Target Encoding for Object Orientation Prediction in Smart Manufacturing

Antonio Gambale, Sonya Coleman, Dermot Kerr, Philip Vance
School of Computing, Engineering & Intelligent Systems,
Ulster University, Londonderry, Northern Ireland
e-mail: {a.gambale | sa.coleman | d.kerr | p.vance}@ulster.ac.uk

Emmett Kerr
Dept. of Electronic & Mechanical Engineering,
Atlantic Technological University, Letterkenny, Ireland
e-mail: emmett.kerr@atu.ie

Cornelia Fermüller, Yiannis Aloimonos
Institute for Advanced Computer Studies,
University of Maryland, USA
e-mail: {fermulcm | yiannis}@umd.edu

***Abstract*—Accurate planar object orientation prediction is critical for enabling robots to interact effectively with their environment, supporting advanced manipulation tasks beyond simple grasping. However, most existing approaches emphasize gripper pose rather than object-centric orientation, which limits both generalisability and downstream utility. Furthermore, model selection for orientation prediction is often heuristic, lacking systematic comparison or theoretical justification. This work introduces novel encoding schemes and integration strategies for robust planar orientation estimation under a unified 360° single-angle representation. Four new target encodings are proposed: base trigonometric, quadrant enhanced, polar augmented, and full composite; in conjunction with two complementary integration strategies, vector based and branched. Together, these methods address angular discontinuity challenges that hinder the performance of direct regression models.To mitigate the lack of manufacturing-oriented pose datasets, an annotation enrichment pipeline is presented, extending the MetaGraspNet dataset with geometrically derived orientation labels and rigorously annotated real-world data to assess domain transfer. Using features extracted from a pre-trained ResNet50, multiple regression architectures are benchmarked across all encoding and integration configurations. Performance is evaluated using the Mean Absolute Angular Error (MAAE) between predicted and ground truth angles. Experimental results show that the quadrant enhanced encoding combined with the vector based integration strategy achieves the highest accuracy. The XGBoost 1.7 model attains a MAAE of 8.15°, while maintaining robustness to angular discontinuities and strong computational efficiency across both synthetic and real-world testing. These findings provide practical guidance for implementing reliable and generalisable orientation predictors within industrial robotic manipulation systems.**

***Keywords*-Computer Vision; Robotics; Manipulation; Machine Learning; Smart Manufacturing.**

I. Introduction

This paper is an extended version of the work presented at the IARIA ADVCOMP 2025 conference [1]. While the core findings remain consistent, this journal version provides a comprehensive methodological expansion, including detailed annotation enrichment procedures, hyperparameter optimisation protocols, and an increased in-depth analysis of encoding and integration strategy interactions. By addressing these methodological aspects, this work thereby advances the practical deployment of orientation estimation in robotic manipulation systems.

The motivation for developing robust orientation estimation methods stems from fundamental limitations in contemporary robotic grasping systems. Current approaches frequently generate arbitrary grasp poses without considering the object's pose for subsequent manipulation requirements [2], [3]. Contemporary robotic grasping research typically focuses on two-finger parallel grippers [2], [3], opting to use complex multi-dimensional grasp representations [4], [5], often referred to as grasping rectangles [6]. These representations cannot easily be implemented on vacuum or three-finger manipulators as they are inherently designed for use with parallel style grippers. Whilst these methodologies demonstrate efficacy in achieving stable grasps, they frequently neglect the requirements of subsequent manipulation tasks, such as tool manipulation or object assembly, as they focus solely on the initial grasp pose of the parallel gripper relative to the object, rather than the object pose from which a grasp can be deduced. Furthermore, many of these works require complex and expensive annotation and validation of multiple positive and negative grasping rectangles for each object [7]. These multi-parameter grasp rectangles necessitate complex approaches to predict five or more parameters, which can be computationally demanding and often result in methods unsuitable for real-time execution in resource constrained environments or embedded systems [8].

Some works have explored various approaches to simplify grasp parameters whilst maintaining effectiveness. Notably, the work in [7] has demonstrated success in representing object orientation using a single angle around the normal vector of a planar surface which is predicted through use of the ImageNet Convolutional Neural Network (CNN) for feature extraction along with a Support Vector Machine (SVM) for angle prediction. However, this approach relies on RGB-D data and complex hierarchical regression techniques to handle angle discontinuity at the 0°/360° boundary. A two-step process is employed where an SVM classifies the angle into one of four 90° intervals, followed by training separate RBF kernel support vector regressors for each interval to predict the precise angle.

While effective, this method introduces additional complexity and computational overhead compared with direct or trigonometric encoded angle regression approaches. Moreover, the adoption of less conventional grasp representations, such as singular angles, over traditional grasping rectangles further constrains the already limited pool of comprehensive annotated datasets for robotic grasping applications [7], [8]. Consequently, for such an approach to be viable it must either involve a method to deduce single angles from existing datasets or advocate for the inclusion of more comprehensive angular information in future grasping datasets. This highlights the need for innovative data generation and annotation techniques to support advanced object orientation prediction methods.

This research builds upon these prior works with the aim of enhancing post-grasp task execution, focussing on the object pose rather than grasp pose, through the use of a methodology which represents object orientation as a single angle in a 360° coordinate system. This approach provides orientation information that not only facilitates the intended post-grasp use of the object but also enables derivation of the appropriate grasp angle. Additionally, this representation enables learning architectures to focus specifically on orientation prediction of one value (θ), circumventing the complexity of multiple parameter estimations and facilitating grasping with manipulators of all types.

To address the scarcity of annotated grasping datasets suitable for single angle object pose estimation [7], [8], this work draws direct inspiration from [9], who utilised traditional computer vision with object geometric data to generate object poses. Here, a modified version of the methodology in [9] is employed to augment the MetaGraspNet dataset: rather than relying on edge maps, ground-truth object segmentation masks are used to extract geometric features such as centroid, handle-tip displacement, and axis of symmetry, enabling the calculation of precise orientation annotations for each object instance. This automated annotation process was validated through manual inspection to ensure geometric fidelity and orientation accuracy, resulting in a high-quality synthetic dataset well-suited for model training. Recognising that synthetic datasets alone cannot capture the full complexity of real-world scenarios, a complementary real-world dataset was created and rigorously annotated. This secondary dataset, was curated to reflect a range of object configurations, occlusions, and lighting conditions encountered in practical smart manufacturing applications. The combination of these two datasets (a large, systematically annotated synthetic set and a smaller, carefully validated real-world set) enables robust evaluation of model generalisation and domain transfer, providing a more comprehensive assessment of each method's practical viability. This research proposes a suite of novel encoding schemes and integration strategies, together with a systematic evaluation of regression architectures, to advance orientation prediction performance using these enriched datasets. Cross-validation, computational benchmarking, and error analysis are then carried out to determine practical suitability, advancing the current state of planar orientation prediction for robotic manipulation by presenting a clear in depth comparison of these existing shallow learning models and clarifying the interplay between encoding, integration, and model architecture, while also providing actionable insights for deploying robust, efficient orientation predictors in real-world robotic pipelines.

II. Methodology

This section describes the methodological pipeline for planar object orientation prediction, including annotation enrichment, novel encoding strategies, integration approaches, feature extraction, model selection, and optimisation.

A. Object Orientation Prediction Pipeline

The novel pipeline presented in this work is summarised in Figure 1 and presents two solutions, hereafter referred to as integration strategies, for the task of object orientation prediction. In this orientation pipeline, an image of the object is first segmented, and learned features are extracted. These features are then input into a machine learning model as a feature vector. During training and evaluation, target variables are also provided to the models that are derived from ground truth values representing the known pose of each object instance, and represent the values the model will be predicting based on the provided learned features from the CNN.

The novelty of this work centres on the definition and structure of the target values, which specify what the model is trained to predict and constitute the ground truth used to guide predictions. For the remainder of this work these target variables are referred to as encoding schemes, and the manner in which they are presented to the shallow learning models is referred to as the integration strategy.

