DIJE: Dense Image Jacobian Estimation for Robust Robotic Self-Recognition and Visual Servoing

Yasunori Toshimitsu[1], Kento Kawaharazuka[1], Akihiro Miki[1], Kei Okada[1], and Masayuki Inaba[1]

***Abstract*— For robots to move in the real world, they must first correctly understand the state of its own body and the tools that it holds. In this research, we propose DIJE, an algorithm to estimate the image Jacobian for every pixel. It is based on an optical flow calculation and a simplified Kalman Filter that can be efficiently run on the whole image in real time. It does not rely on markers nor knowledge of the robotic structure. We use the DIJE in a self-recognition process which can robustly distinguish between movement by the robot and by external entities, even when the motion overlaps. We also propose a visual servoing controller based on DIJE, which can learn to control the robot's body to conduct reaching movements or bimanual tool-tip control. The proposed algorithms were implemented on a physical musculoskeletal robot and its performance was verified. We believe that such global estimation of the visuomotor policy has the potential to be extended into a more general framework for manipulation.**

I. Introduction

While robots have predominantly been used in factory environments for decades, they have only recently started to be used in everyday environments, which can be unstructured and unpredictable. These uncertainties exist both in the external environment and the robot's own body. In particular, we consider that it is important for the robot to be able to discover its own body while actively moving within the environment.

We can consider the problem of visual self-body control as a two component problem. For one, the robot must be able to recognize the *extent of its own body*, i.e. a self-recognition problem. It must be generalize to recognize novel visual features such as new tools in its hands, while ignoring external movements not caused by the robot. Also, the robot must recognize *how to control its own body*. This is akin to acquiring an internal model of the robot movement usable in control, a simple example being the manipulator Jacobian matrix.

To achieve this, a holistic understanding of the visuomotor system is required, rather than calculating values discretely for predetermined keypoints or markers along the robot. To that end, we propose DIJE (Dense Image Jacobian Estimator), a method to densely (i.e. for every pixel of the image) estimate the image Jacobian based on exteroceptive camera images and proprioceptive joint sensor data. It does not require *a priori* knowledge of the robot's kinematic structure, and the resulting dense image Jacobian encodes the global relationship between visual features and joint information for the whole image.

Multiple image Jacobian estimation methods have been proposed [1]–[5], but to our knowledge, this work is the first to apply it densely on the image. To enable this, we propose a simplified Kalman Filter-based estimation algorithm for the image Jacobian, and an update rule to shift the image Jacobian estimation value in each timestep based on the robot movement. DIJE enables a unified approach to self-recognition and visual servoing control. We propose using the output of DIJE to a self-recognition algorithm which can robustly predict the extent of the robot's own body even in the presence of external movements, and to a markerless visual servoing controller which can learn to control the robot's body and the tools in its hands.

The contributions of this work are as follows:

- a dense estimation method for the image Jacobian implemented by a recursive algorithm, based on the Kalman Filter and a dense update method across timesteps, robust to non robot-induced external movements
- a dense binary labeling algorithm of the robot's self body based on the output of DIJE, capable of distinguishing between self-induced and external movement
- a markerless visual servoing controller based on the output of DIJE

All of the algorithms presented require no prior knowledge of the robot structure, and can be run in real time on a conventional notebook PC.

II. Related Work

A. Robotic Self Recognition

In a robotic self recognition system, the robot attempts to locate and distinguish its own body from the environment in the visual (or depth) image. They can be split into methods requiring prior knowledge of the robotic structure [6]–[9], and those that don't [10]–[12].

For methods requiring *a priori* knowledge, the recognition algorithm is trained based on datasets synthesized from a simulation of the robot. Machine learning algorithms such as Mask R-CNN [6], CNN [7], or random forest [8], [9] has been used for the recognizer.

In the simulator, perfect knowledge of the robot state is available. These recognition models can take advantage of that fact to estimate other features from the image and not just the location or binary mask of the robot body, such as joint angle pose [7]–[9]. However, since a training process

[1]The authors are with the Department of Mechano-Informatics, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan. [toshimitsu, kawaharazuka, miki, k-okada, inaba]@jsk.t.u-tokyo.ac.jp

is required, the application of these methods to recognition of novel objects attached to the robot (such as tools) is not possible.

