valid pixels compared to *libfreenect2*. See figure 1 for a qualitative comparison. For 3D modelling with Kinect fusion [3], this results in fewer outlier points and more complete scene details. While the method is designed with the Kinect v2 in mind, it is also applicable to multi-frequency ranging techniques in general.

### 1.1 Related Work

The classic solution to the multi-frequency phase unwrapping problem, is to use the Chinese reminder theorem (CRT). This method is fast, but implicitly assumes noise free data, and in [1] it is demonstrated that by instead generating multiple unwrappings for each frequency, and then performing clustering along the range axis, better robustness to noise is achieved. However, due to its simplicity, CRT is still advocated, e.g. in [4, 5], and is also used in the Kinect v2 drivers.

Simultaneous unwrapping of multiple phases with different frequencies is a problem that also occurs in fringe pattern projection techniques [6, 5]. The algorithms are not fully equivalent though, as the phase is estimated by different means, and the relationship between phase and depth is different.

Another way to unwrap the time-of-flight phase shift is to use surface reflectivity constraints. As the amplitude associated with each phase measurement is a function of object distance and surface reflectivity, a popular approach in the literature is to assume locally constant reflectivity. Under this assumption, the depth can be unwrapped using e.g. a Markov Random Field (MRF) formulation with a data term and a reflectivity smoothness term. In [7], many different such unwrapping methods are discussed. A recent extension of this is [8], where distance, surface albedo and also the local surface normal are used to predict the reflectance.

The multi-frequency and reflectivity approaches are combined in [9] where a MRF with both reflectivity, and dual frequency data terms are used.

Detection of *multipath interference* (i.e. measurement problems due to light reflected from several different world locations reaching the same pixel) is studied in [10]. If four or more frequencies are used, pixels with multipath effects can be detected and suppressed. Recently in [11], a multipath detection algorithm based on blind source separation was applied to the Kinect v2. This required the firmware of the Kinect v2 to be modified to emit and receive at 5 frequencies instead of the default 3. As firmware modification currently requires reverse engineering of the transmission protocol we have not pursued this line of research.

In [12] a simulator for ToF measurements is developed and used to evaulate performance of a MRF that does simultaneous unwrapping and denoising using a wavelet basis. The performance on real data is however not shown.

Noise on the phase measurements is analyzed in [13] and it is suggested that the variance of phase is predicted by sensor variance divided by the phase amplitude squared. In this paper we derive a new model for phase noise that fits better with real data and utilize it as a measure of confidence for the measurements. In [12] a Gaussian mixture model for sensor noise is also derived, but its efficacy is never validated on real sensor data.

### 1.2 Structure

The paper is organized as follows: In section 2 we describe how multi-frequency time-of-flight measurements are used to sense depth. In section 3 we describe how we extend this by generating multiple hypotheses and selecting one based on kernel density estimation. We give additional implementation details and compare our method to other approaches in section 4. The paper concludes with a discussion and outlook in section 5.

## 2 Depth Decoding

In time-of-flight sensors, an amplitude modulated light signal is emitted to be reflected on objects in the environment. The reflected signal is then captured in the pixel array of the sensor, where it is correlated with the reference signal driving the light emitter. On the Kinect v2 this is achieved on the camera chip by using quantum efficiency modulation and integration[14, 15] resulting in a voltage value $v_k$. In the general case $N$ different reference signals are used, each phase shifted $\frac{2\pi}{N}$ radians from the others [13]. Often $N = 4$ is used [12], but in the Kinect v2 we have $N = 3$. The voltage values are used to calculate the phase shift between the emitted and the received signals using the complex phase

$$\mathbf{z} = \frac{2}{N} \sum_{k=0}^{N-1} v_k e^{-\mathbf{i}(p_o + 2\pi k/N)}\,, \tag{1}$$

where $p_o$ is a common phase offset. This expression is derived using least squares [13], and the actual phase shift and its corresponding amplitude are obtained as

$$\phi = \arg \mathbf{z} \quad \text{and} \quad a = |\mathbf{z}|\,. \tag{2}$$

The amplitude is proportional to the reflected signal strength, and increases when the voltage values make consistent contributions to $\mathbf{z}$. It is thus useful as a measure of confidence in the decoded phase.

From the phase shift $\phi$ in (2) the time-of-flight distance can be calculated as

$$d = \frac{c\phi}{4\pi f_m}\,, \tag{3}$$

where $c$ is the speed of light, and $f_m$ is the used modulation frequency (see e.g. [2]). This relationship holds both in multi-frequency RADAR [1] and RGB-D time-of-flight. In fringe projection profilometry [6], phase and amplitude values are also obtained for each frequency of the fringe pattern, resulting in a very similar problem. However, the relationship between phase and depth is different in this case.

