Black-Box Techniques for 3D Orientation Estimation

Gilad Freidkin — Guy Cohen

Abstract

How can we determine the 3D orientation of an object from just a single 2D image? This fundamental problem has many applications in robotics and computer vision, yet it remains challenging to this day. We propose a novel formulation of this task as a black-box optimization problem, enabling the use of black-box optimization algorithms. In addition, we propose a novel symmetry-invariant evaluation object views which we use to asses different algorithms performance. This work presents a comprehensive evaluation of over 200 black-box optimization strategies on this problem, along with extensive analysis that sheds light on the strengths and limitations of applying black-box optimization to this intricate task. Our analysis shows that some strategies outperform a classical baseline method, identifying promising algorithms and avenues for future research.

Introduction

Estimating the 3D orientation of an object from a single 2D image is a significant challenge in robotics and computer vision tasks. This problem is difficult because inferring the pose from a 2D projection is inherently ambiguous, with multiple possible 3D orientations potentially explaining the same 2D observation, which arises from multiple symmetric views. Previous approaches have used classical feature-matching techniques and recent methods use learning-based deep neural networks, but each has its limitations. In this work, we propose tackling the 3D orientation estimation task through a unique lens of black-box optimization algorithms, which remains largely unexplored to this day. We develop a framework that enables black-box algorithms to iteratively render synthetic views of the 3D object model at different orientations and compare them to the input image, until finding the best match, and from it inferring the orientation of the original input. We evaluate several similarity methods and perform systematical analysis of over 200 different black-box optimization strategies on this problem. In addition we introduce a novel orientation similarity metric tailored for symmetric object views which we use to evaluate the different algorithms. Our extensive analysis sheds light on the strengths and limitations of applying black-box optimization to this challenging problem.

Example 1 *Imagine the following scenario: A factory has a conveyor belt which transports objects from one machine to the next, when suddenly one of the transported items falls. A robotic arm stationed nearby is tasked with picking up this object, but in order to get a firm grip the robotic-arm needs to know the fallen object's location. Thankfully, there is a range sensor present, which assists the robotic arm to calculate the spatial location of the object. What's left to determine is the orientation of the fallen object, so the arm could adjust and achieve a firm grip. Here, a camera, which monitors the conveyor belt, comes into play, as it captured an image of the fallen object. An algorithm processes this cameras footage, and manages to determine this object's orientation on the ground, which is afterwards used to calculate the best grip position.*

Related Work

The task of determining the orientation of 3D objects has garnered substantial interest in research endeavors. A significant portion of the research efforts have been dedicated to detecting the 6D pose [7] [3] [12], encompassing both the spatial positioning and rotational orientation. However, many of the techniques developed for this comprehensive pose estimation task are also applicable to the more specific objective of orientation detection, which is the focus of our work. The approaches for this task can be broadly classified into two primary categories: those that operate on intensity images, such as RGB or grayscale images, and those that leverage depth images.

Within the realm of intensity image-based methods, there are two prominent subcategories: learning-based techniques and feature matching approaches. We will delve into each of these in greater detail.

Feature Matching Approaches

Approaches in this category rely on detecting and matching distinctive 2D features within the image. These feature-based techniques typically involve identifying and describing notable features like corners, edges, or blob-like structures in the input image. The extracted 2D features are then matched against a database of known 3D features associated with the target object. The correspondences between the 2D and 3D features established through this matching process are subsequently used to estimate the pose of the object relative to the camera. This is achieved using algorithms such as Perspective-n-Point, for example EPnP [5].

Although feature-based approaches have demonstrated robustness in various scenarios, their performance can be affected by factors such as occlusion, clutter, and lack of texture in the input images.

Learning-Based Techniques

In recent years, there has been a notable shift towards the adoption of deep learning methods, particularly convolutional neural networks (CNNs) for tackling this problem, such as PoseCNN [11]. Numerous works have proposed solutions that leverage the powerful capabilities of CNNs for this task. One intuitive way to harness the capabilities of CNNs is to directly regress the orientation parameters and consequently directly predict the orientation; this approach takes advantage of the available data labels in a straightforward manner. Other works employ generative methods to overcome occlusions and even as a similarity measure between a between a given image and a database [9]. It is worth noting that a notable disadvantage of these learning-based methods is the requirement for extensive datasets and time-consuming training processes. Also, usually the decision making process of learning based techniques is hard to explain and reason about, which hinders the ability of providing any formal performance guarantees.

Contribution

We decided to investigate a new unexplored area of algorithms for this problem, called *Black-Box Optimization Algorithms*. We investigate the feasibility of using these methods, and perform a wide comparative analysis of algorithms from this field.

To this end, we created a broad framework for comparing and evaluating black-box algorithms, which can be easily extended beyond the use cases of our work. This framework immensely simplifies the workflow of using optimization algorithms with data from a simulated environment.

Additionally, we present a novel symmetry-invariant evaluation function for algorithms and provide some theoretical properties of it.

We conclude that using black-box algorithms is a new viable approach for the orientation estimation task, and that is a brand new and exciting research route.

Background

Black-Box Optimization

Black-box optimization refers to a class of algorithms designed to address optimization problems where the objective function l lacks explicit analytical representation and is usually characterized by complexity. These algorithms operate under the premise that only evaluations of the objective function are accessible, without any knowledge of its analytical form or access to its gradient information. The fundamental aim of black-box optimization is to efficiently locate optimal or near-optimal solutions within a given search space by iteratively exploring and exploiting information gained from function evaluations. This approach is particularly valuable in areas where traditional optimization methods are infeasible or impractical due to the opaque nature of the objective function, which is exactly the case in this work. We discuss the objective function later, and the non-analytical nature of it will become clear.

A general purpose black-box algorithm Alg that uses strategy π to minimize an objective function $l : X \to \mathbb{R}$ can be described as follows:

Algorithm 1: Generic Black-Box Algorithm $\boldsymbol{Alg(l)}$

```
1. **Let** $\hat{x} = None$
2. **Let** $\hat{y} = \infty$
3. **Until** stop criterion met **do**:
   (a) **Choose** $\tilde{x} \in X$ by algorithm's strategy $\pi$
   (b) **Let** $\tilde{y} = l(\tilde{x})$
   (c) **If** $\tilde{y} < \hat{y}$ **do**:
       i. **Set** $\hat{x} := \tilde{x}$
       ii. **Set** $\hat{y} := \tilde{y}$
4. **Return** $\hat{x}$
```

Object Orientation

The orientation of a body in a coordinate system can be described in various ways. Throughout this work, we refer to the orientation using two different conventions: Euler Angles and Axis-Angle representation.

Euler Angle Convention describes an orientation as three successive rotations around three global axes, commonly referred to as the X, Y, and Z. The order of rotations around the axes is not commutative, and we use the rotation order XYZ. The ranges of the Euler angles are as follows:

- X axis rotation ϕ - the range of the ϕ angle typically lies within the interval $[0, 2\pi)$, representing a full revolution around the X-axis.
- Y axis rotation θ - the range of the θ angle is restricted to the interval $[0, \pi]$, representing the rotation around the Y-axis.
- Z axis rotation ψ - similar to θ, the range of the ψ angle lies within the interval $[0, 2\pi)$, again representing a full revolution around the Z-axis.

Axis-Angle Convention describes an orientation by a rotation angle θ about a unit vector $\vec{v}$, which represents the axis of rotation. A rotation in this convention can be described by a single vector $\theta\vec{v}$. This convention consists of two components:

- The axis of rotation $\vec{v}$ - this is a 3-dimensional unit vector ($\|\vec{v}\| = 1$) that defines the direction of the axis around which the rotation occurs.
- Angle of Rotation θ - this is the angle that specifies the amount of rotation about the axis of rotation $\vec{v}$. Angle θ is restricted to the interval $[0, \pi]$.