Model-less methods for self-recognition usually utilize the motion of the robot to segment its body from the background. In the work by Metta and Fitzpatrick, the optical flow is used to detect motion, and as its direction changes depending on joint movement, it is compared to the joint angle measurement to detect the location of the arm [10]. Other methods similarly use the detected motion and correlate it with the joint movement, by calculating the mutual information [12] or comparing the temporal correlation [11] to determine whether to attribute it to the robot.

B. Uncalibrated Visual Servoing

In uncalibrated visual servoing, the manipulator model and camera parameters are not known *a priori* and must be learned online while the robot moves. The image Jacobian J describes the differential relationship between a point in the image of the robot's body and the robot's joint pose.

$$\dot{\boldsymbol{p}} = \boldsymbol{u} = J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{1}$$

where $\boldsymbol{p}$ is the time-varying image coordinate of a point on the robot, $\boldsymbol{u}$ is its time derivative, i.e. the optical flow, and $\dot{\boldsymbol{q}}$ is the joint velocity. As the pseudoinverse of the image Jacobian can be used to control the image coordinates of the robot, the online estimation of J has been intensely researched for uncalibrated visual servoing, with methods being proposed based on Broyden updating [1], Kalman Filtering [3], [5], support vector regression [2], or least-squares fitting [4]. In these works, the estimated image Jacobian has been successfully applied to visual servoing to control robots whose kinematic structures are unknown.

However, many of the uncalibrated visual servoing methods assume that the point of interest being controlled is known *a priori*, for example by markers [1], [5] or is only verified on a simulated model [2]–[4] where perfect knowledge is available for where each point of the robot is located in the camera image.

III. DIJE: Dense Image Jacobian Estimation

In this section, we describe the algorithms used in the estimation of the dense image Jacobian. An overview of the process is shown in Fig 1. In Section III-A we describe the Kalman Filter-based estimator that is run on every pixel that estimates the image Jacobian. Then in Section III-B, the update algorithm is introduced, which is critical to apply the estimation densely across the entire image.

A. Image Jacobian Estimation Formulated as a memory-efficient Kalman Filter

Here, the Kalman filter formulation for the image Jacobian estimation is described. This is run separately for each pixel, and thus the computation can be parallelized. The linear relationship between the optical flow $\boldsymbol{u}$, image Jacobian J, and joint velocity $\dot{\boldsymbol{q}}$ shown in (1) is exploited to formulate it as a Kalman filter (KF). The state estimated by the KF is the image Jacobian matrix concatenated into a vector $\boldsymbol{j}$, as

$$\begin{aligned} J &= \begin{bmatrix} \boldsymbol{j}_x & \boldsymbol{j}_y \end{bmatrix}^T \in \mathbb{R}^{2\times N_j} \\ \boldsymbol{j} &:= \begin{bmatrix} \boldsymbol{j}_x \\ \boldsymbol{j}_y \end{bmatrix} \quad\quad \in \mathbb{R}^{2N_j} \end{aligned} \tag{2}$$

where N_j is the number of joints. Then, by using the optical flow $\boldsymbol{u} \in \mathbb{R}^2$ as the observation, the observation model can be formulated using an observation matrix M_q which is created from the joint velocity data:

$$\begin{aligned} \boldsymbol{u} &= M_q \boldsymbol{j} \\ M_q &:= \begin{bmatrix} \dot{\boldsymbol{q}}^T & \\ & \dot{\boldsymbol{q}}^T \end{bmatrix} \end{aligned} \tag{3}$$

Thus, the model for the KF can be formulated as follows:

$$\begin{aligned} \boldsymbol{j}_k &= update(\boldsymbol{j}_{k-1}) + \boldsymbol{w}_k &&\text{: state model} \\ \boldsymbol{u}_k &= M_q \boldsymbol{j}_k + \boldsymbol{v}_k &&\text{: observation model} \end{aligned} \tag{4}$$

Here, the subscript k refers to data for timestep k. $\boldsymbol{w}_k$ is the process noise which follows a zero mean Gaussian distribution with covariance Q and $\boldsymbol{v}_k$ is the observation noise which follows a zero mean Gaussian distribution with covariance R. The estimated state $\boldsymbol{j}_k$ has covariance P_k. $update(\boldsymbol{j}_{k-1})$ is the result of the state transition model for the dense image Jacobian, which updates the value of the estimated image Jacobian of each pixel based on the robot's movement, whose process is described in Section III-B. We note that if we were to do a conventional sparse estimation of the image Jacobian, the previous value itself $\boldsymbol{j}_{k-1}$ can be used instead of $update(\boldsymbol{j}_{k-1})$. This is because the point at which the image Jacobian is estimated follows the robot's movement and always represents the same point on the robot, and thus the image Jacobian can be considered to be constant over time.