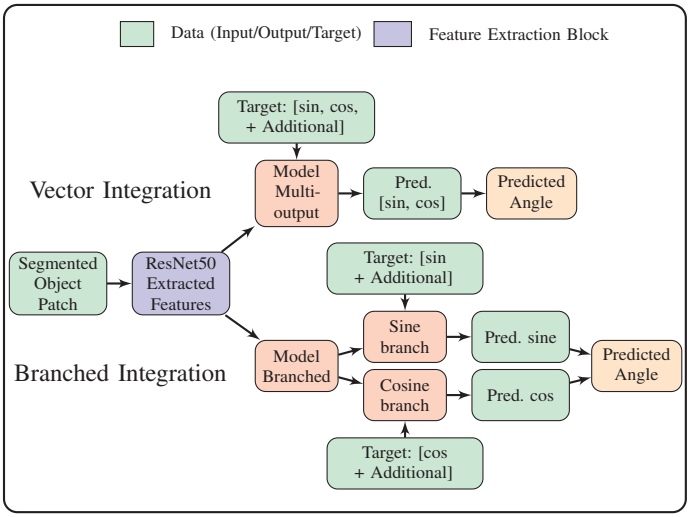


Figure 1. Object orientation prediction pipeline.

B. Target Encoding Schemes

In this study, we are encoding an object's pose angle in degrees which represents the planar orientation of the object, commonly referred to as the azimuth angle, defined over the range 0° to 360°. This angle, denoted by θ, is first encoded through trigonometric decomposition to yield sine and

cosine components. Additional encodings are also derived for robustness, resulting in four proposed encoding configurations:

- **Base**: The fundamental trigonometric representation:

$$\mathbf{y}_{\text{base}} = [\sin(\theta), \cos(\theta)] \tag{1}$$

  This encoding allows the model to output trigonometric components from which the orientation angle can be uniquely reconstructed, whilst naturally handling the periodic nature of angular data.
- **Quadrant**: The base encoding with quadrant confidence indicators:

$$\mathbf{y}_{\text{quad}} = [\sin(\theta), \cos(\theta), \mathbf{1}_{Q1}, \mathbf{1}_{Q2}, \mathbf{1}_{Q3}, \mathbf{1}_{Q4}] \tag{2}$$

  Here, $\mathbf{1}_{Qi}$ denotes the ground truth one-hot indicator for each quadrant, where the appropriate quadrant is assigned a value of 1 and all others are set to 0. The quadrants are defined as:
  - $Q1$: $0^\circ \leq \theta < 90^\circ$
  - $Q2$: $90^\circ \leq \theta < 180^\circ$
  - $Q3$: $180^\circ \leq \theta < 270^\circ$
  - $Q4$: $270^\circ \leq \theta < 360^\circ$

  Although the ground truth labels are strictly one-hot, the model is trained to predict a confidence value for each quadrant, where higher values indicate greater model certainty. In practice, these predicted confidences may not reach exactly 1 or 0; the final quadrant prediction is taken as the one with the highest confidence score. This approach allows for uncertainty around quadrant boundaries, providing smoother gradients and more robust learning near ambiguous angles.
- **Polar**: The base encoding is augmented with an explicit radian angle value:

$$\mathbf{y}_{\text{polar}} = [\sin(\theta), \cos(\theta), \theta_{\text{rad}}] \tag{3}$$

  The term θ_{rad} denotes the orientation angle in radians ($0 \leq \theta_{\text{rad}} < 2\pi$), providing a continuous scalar representation in addition to the trigonometric components, helping the model capture fine-grained angular differences.
- **Full**: A comprehensive encoding combining trigonometric, quadrant, and polar representations:

$$\mathbf{y}_{\text{full}} = [\sin(\theta), \cos(\theta), \mathbf{1}_{Q1}, \mathbf{1}_{Q2}, \mathbf{1}_{Q3}, \mathbf{1}_{Q4}, \theta_{\text{rad}}] \tag{4}$$

  This composite form gives the model access to all available encoded representations for each angle.

These encodings define the set of target variables that the shallow learning models are trained to predict and during inference the models only receive the feature vector derived from the input images and produce predictions for these encoded target values based on the learned features supplied by the chosen CNN. While encodings may include multiple elements (for example, both trigonometric and one-hot components in the quadrant encoding), the final predicted angle is always decoded and presented in degrees from 0° to 360°, representing the object's azimuth orientation.

C. Integration Strategies

In addition to the various target encoding schemes, two model integration strategies were proposed. These strategies determine the architectural approach taken by each model when predicting multiple target variables, structuring the regression process to either provide all outputs jointly with one model (vector-based integration) or via distinct independent models for each target variable (branched integration). The two integration strategies are depicted in Figure 1 and described below.

1) Branched Integration Strategy: In the branched integration strategy, multiple parallel models are trained independently to predict different components of the target encoding scheme. For example, one model may predict the sine component while another predicts the cosine component, with further models included as required for additional encoding targets such as quadrant confidences or polar values. Each model receives the necessary input features and auxiliary target variables relevant to its assigned output. Following prediction, the outputs, such as predicted sine and cosine values, are combined using the inverse tangent function ($\tan^{-1}$) to provide the final inferred angle. This approach allows for flexible extension to more than two models, depending on the chosen encoding configuration. The overall structure is illustrated in Figure 1.

For clarity, consider the **full** encoding:

$$\mathbf{y}_{\text{full}} = [\sin(\theta), \cos(\theta), \mathbf{1}_{Q1}, \mathbf{1}_{Q2}, \mathbf{1}_{Q3}, \mathbf{1}_{Q4}, \theta_{\text{rad}}] \tag{5}$$

In the branched integration configuration, separate models are trained to predict specific components of this target vector. For example, one model predicts the sine component along with quadrant confidences and polar angle, while a second model predicts the cosine component, also together with quadrant confidences and polar angle. Each model shares all auxiliary variables for the selected encoding, except for the trigonometric function assigned to the other model. The predicted components are then combined after inference to estimate the final orientation angle.

2) Vector Integration Strategy: The vector integration strategy employs a single model to directly predict all target variables in a unified manner. This enables the model to jointly capture dependencies and relationships among all components of the encoding. As such, all encoding targets, including both trigonometric components and any additional variables, are predicted together, sharing the same optimisation process, loss function, and model parameters [10]. Following prediction, the sine and cosine outputs are combined using the inverse tangent function ($\tan^{-1}$) to recover the angle estimate.

D. Annotation Enrichment

This section details the methodology employed for enriching annotations within the MetaGraspNet dataset framework by determining object orientations and thus providing the object orientation angles that are used as the ground truth for the target encodings discussed in Section II-B. The approach adapts principles from prior work on edge-based orientation detection [9] to segmentation-based annotation data, enabling precise angular determination of elongated objects such as screwdrivers.

This approach addresses the lack of applicable datasets in planar object orientation estimation for manufacturing-centric applications by automating orientation determination, making it viable to efficiently add detailed annotations to existing datasets such as MetaGraspNet [11].

The annotation enrichment pipeline, as depicted in Figure 2, comprises five sequential stages: (1) COCO-format polygon parsing to extract segmentation data from annotation; (2) centroid computation via spatial moments; (3) principal axis determination using Principal Component Analysis (PCA); (4) orientation direction disambiguation; and (5) final orientation and centroid annotation.

Figure 2. Illustrative example of the orientation annotation workflow, demonstrating the key stages from input image to refined angle estimation for a segmented object.

The annotation enrichment process begins with segmentation polygons from MetaGraspNet's COCO-format annotations [11], where each object instance is defined by boundary points $\mathbf{p}_i = (x_i, y_i)$. These coordinates are first used to calculate a minimum bounding rectangle, creating a localised coordinate system with $(x_{\min}, y_{\min})$ as the origin. All polygon points are shifted to this local system: $\mathbf{p}'_i = (x_i - x_{\min}, y_i - y_{\min})$. A binary mask of the object is then generated by rasterising the shifted polygon [12] onto a blank image sized to the bounding rectangle dimensions, where pixels within the object are set to 1 and pixels outside the object representing the background are set to 0. The geometric centroid is calculated from this mask using spatial moments [13], then transformed back to the original image coordinate system providing the reference point for orientation analysis.