Problem Statement

Imagine we have a single observation of a known object. How can we determine this object's orientation from this single observation? In this section we will describe the formal framework of describing this problem, and introduce some notations which will be used throughout this paper.

Let $\mathcal{O}$ be an object of a known shape, and let $\tilde{\mathcal{O}}$ be a 3D CAD model of this object. The state of object $\mathcal{O}$ relative to a viewer can be described with spatial (translation) and rotational coordinates (relative to the viewer), which we denote as t and r respectively . Throughout this work we assume that the spatial location of the object is known, therefore we omit t from object state description.

Let F be a function that receives an observation of the object's state (and perhaps other auxiliary data), and returns a predicted orientation. Formally speaking:

$$F : \mathcal{I}_r; \tilde{\mathcal{O}} \rightarrow \hat{r}$$

where:

- $\mathcal{I}_r$ - an RGB image of an instance of object $\mathcal{O}$ with orientation r. This is the observation of the object that F receives as input.
- $\hat{r}$ - the predicted orientation of the object.
- F uses the 3D model $\tilde{\mathcal{O}}$ as auxiliary information.

Our goal is to find function F that achieves "low" error, meaning that that it predicts the orientation accurately, by some metric. The question of which metric is used to measure algorithmic error is not a trivial one, and we propose such metric later.

Now that we've defined all the necessary components, lets lay some ground rules for notation usage throughout this work:

- r refers to the ground truth orientation of object $\mathcal{O}$.
- $\tilde{r}$ refers to a simulated orientation of an object model $\tilde{\mathcal{O}}$.
- $\hat{r}$ refers to the predicted orientation by the function F.

Example 2 (continued) *Recall, that we mentioned a scenario where an algorithm was used to predict the orientation of an object which fell from a conveyor, using an observation from a camera. Applying the notations we defined above we can say that the fallen object is of type $\mathcal{O}$, the 3D model of the object is $\tilde{\mathcal{O}}$, the orientation in which this object is now lying down is r and the observation that the camera made of this object's state is $\mathcal{I}_r$. The algorithm that predicted the orientation is F , and it's prediction is $\hat{r} = F(\mathcal{I}_r; \tilde{\mathcal{O}})$.*

Method

As we've said before, we want to find function F that receives an image as input and returns a predicted orientation $\hat{r}$. We focus on one type of such functions, which are called black-box optimization algorithms. The basic idea behind this approach is that given an input image $\mathcal{I}_r$ algorithm F probes different orientations $\tilde{r}$ by some strategy π, using the CAD model it generates a synthetic image $\tilde{\mathcal{I}}_{\tilde{r}}$, and measures the similarity $l(\mathcal{I}_r, \tilde{\mathcal{I}}_{\tilde{r}})$ (l is the objective function) between these two images. Eventually, the algorithm returns the value of $\tilde{r}$ that achieved the best similarity (i.e. the lowest value of l, since we formalize it as a minimization task).

We propose a way to modify any black-box algorithm with strategy π for the task of orientation estimation. Given an objective function $l : \mathcal{I} \times \mathcal{I} \rightarrow \mathbb{R}$, a black-box algorithm $F : \mathcal{I}_r; \tilde{\mathcal{O}} \rightarrow \hat{r}$ with strategy π for the task of orientation estimation operates as follows:

Algorithm 2: Orientation Black-Box Function $\boldsymbol{F(\mathcal{I}_r)}$

```
1. Let $\hat{r} = None$
2. Let $\hat{y} = \infty$
3. Until stop criterion met do:
   (a) Choose orientation $\tilde{r}$ by algorithm's strategy $\pi$
   (b) Using CAD model $\tilde{\mathcal{O}}$ generate simulated image $\tilde{\mathcal{I}}_{\tilde{r}}$
       in which object $\tilde{\mathcal{O}}$ appears at orientation $\tilde{r}$
   (c) Let $\tilde{y} = l(\mathcal{I}_r, \tilde{\mathcal{I}}_{\tilde{r}})$
   (d) If $\tilde{y} < \hat{y}$ do:
       i. Set $\hat{r} := \tilde{r}$
       ii. Set $\hat{y} := \tilde{y}$
4. Return $\hat{r}$
```

XorDiff Evaluation Metric

Motivation: In order to evaluate a black-box algorithm F we need to establish a metric which measures algorithm's prediction performance. This metric measures the error between the original orientation r and the predicted orientation $\hat{r}$.

First, we explain why non-symmetry-invariant evaluation metrics do not suffice. Objects with multiple symmetric views appear similar in two different images, while in reality their real-world orientation is completely different. As result, such different orientations, say r, r', would produce the same input image $\mathcal{I} = \mathcal{I}_r = \mathcal{I}_{r'}$. Since the algorithm only receives the image $\mathcal{I}$ as input, it has no way of knowing which of these ambiguous orientations produced that said input. Subsequently, an algorithm would return the same prediction for both inputs $\mathcal{I}_r$ and $\mathcal{I}_{r'}$, and be penalized for its mistake, even though there is no way to avoid it.

To resolve this ambiguity, an algorithm would need additional images from different viewpoints, but that conflicts with out assumption that only a single image $\mathcal{I}_r$ is available. Since this problem is unresolvable under our assumptions, we treat all orientations of symmetric views as correct predictions as well.

Definition: We present our novel evaluation metric XorDiff. That metric is symmetry-invariant, and very computationally cheap. Also, in a mathematical analysis of this metric, we show that its normalized on different types of objects, allowing for its usage to compare performance of algorithms on different types of objects. First, we present the basic notations:

- For an image of size $n \times m$ we denote it's coordinate set as follows: $C = \{(x, y) | 0 \leq x < n, 0 \leq y < m\}$. Instead of writing everywhere (x, y), we will refer to coordinates by the notation x.
- By $d : C \rightarrow \mathbb{R}^+$ we denote a depth-map over coordinates C, for which the distance from the sensor to the an object

at coordinate x is exactly the value $d(x)$.

- By $c \subseteq C$ we denote the subset of coordinates in which the object $\mathcal{O}$ appears. Similarly, $\tilde{c} \subseteq C$ are the coordinates of the simulated CAD model $\tilde{\mathcal{O}}$.
- Let $k > 0$ be a hyper-parameter of the metric. k is the penalty value for points in $(c \cup \tilde{c}) \setminus (c \cap \tilde{c})$. An appropriate value of k is to make sure that the evaluation error for points $c \cap \tilde{c}$ is on the same scale as for the points $c \cup \tilde{c}$.
- Let $p \geq 1$ be a degree of a norm. This is a hyper-parameter of our proposed metric.
- We denote the symmetric difference between sets A and B as $A \ominus B \equiv (A \cup B) \setminus (A \cap B)$

Given the above notations, we define XORDIFF as follows:

$$\mathit{XorDiff}_p(d, \hat{d}) = \frac{\|\alpha(C)\|_p}{k|c \cup \tilde{c}|} = \frac{1}{k|c \cup \tilde{c}|} \left(\sum_{x \in C} |\alpha(x)|^p \right)^{(1/p)}$$

Where $\alpha : C \to \mathbb{R}$ is defined as:

$$\alpha(x) = \begin{cases} d(x) - \hat{d}(x) & x \in c \cap \tilde{c} \\ k & x \in c \ominus \tilde{c} \\ 0 & otherwise \end{cases}$$

The idea of this metric is that for coordinates which are within the intersection of the objects' masks we compute a regular L_p loss, but for coordinates which are not within the intersection (but do belong to some object) we penalize by some constant penalty k. This metric over the depth-maps is preferred over the unmodified L_p loss over the depths, since otherwise the L_p loss values at coordinates $x \in (c \cup \tilde{c}) \setminus (c \cap \tilde{c})$ would dominate the loss in coordinates $x \in c \cap \tilde{c}$, which happens because the depth difference between an object and the background is likely on a different scale than between two objects in the foreground. In addition, this evaluation loss is normalized because of the division by $k|c \cup \tilde{c}|$. Since XORDIFF is modular in p, by setting $p = 1$ or $p = 2$ we achieve improved versions of the MAE (L1) and RMSE (normalized L2) losses, respectively.

