Structuring Motion-Based Communication as a Language for Robotics: Analysis with Dense Optical Flow

Nicholas Albergo[1], Jeongbin Hwang [2], Doyun Lee [3], and Kevin Han [2]

[1]Department of Computer Science, North Carolina State University, North Carolina State University, Raleigh, NC, USA
[2]Department of Civil, Construction, & Environmental Engineering, North Carolina State University, Raleigh, NC, USA
[3]Department of Civil Engineering & Construction, Georgia Southern University, Statesboro, GA, USA

njalberg@ncsu.edu, jhwang25@ncsu.edu, doyunlee@georgiasouthern.edu, kevin_han@ncsu.edu

Abstract -

The construction industry is currently facing a labor shortage, prompting many researchers and practitioners to explore construction automation through advancements in robotics. For successful construction operations with multiple agents, effective and intuitive communication is essential. To this end, this research proposes a Motion-Based Communication (MBC) framework, which enables collaboration using movement. MBC provides a robust alternative that remains unaffected by various types of disruptions. This work contributes to the field of robotics by developing a motion-based language represented through an ontology. The ontological representation is encoded via directional shifts in motion rather than poses and is decoded by analyzing optical flow vectors. The experimental results in a lab environment using a UR5e robot indicate feasibility, with a shift inference accuracy of 98.8%. These findings can contribute to the development of a multi-agent MBC language to support communication in real time during construction operations.

Keywords -

Motion-Based Communication; Construction Automation via Robotics; Cooperating Robots; Dense Optical Flow

1 Introduction

Traditional robot-to-robot communication consists of physical data transfer via networking. However, in the event of a network failure or disruption, this modality of communication cannot be relied on. Additionally, networked communication efficacy is contingent on condition of the environment. For instance, in the case of large infrastructure construction projects such as dams, roads, and bridges, the installation of Internet networks (i.e., LTE and 5G) may be challenging, making seamless online communication difficult. Additionally, there are wireless networks that do not rely on Internet connections, such as Bluetooth [1] and Wi-Fi [2], which are relatively easy to install and can address some of these shortcomings. However, these networks are limited to specific areas and require manual connection each time they are used. As construction sites physically constantly change over time, wired networks must be reconfigured to consider the site conditions, which requires significant time and cost.

Motion-based communication (MBC) serves as a potential solution for non-networked communication for several reasons. First, it can potentially allow communication between human and computer as motion can be used as a common language channel that is understandable by humans [3]. In addition, compared to other multi-modal robot-to-human communication methods [4], displays and LED-based systems suffer from areas of high contrast or light intensity, while audio communication is evidently ineffective in vacuums or underwater. Lastly, motion is not a unique attribute and has the potential to facilitate a universal language channel in both organic and robotic agents that are capable of some form of actuation.

Therefore, the authors propose implementing MBC as a pseudo-language. This is done in two steps. First, an ontology is created by performing *open coding* [5] on several robot action research papers and corpora and ontologies [6, 7, 8] represented by OWL files (W3C Web Ontology Language [9]). The coding process results in an ontology representing a pseudo-language consisting of **action categories**, **actions types**, **entity types**, and **value types**, and **atomic types** (referred to collectively as **symbols**).

Afterwards, symbols in the ontology are encoded through alternating clockwise and counter-clockwise movement inside a communication window from a UR5e arm. A dense optical flow based motion analysis framework coupled with contour detection and tracking supplemented by Kalman filter is evaluated as the proposed methodology for decoding the motion-based symbols using a Logitech C270 camera with a resolution of 640 by 480. The efficacy of this method is compared to state-of-the-art (SOTA) MBC research. For brevity, this paper focuses on the actual communication framework and decoding rather than the ontology development. An overview of the steps taken can be seen in Figure 1.

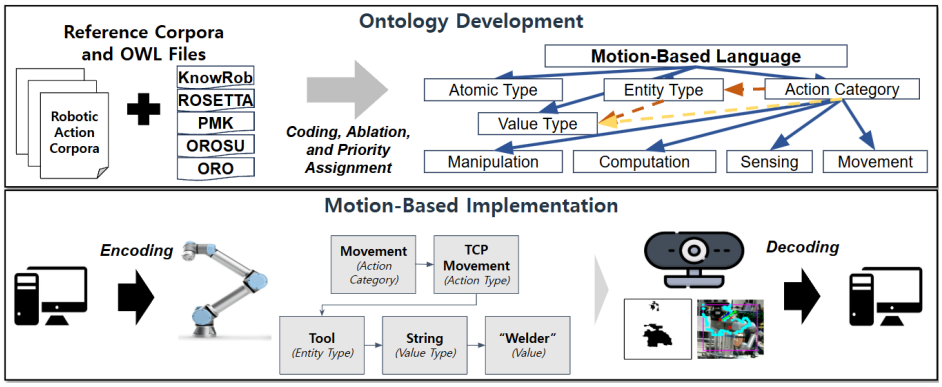

Figure 1. Research overview consisting of an open coding process to construct the language ontology (top) and the communication process with an example message (bottom).