Principal Component Analysis (PCA) [14], [15] is then applied to identify the primary axis of elongation through eigenvector decomposition of the boundary points. The boundary points are first centred by subtracting the computed centroid creating a zero-mean coordinate system. The 2×2 covariance matrix is then computed from these centred coordinates and eigenvalue decomposition then yields two eigenvectors: the primary eigenvector corresponding to the largest eigenvalue represents the direction of maximum variance (elongation), while the secondary eigenvector represents the perpendicular direction. The initial orientation angle is calculated as $\theta = \tan^{-1}\left(\frac{v_{1y}}{v_{1x}}\right)$, where (v_{1x}, v_{1y}) are the components of the primary eigenvector. However, this PCA-derived angle presents directional ambiguity for asymmetric objects such as screwdrivers or bolts, as the eigenvector can point in either direction along the elongation axis [9]. This ambiguity arises because PCA identifies the axis of maximum variance but cannot distinguish between the tool tip and handle ends of the elongated object. To resolve this ambiguity and determine the true tip orientation, a final refinement stage is applied based on the geometric assumption that the object centroid is towards the handle due to asymmetric mass distribution in tools like screwdrivers [9].

This refinement stage defines two search corridors 10 pixels wide along directions θ (initial angle from PCA) and $\theta + 180^\circ$. For each corridor, the algorithm casts a ray from the centroid and identifies boundary points that fall within the corridor width by calculating their perpendicular distance from the ray centreline. Within each valid corridor, the furthest boundary point is identified by computing the projected distance along the ray direction using the dot product. The maximum projected distances d_+ and d_- for the two opposing directions are compared: if $d_+ > d_-$, the original PCA angle θ is retained; otherwise, θ is adjusted by 180° to point toward the tool tip (the direction with the greater extent). This refinement approach aims to resolve the directional ambiguity by exploiting the asymmetric geometry of elongated tools, where the tip extends further from the centroid than the handle region [9].

The complete pipeline processes each annotation by parsing COCO-format segmentation polygons, extracting boundary points, computing the geometric centroid through spatial moments, determining the principal elongation axis via PCA eigenvalue decomposition, resolving tip direction through ray-casting comparison, and appending the final orientation angle θ and centroid coordinates (c_x, c_y) as new annotation fields to the JSON annotation file. This automated enrichment provides orientation ground truth data that enables training of direction-aware orientation models while maintaining compatibility with existing dataset frameworks and COCO annotation standards.

E. Feature Extraction

Feature extraction serves to transform raw image data into numerical representations suitable for input into shallow learning models as part of the object orientation pipeline outline in Figure 1. A ResNet50 convolutional neural network, pre-trained on ImageNet with the classification head removed and global average pooling applied, was employed to generate consistent feature vectors for all object instances [16], [17]. The feature extraction pipeline, depicted in Figure 1, operates as the bridge between raw image data and the regression models, providing the input vectors upon which the shallow learning algorithms operate to predict the target encoding variables described Section II-B.

The feature extraction process comprises four sequential stages: (1) object patch extraction using segmentation masks, (2) background normalisation and padding, (3) standardised resizing, and (4) deep feature computation via the pre-trained CNN as described below.

Object Patch Extraction: Each object instance is isolated from its source image using the corresponding segmentation polygon from the dataset annotations, after which a bounding rectangle is computed with an additional 10% padding factor applied to ensure complete object capture. The masked object is then extracted and placed onto a uniform white background, removing potential confounding factors from the surrounding scene whilst preserving the object's visual characteristics.

Aspect-Preserving Normalisation: To maintain geometric relationships whilst ensuring consistent input dimensions, the extracted object patches undergo aspect ratio preserving resizing. Each patch is scaled such that its largest dimension fits within 224×224 pixels, with the scaled image centred and the remaining space filled with white pixels. This approach prevents distortion whilst providing the standardised input size required by the ResNet50 architecture.

Deep Feature Generation: The normalised patches are processed through the ResNet50 network pre-trained on ImageNet. The network's convolutional layers extract hierarchical visual features, which are then aggregated via global average pooling to produce a fixed-length 2048-dimensional feature vector for each object instance [18]. These resulting feature vectors constitute the input data for the shallow learning models, providing a consistent numerical representation derived from the visual content of segmented object instances. These features, combined with the target encoding variables outlined in Section II-B, enable the regression models to learn mappings from visual appearance to planar orientation estimates. For testing on real-world data, the same pipeline is applied using the custom Real-world Screwdrivers dataset, with segmentation data provided by the Segment Anything Model 2.0 (SAM) [19].

F. Model Selection and Evaluation Framework

The enriched angular annotation data derived in Section II-D enables the training and evaluation of machine learning models for planar object orientation prediction on a refined subset of MetaGraspNet. Owing to the moderate scale of the enhanced dataset, this work intentionally focuses on shallow learning models [20], which offer notable advantages in computational efficiency and interpretability over deep learning approaches. Accordingly, a diverse suite of shallow regression architectures was selected: Random Forest (RF), Support Vector Regression (SVR), Multiple-Output Support Vector Regression (M-SVR), and both the standard (single-output) and multi-output variants of XGBoost. These models were chosen for their established efficacy with moderate data volumes, rapid inference speeds, and transparency in decision-making [7], [10]. Each operates on the input features produced by the CNN feature extraction pipeline (Section II-E) and learns to predict the target variables specified by the encoding schemes and integration strategies discussed in Section II-B.

The investigative framework thus systematically evaluates:

- **Four novel target encoding schemes:** From basic trigonometric representations to rich composite encodings including quadrant and polar variables.
- **Two novel integration strategies:** Branched (predicting sine and cosine individually) and vector-based (joint multi-output regression).
- **Three regression architectures:** Decision tree-based (RF, XGBoost), kernel-based (SVR), and their multi-output extensions.

This approach supports a comprehensive exploration of how encoding design, integration strategy, and model architecture jointly influence orientation prediction accuracy and efficiency, ultimately providing actionable guidance for robust, interpretable robotic perception systems. The shallow learning models considered in this work are summarised in Table I.

TABLE I. Multi-Output Implementation of Shallow Learning Models

Model	Multi-Output Implementation
Random Forest	Native (single model, vector outputs)
SVR	Wrapper (multiple independent regressors)
M-SVR	Native (unified loss function)
XGBoost 1.7	Wrapper (multiple independent regressors)
XGBoost 2.0	Native (vector leaf nodes)

Multi-Output Prediction Implementation

The architecture of each regression model fundamentally determines how multiple target variables are handled, with critical implications for both integration strategies:

- **Native Multi-Output Models (RF, M-SVR, XGBoost 2.0):** These models inherently support simultaneous prediction of multiple targets within a single unified framework. Random Forest achieves this through ensemble averaging, where each tree can produce multi-dimensional outputs at leaf nodes. For example, when predicting the **quadrant** encoding scheme $[\sin(\theta), \cos(\theta), Q_1, Q_2, Q_3, Q_4]$, a single Random Forest model produces all six values simultaneously in one forward pass. M-SVR extends the traditional SVR with a multi-dimensional ε-insensitive loss function that jointly optimises all targets, whilst XGBoost 2.0 implements vector-valued leaf nodes where each leaf stores a complete output vector spanning all prediction targets. This unified approach enables the model to learn cross-target dependencies and implicitly enforce mathematical relationships such as $\sin^2\theta + \cos^2\theta = 1$ [10].
- **Wrapper-Based Multi-Output (SVR, XGBoost 1.7):** Models lacking native multi-output support utilise wrapper frameworks that create independent regressors for each target variable [10]. For instance, when predicting the quadrant encoding scheme, the wrapper creates six separate SVR models: one each for $\sin(\theta)$, $\cos(\theta)$, Q_1, Q_2, Q_3, and Q_4 quadrant indicators. Crucially, these regressors share identical hyperparameters (e.g., regularisation strength, kernel parameters) and receive the same input features, but each is trained independently with its respective encoded target variable output. This approach differs from a fully branched integration strategy in that hyperparameters are globally optimised across all outputs using composite scoring functions, this would mean optimisation across all target encodings for vector approaches, or separate optimisation for sine and cosine branches. However, the independent training prevents explicit enforcement of cross-target relationships, requiring any trigonometric constraints to be learned implicitly through the shared feature space [10], [21], thus the relationship $\sin^2\theta + \cos^2\theta = 1$ is not inherently enforced.
- **Practical Example - Quadrant Prediction:** To illustrate these differences: when predicting quadrant membership, native multi-output models produce confidence scores for all four quadrants (Q_1 to Q_4) simultaneously alongside sine

and cosine values, enabling the model to consider quadrant-trigonometric relationships during training. Wrapper-based approaches train separate regressors for each quadrant confidence score, where the final quadrant prediction is determined by selecting the regressor output with the highest confidence value. Both approaches ultimately yield the same prediction format, but the learning dynamics and cross target consistency differ substantially.