You've probably noticed that we defined the XORDIFF metric on depth-maps and not on orientations. To evaluate two orientations r and $\hat{r}$ simply use the following procedure:

Algorithm 3: Evaluation Procedure **Eval($r, \hat{r}$)**

```
1. Generate reference depth-map $d$ using CAD model $\tilde{\mathcal{O}}$ at orientation $r$
2. Generate prediction depth-map $\hat{d}$ using CAD model $\tilde{\mathcal{O}}$ at orientation $\hat{r}$
3. Set $v = \mathit{XorDiff}_1(d, \hat{d})$
4. Return $v$
```

Exploring Different Approaches

Since a black-box algorithm for our task is comprised of a strategy and a similarity function, the search space for exploring different algorithmic approaches is 2-fold:

- Explore different objective functions l, and find which ones are fitting.
- Investigate different sampling strategies π, and determine which ones yield good results.

Qualities of a Good Objective Function: Since we seek to minimize the evaluation function $Eval$, which represents the "ground truth" error, ideally an algorithm would perform minimization of $Eval$ function directly. Unfortunately, calculating $Eval$ requires depth-information that the algorithm doesn't have. As result, we seek to find an objective function l that works on RGB inputs, which serves as a good approximation to the XorDiff evaluation function.

Using the evaluation function $Eval(r, \tilde{r})$, we describe the qualities of a good choice of l:

1. **Mandatory Condition:** l is monotonically correlated to $Eval$. Otherwise, minimizing l doesn't imply minimization of the ground truth evaluation metric. This condition guaranties that minimizing l will lead to a local minimum of $Eval$.
2. **Best Case Scenario:** l is linearly correlated to $Eval$. This directly implies the the above point, but also means that the black-box algorithm is directly minimizing the $Eval$ function itself (which is unknown to it), since $\arg\min x = \arg\min \beta x$ for $\beta > 0$. As result, the orientation resulting a global minima of the objective function also results the global minima in the $Eval$ error metric.

Evaluating Objective Functions: To determine if an objective function l is suitable for the task, as defined in the previous section, we begin by generating large amount N of random pairs of orientations $R = \{(q, q')\}_{1..N}$. For each orientation pair $(q, q') \in R$ we render images of the CAD model in these orientations $(\tilde{\mathcal{I}}_q, \tilde{\mathcal{I}}_{q'})$, calculate the evaluation metric $Eval(q, q'; \tilde{\mathcal{O}})$ and the matching objective loss $l(\tilde{\mathcal{I}}_q, \tilde{\mathcal{I}}_{q'})$. This calculation (over the same R) is performed over a wide variety of objects.

Now, we have all the data to calculate correlation between the $Eval$ metric and the objective function l. For each object individually, we calculate Pearson correlation (which measures linear correlation), and also Spearman [8] and Kendall's [4] rank-order correlations (which measures monotonic correlation). Afterwards, we average all of these measures among all of the objects, resulting in a single value per measure. By calculating these correlations among a wide variety of different objects, the resulting values after averaging become quite robust. Using these measures we can determine how well l holds for each of the qualities of a good objective function, which we mentioned earlier.

Analyzing an Orientation Sampling Strategy π:

To compare between different sampling strategies, we begin by fixing some objective function l. Let π be a sampling strategy, and let F_π denote a black-box algorithm that uses the objective function l and the sampling strategy π.

Let $\mathcal{B} = \{\tilde{\mathcal{O}}_1, \tilde{\mathcal{O}}_2, ...\}$ be a set of CAD models. The error of a strategy π w.r.t $\mathcal{B}$ is defined as follows:

$$Err(\pi) = \mathop{\mathbb{E}}_{\tilde{\mathcal{O}}\in\mathcal{B}} \mathop{\mathbb{E}}_{r} \left[Eval(r, F_\pi(\tilde{\mathcal{I}}_r; \tilde{\mathcal{O}})) \right]$$

where:

- r is an orientation drawn uniformly from the orientation space
- $\tilde{\mathcal{I}}_r$ is a render of CAD model $\tilde{\mathcal{O}}$ at orientation r

Since in practice calculating the expected value over all orientations is impossible, we calculate the empirical mean over a large set of orientations. Additionally, by choosing the object set $\mathcal{B}$ to be composed from a wide variety of different objects with different characteristics, this Err measure becomes more robust and generalized.

Formal Analysis

Here we present a formal analysis of the mathematical properties of our evaluation metric XorDiff.

Symmetry: It is easy to see that this metric is symmetric, meaning that:

$$\textit{XorDiff}(d, \hat{d}) = \textit{XorDiff}(\hat{d}, d)$$

Lower Bound: There is a universal lower bound for our evaluation function. XorDiff is strictly larger than the following:

$$\textit{XorDiff}_p(d,\hat{d}) = \left\| \frac{\alpha(C)}{k|c\cup\tilde{c}|} \right\|_p \geq \left\| \frac{k|c\ominus\tilde{c}|}{k|c\cup\tilde{c}|} \right\|_p =$$

$$\left\| \frac{|c\cup\tilde{c}| - |c\cap\tilde{c}|}{|c\cup\tilde{c}|} \right\|_p = \|1 - \text{IOU}(c,\tilde{c}))\|_p$$

Upper Bound: For an object $\mathcal{O}$ and it's matching CAD model $\tilde{\mathcal{O}}$, let $d, \hat{d}$ be depth-maps of $\mathcal{O}$ and $\tilde{\mathcal{O}}$, respectively. We define k as follows:

$$k = \max_{d,\hat{d}} \{ |d(x) - \hat{d}(x)| : x \in c\cap\tilde{c} \}$$

From the definition of α we get that $\alpha(x) \leq k$ for any point $x \in c\cup\tilde{c}$, and otherwise 0. As result, it holds that:

$$\textit{XorDiff}_p(d,\hat{d}) = \left\| \frac{\alpha(C)}{k|c\cup\tilde{c}|} \right\|_p \leq \left\| \frac{k|c\cup\tilde{c}|}{k|c\cup\tilde{c}|} \right\|_p = 1$$

Expectation Upper Bound: For an object $\mathcal{O}$ and it's matching CAD model $\tilde{\mathcal{O}}$, let $d, \hat{d}$ be random depth-maps, which were acquired by uniformly sampling random orientations of objects $\mathcal{O}$ and $\tilde{\mathcal{O}}$ respectively. In contrast to before, where k was defined as the global max depth-difference, we define a less strict value, as follows:

$$k = \mathop{\mathbb{E}}_{d,\hat{d}} \left[\max\{ |d(x) - \hat{d}(x)| : x \in c\cap\tilde{c} \} \right]$$

Using that new definition of k, in the appendix we show:

$$\mathop{\mathbb{E}}_{d,\hat{d}} \left[\textit{XorDiff}_{p=1}(d,\hat{d}) \right] \leq 1$$

Empirical Evaluation

Remember, that to find a black-box algorithm we perform a search over the following 2 parameters: the objective function l and an orientation sampling strategy π. In this section we discuss how to choose a fitting objective function, and afterwards we analyze the evaluation results of different sampling strategies, with the goal of understanding how different approaches compare to each other, and determine if they are viable for the task.

Dataset

To comprehensively evaluate the performance of various black-box algorithms, we constructed three datasets, each serving a distinct purpose in our analysis. These datasets encompass a collection of orientations, on which the algorithms are evaluated.