2 Background

2.1 Motion-based communication

MBC deals with encoding a message between multiple agents via pose or motion and interpreting the message using computer vision techniques. These agents can consist of robots, humans, or some mix of the two. The capability to engage in multi-agent (i.e. both human and robot) communication with both human and robotic agents is a crucial benefit of MBC, along with the aforementioned resilience to networking impediments. Modern MBC research, whether dealing with human-to-robot, robot-to-robot, or multi-agent communication, utilizes either gestures [10, 11], poses [4, 3], trajectories [12], or a combination [13] of these to encode information.

2.1.1 Gesture-based

Gestures have been used both in the context of aligning common objects in multi-robot environments [10] as well as a common language channel between humans and robots. Vanc et al. [13] propose a pseudo-language for human-robot interaction (HRI) by implementing a framework using a combination of deictic, mimetic, and quantification gestures in addition to teleoperation. This setup requires Leap Motion Controller [14] and a gesture classification model [11] to recognize the encoded gesture-based signals real-time. Although this framework frames the MBC paradigm as a language, the usage of a model limits the language to communication from a human to a robot.

2.1.2 Pose-based

In the context of this research, the salient components of the pose of an agent are the roll, pitch, and yaw orientations of its body (or end effector, in the case of a robotic manipulator). Encoding from Koreitem et al. [15] is done using a set of ordered lists, called a *code*, where each element is a particular pose configuration (i.e., roll, pitch, yaw) in which the orientation space is binned. Their work focuses on minimizing a cost function across all codes. For each possible pose configuration, the cost is calculated by three factors: probability of occurrence, detection error of their orientation regressor, and time to execute. Decoding is performed via convolutional neural network (CNN) based pose estimation. As the main focus is the optimization of the encoded messages with respect to time, energy, and accuracy, there is less of an emphasis on flexibility. Using a pose estimation model limits generality, and each pose encodes a pre-built, holistic message.

2.1.3 Trajectory-based

Although using pose as a method of encoding messages is popular in modern MBC research, it can be supplemented with trajectory information as well. SOTA research from Fulton et al. [4] and Enan et al. [3] mainly use pose-based encodings, but some of their messages also contain trajectories. For instance, in [3], the message "GO TO LOCATION" is defined by rolling 45-degrees, moving forward and then backward twice, and then proceeding to the desired location. Their research is the first to create a unified HRI and robot-robot interaction (RRI) motion-based framework in which both the human operator and robot can interpret the gestures. In addition, it is the first MBC research to utilize self-attention [16] for decoding. User-defined gestures can also be defined to extend the language via configuration files.

However, as previously mentioned, MBC research has not taken full advantage of the potential generality of motion. That is, there is information contained in the motion itself rather than the pose that can be used as a sort of lexicon. In the example from [3], the individual words *go to* and *location* and their semantics are not encoded. There-

fore, adding another unique message relating to movement, for instance "GO TO AGENT," would require an entirely unique encoding, potentially even unrelated to the original one. Additionally, while the usage of deep-learning models enables potentially high precision, it comes at a high computational cost and can limit the ability to generalize the communication framework to other robots.

3 Methodology

The following sections briefly outline the methodology used in developing the baseline ontology used in this framework as well as covering the decoding process used to interpret messages.

3.1 Ontology coding process

A rough initial ontology is built based upon coding actions, categories, and frames from previous robotic action studies [8, 7, 6] to guide the coding process. The initial ontology consists of mainly action categories and action types with some initial entity types and values. Then, five OWL files (studied in [17]) are inspected using Protégé [18] and all leaves (a class without any subclasses) are extracted. Irrelevant classes, such as those not related to actions, objects, or values, are ablated. Each class is coded as either a type of action, entity, or generic value using the preliminary ontology as a reference.