The phase shift obtained from (2) is the true phase shift $\tilde{\phi}$ modulo $2\pi$. Thus $\phi$ is ambiguous in an environment where $d$ can be larger than $c/(2f_m)$. Finding the correct period, i.e. $n$ in the expression

$$\tilde{\phi} = \phi_{\text{wrapped}} + 2\pi n\,, \quad n \in \mathbb{N}\,, \tag{4}$$

is called *phase unwrapping*. To reduce measurement noise, and increase the range in which $\phi$ is unambiguous, one can combine the phase measurements from multiple modulated signals with different frequencies.

Figure 2 shows the phase to distance relation for the three amplitude-modulated signals, with frequencies 16, 80 and 120 MHz, which is the setup used in the Kinect v2 [2]. For each of the three frequencies, three phase shifts are used to calculate a phase according to (2), and thus a total of nine measurements are used in each depth calculation. In the figure, we see that if the phase shifts are combined, a common wrap-around occurs at 18.75 meters. This is thus the maximum range in which the Kinect v2 can operate without depth ambiguity.

As a final step, the phase shifts from the different modulation frequencies are combined using a phase unwrapping procedure and a weighted average.

It is of critical importance that the phase is correctly unwrapped, as choosing the wrong period will result in large depth errors. This is the topic of the following sub-sections.

### 2.1 Phase unwrapping

Consider phase measurements of $M$ amplitude modulated signals with different modulation frequencies. From (3) we get the following relations:

$$d = \frac{c\,(\phi_0 + 2\pi n_0)}{4\pi f_0} = \frac{c\,(\phi_1 + 2\pi n_1)}{4\pi f_1} = \cdots = \frac{c\,(\phi_{M-1} + 2\pi n_{M-1})}{4\pi f_{M-1}} \iff \tag{5}$$

$$\frac{k_0}{2\pi}\phi_0 + k_0 n_0 = \frac{k_1}{2\pi}\phi_1 + k_1 n_1 = \ldots = \frac{k_{M-1}}{2\pi}\phi_{M-1} + k_{M-1} n_{M-1} \tag{6}$$

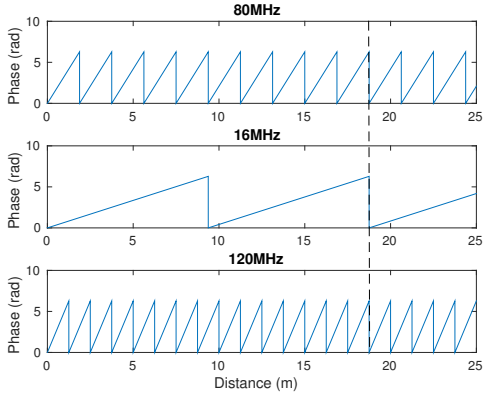



**Fig. 2.** Wrapped phases for Kinect v2, in the range 0 to 25 meters. Top to bottom: $\phi_0$, $\phi_1$, $\phi_2$. The dashed line at 18.75 meters indicates the common wrap-around point for all three phases. Just before this line we have $n_0 = 9$, $n_1 = 1$, and $n_2 = 14$.

where $\{k_m\}_{m=0}^{M-1}$ are the least common multiples for $\{f_m\}_{m=0}^{M-1}$ divided by the respective frequency and $\{n_m\}_{m=0}^{M-1}$ are the set of sought unwrapping coefficients. Now (6) can be simplified to a set of constraints on pairs of unwrapping coefficients $(n_i, n_j)$:

$$k_i n_i - k_j n_j = \frac{k_j}{2\pi}\phi_j - \frac{k_i}{2\pi}\phi_i, \forall i, j \in [0, M-1] \text{ and } i > j. \tag{7}$$

In total there are $M(M-1)/2$ such equations. As the system is redundant, the correct unwrapping cannot be obtained by e.g. Gaussian elimination and in practice the equations are unlikely to hold due to measurement noise. The constraints can however be used to define a likelihood for a specific unwrapping.

### 2.2 CRT based unwrapping

The ambiguity of the phase measurements can be resolved by applying a variant of the Chinese reminder theorem (CRT) [4, 5] to one equation at a time in (7):

$$n_i = k_i \cdot \text{round}\left(\frac{k_j\phi_j - k_i\phi_i}{k_i 2\pi}\right) \tag{8}$$

$$\tilde{\phi}_i = \phi_i + 2\pi n_i \tag{9}$$

In the case of more than two frequencies the unwrapped phase $\tilde{\phi}_i$ could be used in (8) for the next equation in (7) to unwrap the next phase. This is suggested and described in [5] and is also used in *libfreenect2*. In the end when all equations have been used, the full unambiguous range of the combined phase

measurements has been unwrapped. The CRT method is fast but sensitive to noise as it unwraps each of the phase measurements in sequence. The consequence of this is that an error made early on will be propagated.