1. Dataset $\mathcal{D}_1$ with 20 randomly sampled orientations for initial algorithm assessment and selection.
2. Dataset $\mathcal{D}_2$ with 20 randomly sampled orientations for hyper-parameter tuning of selected algorithms.
3. Dataset $\mathcal{D}_3$ with 27 equidistantly spaced orientations for in-depth analysis of the promising algorithms. This dataset is constructed by placing 27 evenly-spaced points in the orientation space.

In our work, we consider nine different object types. Consequently, when evaluating an algorithm using one of these datasets, the algorithm is evaluated on each object type, resulting in a total of $9N$ evaluation data points for a dataset of length N. Despite the seemingly small size of our datasets, the actual number of evaluation data points is nearly an order of magnitude larger.

It is important to note that sampling uniformly from the orientation space or obtaining equidistant orientations is highly non-trivial. We provide a more detailed discussion on the methodology employed for achieving this in the the appendix.

Experimental Setup

First, we decided to vary the object set $\mathcal{B}$ on which the algorithms are evaluated, to achieve robust results. The objects we selected are the following: *'Android Model', 'Dinosaur', 'Hammer', 'Mug', 'Nescafe Jar', 'Screwdriver', 'Shoe', 'Miniature Sofa' and 'Stack-Rings'*. Most of these models were obtained from the Google Scanned Objects Dataset [2]. This dataset contains scanned objects rather than ones modeled by hand, which consequently realistically reflects real object properties and not an idealized recreations.

For each object type, we create two distinct CAD models. The first model is used to simulate the object in real-world conditions. Therefore, this CAD model is simulated with shadows, different lighting conditions and with reflective materials. This model represents the actual "real-world" object $\mathcal{O}$, and using this model we generate the images $\mathcal{I}_r$ which are provided as an input for the black box algorithms. Images rendered from this model are the images on which the black-box algorithms are tested on. On the other hand,

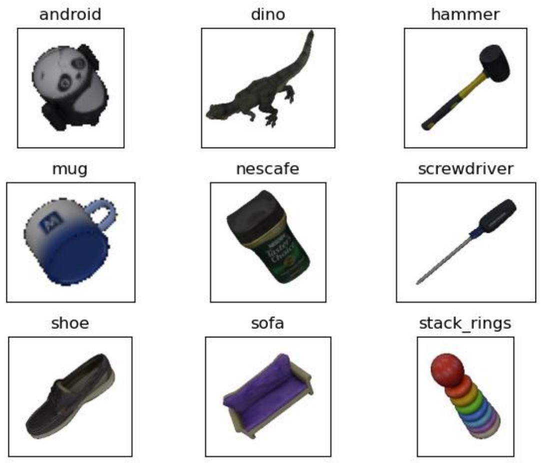

Figure 1: All object on which the algorithms were tested

the second model is the object simulated in simplistic manner. This model is simulated without any reflections, shadows or any other effects whatsoever. This CAD model serves as the abstract object representation $\tilde{\mathcal{O}}$, from which algorithms samples different views $\tilde{\mathcal{I}}_{r'}$ (which it compares with the target image $\mathcal{I}_r$), in an attempt to find the original orientation. The reason for creating two different CAD models for each object type is to resemble more closely real-world conditions, since, for example, it's quite unlikely that the lighting conditions in a real world input image $\mathcal{I}_r$ would match to the ones in a simulation $\tilde{\mathcal{O}}$. Figure 2 depicts the difference between the two object models.

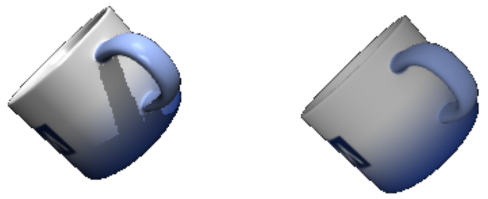

Figure 2: Left - 'Mug' object rendered by the first CAD model, which resembles real-world conditions. This image serves as the input $\mathcal{I}_r$ for a black-box algorithm. Right - the same object rendered by the second CAD model, representing the abstract object representation $\tilde{\mathcal{O}}$. Rendered views from this CAD model $\tilde{\mathcal{I}}_{r'}$ are compared with the original image $\mathcal{I}_r$ using some objective function l.

Now that we've defined everything needed to conduct the empirical evaluation. The evaluation flow goes as following:

1. For each object, we select an appropriate value for the parameter k in the XorDiff evaluation metric.
2. We perform the evaluation of various objective functions, to identify those that satisfy our predefined conditions for a suitable objective function.
3. We perform initial evaluation of different sampling strategies using dataset $\mathcal{D}_1$, with up to 15 seconds run-time per sample (i.e. input image $\mathcal{I}_r$) permitted. This results in total compute time of about 160 hours.
4. A subset of algorithms is chosen to be investigated into greater depths. This subset consists of the best-performing algorithms among all the ones benchmarked using $\mathcal{D}_1$, and also a selection of classic techniques, along with our baseline strategy.
5. The selected algorithms' hyper-parameters are tuned using dataset $\mathcal{D}_2$. Again, each algorithm is permitted to run up to 15 seconds per sample.
6. Finally, an extensive analysis is conducted of the selected few algorithms. The analysis is conducted using dataset $\mathcal{D}_3$, with permitted run-time of up to 4 minutes per sample . This results in a total compute time of 145 hours.

The results obtained from the final evaluation will be compared against a baseline algorithm, *'UniformSampling'*. This algorithm employs a very simple orientation-sampling strategy, of sampling some predetermined amount N of different equidistant orientations from the orientation space. This strategy is very intuitive, since the orientation space is sampled in even intervals all across, which means that with large enough N this strategy will sample an orientation from which the view is similar to the original one. At first glance it might seem easy to create such sampling strategy, but surprisingly, that's not the case. As mentioned earlier, we discuss this in greater detail, along with technical explanation of how to do so, in the appendix.

Regarding computational resources, we employ a combination of desktop computers and a high-performance server with a substantial number of CPU cores to facilitate the necessary computations. Using the high amount of CPUs allows us to perform the evaluation process in parallel, reducing by an order of magnitude the time that is required to perform it.

Results

Choosing an Appropriate k Value for XorDiff Let $d, \hat{d}$ be random depth-maps acquired by uniformly sampling random orientations of object $\tilde{\mathcal{O}}$. For every such object in our object set $\mathcal{B}$ we chose to calculate the hyper-parameter $k_{\tilde{\mathcal{O}}}$ of XorDiff in the following manner:

$$k_{\tilde{\mathcal{O}}} := \mathop{\mathbb{E}}_{d,\hat{d}}\Big[\max\{|d(x) - \hat{d}(x)| : x \in c \cap \tilde{c}\}\Big]$$

In practice, we approximate the expectation using the empirical mean with a large sample size. Given the choice of k described above, in the appendix we show the following:

$$\mathop{\mathbb{E}}_{d,\hat{d}}\Big[\mathit{XorDiff}_1(d, \hat{d})\Big] \leq 1$$

The following Figure 5 illustrates the distribution of XorDiff values among different objects. This distribution was constructed by randomly generating N pairs of uniform orientations, rendering their depth-maps and calculating the corresponding XorDiff of each pair. This was done for each type of object in $\mathcal{B}$.

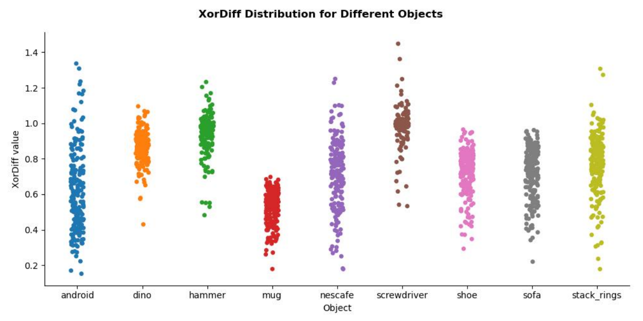

Figure 5: XorDiff distribution for different objects using $k_{\tilde{O}}$

Indeed, we notice that for all objects, the expected XorDiff loss is at most 1, in accordance with our claim.