From largest to smallest, the OWL files used in this research come from KnowRob [19, 20], ROSETTA [21], OROSU [22], ORO [23], and PMK [24]. These five OWL files are selected as they are among the 5 largest, open-source, industrial robotic action ontologies. For each of the 5 OWL files, only classes (i.e. subclasses of **owl:Thing**, not properties or individuals) relating to actions and objects were extracted and coded from the five OWL files.

This entire coding process is repeated for a total of five full runs, refining the codes and resulting in added granularity in the form of introducing action categories and specific value types. The final ontology for the proposed language consists of the following levels: **action categories**, **action types**, **entity types**, **value types**, and **atomic types**. Action categories are broad classifications of actions (*Manipulation*, *Movment*, etc.) while action types are specific varieties of those actions (*Handling*, *Rotation*, etc.). Entities are objects that can be observed or interacted with are children of action types that use them. Value types correspond to the possible arguments that can be used for specific messages. All symbols, aside from literal values, are assigned a priority with respect to other symbols in the same level of the tree based on their frequency in the surveyed literature.

3.2 Optical flow-based decoding

The communication process begins when one robot sends a *ToggleCommunications* symbol which consists of a unique cross pattern (discussed in Section 3.2.5). Then, the communicating robot begins sending a sequence of symbols followed by a final *ToggleCommunications* symbol which terminates communication. Motion detection and tracking for interpreting symbols (encoded as aforementioned **shifts**) is done using Gunnar-Farnebäck optical flow [25] coupled with contour finding [26]. A Kalman Filter [27] is also used to help estimate the center of the tracked objects.

In this language, symbols and literal values are encoded using a discrete quantity of **lexical motion units** (LMU). The magnitude of this motion (i.e. one LMU) is calculated dynamically when an agent initiates communication (see 3.2.5). The motion consists of a circular trajectory inside the communication window and alternates between counterclockwise and clockwise movements for each symbol. This ensures symbols are distinct and identifiable with a definitive **shift** in the direction of movement. In other words, the *direction* of the movement is independent of the symbol. Rather, the direction of motion flips after each symbol such that the overall message consists of a series of directional shifts along a fixed circle. Each symbol's encoding (aside from literal values) is equivalent to the calculated LMU value multiplied by its priority from the open coding process. Literal values have their own encoding rules using a binary representation, which is left out for brevity.

3.2.1 Farnebäck method

Farnebäck's method falls under the category of differential optical flow as it utilizes partial differential equations to model the flow field and minimize an energy function to perform motion estimation. It is chosen as the base motion detection component for several reasons. It does not rely on deep learning or training data and as such allows for better generalization and faster inference time compared to SOTA MBC decoding paradigms that implement pose-based CNNs [15, 4, 3]. This is a result of using a model-free method that only uses motion where there is no need to query any models. Lastly, the OpenCV [28] libraries provide the Farnebäck algorithm, permitting a simpler and more portable framework, which are attributes that are important in resource-limited environments that can often occur in construction.

Results from dense optical flow in environments with inconsistent lighting or a large quantity of background motion can be noisy. Flow vector magnitude thresholding, Gaussian blurring, and morphological operations (such as erosion and dilation) are all applied to yield a cleaner

image before processing continues.

3.2.2 Contour tracking

Dense optical flow yields both a magnitude $m(x, y)$ and direction $\theta(x, y)$ for each (x, y) pixel in the image. However, contours do not inherently contain optical flow information as they only consist of a set of points that define a boundary. Therefore, given a set of optical flow vectors **F** and contours **Q** inside an image **I**, the mapping from pixels in **Q** to **F** is described by:

$$\forall (x, y) \in \mathbf{Q},\ \exists\, (m(x, y), \theta(x, y)) = \mathbf{F}(x, y) \qquad (1)$$

Each contour is then assigned to an **Optical Flow Contour** object (OFC). This is done by comparing the central moment of a contour to each previously found OFC. If the center is within a tracking threshold, T_t (chosen as 22.5 pixels given the experimental setup and camera resolution), the OFC object is updated with the new information from the contour via a Kalman filter. If there are no current $OFCs$ or the current contour's center or area is beyond the thresholds, a new OFC object is created.