### 2.3 Phase fusion

The unwrapped phase measurements are combined by using a weighted average:

$$t^* = \sum_{m=0}^{M-1} \frac{k_m \tilde{\phi}_m}{(k_m \sigma_{\phi_m})^2} \Big/ \left( \sum_{m=0}^{M-1} \frac{1}{(k_m \sigma_{\phi_m})^2} \right) , \tag{10}$$

where $\sigma_{\phi_m}$ is the standard deviation of the noise in $\phi_m$. The *pseudo distance* estimate $t^*$ is later converted to a depth (i.e. distance in the forward direction), using the intrinsic camera parameters.

## 3 Kernel Density based Unwrapping

In this paper, we propose a new method for multi-frequency phase unwrapping. The method considers several fused pseudo distances $t^*$ (see (10)) for each pixel location $\mathbf{x}$, and select the one with the highest kernel density value [16]. Each such hypothesis $t^i(\mathbf{x})$ is a function of the unwrapping coefficients $\mathbf{n} = (n_0, \ldots, n_{M-1})$. The kernel density for a particular hypothesis $t^i(\mathbf{x})$ is a weighted sum of all considered hypotheses in the spatial neighbourhood:

$$p(t^i(\mathbf{x})) = \frac{\sum_{j \in \mathcal{I}, k \in \mathcal{N}(\mathbf{x})} w_{jk} K(t^i(\mathbf{x}) - t^j(\mathbf{x}_k))}{\sum_{j \in \mathcal{I}, k \in \mathcal{N}(\mathbf{x})} w_{jk}} . \tag{11}$$

Here $K(\cdot)$ is the kernel, and $w_{jk}$ is a sample weight. The sets of samples to consider are defined by the hypothesis indices $\mathcal{I}$ (e.g. $\mathcal{I} = \{1, 2\}$ if we have two hypotheses in each pixel), and by the set of all spatial neighbours $\mathcal{N}(\mathbf{x}) = \{k : \|\mathbf{x}_k - \mathbf{x}\|_1 < r\}$ where $r$ is a square truncation radius. The hypothesis weight $w_{ik}$ is defined as

$$w_{ik} = g(\mathbf{x} - \mathbf{x}_k, \sigma) p(t^i(\mathbf{x}_k) | \mathbf{n}_i(\mathbf{x}_k)) p(t^i(\mathbf{x}_k) | \mathbf{a}_i(\mathbf{x}_k)) . \tag{12}$$

The three factors in $w_{ik}$ are:

- the *spatial weight* $g(\mathbf{x} - \mathbf{x}_k, \sigma)$, which is a Gaussian that downweights neighbours far from the considered pixel location $\mathbf{x}$.
- the *unwrapping likelihood* $p(t^i(\mathbf{x}) | \mathbf{n}_i(\mathbf{x}))$, that depends on the consistency of the pseudo-distance estimate (10) given the unwrapping vector $\mathbf{n} = (n_0, \ldots, n_{M-1})$.
- the *phase likelihood* $p(t^i(\mathbf{x}) | \mathbf{a}_i(\mathbf{x}))$, where $\mathbf{a}_i = (a_0, \ldots, a_{M-1})$, are the amplitudes from (2). It defines the accuracy of the phase before unwrapping.

The kernel in (11) is defined as:

$$K(x) = e^{-x^2/2h^2}\,, \tag{13}$$

where $h$ is the kernel scale.

In the following sub-sections we will describe the three weight terms in more detail. For simplicity of notation, we will drop the pixel coordinate argument $\mathbf{x}$, and e.g. write $p(t^*)$ instead of $p(t^*(\mathbf{x}))$.

### 3.1 Unwrapping likelihood

Due to measurement noise, the constraints in (7) are never perfectly satisfied. We thus subtract the left-hand side from the right-hand side of these equations to form residuals $\epsilon_k$, one for each of the $M(M-1)/2$ constraints. These are then used to define a cost for a given unwrapping vector $\mathbf{n} = (n_0, \ldots, n_{M-1})$:

$$J(\mathbf{n}) = \sum_{k=1}^{M(M-1)/2} \epsilon_k^2/\sigma_{\epsilon_k}^2\,. \tag{14}$$