Furthermore, different objects have different distributions of XorDiff errors. This phenomenon can be explained by the different properties each object holds. For example, XorDiff values of *'mug'* object has a low mean error. This can be explained by the fact that for most pairs of orientations, most of the pixels where one mug appears the second one appears as well, because of the spherical and quite symmetrical shape of this object.

In contrast, highly asymmetrical objects, and especially ones with non-spherical shapes, receive high mean of XorDiff error. These objects in different orientations have a very small amount area of intersection, resulting in large amount of XorDiff penalties (of value k) in the areas where one object appears but not the other. Example for such objects, that can be seen in figure 5, are *'dino'*, *'screwdriver'* and *'hammer'*.

Choosing an Objective Functions To evaluate different objective functions we followed the method described in the 'Evaluating Objective Functions' part. We set $N = 500$ and the set of objects is the same as in seen in figures 1 and 5, which means that each objective function was evaluated on 4500 data-samples.

	Spearman	Kendall	Pearson
IOU	0.55	0.42	0.66
MSE	0.27	0.19	0.35
RMSE	0.40	0.29	0.50
NMI	0.47	0.34	0.65
PSNR	0.27	0.19	0.42
SSIM	0.16	0.11	0.24
Hausdorff	0.60	0.45	0.56
ARE	0.37	0.26	0.46
VI	0.27	0.19	0.34

Table 1: Different correlation metrics, first measured per object and afterwards averaged across all objects.

As can be seen in Table 1 and in Figure 3, among the monotonic correlations there are 2 clear best candidates objective functions: Hausdorff Distance and IOU. As result, these functions meet our first quality for a good objective function.

We can observe that Husdorff's monotonic correlation measures are slightly higher than of IOU's. On the other hand, computing Hausdorff is much more computationally intense and time consuming compared to IOU, which results in significant performance margins between the two. Additionally, the linear (Pearson) correlation of IOU with the evaluation function is significantly higher than of Hausdorff, giving it another edge, according to our second quality of a good objective function.

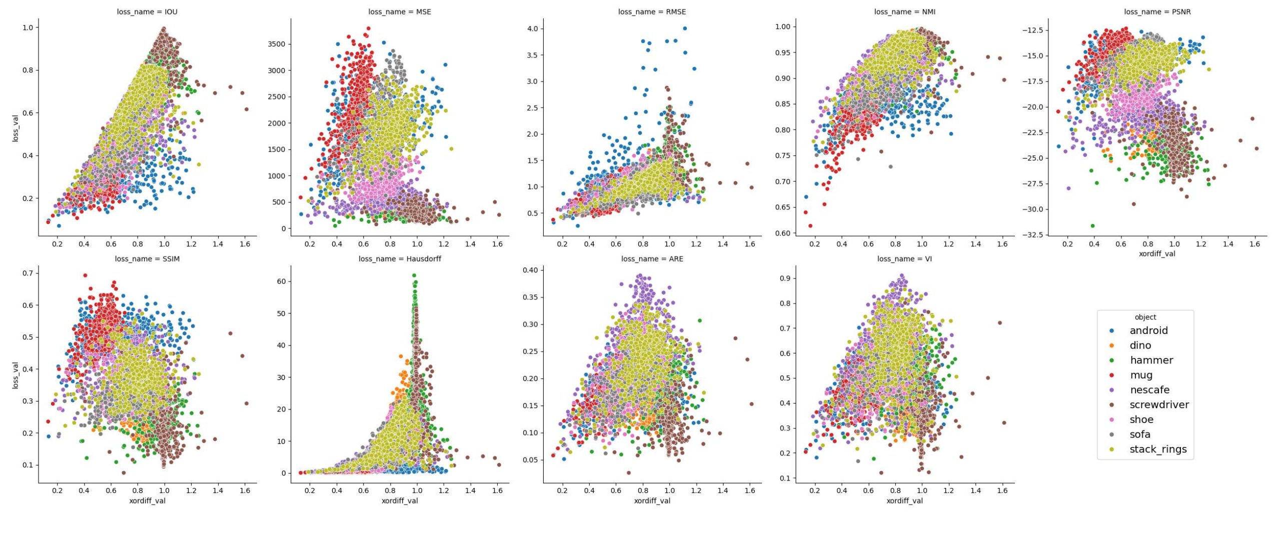

Figure 3: Different objectives as function of XorDiff evaluation error. Objective functions in left-right, top-bottom ordering: IOU, MSE, Root-MSE (Normalized MSE), NMI (Normalized Mutual Information), PSNR (Peak Signal Noise Ratio), SSIM (Structural Similarity), Hausdorff (Hausdorff Distance), ARE (Adapted Rand Error), VI (Variation of Information).

For time considerations, we decided to choose a single objective function for the rest of this work. Taking the above into consideration, we chose IOU as our objective function, since it has the best linear correlation, it is very cheap to compute, and has very good monotonic correlation with the evaluation metric.

The definition of IOU objective function is as follows:

```
Algorithm 4: Objective Function IOU($\mathcal{I}_1, \mathcal{I}_2$)
1. Calculate mask $c_1$ from image $\mathcal{I}_1$, where $c_1$ is the set of all pixels in which the object of interest appears.
2. Calculate mask $c_2$ from image $\mathcal{I}_2$, where $c_2$ is the set of all pixels in which the object of interest appears.
3. Set $v = \frac{|c_1 \cup c_2| - |c_1 \cap c_2|}{|c_1 \cup c_2|}$
4. Return $v$
```

Note, that the this definition of IOU is minimized whenever the size of the intersection $|c_1 \cap c_2|$ becomes bigger, unlike in the standard definition. If we denote the standard definition by IOU', we get $\text{IOU}(\mathcal{I}_1, \mathcal{I}_2) = 1 - \text{IOU'}(\mathcal{I}_1, \mathcal{I}_2)$. The reason that this version of IOU is minimized is because we formulated algorithm's task as a minimization problem.

Initial Algorithm evaluation We curated a set of 212 different algorithms (orientation sampling strategies) and evaluated them using the initial evaluation dataset $\mathcal{D}_1$ and object set $\mathcal{B}$, using the IOU objective function. We implemented 3 strategies, referred as *'UniformSampling', 'IDUniformSampling'* and *'RandomSampling'*. The rest of the strategies were imported from the mealpy [10] library, which is a library dedicated to meta-heuristic algorithms (which is a specific subset of black-box algorithms).

Examining the results, displayed in Figure 4, we can see that while many strategies performed roughly the same, some strategies clearly outperform the others. The baseline algorithm 'UniformSampling' appears in the middle of the upper graph, indicating that there are better strategies if the permitted run-time is 15 seconds.

In order to perform an in-depth evaluation and analysis, we have chosen some of the top performing algorithms: *'OriginalICA', 'OriginalSCSO', 'ImprovedSFO', 'OriginalMPA'*, along with some of the classics: *'BaseGA', 'OriginalDE', 'OriginalHC', 'OriginalPSO'* and our baseline strategy *'UniformSampling'*.

In Depth Evaluation Before Doing the in-depth evaluation, for each one of the chosen algorithms we ran a grid-search on different combinations of hyper-parameters, and using dataset $\mathcal{D}_2$ chose the parameters that yielded the best results.