3.2.3 Optical flow contour objects

Each OFC contains the current magnitude $\bar{m}_i$, current angle $\bar{\theta}_i$, and center (c_i^x, c_i^y) of the contour it is assigned or updated with. To calculate $\bar{m}_i$ for the OFC, the mean of all *non-zero* magnitude pixels inside the assigned contour is found, filtering small noisy measurements. The angle $\bar{\theta}_i$ is computed by taking the circular mean of the angle vector, which also only includes pixels that are included in the mean magnitude calculation. Therefore, using the remaining n pixels inside the assigned contour, the magnitude and angle are given by:

$$\bar{m}_i = \frac{1}{n}\sum_{k=1}^{n} m_k;\ \bar{\theta}_i = \arg\left(\frac{1}{n}\sum_{k=1}^{n} e^{\mathrm{i}\theta_k}\right) \qquad (2)$$

Where i is the imaginary unit and $\arg(z)$ is the argument (angle) of the complex number z. Lastly, the center is found by taking the weighted center of pixel intensity (moment) from the convex hull of the contour. (c_i^x, c_i^y) and $\bar{m}_i$ are passed into the Kalman filter as measurements. This is used to compute the estimated state, or center of the OFC. In this way, the Kalman filter assists in the tracking of an OFC, but does not directly calculate its magnitude or angle. For each frame an arbitrary OFC is detected, $\bar{m}_i$ is added to that OFC's **shift magnitude**, ρ_i. The shift magnitude then contains the sum of all non-zero mean magnitudes, but is reset once a shift is detected for the current OFC. Motion - specifically the motion of the sender in between each shift - is then quantified by using ρ_i.

3.2.4 Shift detection

Shifts are considered changes in the angular component of the optical flow vector of $\frac{\pi}{2}$, also know as the angular threshold T_θ. The decoding process then consists of analyzing the motion segments in between each directional shift and estimating how many discrete LMUs make up each motion segment.

To determine a shift, $\Delta\bar{\theta}_i$ is acquired by finding the angular difference between $\bar{\theta}_i$ and $\bar{\theta}_{i-1}$. A maximum of three angular differences are stored in memory as a deque (double-ended queue) for each OFC to detect shifts; this is known as the **angle history**, Θ, of the OFC. A maximum deque for an arbitrary OFC is then given by:

$$\Theta = \{\Delta\bar{\theta}_{i-2}, \Delta\bar{\theta}_{i-1}, \Delta\bar{\theta}_i\} \qquad (3)$$

Each time an OFC object is matched with a viable contour in the current frame and updated, the angular difference is calculated and added to its angle history. The angles inside Θ are summed and compared to T_θ:

$$f(\Theta) = \sum_{\Delta\bar{\theta}\in\Theta} \Delta\bar{\theta} \qquad (4)$$

If $f(\Theta)$ exceeds T_θ, a shift is detected. $\bar{\theta}_i$, ρ_i, and (c_i^x, c_i^y) are then added to a dequeue known as the **shift history**, S, up to a maximum of seven shifts (the number of shifts required to detect the communication window). When a shift occurs, both Θ_i and the current shift magnitude ρ_i are cleared for the current OFC. Then ρ_i is used to determine the current symbol by rounding it to the nearest whole LMU. For example, given an LMU of 5.5 and final ρ_i of 10.2, the symbol is interpreted as 2 LMUs; the current direction of motion is irrelevant. Figure 2 shows shift detection between two frames.

3.2.5 ROI estimation for communication window

The symbol used to request and end communication, $ToggleCommunications$, serves two main roles. First, to signal the start and end of the motion-based communication chain. Secondly, it establishes the *communication window* and LMU estimation used by the receiver. The purpose of the communication window is also two-fold. It reduces noise from the environment by focusing on a specific region of interest (ROI) within the camera view and it reduces computation by disregarding pixels outside this ROI (see Section 4.1). This has the advantage of increasing frame rate while still avoiding the "aperture problem" caused by local measurements within a small window since the optical flow component continues to track the entire scene.

The sender and receiver are assumed to both be aware of the expected radius of the communication window, r_c,