This cost function corresponds to the following *unwrapping likelihood*:

$$p(t^*|\mathbf{n}) \propto e^{-J(\mathbf{n})/(2s_1^2)}\,, \tag{15}$$

where $t^*$ is the fusion of the three unwrapped pseudo-distances, see (10), and $s_1$ is a scaling factor to be determined. For normally distributed residuals, and the Kinect v2 case of $M = 3$, the constraints in (7) imply:

$$\sigma_{\epsilon_1}^2 = \left(\frac{k_1\sigma_{\phi_1}}{2\pi}\right)^2 + \left(\frac{k_0\sigma_{\phi_0}}{2\pi}\right)^2 \tag{16}$$

$$\sigma_{\epsilon_2}^2 = \left(\frac{k_2\sigma_{\phi_2}}{2\pi}\right)^2 + \left(\frac{k_0\sigma_{\phi_0}}{2\pi}\right)^2 \tag{17}$$

$$\sigma_{\epsilon_3}^2 = \left(\frac{k_2\sigma_{\phi_2}}{2\pi}\right)^2 + \left(\frac{k_1\sigma_{\phi_1}}{2\pi}\right)^2\,. \tag{18}$$

This gives us the weights in (14). The values of $\sigma_{\phi_m}$ could be predicted from the phase amplitude $a_m$ (more on this later), but they tend to deviate around a fixed ratio, and we have observed better robustness of (15) if the ratio is always fixed. We assume that the phase variances is equal for all modulation frequencies. This assumption gives us their relative magnitudes, but not their absolute values, which motivates the introduction of the parameter $s_1$ in (15).

### 3.2 Multiple hypotheses

In contrast to the CRT approach to unwrapping, see section 2.2, we will consider all meaningful unwrapping vectors $\mathbf{n} = (n_0, \ldots, n_{M-1})$ within the unambiguous

range. A particular depth value corresponds to a unique unwrapping vector, but with the introduction of noise, neigbouring unwrappings need to be considered at wrap around points. For example, looking at the Kinect v2 case shown in figure 2, if $n_0 = n_1 = 0$, $n_2$ should either be 0 or 1. In total 30 different hypotheses for $(n_0, n_1, n_2)$ are constructed in this way. These can then be ranked by (15).

Compared with the CRT approach, that only considers one hypothesis, the above approach is more expensive. On the other hand, the true maximum of (15) is guaranteed to be checked.

In the low noise case, we can expect the hypothesis with the largest likelihood according to (15) to be the correct one. This is however not necessarily the case in general. Therefore a subset $\mathcal{I}$ of hypotheses with high likelihoods are saved for further consideration, by evaluating the full kernel density (11).

### 3.3 Phase likelihood

The amplitude, $a$, produced by (2) can be used to accurately propagate a noise estimate on the voltage values to noise in the phase estimate. In [13] this relationship is analysed and an expression is derived that can only be computed numerically. For practical use, [13] instead propose $\sigma_\phi^2 = 0.5(\sigma_v/a)^2$ as approximate propagation formula (for $N = 4$). For constant but unknown noise variance on the voltage values $\sigma_v^2$, the phase noise can be predicted from the amplitude, as:

$$\sigma_\phi = \gamma/a\,, \tag{19}$$

where $\gamma$ is a parameter to be determined. While propagation of noise from voltage values to the complex phase vector $\mathbf{z}$ is linear, the final phase extraction is not, and we will now derive a more accurate approximation using sigma-point propagation [17]. Geometrically, phase extraction from the phase vector (2) is a projection onto a circle, and thus the noise propagation is also a projection of the noise distribution $p(\mathbf{z})$ onto the circle, see figure 4 (a). $p(\mathbf{z})$ is centered around the true amplitude $a$, and sigma-point candidates are located on a circle with radius $\sigma_z$. By finding the points where the circle tangents pass through the origin, we get an accurate projection of the noise distribution.

The points of tangency can be found using the pole-polar relationship [18]. For points $(x, y)$ and $(x, -y)$ we get the expressions:

$$x = (a^2 - \sigma_z^2)/a \quad \text{and} \quad y = \frac{\sigma_z}{a}\sqrt{a^2 - \sigma_z^2}\,. \tag{20}$$

From these expressions, the phase noise can be predicted as:

$$\hat{\sigma}_\phi = \tan^{-1}(y/x) = \tan^{-1}(\sqrt{1/((a/\sigma_z)^2 - 1)})\,, \tag{21}$$

where $\sigma_z$ is a model parameter to be determined. Values of $a < \sigma_z$ invalidate the geometric model in figure 4 (a), and for these we use (19) with $\gamma = \sigma_z\pi/2$.