For the in-depth evaluation of the chosen algorithms dataset $\mathcal{D}_3$ is used. In Figure 6 we can see that the extra run-time was beneficial to the algorithms, as the evaluation error decreased significantly compared to the same algorithms in the initial evaluation. Analyzing Figure 6 further, we spot performance differences between the algorithms. We observe that both *'OriginalISCSO'* and *'OriginalICA'* clearly outperform the rest of the algorithms in terms of final evaluation loss. Additionally, the variance of these algorithms is also comparatively small, indicating that regardless of the input they manage to converge towards a good solution. Another notable algorithm here is 'OriginalPSO', which lacks

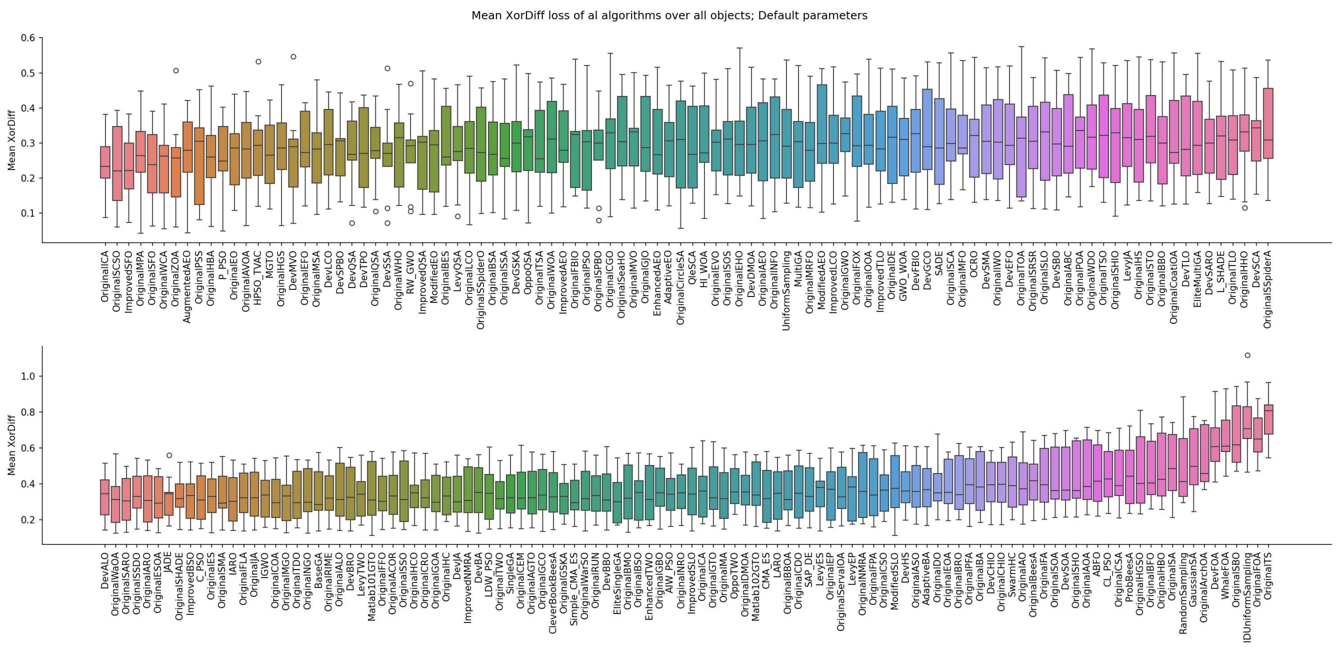

Figure 4: Mean XorDiff values of different strategies (algorithms) over object set $\mathcal{B}$ and evaluation dataset; IOU objective function; run for about 15 second per sample

.

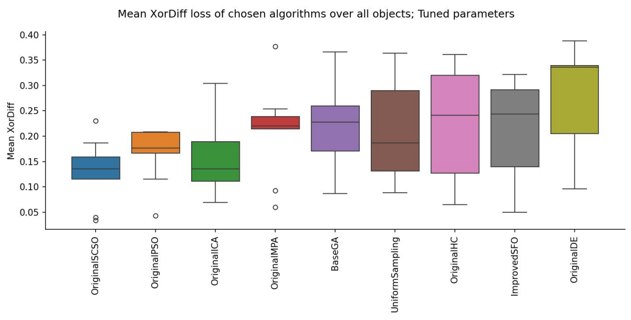

Figure 6: Evaluation results of the algorithms chosen in the last section; evaluated over $\mathcal{D}_3$ dataset; 4 minutes per sample.

behind the previously mentioned two, but still achieves very good results compared to the rest. What's interesting is that PSO (particle-swarm-optimization) strategy is pretty simple, yet it achieves quite good results. Since the evaluation was performed on different objects, that indicates that 'OriginalISCSO' , *'OriginalICA'* and 'OriginalPSO' perform well on a wide variety of objects, making them good candidates for recognizing orientations of new objects that were not included in out object set $\mathcal{B}$.

In contrast to the three strategies (algorithms) mentioned before, the rest of the algorithms suffers either from high variance or high median of the evaluation error. That means that these algorithms perform worse overall, and that their performance varies significantly between different runs on different inputs, making them less robust. Interestingly enough, our baseline strategy 'UniformSampling' suffers from the same faults, even though it samples the orientation space evenly in very small intervals (4 minus of runtime allows for very fine-grained sampling of the orientation space, resulting in about 10,000 evenly spaced orientations).

From this figure we conclude that *'OriginalISCSO'* and *'OriginalICA'* algorithms are superior to our baseline approach, which shows merit in investigating and researching more complex sampling strategies for this problem. Also, 'OriginalPSO' is not very far behind the leading strategies, which shows that even simple algorithms such as Particle-Swarm-Optimization can be superior to our baseline approach.

Figure 7 depicts the objective loss (*IOU*) of each algorithm as a function of time, averaged over all objects and all samples tested. We can see that algorithms that performed well w.r.t XorDiff also performed well w.r.t the objective function (IOU) and those that performed poorly did so in both cases as well. This supports our analysis and decision of IOU as an objective function.

Analyzing Figure 7 further, we observe that different algorithms differ quite significantly in their convergence rate. Given a time limit of up-to 50 seconds, OriginalPSO demonstrates significantly lower IOU loss because of its high convergence rate. Looking at 50 seconds run-time and beyond, the roles change, and OriginalICA and OriginalSCSO take the lead.

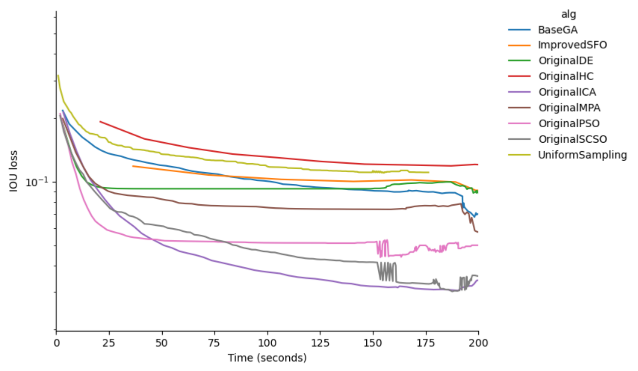

Figure 7: Chosen algorithms objective function(IOU) loss over time; averaged over all objects and samples in $\mathcal{D}_3$; logarithmic scale in y-axis

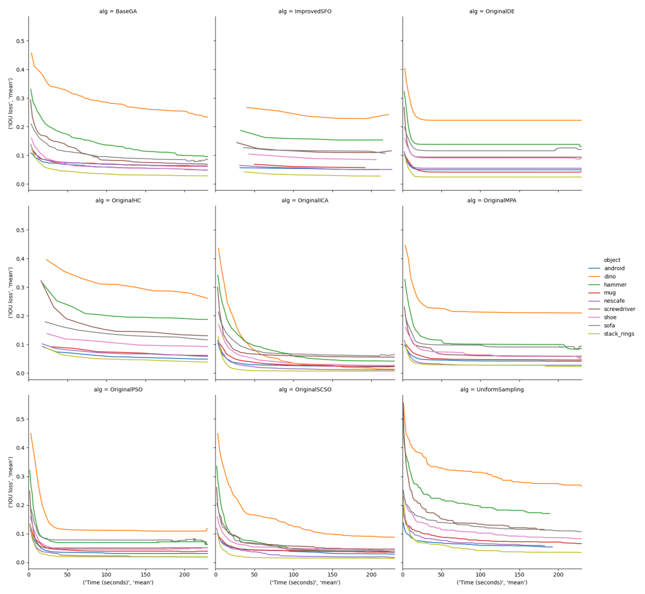

Figure 8: Chosen algorithms objective function(IOU) loss over time per object; averaged over all samples in $\mathcal{D}_3$

Looking closer at the algorithms' IOU loss over the time in Figure 8, we can see that the algorithms consistently performed poorly on certain objects like the *dino* or well on objects like the *stack rings*. This can be explained by different properties of the objects such as how many symmetric orientations it has (that leads to multiple minimums), or how the intersection of the object changes on different pairs of orientations (which changes the IOU distribution over them) which can make the task of minimizing the objective function by sampling more difficult. This shows that our set of objects is indeed diverse and beneficial to evaluating how the algorithms generalize.