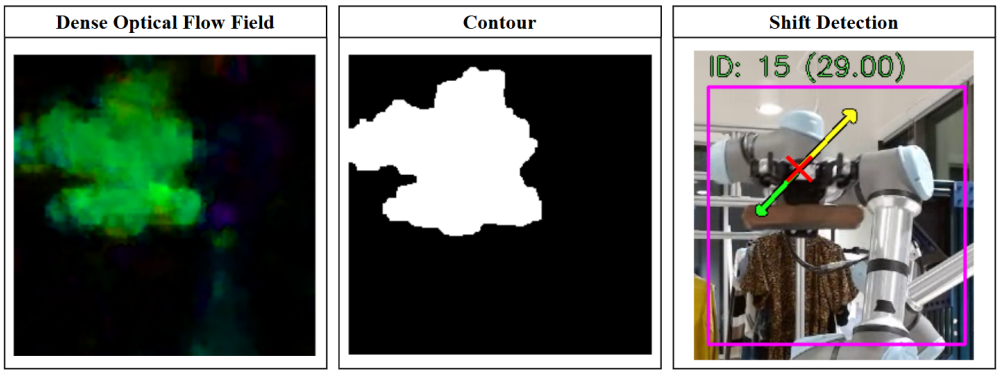

Figure 2. Optical flow field (left), contour (middle) and annotated frame (right) during a detected shift, marked with a red 'X' inside the ROI (purple box). Yellow and green arrows represent magnitude and direction ($\bar{m}$, $\bar{\theta}$) for $i-1$ and i, respectively. The ID of the OFC and current shift magnitude ρ_i are displayed above the ROI.

and an expected **radius ratio**, α, which is used to designate the ratio between ROI_r and r_c. In other words, the $ToggleCommunications$ symbol is meant to be sent within a smaller portion of the overall communicating window. This is used to help establish the ROI and LMU computation between sender and receiver, regardless of distance. The chosen r_c in this case is 10 cm, and α is selected to be 1/3. The $ToggleCommunications$ symbol consists of a series of movements in a cross pattern: $\{down, left, right, down, up, right, left\}$. For each OFC with a fully saturated shift history deque, the receiver actively processes these shifts until this sequence is found. It designates the $ToggleCommunications$ symbol, which sets the OFC that sent the symbol to *active*, meaning the receiver is now actively listening to (or more appropriately, watching for) encoded symbols from this OFC.

The ROI for the communication window is also established during this process. To find the window, the center of the points from the shifts are used to calculate both the center (origin) of the window (ROI_c), its radius (ROI_r), and its average magnitude (ROI_ρ). The center is found by taking the mean (x, y) coordinates and the radius by taking the average distance between each shift s in the $ToggleCommunications$ symbol for an arbitrary OFC:

$$ROI_c = \frac{1}{n}\sum_{k=1}^{n} s_k^{(c_x, c_y)} \tag{5}$$

$$ROI_r = \frac{1}{n-1}\sum_{k=1}^{n-1} \sqrt{(s_{k+1}^{c_x} - s_k^{c_x})^2 + (s_{k+1}^{c_y} - s_k^{c_y})^2} \tag{6}$$

The initial ROI_r is then multiplied by the reciprocal of the radius ratio $\frac{1}{\alpha}$ to yield the final expected ROI_r. An ROI is created around the origin ROI_c with radius ROI_r in which **only motion inside this window** from the currently **active** OFC is considered for symbol interpretation. Lastly, a weighted average shift magnitude of the radius is used in the computation of acquiring the expected magnitude for a singular lexical motion unit. At any time, the sending robot can signal $ToggleCommunications$ to remove the ROI and active status, thus allowing the receiver robot to resume actively observing for the start of a new $ToggleCommunications$ symbol.

4 Experimental results

To evaluate the overall accuracy of the system, experimental testing is performed on recordings of 13 distinct messages - one from each action type, with arbitrary entity and value parameters, and a message composed of several other messages. Each message is recorded from 1, 2, 3, 4, and 5 meters. Below is a list of example chosen messages along with their meaning:

M05 $Handling \rightarrow Manipulation \rightarrow Tool \rightarrow None \rightarrow None \rightarrow 0$: Pick up the nearest $Tool$.
M08 $Movement \rightarrow TCPMovement \rightarrow None \rightarrow Integer \rightarrow 20$: Move TCP forward 20 cm.
M09 $Movement \rightarrow TCPRotation \rightarrow None \rightarrow IntegerVector \rightarrow [90, 0, 0]$: Rotate TCP by 90 degrees in the x-axis.
M12 $Halt$: Stop all movement/actions and cancel all previous commands.
M13 A message consisting of 3 other messages: pick up the nearest tool, move the TCP to the nearest obstacle, and activate held tool