So far, the formulation of the KF to estimate the image Jacobian is the same as what has been proposed by Qian and Su in 2002 [5], apart from the introduction of $update(\boldsymbol{j}_{k-1})$. However, in this research, the KF must be run on every pixel. In order to make the calculation tractable, we propose to approximate the state covariance matrix P_k as a diagonal matrix, and therefore calculate only the diagonal elements. In other words, only the variance is considered in the model, and the covariance is disregarded. Without this assumption, the memory size required to save the covariance matrix becomes $O(N_j^2)$, which becomes intractable as the number of joints N_j increases[1]. By calculating only the diagonal elements of the covariance matrix, the memory requirement becomes $O(N_j)$. This approximation is important to ensure the scalability of DIJE to multi-joint structures.

Thus, the covariance matrix of the image Jacobian vector $\boldsymbol{j}_k$, P_k, can be written as:

$$P_k = \text{diag}(p_{k,1}, p_{k,2}, \ldots, p_{k,2N_j}) \tag{5}$$

[1]for example, saving the covariance matrix for each pixel in a 320×240 image for $N_j = 5$ with 8-byte floats already amounts to 61MB of data.

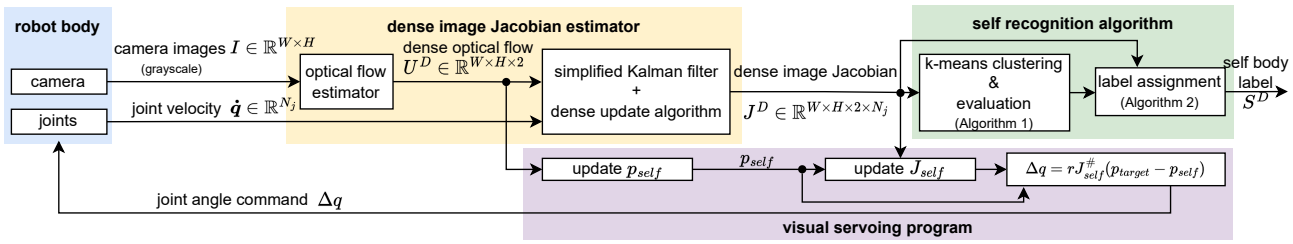


Fig. 1: System diagram of DIJE. It combines exteroceptive data (camera images) and proprioceptive data (joint states) to compute the image Jacobian for every pixel.

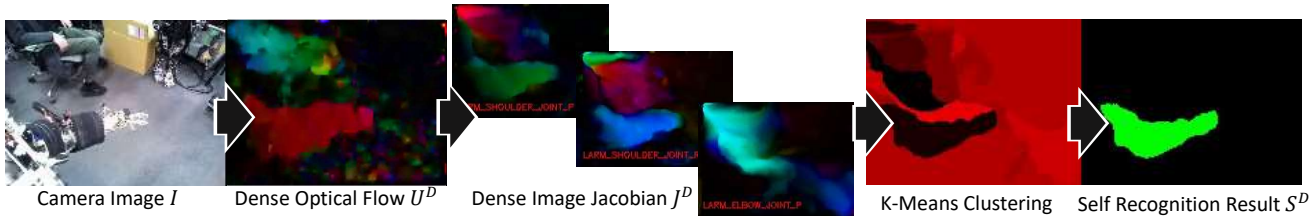


Fig. 2: Visualization of each processing step in DIJE and the dense self recognition algorithm. U^D and each column of J^D are visualized with the hue and brightness in HSV color space. Each cluster is visualized with different shades of red. Even when there is movement in the background (person moving in a chair) the self recognition algorithm can correctly identify which part of the image belongs to the robot.