Another insight from the graph is that the ability of some algorithms like OriginalHC, OriginalDE to minimize the objective loss is very dependent on the object while others, like *OriginalICA*, *OriginalPSO* and *OriginalSCSO* manage to minimize the objective loss quite consistently. This shows that some black-box algorithms like *OriginalICA* can generalize the task of minimizing the IOU objective function (which is used as a proxy of XorDiff) to different objects.

Conclusion

This work explored the novel formulation of 3D orientation estimation as a black-box optimization problem and presented a comprehensive empirical evaluation of over 200 different optimization strategies on this challenging task. The extensive analysis sheds light on the strengths and limitations of applying black-box techniques to this domain.

On the positive side, our results demonstrate that some black-box optimization algorithms like OriginalICA, OriginalSCSO, and OriginalPSO can effectively minimize the proposed symmetry-invariant XorDiff evaluation metric by iteratively rendering synthetic views and comparing them to the input image. Their strong performance across a diverse set of 3D object models suggests they can generalize well to previously unseen objects.

However, black-box optimization for this task is not without its drawbacks. One limitation is the use of the IOU objective function as an approximation to the XorDiff evaluation metric. While IOU demonstrated good correlation properties, directly minimizing XorDiff would potentially yield better orientation estimates. The computational overhead of rendering synthetic views for each function evaluation can also become prohibitive for resource-constrained systems.

Despite the limitations, this work established black-box optimization as a viable and promising new paradigm for tackling the 3D orientation estimation challenge. The strong results from some algorithms like ICA and PSO provide an exciting blueprint for further research in this area. Future work overcoming the drawbacks identified here could potentially enable highly robust and efficient orientation estimation solutions.

Future Work

There are several promising directions in which this research could be expanded. Let's explore some of the potential paths forward.

To begin with, the objective function used in this study - IOU, has a significant limitation. While it captures the area overlap between the objects, it completely disregards color information and perceptual differences beyond the object's outline. As a result, valuable visual cues that could aid orientation estimation are lost. It would be worthwhile to investigate alternative objective functions that incorporate color and perceptual features, perhaps by combining multiple objectives through a weighted sum. This approach could lead to improved performance.

Additionally, our evaluation method assumed a uniform distribution of orientations. However, in real-world scenarios, objects tend to rest in stable poses, resulting in a non-uniform orientation distribution. By leveraging the our API, one could evaluate the algorithms' performance on this more realistic orientation distribution, potentially improving their generalization to practical applications.

For a truly accurate measure of real-world performance, the performance of the algorithms should be assessed on a dataset of real-world images, where the input images are photographs of physical objects. While this would provide the most reliable assessment, it would require a significant effort in curating such non-synthetic dataset with corresponding annotations.

The algorithms investigated in this work belong to the meta-heuristic category, a subcategory of black-box optimization methods. It may be insightful to explore alternative black-box algorithm classes beyond meta-heuristics to assess their suitability for this problem.

Our work focused on the case of RGB input images, which is indeed a common scenario. However, depth sensors are often available, providing additional geometric information. Evaluating the performance of these black-box algorithms on inputs containing depth information could yield valuable insights and potentially improve their effectiveness.

Finally, while this study addressed object orientation estimation, spatial localization (x, y, z) is frequently an additional requirement in practical applications. Extending the problem formulation to the 6DoF case, where both the orientation and position of objects are unknown, would broaden the scope and applicability of this line of research.

References

[1] Deserno, M. 2004. How to generate equidistributed points on the surface of a sphere.

[2] Downs, L.; Francis, A.; Koenig, N.; Kinman, B.; Hickman, R.; Reymann, K.; McHugh, T. B.; and Vanhoucke, V. 2022. Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items.

[3] Guan, J.; Hao, Y.; Wu, Q.; Li, S.; and Fang, Y. 2024. A Survey of 6DoF Object Pose Estimation Methods for Different Application Scenarios. *Sensors*, 24(4).

[4] Kendall, M. G. 1938. A New Measure of Rank Correlation. *Biometrika*, 30(1/2): 81–93.

[5] Lepetit, V.; Moreno-Noguer, F.; and Fua, P. 2009. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81.

[6] Miles, R. E. 1965. On Random Rotations in R^3. *Biometrika*, 52(3/4): 636–639.

[7] Sahin, C.; Garcia-Hernando, G.; Sock, J.; and Kim, T.-K. 2020. A review on object pose recovery: From 3D bounding box detectors to full 6D pose estimators. *Image and Vision Computing*, 96: 103898.

[8] Spearman, C. 1987. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology*, 100(3/4): 441–471.

[9] Sundermeyer, M.; Marton, Z.-C.; Durner, M.; Brucker, M.; and Triebel, R. 2019. Implicit 3D Orientation Learning for 6D Object Detection from RGB Images. arXiv:1902.01275.

[10] Van Thieu, N.; and Mirjalili, S. 2023. MEALPY: An open-source library for latest meta-heuristic algorithms in Python. *Journal of Systems Architecture*.

[11] Xiang, Y.; Schmidt, T.; Narayanan, V.; and Fox, D. 2017. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *ArXiv*, abs/1711.00199.

[12] Zhu, Y.; Li, M.; Yao, W.; and Chen, C. 2022. A Review of 6D Object Pose Estimation. In *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 10, 1647–1655.

Appendices

XorDiff Expectation Upper Bound

Let $d, \hat{d}$ be random depth-maps, which were acquired by uniformly sampling random orientations of some object $\mathcal{O}$. Define:

$$k \;= \underset{d,\hat{d}}{\mathbb{E}}\left[\max\{|d(x)-\hat{d}(x)| : x \in c \cap \tilde{c}\}\right]$$

We claim, that using k as defined above, the following holds:

$$\underset{d,\hat{d}}{\mathbb{E}}\left[\mathit{XorDiff}_{p=1}(d,\hat{d})\right] \leq 1$$

Proof:

$$\underset{d,\hat{d}}{\mathbb{E}}\left[\mathit{XorDiff}_1(d,\hat{d})\right] = \underset{c,\tilde{c}}{\mathbb{E}}\left\|\frac{\alpha(C)}{k|c\cup\tilde{c}|}\right\|_1 =$$

$$\underset{c,\tilde{c}}{\mathbb{E}}\left[\sum_{x\in c\cap\tilde{c}}\frac{|d(x)-\hat{d}(x)|}{k|c\cup\tilde{c}|}\right] + \underset{c,\tilde{c}}{\mathbb{E}}\left[\sum_{x\in c\ominus\tilde{c}}\frac{k}{k|c\cup\tilde{c}|}\right] =$$

$$\frac{1}{k}\underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{1}{|c\cup\tilde{c}|}\sum_{x\in c\cap\tilde{c}}|d(x)-\hat{d}(x)|\right] + \underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{|c\ominus\tilde{c}|}{|c\cup\tilde{c}|}\right]$$