In *libfreenect2*, a bilateral filter is applied to the $\mathbf{z}$ vectors. The noise attenuation this results in is amplitude dependent, but it can be accurately modelled as a quadratic polynomial on $a$.

$$\hat{\sigma}_{\phi,\text{bilateral}} = \tan^{-1}(y/x) = \tan^{-1}(\sqrt{1/((\gamma_0 + a\gamma_1 + a^2\gamma_2)^2 - 1)})\,, \tag{22}$$

We use the predicted phase noise to define a *phase likelihood*:

$$p(t^*|\mathbf{a}) = \prod_{m=0}^{M-1} p(t^*|a_m)\,, \text{ where } p(t^*|a_m) \propto e^{-0.5\hat{\sigma}^2_{\phi m}/s_2^2}\,. \tag{23}$$

where $s_2$ is a parameter to be tuned. The phase likelihood encodes the accuracy of the phases *before* unwrapping.

### 3.4 Hypothesis selection

In each spatial position $\mathbf{x}$, we rank the considered hypotheses $t^i$, using the KDE (kernel density estimate) defined in (11). The final hypothesis selection is then made as:

$$i^* = \arg\max_{i \in \mathcal{I}} p(t^i)\,. \tag{24}$$

For the selected hypothesis, $p(t^{i^*})$ is also useful as a confidence measure that can be thresholded to suppress the output in problematic pixels. However, if the spatial support is small, e.g. $3 \times 3$, the weighted KDE occasionally encounters sample depletion problems (only very bad samples in a neighbourhood). This can be corrected by regularizing the confidence computation according to:

$$\text{conf}(t^i) = \frac{\sum_k w_k K(t^i - t^k)}{\max(p_{\min}, \sum_k w_k)} \approx p(t^i)\,, \tag{25}$$

where $p_{\min}$ is a small value, e.g. 0.5.

### 3.5 Spatial selection versus smoothing

The proposed KDE approach, see (11), selects the best phase unwrapping by considering the distribution of hypotheses in the spatial neighbourhood of a pixel. Note that the spatial neighbourhood is only used to *select* among different hypotheses. This is different from a spatial smoothing, as is commonly used in e.g. depth from disparity [19]. A connection to kernel based smoothing approaches, such as channel smoothing [20, 21] can be made by considering the limit where the number of hypotheses is the continuous set of $t$-values in the depth range of the sensor. The discrete selection in (24) will then correspond to decoding of the highest peak of the PDF, and thus to channel smoothing. In the experiments we will however use just $|\mathcal{I}| = 2$, or 3 hypotheses per pixel, which is far from this limit. After selection, the noise on each pixel is still uncorrelated from the noise of its neighbours, and each pixel can thus still be considered an independent measurement. This is beneficial when fusing data in a later step, using e.g. Kinect Fusion [3].


**Fig. 3.** Unwrapping ground truth for the three datasets. Top row: ground truth depth maps. Green pixels are suppressed, and not used in the evaluation. Bottom row: corresponding images from the RGB camera.

## 4 Experiments

We apply the method to depth decoding for the Kinect v2 sensor, and compare it to the *Microsoft Kinect SDK*[2] in the following denoted *Microsoft* and to the open source driver *libfreenect2*. A first visual result is shown in figure 1. As can be seen in the figure, the proposed method has a better coverage in the depth images than *libfreenect2*. Another clear distinction between the methods is that *libfreenect2* produces salt and pepper noise all over the image. See also [22] for more examples, and corresponding RGB frames.

### 4.1 Implementation

The algorithm was implemented by modifying the *libfreenect2*[3] code for depth calculations using *OpenCL* [23] for GPU acceleration. When running the proposed pipeline with $|\mathcal{I}| = 2$ on a *Nvidia GeForce GTX 760* GPU, the frame rate for the depth calculations is above 30 fps for spatial supports up to $17 \times 17$. For e.g. a $3 \times 3$ support our method operates at 200 fps, which is marginally slower than *libfreenect2* (which also operates in a $3 \times 3$ neighbourhood) at 245 fps. The current implementation is however designed for ease of testing, and further speed optimization is be possible.

[2] Version 2.0.1409
[3] As in *opencl_depth_packet_processor.cl* Feb 18 2016 commit: 1d06d2db04a9

### 4.2 Ground Truth for Unwrapping

We construct our own ground truth data, which is used for quantitatively evaluating the correctness of the phase unwrappings. The accuracy of the ground truth must be good enough to tell a correct unwrapping from an incorrect one. As we have 30 unwrapping candidates in an 18.75m range, the distance between the candidates is on average 60cm. To ensure that no incorrect unwrappings are accidentally counted as inliers, we require an accuracy of at least half the candidate distance, i.e. better than 30cm.