We make some further assumptions; we assume that the first N_j elements respectively match the last N_j elements, i.e.:

$$\begin{aligned} p_{k,1} &= p_{k,1+N_j} \\ p_{k,2} &= p_{k,2+N_j} \\ &\vdots \\ p_{k,N_j} &= p_{k,2N_j} \end{aligned} \tag{6}$$

This indicates that for each joint, the variance for the horizontal and vertical direction is the same. Then, P_k can be written with a vector $\boldsymbol{p}_k \in \mathbb{R}^{N_j}$, as

$$P_k = \mathrm{diag}(\begin{bmatrix}\boldsymbol{p}_k \\ \boldsymbol{p}_k\end{bmatrix}) \tag{7}$$

By running this formulationn for P_k through the Kalman Filter calculations, it can shown though mathematical induction that (6) holds for every step k, as long as it is true for the initial value.

Using the formulations and assumptions introduced above, the actual calculation of the KF can be formulated as follows. First, in the prediction step, the *a priori* image Jacobian estimate and its variance is calculated.

$$J_{k|k-1} = update(J_{k-1}) \tag{8}$$

$$\boldsymbol{p}_{k|k-1} = \boldsymbol{p}_{k-1} + q\mathbf{1} \tag{9}$$

$\mathbf{1}$ is a vector whose elements are all 1. q is the variance of the process noise, and is assumed to be the same for all elements of the state vector. The process for $update(J_{k-1})$ is described in Section III-B. Next, in the update step, the *a priori* estimate $J_{k|k-1}$ and variance $\boldsymbol{p}_{k|k-1}$, the joint velocity $\dot{\boldsymbol{q}}$, and the observed optical flow $\boldsymbol{u}_k$ is used to calculate the *a posteriori* estimate. The full derivation of these equations is omitted here for the sake of space.

$$J_k = J_{k|k-1} + \frac{(\boldsymbol{u}_k - J_{k|k-1}\dot{\boldsymbol{q}})(\boldsymbol{p}_{k|k-1} \odot \dot{\boldsymbol{q}})^T}{\boldsymbol{p}^T_{k|k-1}(\dot{\boldsymbol{q}} \odot \dot{\boldsymbol{q}}) + r} \tag{10}$$

$$\boldsymbol{p}_k = \boldsymbol{p}_{k|k-1} \odot (\mathbf{1} - \frac{\boldsymbol{p}_{k|k-1} \odot \dot{\boldsymbol{q}}^2}{r + \boldsymbol{p}^T_{k|k-1}\dot{\boldsymbol{q}}^2}) \tag{11}$$

Here, r is the variance of the observation noise.

We note that this simplified KF-based estimation of the image Jacobian yields an update rule similar to the one proposed by Hosoda et al. in 1994 (some of the notation has been modified to match that of this paper) [1]:

$$\Delta J = \frac{(\boldsymbol{u} - J\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}^T W}{\rho + \dot{\boldsymbol{q}}^T W \dot{\boldsymbol{q}}} \tag{12}$$

We can see this as a special case of (10), where the observation noise variance r corresponds to the forgetting factor ρ, and where the state estimate variance $\boldsymbol{p}$ has a fixed value across every step and corresponds to the weighting matrix W.

B. Updating the dense image Jacobian estimate across timesteps

In this section, we will detail the algorithm to obtain $J_{k|k-1}$ used in (8), which updates the dense image Jacobian across timesteps. The superscript D will be used to indicate a dense value that is calculated for every pixel. Here, the ideal update method would be to set the same value of the image Jacobian at step k to that of the same point on the robot at step $k-1$. Thus, we can first consider using the observed optical flow to update the dense image Jacobian,

since they describe the movement caused by the robot. This can be done using grid interpolation for each pixel i, as

$$J_{k|k-1,i} \leftarrow \text{interpolate}(J^D_{k-1}, \boldsymbol{x}_i - \boldsymbol{u}_{k,i}) \tag{13}$$

where $\boldsymbol{x}_i$ is the image coordinates of pixel i, and thus $\boldsymbol{x}_i - \boldsymbol{u}_{k,i}$ is the location of the point appearing at $\boldsymbol{x}_i$ at the previous timestep. The interpolate$(\cdot)$ function interpolates on a regular grid, and thus efficient bilinear interpolation functions such as that implemented in scipy.interpolate.interpn[2] can be used to calculate this interpolation for all pixels in real time.

However, this method has the issue that movement in the environment may be picked up to falsely update the dense image Jacobian, despite it not being caused by the robot. The optical flow caused by such background movement causes the Jacobian value estimated for the robot's body to "leak out" into the environment.