Through this systematic evaluation of model architectures and their multi-output implementations, this study provides granular insights into how structural choices impact both prediction performance and the effective exploitation of object orientation encodings within moderate-scale datasets.

G. Optimisation

Hyperparameter optimisation was carried out using grid search across the models. The parameter spaces for each model are detailed in Table II. This approach balances thorough exploration of the hyperparameter space with computational feasibility [22], [23].

TABLE II. Hyperparameter Search Spaces

Model	Hyperparameters (Values)
SVR	C: 0.01, 0.1, 1, 10, 100, 1000, 10000 Kernel: rbf, linear, sigmoid Gamma: auto, 0.001, 0.01, 0.1, 1, 10, 100, 1000 Epsilon: 10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.05, 0.1 Tolerance: 10^{-5}, 10^{-4}, 5×10^{-4}, 10^{-3}, 10^{-2}
Random Forest	Number of estimators: 100, 200, 300, 400, 500 Maximum depth: 3, 5, 10, 15, 20, 25, 30 Minimum samples split: 2, 5, 8, 11 Minimum samples leaf: 1, 2, 3, 4, 5 Maximum features: sqrt, log2 Bootstrap: True, False
M-SVR	C: 0.01, 0.1, 1, 10, 100, 1000, 10000 Kernel: rbf, linear, sigmoid Gamma: auto, 0.001, 0.01, 0.1, 1, 10, 100, 1000 Epsilon: 10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.05, 0.1 Tolerance: 10^{-5}, 10^{-4}, 5×10^{-4}, 10^{-3}, 10^{-2}
XGBoost (1.7/2.0)	Number of estimators: 100, 200, 300, 400, 500 Maximum depth: 3, 5, 10, 15, 20, 25, 30 Learning rate: 0.005, 0.01, 0.05, 0.1, 0.2 Subsample: 0.5, 0.7, 0.8, 1.0 Column sample by tree: 0.5, 0.7, 0.8, 1.0 Minimum child weight: 1, 3, 5, 7 Gamma: 0, 1, 3, 5 Regularization alpha: 0, 0.1, 0.5, 1 Regularization lambda: 0.5, 1, 1.5, 2 Tree method: hist

The optimisation process employed 5-fold cross validation with a custom angular distance scorer designed for circular data. This scorer calculates the minimum circular difference between predicted and true angles, handling the periodic nature of angular data where 359° and 0° are only 1° apart rather than 359° apart. While all models use this angular distance scorer for optimisation, they employ different loss functions: Random Forest uses Mean Squared Error (MSE), SVR utilises an epsilon-insensitive loss function, M-SVR incorporates a multi-target epsilon-insensitive loss function, and both XGBoost versions employ MSE, however only 2.0 allows for multiple targets. To assess the predictive performance of the tested models, integration strategies, and encoding approaches, the Mean Absolute Angular Error (MAAE) was employed as the primary metric. This measure was computed independently for each encoding method and dataset, ensuring a granular evaluation of angular accuracy. A comprehensive computational benchmarking analysis was also conducted to evaluate inference speed. For each model, inference was performed on identical image features, with each test comprising 1000 repetitions per run and five aggregate runs to account for variability. All experiments were executed on a Linux 6.8.0 system equipped with Python 3.8.3, an Intel Xeon w7-3445 CPU (20 cores, 40 threads), and 62.3 GB of RAM. This rigorous protocol facilitates a thorough and equitable comparison of both predictive accuracy and computational efficiency across all models and integration strategies.

III. Datasets

To address the scarcity of orientation annotations in existing datasets [24], this work employs a synthetic-to-real transfer approach using two distinct datasets. The first is a modified version of an existing synthetic dataset, used for training and validation. The second is a custom real-world dataset, created by the authors to evaluate model performance under conditions that closely resemble practical applications.

Enhanced MetaGraspNet: The initial training of the proposed methods utilised a carefully curated subset of the MetaGraspNet dataset [25] reduced to focus exclusively on Phillips and flat-head screwdrivers. The subset selected for this work corresponds to the single-class, multiple-instance configuration within the MetaGraspNet framework. The choice of screwdrivers as the focal object class is based on the fact that these objects possess geometric properties that satisfy the requirements outlined in [9] and the annotation enrichment methodology detailed in Section II-D.

The resulting curated dataset provides 7,932 annotations across 2,691 images of size 1200 x 1200 pixels [25]. To validate the calculated object orientations created through the dataset enrichment process, 10% of the datasets ground truth angle data were manually inspected with an allowable deviation threshold of 10°, following [9]. This process involved extracting objects using their segmentation data, rotating them by their calculated ground truth angle to achieve horizontal alignment, and finally a manual verification of object orientation. To aid this process, reference lines indicating the calculated angle and image horizontal were also superimposed, as illustrated in Figure 3.

This validation revealed that objects with areas below 10,000 pixels exhibited significant orientation errors (up to 180° in 184 cases). These anomalies occurred primarily in Difficulty level 5 images across all camera angles, consistent with the increased object density and occlusion at this difficulty level.

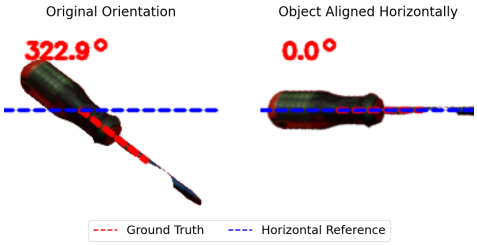


Figure 3. Illustrative example of angular validation.

Figure 4 shows an example of an object removed due to occlusion induced area reduction. To ensure that the calculated ground truth angular data used to train the models were robust, all objects with areas less than 10,000 pixels were therefore removed, resulting in the removal of 2,224 outliers and reducing dataset from 7,932 annotations to 5,709. Angular accuracy was then assessed again on a 10% sample (572 annotations).

Figure 4. Example of occluded object (highlighted in green) removed due to having an area less than 10,000 pixels.

Evaluation of the reduced dataset (5,709 annotations) revealed 55 objects with incorrect detected angles, with angular errors ranging from 1° to 9°. Producing a MAAE across the 10% sample (572 annotations) of 0.313°. Table III presents the full results of the objects with errors, where Cθ refers to the camera angle, Oθ refers to the calculated ground truth angle of the objects, and Δ represents the disparity between the detected angle and the manually validated angle.