The required accuracy can easily be met using the Kinect sensor itself. By fusing many frames from the same camera pose, we can reduce the amount of unwrapping errors, and also increase the accuracy in correctly unwrapped measurements. For a given scene we place the camera at different locations corresponding to a spatial $3 \times 3$ grid. By fusing data from these poses we can detect and suppress multipath responses, which vary with camera position. Further details on the dataset generation can be found in the supplemental materials [22].

### 4.3 Datasets

We have used the procedure in section 4.2 to collect three datasets with ground truth depth, shown in figure 3. The **kitchen** dataset has a maximal depth of 6.7m, and is used to test the Kinect v2 under the intended usage with an 8m depth limit. The **lecture** dataset has a maximal depth of 14.6m and is used to evaluate methods without imposing the 8m limit. The **library** dataset is used for parameter tuning, and has a maximal range of 17.0m. For each dataset, we have additionally logged 25 raw-data frames from the central camera pose, using a data logger in Linux, and another 25 output frames using the *Microsoft SDK v2* API in Windows.

### 4.4 Comparison of noise propagation models

The tuning dataset **library** was used to estimate the standard deviations $\sigma_\phi$ of the individual phase measurements over 40 frames. The model parameters in (19), (21) and (22) were found by minimizing the residuals of the corresponding inverted expressions using non-linear least squares over all amplitude measurements $a$. The inversion of the expressions reduced bias effects due to large residuals for small amplitudes.

This procedure was performed for $\mathbf{z}$ with and without bilateral filtering (as implemented in *libfreenect2*). Figure 4 ((b) and (c)) shows the resulting predictions overlaid on the empirical distributions of the relation between the amplitude and the phase standard deviation. We see that the models proposed in (21) and (22) have a slightly better fit to the empirical distribution than [13] on raw phase measurements. However, for bilateral filtered $\mathbf{z}$, the quadratic model suggested in expression (22) has the best fit. As bilateral filtering improves the final performance this is the model used in our method.

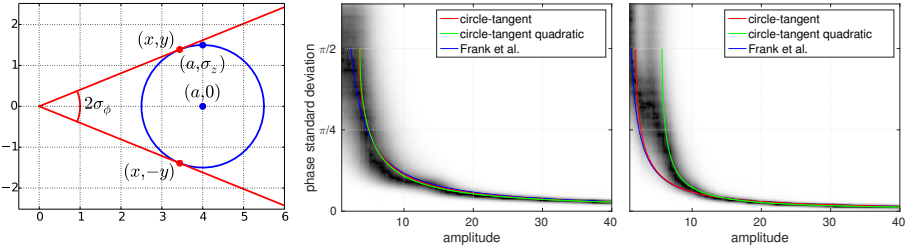



**Fig. 4.** (a): Geometrical illustration of the circle-tangents. (b): Predictions from raw phase overlaid on empirical distribution. (c): Predictions from bilateral-filtered phase.

### 4.5 Outlier Rejection

***libfreenect2***: outlier rejection is performed at several steps, each with one or several tuned thresholds:

- Pixels where any of the amplitudes is below a threshold are suppressed.
- Pixels where the pseudo-distances differ in magnitude are suppressed. The purpose of this is similar to (15), but an expression based on the cross-product of the pseudo phases with a reference relation is used.
- Pixels with a large depth, or amplitude variance in their $3 \times 3$ neighbourhood are suppressed.
- Pixels that deviate from their neighbours are suppressed.
- Pixels on edges in the voltage images are suppressed.

**Proposed method**: a single threshold is applied on the KDE-based confidence measure in (25).

### 4.6 Parameter settings

The proposed method introduces the following parameters that needs to be set:

- the scaling $s_1$ in (15)
- the scaling $s_2$ in (23).
- the kernel scale $h$ in (13).
- the spatial support $r$. (the Gaussian in (12) has a spatial support of $(2r + 1) \times (2r + 1)$ and $\sigma = r/2$.)
- the number of hypotheses $|\mathcal{I}|$.

The method is not sensitive to the selection of $s_1$, $s_2$ and $h$, and thus the same setting is used for all experiments. Unless otherwise stated, the parameters $r = 5$ and $|\mathcal{I}| = 2$ are used. The effects of these parameters are discussed further in the supplemental material [22].