Let us focus on the first component:

$$\frac{1}{k}\underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{1}{|c\cup\tilde{c}|}\sum_{x\in c\cap\tilde{c}}|d(x)-\hat{d}(x)|\right] = \qquad \text{(def. of } k\text{)}$$

$$1/\underset{c,\tilde{c}}{\mathbb{E}}\left[\max_{x\in c\cap\tilde{c}}|d(x)-\hat{d}(x)|\right]\underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{1}{|c\cup\tilde{c}|}\sum_{x\in c\cap\tilde{c}}|d(x)-\hat{d}(x)|\right] \leq$$

(using Jensen's inequality, we get $1/\mathbb{E}[m] \leq \mathbb{E}[1/m]$)

$$\underset{c,\tilde{c}}{\mathbb{E}}\left[1/\max_{x\in c\cap\tilde{c}}|d(x)-\hat{d}(x)|\right]\underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{1}{|c\cup\tilde{c}|}\sum_{x\in c\cap\tilde{c}}|d(x)-\hat{d}(x)|\right] =$$

$$\underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{1}{|c\cup\tilde{c}|}\sum_{x\in c\cap\tilde{c}}\frac{|d(x)-\hat{d}(x)|}{\max_{y\in c\cap\tilde{c}}|d(y)-\hat{d}(y)|}\right] \leq \underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{|c\cap\tilde{c}|}{|c\cup\tilde{c}|}\right]$$

Using what we showed so far, we get the following:

$$\underset{d,\hat{d}}{\mathbb{E}}\left[\mathit{XorDiff}_1(d,\hat{d})\right] \leq \underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{|c\cap\tilde{c}|}{|c\cup\tilde{c}|}\right] + \underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{|c\ominus\tilde{c}|}{|c\cup\tilde{c}|}\right] =$$

$$\underset{c,\tilde{c}}{\mathbb{E}}\left[\frac{|c\cap\tilde{c}|+|c\ominus\tilde{c}|}{|c\cup\tilde{c}|}\right] = 1$$

Software Framework

To facilitate the empirical evaluation and comparative analysis of different optimization algorithms, we developed a modular and extensible software framework [1]. This framework served as an application programming interface (API) that streamlined our workflow and enabled the effortless assessment of various algorithms. The key features of our API are as follows:

- **Object Rendering:** The API incorporates a convenient wrapper around MuJoCo, a physics engine which used for rendering CAD models of different objects. This wrapper significantly simplifies the interaction with MuJoCo, providing an easy and straightforward approach to object manipulation within the simulation. Through this wrapper, we can effortlessly rotate the object of interest to a desired orientation r and obtain the rendered image $\tilde{\mathcal{I}}_r$ or the corresponding depth map.
- **Black-Box Algorithm Implementation:** Our API offers out-of-the-box support for the Mealpy [10] library, which implements numerous state-of-the-art black-box algorithms (in our context - orientation sampling strategies). Leveraging the vast array of algorithms implemented in Mealpy, we can evaluate a comprehensive set of different strategies π. Additionally, we provide built-in implementations for several relatively simple algorithms, one of which serves as the baseline strategy.
- **Objective Function Variety:** The API supports a wide range of objective functions l. Furthermore, the modular design of the code allows for seamless integration of new objective functions into our API.
- **Evaluation Functions:** We provide a built-in implementation for our evaluation function XorDiff. Moreover, it is straightforward to implement new evaluation metrics and effortlessly integrate them into our API.
- **Dataset Support:** There is a direct support for creating different types of datasets. Utilizing this capability, we were able to generate the datasets discussed in the 'Dataset' section, enabling us to evaluate the performance of numerous algorithms with ease.
- **Algorithm Evaluation:** Our API provides an easy-to-use evaluator class, which we employed to assess the performance of different algorithms.
- **Logging and Visualization:** The API includes logging support to record the execution of different algorithm runs, which can be used later for analysis. Additionally, the API allows for graphical visualization of an algorithm's run, providing a real-time view of the sampling strategy π during the optimization process.

[1]https://github.com/giladfrid009/Robotics-Proj

Uniformly Sampling Orientations

First, one needs to understand that it's not correct to simply sample a random orientation (in Euler form) from a uniform distribution over $[0, 2\pi] \times [0, \pi] \times [0, 2\pi]$ space. We leverage previous work that was done in the area of sampling points from the surface of spheres [1] [6], to perform both uniform random sampling and dividing the orientation space into equidistributed points.

To randomly sample an orientation r in a (pseudo) uniform manner, one may use the following algorithm:

Algorithm 5: Random Uniform Sampling of Orientation r

1. **Sample** point $\vec{v}$ from a uniform distribution over the surface of a unit sphere, as described in [1]
2. **Sample** rotation $\theta \sim U[0, \pi]$
3. **Return** $\theta\vec{v}$, which is a random orientation in the *Rotation Vector (axis angle)* representation.

Note, that according to [6], to achieve truly uniform orientation sampling, one must sample θ according to the CDF $(\theta - \sin\theta)/\pi,\ 0 \leq \theta \leq \pi$. The sampling of vector $\vec{v}$ as described in the above algorithm is in accordance to [6], but for simplicity's sake we sample θ from $U[0, \pi]$ instead. Additionally, using that CDF wouldn't allow us to (simply) divide the orientation space into equidistant parts, as described next.

To sample N pseudo-equidistant orientations, one may use the following algorithm:

Algorithm 6: Sample of N Pseudo-Equidistant Orientations

1. **Sample** $N^{(2/3)}$ equidistributed points V from the surface of a unit sphere, as described in [1]
2. **Sample** Divide $[0, \pi]$ into $N^{(1/3)} + 1$ equal-length intervals $I = \{i_1, i_2, ...\}$.
3. **Let** Θ be the set of all starting points of intervals in I, excluding the first interval i_1.
4. **Return** $\{\theta\vec{v} \mid \vec{v} \in V, \theta \in \Theta\}$, which is a set of N equidistant orientations from the orientation space, in the *Rotation Vector (axis angle)* representation.

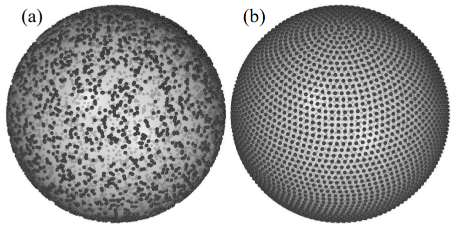

Figure 9: Image from [1]. Points in (a) are sampled uniformly-random, while in (b) are equidistributed. Each point on the sphere represents the axis in the axis-angle orientation representation.

Orientation Odyssey

In accordance with our centuries long tradition, each great work requires an accompanying great song. With words by Claude and ChatGPT, blood, sweat and tears are by us, we present our greatest masterpiece:

The robots were baffled, they scratched their metal domes
Not knowing which way all these objects were combed
They tried and they tried to determine the poses
But just ended up bending their circuit-board noses

That's when the black boxes stepped in feeling smug
Saying "Step aside buckets, and let us debug We'll sample some orientations with our own special sauce
Using techniques so complex, they'll make your ram toss!"

They rendered those objects from models they had
Comparing the images, good, better, or bad
"Hey IOU metric, what's the similarity score?"
"Not high enough yet, we must sample some more!"

And what did they find, in this lengthy discourse?
Some algorithms shone, with power and force.
ICA and ICSCO, the stars of the show,
Outshining the rest, their prowess aglow.

With XorDiff in hand, they sliced through the fog,
A beacon of clarity, no longer agog.
"Behold the results!" they declared with a cheer,
"Our functions are sharp, the answers are clear!"

The robots rejoiced, their domes now uncreased,
The riddle of poses had finally ceased.
With algorithms faithful, and metrics so true,
The paper concluded, a journey anew.

So hats off to the team, their efforts now penned,
In pages so many, they seem to have no end.
But within every line, a truth is unfurled,
A symphony of data, a 3D world.

Credits to Claude, ChatGPT and us of course :)