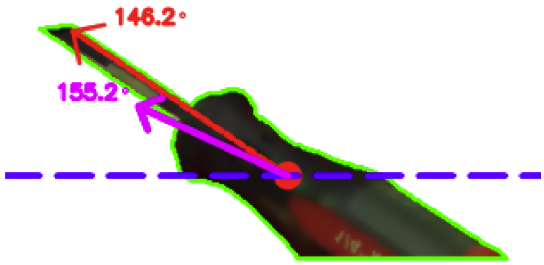
Analysis of Table III reveals several noteworthy trends:

- Error distribution across camera angles is relatively even, with no strong bias towards any particular angle. However, camera angle 0 (top-down view) shows the least error, likely due to minimal occlusion from the tray.
- Object angles cover the full spectrum from 0° to 359° with no clear clustering or preference for specific orientations.
- Most angular errors are minor, typically ranging from 1-3° (Δ). The maximum angular error of 9° occurred in scene 14771, where manual validation revealed significant occlusion at both the screwdriver's tip and handle, as illustrated in Figure 5.
- Several scenes (e.g., 11131, 11519, 13228) appear multiple times with different camera angles. These cases consistently show errors related to occlusion, either from other objects or the tray sides. While these scenes maintain sufficient area (>10,000 pixels), the mass distribution is uneven and therefore affects angle calculations.

TABLE III. Scene Camera Angle Analysis

Scene	Cθ	Oθ	Δ	Scene	Cθ	Oθ	Δ
11091	6	46.57	1	12546	1	81.38	5
11124	0	234.49	1	12720	8	22.37	2
11131	1	128.99	1	12760	4	115.57	6
11131	5	210.32	1	12879	6	189.10	2
11131	8	166.57	1	12976	6	58.21	6
11165	6	348.91	2	12979	3	148.46	1
11189	7	218.57	1	13040	8	287.78	2
11353	5	150.20	6	13169	5	25.17	5
11377	7	73.41	3	13228	2	275.33	1
11377	8	29.90	2	13228	7	34.71	5
11501	4	135.76	8	13228	8	4.81	3
11519	5	359.15	5	13292	2	134.87	2
11519	6	235.90	1	13417	1	63.83	5
11519	8	150.27	5	13499	5	134.26	1
11538	2	345.07	2	13711	6	152.91	6
11538	6	3.84	4	13787	3	60.84	3
11598	1	50.02	2	13794	3	77.88	4
11623	4	109.74	5	13811	2	53.26	3
11806	3	265.60	4	13943	8	355.62	4
11933	4	341.34	2	14120	4	26.41	5
11933	6	183.35	2	14588	7	251.76	2
11933	7	147.01	4	14606	2	40.96	6
11962	1	69.80	2	14634	3	68.50	3
12129	3	198.57	3	14656	3	287.13	3
12323	3	188.74	4	14670	3	115.38	2
12441	7	136.19	4	14771	7	155.19	9
12458	4	21.96	1	14811	4	141.74	2
12488	2	48.29	3	14811	6	346.61	1

(a) Full scene visualisation with the screwdriver producing the maximum angular error highlighted in green.



(b) Isolated segmentation mask with overlays to visualise predicted (Pink) and ground truth (red) angles.

Figure 5. Occluded Screwdriver from scene 14771, camera angle 7. (a) Full scene visualisation with screwdriver mask and angle overlay. (b) Isolated object mask with overlays for predicted (pink) and ground truth (red) angles and corresponding angle values.

It is important to note that both the maximum angular error and the MAAE are less than the 10° angle variation threshold delineated in [9]. As such, these data would not significantly compromise robotic grasping performance [26]. Therefore, the final dataset comprises 4,567 training samples and 1,142 testing samples with associated images and annotations representing an 80/20 split.

Since the MetaGraspNet dataset was not originally designed for this specific application, a final assessment of the object distribution across the 360° angular range was conducted. To accomplish this, the generated ground truth annotations were binned into 10° intervals, revealing an average of 127 images per bin for the training set and 32 objects per bin for the test set. The analysis demonstrated a consistent distribution of angles between the training and test sets, as illustrated in Figure 6. Additionally, the training set exhibited a slight imbalance between 300° and 309°, containing only 62 images compared with the average. This imbalance was less pronounced in the test set but remained present. Since the distribution of objects is not controlled in the original dataset, this imbalance will not be addressed; however, careful attention will be given to assessing model performance in these under-represented angular ranges to evaluate their impact on overall system performance.

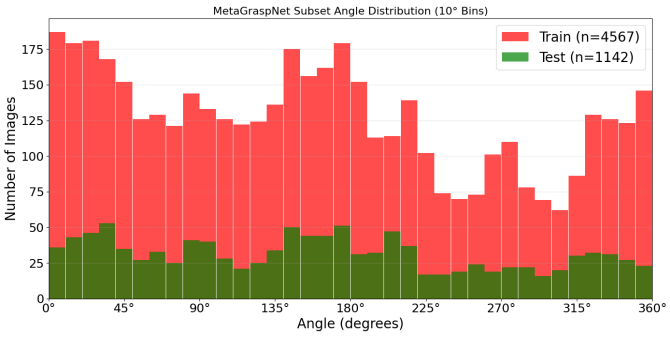


Figure 6. Distribution of ground truth angles across 10° bins for the MetaGraspNet subset.

Real-world Dataset:

To facilitate a simulation-to-reality transfer methodology, a second dataset was created for this work. This real-world dataset was constructed to mirror the structure of the curated MetaGraspNet subset used for training, closely adhering to the difficulty levels defined in [25], but utilises real images of screwdrivers rather than synthetic data.

In line with the MetaGraspNet difficulty levels, which are designed to progressively increase in complexity, levels 1 and 2 represent single objects with no occlusion, while higher levels introduce multiple objects and increasing degrees of occlusion and clutter. Consistent with the original subset selection from MetaGraspNet, which also excluded images at this level, the dataset did not include images from difficulty level 3. The annotation and data distribution by difficulty of this dataset can be seen in Table IV.

This real-world screwdriver dataset was acquired using a camera equipped with a Samsung ISOCELL GN9 sensor,

TABLE IV. Summary of images and annotations by difficulty level

Level	No. of Images	No. of Annotations
1-2	8	8
4	5	17
5	14	56

Figure 7. Example images from the secondary dataset.

capturing images of size 3072 × 4080 pixels, with a lens aperture of f/1.9, an exposure time of 1/100 second, and an ISO sensitivity of 386, with images being collected from both overhead and 45° angled perspectives. The dataset features three screwdriver variants, red/black Torx, blue/black flathead/starhead, and solid black starhead, randomly arranged on a white backdrop under uncontrolled ambient lighting conditions. To ensure natural illumination variability, 27 images were captured across daylight hours, resulting in 81 annotations; examples of such images can be seen in Figure 7.

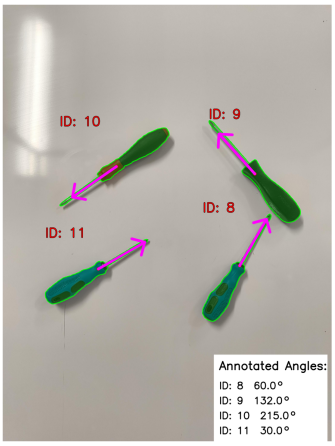


Figure 8. Sample image from the real-world dataset visualising four annotated screwdrivers.

Ground truth segmentation masks and orientation angles were annotated by three domain experts [27]. Inter-annotator agreement on segmentation masks was assessed using mean

IoU, yielding a high consistency of 0.95. Discrepancies were resolved via consensus review to refine the segmentation boundaries. For orientation, annotators measured shaft angles against the right horizontal axis, with a mean absolute difference of $\pm 1.8^{\circ}$ between predictions. Disagreements exceeding 5° were resolved by taking the circular mean for the affected objects; otherwise, the final orientation annotations were selected from a random sample of the independent measurements, except where consensus averaging was required as described. Examples of these annotations can be seen superimposed on one of the test images from the real world dataset in Figure 8 in which the segmentation masks are outlined in green, object IDs are labelled in red, and annotated orientation angles are indicated by pink arrows. The corresponding orientation values (in degrees) are listed in the lower right corner.

TABLE V. Default parameters used for the Segment Anything Model (SAM) during mask generation.