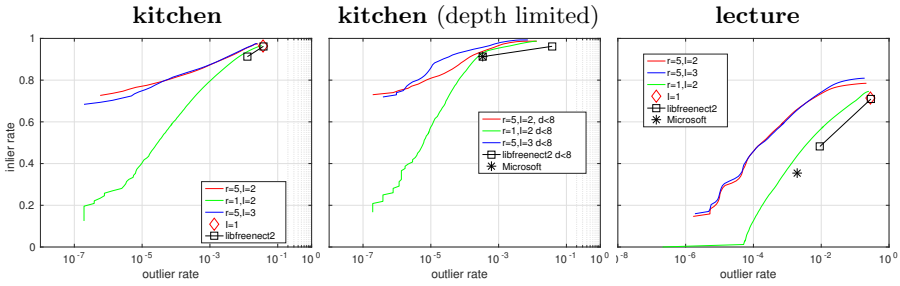



**Fig. 5.** Inlier and outlier rate plots. Each point or curve is the average over 25 frames.

### 4.7 Coverage Experiments

We have used the unwrapping datasets described in section 4.3 to compare the methods in terms of inliers (correctly unwrapped points), and outliers (incorrectly unwrapped points). A point is counted as an inlier when a method outputs a depth estimate which is closer than 30cm to the ground truth, and an outlier otherwise. These counts are then divided by the number of valid points in the ground truth to obtain inlier and outlier rates.

Figure 5 shows plots of inlier rate against outlier rate for our method, for the full range of thresholds on the output confidence in (25). As a reference, we also plot the output from *Microsoft* and *libfreenect2*, as well as *libfreenect2* without the outlier threshold, and *libfreenect2* where the hypothesis selection is done by minimising (14), instead of using the CRT approach in section 2.2 (labelled $I = 1$ in the legend). As can be seen in figure 5 middle, the performance of *libfreenect2* and *Microsoft* are similar on short range scenes with a depth limit (this is expected, as the *libfreenect2* source mentions it being based on disassembly of the Microsoft SDK).

As can be seen, the proposed method consistently has a higher inlier rate at the same outlier rate, when compared to *libfreenect2* with the same spatial support, i.e. $r = 1$. When the spatial support size is increased, the improvement is more pronounced.

Performance for scenes with larger depth is exemplified with the **lecture** dataset. With the depth limit removed, we get significantly more valid measurements at the same outlier rate. The *Microsoft* method has a hard limit of 8m and cannot really compete on this dataset; it only reaches about 35% inlier rate. The *libfreenect2* method without the depth limit reaches 48% inliers, at a 1% outlier rate. At the 1% outlier rate, the proposed method has a 73% inlier rate, which is an relative improvement of 52% over *libfreenect2*.

The performance is improved slightly for $|\mathcal{I}| = 3$ compared with $|\mathcal{I}| = 2$. While still having frame rates over 30 fps for a spatial support of $r = 5$, we consider the costs to outweigh the small improvement, and thus favour the setting of $|\mathcal{I}| = 2$.

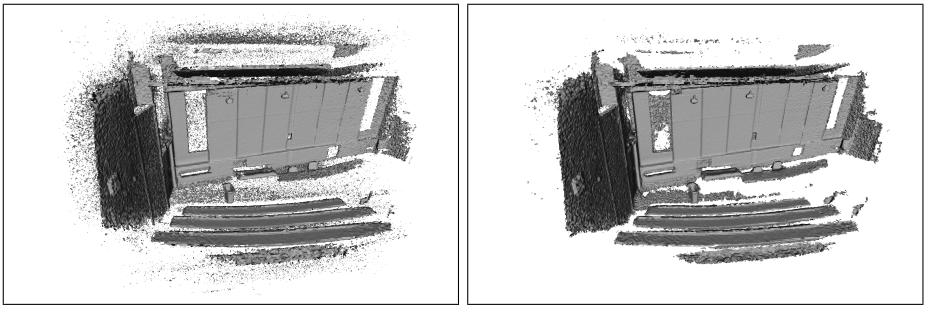


**Fig. 6.** Meshes of **lecture** scene from KinFu. Left: unwrapped with *libfreenect2*. Right: unwrapped with the proposed method.

### 4.8 Kinect Fusion

We have implemented a data-logger that saves all output from the Kinect v2 to a file for later playback. This allows us to feed the Kinect Fusion implementation KinFu in the *Point Cloud Library* [24] with Kinect v2 output unwrapped with both *libfreenect2* and the proposed method. Figure 6 shows two meshes obtained in this way. As can be seen Kinect Fusion benefits from the proposed approach by generating models with fewer outlier points, and consistently more complete scene details. See [22] for more examples.

## 5 Concluding Remarks

This paper introduces a new multi-frequency phase unwrapping method based on kernel density estimation of phase hypotheses in a spatial neighbourhood. We also derive a new closed-form expression for prediction of phase noise and show how to utilize it as a measure of confidence for the measurements.