Results from decoding a message (**M13**) can be seen in Figure 3. Evaluation is performed by comparing the actual ρ_t^a (red dotted line) at each t where a shift occurs against the expected ρ_t^e (green line). Since LMUs are discrete units, ρ_t^a is rounded to the nearest LMU to get the final

decoded result, ρ_t^r (cyan dashed line). The orange dotted lines represent the LMU intervals, with the first (bottom) line equal to one LMU, the next line equal to two LMUs, and so on. The mean residual error is equal to the average difference at each shift ($|\rho_t^a - \rho_t^e|$)

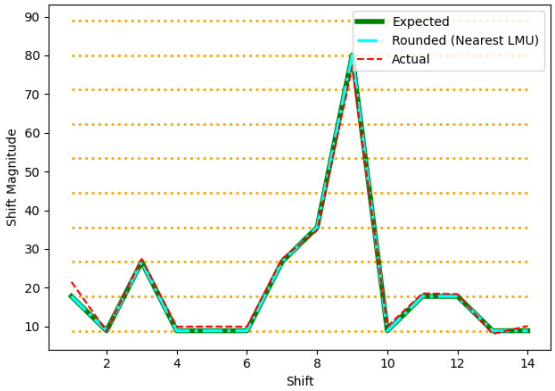

Figure 3. **M13** correctly interpreted at 2m with minimal mean residual error.

Across all 13 messages, there are a total of 167 symbols. Of the distances evaluated, the optimal range of interpreting shifts falls between 1 to 3 meters. From these distances, 98.8% (495 of 501) of symbols are correctly interpreted. Overall, a successful interpretation rate of 94.1% (786 of 835) is achieved from the 1 meter bins between 1 and 5 meters using a 640x480 resolution camera. The accuracy of each messages can be seen in Table 1. Each message along with the number of shifts required for its encoding is provided, along with the shift accuracy for both the optimal range (1, 2, and 3 meters) and all ranges (1, 2, 3, 4, and 5 meters). Figure 4 also shows the performance of messages from 1-3 meters, which is the weighted (0.95) accuracy minus the weighted (0.05) mean residual error that results from each shift.

4.1 Performance

Without downsampling, multi-threading, or hardware acceleration, the decoding algorithm runs at around 18-20 FPS on a AMD Ryzen 7 3700X 8-Core Processor. After the ROI is selected, the FPS reaches between 70 (for larger ROIs, in the case of close proximity to the communicating robot) and 330 FPS. As LMUs are dynamically calculated with respect to the relative motion of the robot, the algorithm is theoretically independent on speed assuming that the velocity of the robot remains relatively constant throughout the signaling.

Dense optical flow can struggle in environments with poor lighting or noisy backgrounds, such as flickering lights or numerous rapidly moving objects. The presence of heavy background noise or complex scene dynamics can be a limitation to the proposed dense optical flow approach for MBC, reducing both accuracy and performance. Although preprocessing and ROI construction help to mitigate noise under the evaluated conditions, the effects of environmental illuminance and highly disruptive noise on the proposed system are currently being studied.

Table 1. Results of the 13 messages evaluated from the optimal range (1 to 3 meters) and all ranges (1 to 5 meters) using a UR5e robotic arm. The values for shift inference accuracy represent the percentage of correctly decoded shifts using a 640x480 resolution camera at optimal range (left, 1-3m) and all evaluated ranges (right, 1-5m).

Message	Shifts	Shift Inference Accuracy (%)
M01	25	98.7/98.4
M02	36	100/94.4
M03	4	100/100
M04	22	100/100
M05	6	100/86.7
M06	18	96.3/93.3
M07	4	100/100
M08	10	96.7/88
M09	17	98.0/96.5
M10	5	100/100
M11	4	100/90
M12	2	100/100
M13	14	97.6/77.14

4.2 Comparison with SOTA

The SOTA in motion-based communication [3] achieves a recognition accuracy of 94.67% in simulation using their RRCommNet model, with an average inference speed of 10 FPS and 12.5 FPS (for RRCommNet-Skip, which averages 88% accuracy as it uses every other frame). Real-world accuracy for the two models is 83.33% and 73.81%, respectively for the regular and frame-skipping versions. By using optical flow vectors without machine learning, the MBC language runs faster and does not incur additional computational costs stemming from querying a model.

Since the language is fundamentally different than SOTA works, the comparison between shift inference accuracy and message inference accuracy is not necessarily one-to-one. However, the individual shift inference accuracy from this research outperforms SOTA real-world and simulation message inference accuracy at 1 through 5 meters using a 640x480 resolution camera without the usage of any type of model. The performance is also higher, with a minimum of about 18 FPS, reaching up to 330 FPS.