Parameter	Value
Model Type	`vit_h`
Checkpoint	`sam_vit_h_4b8939.pth`
Device	`cuda`
Points Per Side	32
Pred IoU Threshold	0.88
Stability Score Threshold	0.95
Crop N Layers	0
Crop Overlap Ratio	0.3413

To ensure relevance to downstream deployment, segmentation masks for inference on the custom real-world dataset were generated using SAM [19], rather than relying on the expert annotated masks. SAM was configured using its default parameters, which are summarised in Table V. The accuracy of these automatically generated masks was validated against the manual annotations, achieving a mean Intersection-over-Union (IoU) of 0.95. This strong agreement supported the use of SAM generated masks for subsequent stages in the pipeline.

IV. Results

The predictive performance of the encoding approaches, integration strategies and shallow learning models was evaluated using the Mean Absolute Angular Error (MAAE), as described in Section II-G. Results were computed independently for each model, encoding method, integration strategy and dataset, enabling a detailed assessment of angular accuracy.

Inference speed was benchmarked using the computational protocol detailed previously, ensuring comparability across models. The following results begin with MAAE measurements using the MetaGraspNet subset, comparing integration strategies (vector vs. branched encoding) and encoding configurations (full vs. reduced target sets). All values represent degrees of angular error, with lower values indicating better performance. The initial results are summarised in Table VI.

Following the MetaGraspNet subset results, Table VII summarises model performance using the real-world dataset and the same MAAE metric to facilitate direct comparison.

TABLE VI. MAAE results on MetaGraspNet dataset (degrees)

Model	Base	Quadrant	Polar	Full
XGBoost 1.7 (Vector)	5.15	5.15	5.15	5.15
XGBoost 1.7 (Branched)	5.15	5.15	5.15	5.15
XGBoost 2 (Vector)	**4.92**	**5.01**	5.46	5.17
M-SVR (Vector)	8.04	8.04	8.04	8.04
SVR (Vector)	5.01	5.12	**5.01**	**5.01**
SVR (Branched)	5.01	5.12	**5.01**	**5.01**
Random Forest(Vector)	5.48	**5.08**	5.87	5.67
Random Forest(Branched)	7.43	5.44	5.93	5.88

TABLE VII. MAAE results using real-world dataset (degrees)

Model	Base	Quadrant	Polar	Full
XGBoost 1.7 (Vector)	**9.61**	**8.15**	**8.96**	**9.61**
XGBoost 1.7 (Branched)	**9.61**	11.66	**9.15**	11.14
XGBoost 2 (Vector)	17.09	13.91	**7.18**	10.46
M-SVR (Vector)	28.86	28.86	28.82	28.82
SVR (Vector)	23.13	23.54	23.14	28.03
SVR (Branched)	23.15	23.54	23.14	23.28
Random Forest(Vector)	14.81	14.49	11.84	17.43
Random Forest(Branched)	16.45	**11.62**	12.90	17.50

Following the analysis of angular accuracy, Table VIII presents inference times per image in milliseconds for each model and condition. These results allow for direct comparison of computational efficiency across integration strategies and model architectures. As with previous metrics, lower values indicate superior performance, in this case, faster processing.

TABLE VIII. Inference time per image patch (milliseconds) by model and condition

Model Type	Base	Quadrant	Polar	Full
XGBoost 1.7 (Vector)	0.76	1.86	0.91	1.73
XGBoost 1.7 (Branched)	0.50	0.83	1.20	3.61
XGBoost 2 (Vector)	0.76	1.86	0.91	0.29
M-SVR (Vector)	17.76	17.78	17.80	17.76
SVR (Vector)	13.48	41.94	20.78	51.25
SVR (Branched)	13.55	70.23	27.68	75.49
Random Forest(Vector)	59.27	56.50	57.90	45.42
Random Forest(Branched)	119.15	117.08	117.83	91.04

The bolded values in Tables VI and VII highlight the configurations yielding the lowest Mean Absolute Angular Error (MAAE) for each dataset and encoding condition, representing the top-performing approaches in terms of angular prediction accuracy. Across both synthetic and real-world evaluations, the results show a clear benefit for methods employing vector-based integration, with XGBoost 2 and SVR providing the best performances on the MetaGraspNet test set, and XGBoost 1.7 (vector-based, quadrant encoding) and XGBoost 2 (vector-based, polar encoding) achieving the most accurate results on real-world data. The initial takeaway is that encoding scheme and integration strategy significantly influence predictive accuracy, with performance differences becoming more pronounced when models are tested on real-world as opposed to synthetic data.

V. Discussion | Evaluation

While the MAAE metrics presented in Section IV form the core of this discussion, it is essential to first address the primary challenge that this work seeks to overcome: the issue of boundary discontinuity. To thoroughly assess each method's performance in this regard a more granular analysis of the inference data on the custom dataset is therefore required, as visualised in Figure 9. Figure 9 displays the prediction error as a function of the ground truth angle for each model and encoding configuration. Specifically, the x-axis represents the ground truth angle of the object, while the y-axis shows the signed angular error, calculated as the shortest difference between the predicted and actual angles, wrapped to the interval $[-180°, 180°]$. Each point corresponds to a single prediction, positioned horizontally by its ground truth angle and vertically by the deviation from the true value.

A detailed inspection of these plots reveals substantial differences in error distributions between models, which are not always reflected in the aggregate MAAE values. The M-SVR model exhibits pronounced and frequent error spikes at the 0° and 359° boundaries, indicating a persistent struggle with boundary discontinuity and a lack of robustness in these critical regions. Beyond isolated boundary failures, M-SVR demonstrates erratic behaviour across much of the angular range, with large, abrupt deviations that suggest poor generalisation and reliability. In contrast, XGBoost 1.7 (both vector and branched variants) stands out for its consistent and stable error profile. Across nearly the entire angular range, prediction errors remain tightly clustered around zero, with only occasional moderate spikes, mostly at angular boundaries. This stability is indicative of a model that not only achieves a low mean error, as seen in Table VII, but also avoids catastrophic failures, making it more suitable for real-world deployment where reliability is paramount.

XGBoost 2, while achieving competitive MAAE values in some configurations, displays a more volatile error pattern. Notably, it exhibits significant errors not only at the 0° and 359° boundaries but also around 180°, suggesting that its generalisation may be compromised at multiple critical angles. This behaviour underscores the importance of evaluating models beyond mean metrics, as a low MAAE can mask underlying instability. The SVR models, both vector and branched, provide mixed results. While their errors are generally moderate, both models are prone to sporadic, large prediction failures at various angles, particularly near the boundaries and occasionally in the mid-range. These large spikes indicate that, although SVR may perform adequately on average, it is susceptible to unpredictable outliers that could undermine its practical utility. Random Forest models show moderate stability, with the vector variant generally outperforming the branched version in terms of error consistency. While occasional spikes are present, these are less frequent and less severe than those observed in M-SVR or SVR, positioning Random Forest as a reasonable compromise between stability and accuracy reflected by its low MAAE when paired with quadrant encoding and a branched integration strategy as seen in Table VII.