Our method was implemented and tested extensively on the Kinect v2 time-of-flight depth sensor. Compared to the previous methods in *libfreenect2* and *Microsoft Kinect SDK v2* it consistently produces more valid measurements when using the default depth limit of 8m, while maintaining real-time performance. In large-depth environments, without the depth limit, the gains are however much larger, and the number of valid measurements increases by 52% at the same outlier rate.

As we have shown, the proposed method allows better 3D scanning of large scenes, as the full 18.75m depth range can be used. This is of interest for mapping and robotic navigation, where seeing further allows better planning. As the method is generic, future work includes applying it to other multi-frequency problems such as Doppler radar [1] and fringing [5, 6].

**Acknowledgements** This work has been supported by the Swedish Research Council in projects 2014-6227 (EMC2) and 2014-5928 (LCMM) and the EU's Horizon 2020 Programme grant No 644839 (CENTAURO).

## References


1. Trunk, G., Brockett, S.: Range and velocity ambiguity resolution. In: IEEE National Radar Conference. (1993) 146–149
2. Sell, J., O'Connor, P.: The Xbox One system on a chip and Kinect sensor. IEEE Micro **34**(2) (March-April 2014)
3. Newcombe et al., R.A.: KinectFusion: Real-time dense surface mapping and tracking. In: IEEE ISMAR'11, Basel, Switzerland (October 2011)
4. Jongenelen, A.P.P., Bailey, D.G., Payne, A.D., Dorrington, A.A., Carnegie, D.A.: Analysis of errors in tof range imaging with dual-frequency modulation. IEEE transactions on instrumentation and measurement **60**(5) (May 2011)
5. Wang, Z., Du, H., Park, S., Xie, H.: Three-dimensional shape measurement with a fast and accurate approach. Applied optics **48**(6) (2009) 1052–1061
6. Gorthi, S.S., Pastogi, P.: Fringe projection techniques: Wither we are? Optics and Lasers in Engineering **48**(2) (2010) 133–140
7. Hansard, M.: Chapter 2: Disambiguation of time-of-flight data. In: Time-of-Flight Cameras, Springer Briefs in Computer Science. (2013)
8. Crabb, R., Manduchi, R.: Fast single-frequenct time-of-flight range imaging. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'15). (2015)
9. Droeschel, D., Holz, D., Behnke, S.: Multi-frequency phase unwrapping for time-of-flight cameras. In: IEEE International Conference on Intelligent Robots and Systems. (2010)
10. Kirmani, A., Benedetti, A., Chou, P.A.: SPUMIC: Simultaneous phase unwrapping and multipath interference cancellation in time-of-flight cameras using spectral methods. In: IEEE International Conference on Multimedia & Expo (ICME'13). (2013)
11. Feigin, M., Bhandari, A., Izadi, S., Rhemann, C., Schmidt, M., Raskar, R.: Resolving multipath interference in Kinect: An inverse problem approach. IEEE Sensors Journal (2015) Accepted, On-line.
12. Mei, J., Kirmani, A., Colaco, A., Goyal, V.K.: Phase unwrapping and denoising for time-of-flight imaging using generalized approximate message passing. In: IEEE International Conference on Image Processing (ICIP'13). (2013)
13. Frank, M., Plaue, M., Rapp, H., Köthe, U., Jähne, B.: Theoretical and experimental error analysis of continuous-wave time-of-flight range cameras. Optical Engineering **48**(1) (January 2009)
14. Bamji, C., Charbon, E.: US 6,515,740 B2: Methods for CMOS-compatible three-dimensional image sensing using quantum efficiency modulation (2004)
15. Bamji et al., C.S.: A 0.13 $\mu$m CMOS system-on-chip for a $512 \times 424$ time-of-flight image sensor with multi-frequency photo-demodulation up to 130 MHz and 2 GS/s ADC. IEEE Journal on Solid-State Circuits **50**(1) (January 2015)
16. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT Press (2012)
17. Uhlmann, J.: Dynamic Map Building and Localization: New Theoretical Foundations. PhD thesis, University of Oxford (1995)
18. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. 2nd edn. Cambridge University Press (2003)
19. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer-Verlag New York, Inc. (2010)
20. Forssén, P.E.: Low and Medium Level Vision using Channel Representations. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden (March 2004) Dissertation No. 858, ISBN 91-7373-876-X.

21. Felsberg, M., Forssén, P.E., Scharr, H.: Channel smoothing: Efficient robust smoothing of low-level signal features. IEEE TPAMI **28**(2) (February 2006) 209–222
22. Anonymous: Efficient multi-frequency phase unwrapping using kernel density estimation, supplemental material. Technical report, A Department (2016)
23. Khronos Group: OpenCL language specification. https://www.khronos.org/opencl/ (2015)
24. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE ICRA. (May 9-13 2011)