5 Conclusions

The authors have presented a novel framework for motion-based communication in which it is framed as a

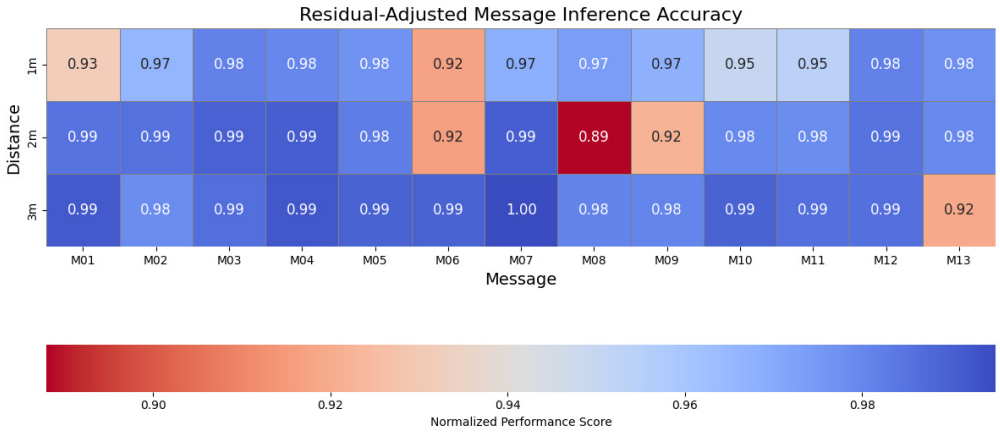

Figure 4. Accuracy across each message from 1-3 meters, accounting for average shift magnitude residuals. Lower values indicate higher mean residual error; orange/red indicates an incorrect shift detection.

language rather than discretely mapping poses to set messages. This is done by developing an ontological structure and encoding its nodes as units of motion along a circle. Experimental testing on a UR5e arm in a real-world lab environment demonstrates the feasibility of implementing this approach with robotic agents. An overall shift inference accuracy of 98.8% between 1 to 3 meters is attained with a 640x480 resolution Logitech C270. Due to the model-free approach, inference is done in real-time without the need for model computation overhead. Additionally, the modularity gained from framing the MBC paradigm as a natural language should allow for enhanced communication flexibility.

Evaluation of the proposed network-free framework in dynamic and complex construction sites where there may be elevated levels of network interference is planned as future work. Since the symbols are encoded and interpreted solely through motion, an additional area of future work is evaluation on other robotic and human agents to verify generality. Due to the extendability of the language and potential for decoding generality, we believe this MBC framework helps lay the groundwork for flexible and robust multi-agent communication going forward.

References

[1] Michael Abner, Peter Kok-Yiu Wong, and Jack C.P. Cheng. Battery lifespan enhancement strategies for edge computing-enabled wireless bluetooth mesh sensor network for structural health monitoring. *Automation in Construction*, 140:104355, 2022. ISSN 0926-5805. doi:https://doi.org/10.1016/j.autcon.2022.104355.

[2] Payam R. Zekavat, Sungkon Moon, and Leonhard E. Bernold. Performance of short and long range wireless communication technologies in construction. *Automation in Construction*, 47:50–61, 2014. ISSN 0926-5805. doi:https://doi.org/10.1016/j.autcon.2014.07.008.

[3] Sadman Sakib Enan, Michael Fulton, and Junaed Sattar. Robotic detection of a human-comprehensible gestural language for underwater multi-human-robot collaboration. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2022-October, pages 3085–3092. Institute of Electrical and Electronics Engineers Inc., 2022. ISBN 9781665479271. doi:10.1109/IROS47612.2022.9981450.

[4] Michael Fulton, Chelsey Edge, and Junaed Sattar. Robot communication via motion: A study on modalities for robot-to-human communication in the field. *ACM Transactions on Human-Robot Interaction*, 11:1–40, June 2022. ISSN 25739522. doi:10.1145/3495245.

[5] John W. Creswell and Cheryl N. Poth. *Qualitative inquiry and research design: Choosing among five approaches*. SAGE Publications, 4 edition, February 2016.

[6] Matteo Pantano, Thomas Eiband, and Dongheui Lee. Capability-based frameworks for industrial robot skills: a survey. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 2355–2362. IEEE Press, 2022. doi:10.1109/CASE49997.2022.9926648.