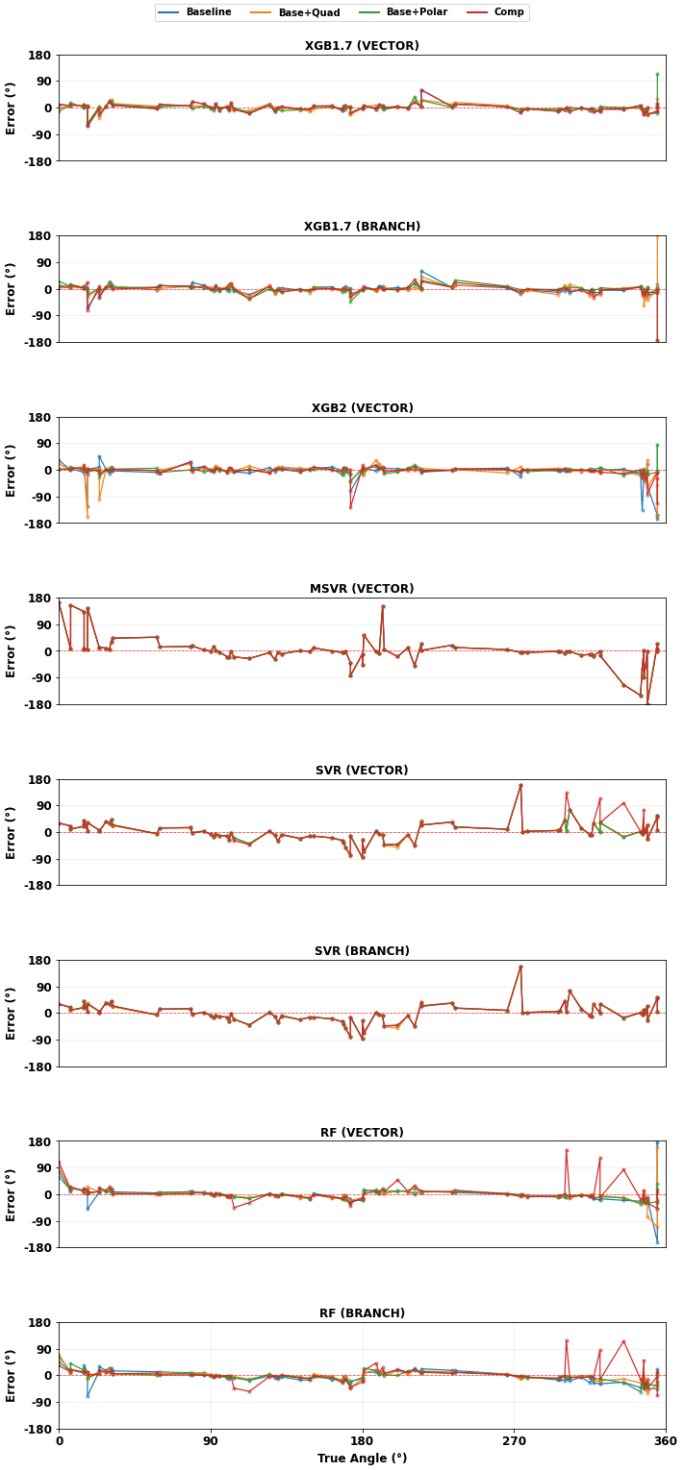


Figure 9. Model prediction errors against ground truth angle on the real-world dataset.

It is important to note that all models were trained exclusively on a synthetic dataset which is inherently simpler and more controlled than the real-world test set. As a result, the MAAE values achieved using the synthetic MetaGraspNet subset (Table VI) are substantially lower across all models, reflecting

the training-domain familiarity. For example, XGBoost 1.7 achieves an MAAE of 5.15° using the synthetic data, compared with 8.15–9.61° using the real-world dataset, while M-SVR and SVR also show marked increases in error when transitioning to real-world evaluation. This domain gap highlights the challenge of generalising from synthetic to real data, and underscores the value of robust error analysis using the real-world test set.

In addition to predictive accuracy, computational efficiency was also evaluated. Table VIII quantifies inference times per image patch, demonstrating XGBoost 1.7's superior performance with configurations predominantly achieving sub-2ms inference times (one outlier excepted). In contrast, the M-SVR/SVR models exhibit 10-100× slower performance (13.48-75.49ms), while Random Forest demonstrates the poorest efficiency, consistently exceeding 45ms and frequently surpassing 100ms. This efficiency advantage positions XGBoost 1.7 as optimal for applications requiring both rapid inference and angular reliability.

To understand the architectural origins of these performance differences, Figure 10 isolates each model's baseline capability using exclusively the fundamental (sin, cos) encoding, before any encoding augmentation is applied. This comparison reveals why certain models benefit substantially from advanced encodings while others show minimal improvement.

XGBoost 1.7's baseline performance (MAAE: 9.6° in both integration strategies) is remarkably strong, exhibiting stable error profiles with no large spikes at the 0°/360° boundary, demonstrating that this architecture inherently manages circular discontinuities effectively even with basic trigonometric features alone. This explains why XGBoost 1.7 maintains consistent performance across encodings in Figure 9: the architecture is already operating near its performance ceiling with baseline encoding. Conversely, M-SVR's baseline (MAAE: 28.9°) exhibits extreme error spikes exceeding ±150° concentrated at boundaries and scattered throughout the range, revealing fundamental architectural incapacity for circular prediction. Even with augmented encodings in Table VII, M-SVR cannot overcome these baseline limitations, remaining above 24° MAAE across all configurations.

Similarly, SVR's baseline (MAAE: 23.1°) shows pronounced spikes including severe failures around 180°, explaining its persistent high error (19-23°) regardless of encoding sophistication. XGBoost 2 (baseline MAAE: 17.1°) and Random Forest (baseline MAAE: 14.8-16.5°) occupy intermediate positions, with XGBoost 2's weaker baseline explaining its greater encoding sensitivity; requiring polar coordinates to achieve its best performance (7.18°) as seen in Table VII. This baseline comparison establishes that models unable to manage circular boundaries with simple (sin, cos) encoding cannot be rescued through encoding sophistication alone, directly informing model selection for angular prediction tasks.

A key focus of this work is the interplay between encoding strategies, model architectures, and integration approaches. The results reveal that model responsiveness to encoding varies considerably:

- **XGBoost 1.7** demonstrates strong robustness across all encodings, but achieves its best real-world performance with quadrant encoding in the vector integration configuration, yielding the lowest MAAE (8.15°). Notably, the addition of quadrant or polar information generally improves performance over the base (sin, cos) encoding, suggesting that XGBoost 1.7 can leverage richer target representations to better manage boundary effects and reduce systematic errors. The full encoding does not consistently outperform quadrant or polar, indicating that it may introduce redundancy or noise for this architecture.
- **XGBoost 2.0** is more sensitive to the encoding choice, achieving its best real-world MAAE (7.18°) with the polar encoding. However, its error distribution is less stable, with notable spikes at both boundary and mid-range angles, suggesting that while certain encodings can lower mean error, they may not guarantee robust predictions.
- **SVR and M-SVR** models show limited benefit from more complex encodings. Both models exhibit high MAAE and frequent large errors regardless of encoding, indicating that their architectures are less capable of exploiting additional target information to improve generalisation, especially in the presence of boundary discontinuities.
- **Random Forest** benefits moderately from quadrant encoding, particularly in the branched configuration. It does not reach the accuracy or stability of XGBoost models but presents performance more consistent than SVR/M-SVR.

The choice between branched and vector integration strategies also influences model performance:

- For **XGBoost 1.7**, the vector strategy paired with quadrant encoding is optimal, while the branched approach is less effective, especially when combined with more complex encodings.
- **Random Forest** shows a preference for the branched strategy when used with quadrant encoding, achieving its lowest MAAE (11.62°), but remains slower and less accurate than XGBoost.
- **SVR** and **M-SVR** do not benefit significantly from either integration strategy, with both approaches yielding high error and erratic predictions.

These results demonstrate that XGBoost 1.7, with vector integration and quadrant encoding, offers the best trade-off between accuracy, robustness to boundary discontinuities, and computational efficiency using real-world data. The baseline analysis reveals that this superiority stems from inherent architectural capacity to handle circular predictions, rather than merely benefiting from sophisticated encoding schemes. While more complex encodings can marginally improve mean error for models with weaker baselines (e.g., XGBoost 2), the quadrant encoding strikes an effective balance between informativeness and generalisation. Models such as M-SVR and SVR are fundamentally limited by their inability to manage angular discontinuities at the architectural level, and consequently do not benefit meaningfully from richer encodings or alternative integration strategies. Random Forest provides moderate stability but cannot match the overall reliability or

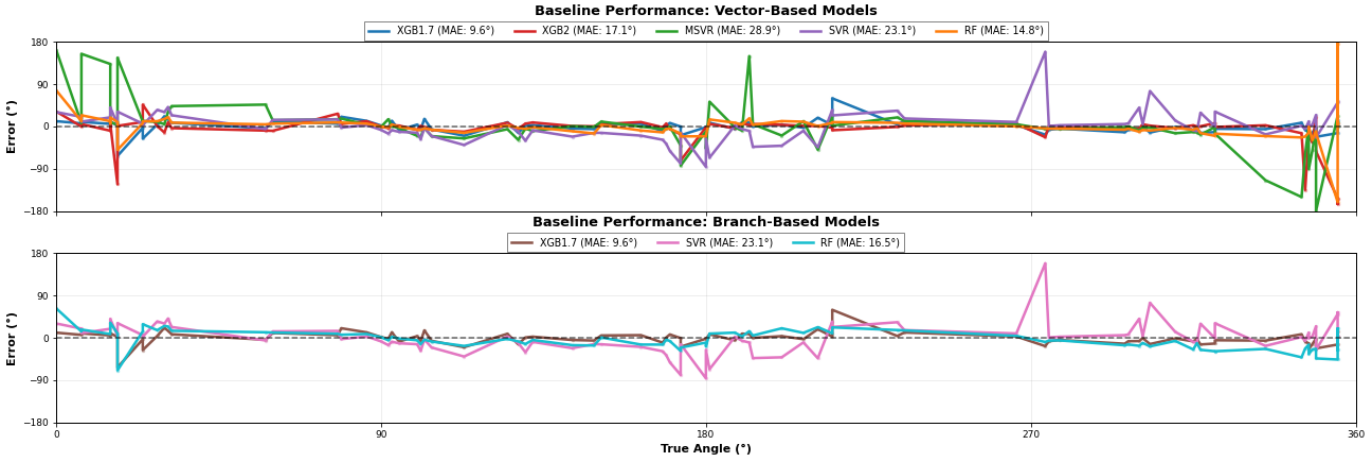


Figure 10. Baseline performance comparison across all models using only (sin, cos) encoding. Top: Vector-based integration strategies. Bottom: Branch-based integration strategies. MAAE values quantify performance without encoding augmentation, revealing inherent architectural capacity for circular prediction.

speed of XGBoost 1.7. These findings underscore the need to consider both aggregate metrics and detailed error distributions when selecting models for applications requiring consistent, timely performance across all angular ranges.

VI. Conclusion and Future Work

This study systematically evaluated novel target encoding schemes, integration strategies, and a variety of shallow regression models for the challenging task of planar object orientation prediction, using both synthetic and real-world datasets. The results demonstrate that among the configurations tested, XGBoost 1.7 combined with the vector based integration strategy and quadrant encoding consistently achieves the best balance of predictive accuracy, robustness to angular discontinuities, and computational efficiency when applied to real-world data. While more complex encodings such as full composite or polar augmented representations can marginally improve mean error for certain models, the quadrant encoding offers superior generalisability by effectively balancing informativeness with simplicity and avoiding overfitting or feature redundancy. Building on these insights, future work will focus on deploying the optimal model configuration within an industrial robotic grasping context, thereby advancing automation reliability and precision in manufacturing systems. These findings provide actionable guidance for designing efficient and robust orientation prediction modules that can integrate seamlessly into intelligent robotic manipulation pipelines in smart factories.

References

[1] A. Gambale *et al.*, "Orientation prediction in robotics: A study of trigonometric decomposition methods across synthetic and real-world datasets", in *Proceedings of the 19th International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP)*, 2025, pp. 1–7.

[2] X. Gao and Z. Wen, "Task-oriented robotic grasping for intelligent manufacturing", in *2023 3rd International Symposium on Artificial Intelligence and Intelligent Manufacturing (AIIM)*, IEEE, 2023, pp. 101–104.

[3] H. Sekkat, O. Moutik, L. Ourabah, B. ElKari, Y. Chaibi, and T. A. Tchakoucht, "Review of reinforcement learning for robotic grasping: Analysis and recommendations", *Statistics, Optimization & Information Computing*, vol. 12, pp. 571–601, 2024.

[4] L. Chen, P. Huang, Y. Li, and Z. Meng, "Detecting graspable rectangles of objects in robotic grasping", *International Journal of Control, Automation and Systems*, vol. 18, pp. 1343–1352, 2020.

[5] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 1316–1322.

[6] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 769–776.

[7] M. Schwarz, H. Schulz, and S. Behnke, "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1329–1335. DOI: 10.1109/ICRA.2015.7139363.

[8] A. Wong, Y. Wu, S. Abbasi, S. Nair, Y. Chen, and M. Shafiee, "Fast graspnext: A fast self-attention neural network architecture for multi-task learning in computer vision tasks for robotic grasping on the edge", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2293–2297.

[9] A. Gambale *et al.*, "A comparative study of hough transform and pca for bolt orientation detection", in *2024 IEEE 22nd International Conference on Industrial Informatics (INDIN)*, IEEE, 2024, pp. 1–6.

[10] J. Brownlee, *Ensemble Learning Algorithms With Python: Make Better Predictions with Bagging, Boosting, and Stacking*. Machine Learning Mastery, 2021.

[11] Y. Chen, E. Z. Zeng, M. Gilles, and A. Wong, "Metagraspnet_v0: A large-scale benchmark dataset for vision-driven robotic grasping via physics-based metaverse synthesis", *arXiv preprint arXiv:2112.14663*, 2021.

[12] G. Bradski, "The opencv library", *Dr. Dobb's Journal of Software Tools*, 2000.

[13] O. Team, *Image moments (ana huamán tutorial)*, https://docs.opencv.org/4.x/d0/d49/tutorial_moments.html, Accessed: 2025-06-26, 2023.

[14] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] K. Pearson, "On lines and planes of closest fit to systems of points in space", *Philosophical Magazine*, vol. 2, pp. 559–572, 1901. DOI: 10.1080/14786440109462720.

[16] J. Zhang, B. Yin, Y. Zhong, Q. Wei, J. Zhao, and H. Bilal, "A two-stage grasp detection method for sequential robotic grasping in stacking scenarios", *Mathematical Biosciences and Engineering*, vol. 21, pp. 3448–3472, 2024.

[17] I. M. Baltruschat, A. Saalbach, M. P. Heinrich, H. Nickisch, and S. Jockel, "Orientation regression in hand radiographs: A transfer learning approach", in *Medical Imaging 2018: Image Processing*, SPIE, vol. 10574, 2018, pp. 473–480.

[18] A. Depierre, E. Dellandréa, and L. Chen, "Scoring graspability based on grasp regression for better grasp prediction", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 4370–4376.

[19] A. Kirillov *et al.*, "Segment anything", in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[20] S. Robles Herrera, M. Ceberio, and V. Kreinovich, "When is deep learning better and when is shallow learning better: Qualitative analysis", *International Journal of Parallel, Emergent and Distributed Systems*, vol. 37, pp. 589–595, 2022.

[21] H. Borchani, G. Varando, C. Bielza, and P. Larranaga, "A survey on multi-output regression", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, pp. 216–233, 2015.

[22] A. Morales-Hernández, I. Van Nieuwenhuyse, and S. Rojas Gonzalez, "A survey on multi-objective hyperparameter optimization algorithms for machine learning", *Artificial Intelligence Review*, vol. 56, pp. 8043–8093, 2023.

[23] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization", *Journal of Machine Learning Research*, vol. 13, 2012.

[24] Z. Wang, Z. Zhang, T. Pang, C. Du, H. Zhao, and Z. Zhao, "Orient anything: Learning robust object orientation estimation from rendering 3d models", *arXiv preprint arXiv:2412.18605*, 2024.

[25] M. Gilles, Y. Chen, T. R. Winter, E. Z. Zeng, and A. Wong, "Metagraspnet: A large-scale benchmark dataset for scene-aware ambidextrous bin picking via physics-based metaverse synthesis", in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2022, pp. 220–227.

[26] A. Gambale, E. Kerr, D. Kerr, and S. Coleman, "Computing the orientation of hardware components from images using traditional computer vision methods", *Engineering Proceedings*, vol. 65, p. 8, 2024.

[27] A. Gambale, *Screwdrivers dataset*, https://app.roboflow.com/mvtec-uihve/screwdrivers-kmij6/2, Accessed: 2025-06-13.