[7] Andrea Vanzo, Danilo Croce, Emanuele Bastianelli, Roberto Basili, and Daniele Nardi. Grounded language interpretation of robotic commands through structured learning. *Artificial Intelligence*, 278, January 2020. ISSN 00043702. doi:10.1016/j.artint.2019.103181.

[8] Irena Markievicz, Jurgita Kapočiūtė-Dzikienė, Minija Tamošiūnaitė, and Daiva Vitkutė-Adžgauskienė. Action classification in action ontology building using robot-specific texts. *Information Technology and Control*, 44:155–164, July 2015. ISSN 2335884X. doi:10.5755/j01.itc.44.2.7322.

[9] Michael Dean, Guus Schreiber, Sean Bechhofer, Frank Van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn A. Stein. Owl web ontology language overview, 2004. Available at: `https://www.w3.org/TR/owl-features/`.

[10] Polychronis Kondaxakis and Khurram Gulzar. Robot–robot gesturing for anchoring representations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 107–114. IEEE, 2018. doi:10.1109/IROS.2018.8502843. URL `https://ieeexplore.ieee.org/document/8502843`.

[11] P. Vanc. Gesture teleoperation toolbox v.0.1, 2022. URL `https://github.com/imitrob/teleopgesturetoolbox/`.

[12] Haruki Nishimura and Mac Schwager. Active motion-based communication for robots with monocular vision. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2948–2955, 2018. ISBN 9781538630815. doi:10.1109/ICRA.2018.8463152.

[13] Petr Vanč, Jan Kristof Behrens, Karla Štěpánová, and Václav Hlaváč. Communicating human intent to a robotic companion by multi-type gesture sentences. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9839–9845, 2023. doi:10.1109/IROS55552.2023.10341944. URL `https://doi.org/10.1109/IROS55552.2023.10341944`.

[14] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(55):6380–6393, May 2013.

[15] Karim Koreitem, Jimmy Li, Ian Karp, Travis Manderson, and Gregory Dudek. Underwater communication using full-body gestures and optimal variable-length prefix codes. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8018–8025, 2019. ISBN 9781538660270. doi:10.1109/ICRA.2019.8793644.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[17] Alberto Olivares Alacros. A review and comparison of ontology-based approaches to robot autonomy. *The Knowledge Engineering Review*, 34, 2019. doi:10.1017/S0269888920000090.

[18] Mark A Musen. The protégé project: a look back and a look forward. *AI Matters*, 1:4–12, June 2015. doi:10.1145/2757001.2757003.

[19] Moritz Tenorth and Michael Beetz. Knowrob — knowledge processing for autonomous personal robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266, 2009. ISBN 9781424438044. doi:10.1109/IROS.2009.5354602.

[20] Moritz Tenorth and Michael Beetz. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *International Journal of Robotics Research*, 32:566–590, April 2013. ISSN 02783649. doi:10.1177/0278364913481635.

[21] Maj Stenmark and Jacek Malec. Knowledge-based industrial robotics. In *Twelfth Scandinavian Conference on Artificial Intelligence*, pages 265–274. IOS Press, 2013. doi:10.3233/978-1-61499-330-8-265.

[22] Paulo J.S. Goncalves and Pedro M.B. Torres. Knowledge representation applied to robotic orthopedic surgery. *Robotics and Computer-Integrated Manufacturing*, 33:90–99, 2015. ISSN 07365845. doi:10.1016/j.rcim.2014.08.014.

[23] Séverin Lemaignan, Raquel Ros, Lorenz Mösenlechner, Rachid Alami, and Michael Beetz. Oro, a knowledge management platform for cognitive architectures in robotics. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3548–3553, 2010. ISBN 9781424466764. doi:10.1109/IROS.2010.5649547.

[24] Mohammed Diab, Aliakbar Akbari, Muhayy Ud Din, and Jan Rosell. Pmk—a knowledge processing framework for autonomous robotics perception and manipulation. *Sensors*, 19, March 2019. ISSN 14248220. doi:10.3390/s19051166.

[25] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370. Springer Berlin Heidelberg, 2003. URL `http://www.isy.liu.se/cvl/`.

[26] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30:32–46, 1985. ISSN 0734-189X. doi:https://doi.org/10.1016/0734-189X(85)90016-7.

[27] R E Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35–45, March 1960. ISSN 0021-9223. doi:10.1115/1.3662552.

[28] Gary Bradski and Adrian Kaehler. Opencv, 2000. Accessed: 2024-07-16.