Thus, we propose to instead use the predicted optical flow calculated from the current estimate of the image Jacobian, to update the dense image Jacobian across timesteps. Thus, the update method for each pixel i is as follows:

$$J_{k|k-1,i} \leftarrow \text{interpolate}(J^D_{k-1}, \boldsymbol{x}_i - J_{k-1,i}\dot{\boldsymbol{q}}) \tag{14}$$

As long as the image Jacobian is appropriately estimated, the predicted optical flow $J_{k-1,i}\dot{\boldsymbol{q}}$ is not affected by movements not caused by the robot. The effect of this proposed update method is verified in the experiments section.

IV. Application of DIJE to recognition and control

A. Dense labeling for self-recognition

In this section, we propose an algorithm for dense binary labeling of the robot's own body, based on the k-means clustering algorithm using the output of DIJE. Algorithms 1 and 2 describe the process. Algorithm 1 does k-means clustering and calculates the likeliness that each cluster belongs to the robot's own body, and Algorithm 2 applies the clustering result to the image to get the dense self-recognition label S^D. The two algorithms are run separately because the k-means calculation in Algorithm 1 cannot be run in real time, while the dense label S^D must be generated in real time. Each step of the data processing pipeline is visualized in Fig. 2.

In Algorithm 1, the image Jacobian of each pixel is treated as a vector (and thus the dense image Jacobian is a list of vectors), for the k-means clustering process, to create N_{kmeans} cluster center vectors (i.e., **KmeansList**). For each cluster, an evaluation value is calculated for each cluster (i.e., **EvalList**) that tries to estimate the likeliness that the cluster belongs to the robot's body.

This evaluation is updated based on an assumption that the elements of the image Jacobian are relatively consistent over time, compared to other parts of the image. The image Jacobian would still be erroneously calculated for external movement (as visible in Fig. 2 for the human moving in a chair in the background), but that movement does not correlate with the robot's movements, so the resulting image Jacobian rapidly changes, and the algorithm can label that part of the image as not belonging to itself.

The similarity of each cluster is calculated by first finding the closest cluster center vector from the previous result of k-means, and inherits the evaluation value from the previous corresponding cluster. The evaluation is updated based on the **Consistency** value. This uses an inverse square root (with a small value added to the denominator to avoid dividing by zero) to reward cluster centers that are closer together with the previous corresponding cluster center. Finally, the evaluated values are normalized to 1, and clusters with an **EvalList** value of above e_{thresh} are considered to be the robot's self body ($e_{thresh} = 0.2$ was used in the experiments), and those indices are saved as **SelfBodyIndices**.

The cluster centers **KmeansList** and its indices labelled as the self-body, **SelfBodyIndices**, are used in Algorithm 2 to assign labels in real time.

Algorithm 1 Automatic evaluation of self body from the estimated dense image Jacobian. This algorithm is run in a separate thread from the main loop.

function FindClosest(Vec, VecList)
  find i where VecList[i] is closest to Vec
  return i, ||Vec - VecList[i]||
Global var: KmeansList, SelfBodyIndices
EvalList $\leftarrow [1, 1, \dots]$
while True **do**
  KmeansList $\leftarrow$ Kmeans(J^D, N_{kmeans})
  for all Center in KmeansList **do**
    i, dist $\leftarrow$ FindClosest(Center, KmeansList_prev)
    Eval $\leftarrow$ EvalList[i]
    Consistency $\leftarrow 1/\sqrt{\frac{||dist||}{||Center||} + 0.1}$
    Eval $\leftarrow$ Eval $\times$ 0.1(Consistency -1) $+ 1$
    NewEvalList.append(Eval)
  EvalList $\leftarrow$ Normalize(NewEvalList)
  KmeansList_prev $\leftarrow$ KmeansList
  SelfBodyIndices $\leftarrow$ Where(EvalList[i] $> e_{thresh}$)

Algorithm 2 Label each pixel based on the result of Algorithm 1 in real time. Assign$(\cdot)$ assigns the closest cluster center to each pixel in the image (e.g. scipy.cluster.vq.vq). **SelfRecogDense** corresponds to S_D in Figs. 1 and 2

Global var: KmeansList, SelfBodyIndices, SelfRecogDense
while True **do**
  IndexList $\leftarrow$ Assign($[\boldsymbol{j}_0, \boldsymbol{j}_1, \dots]$, KmeansList)
  SelfRecogDense $\leftarrow$ IndexList is in SelfBodyIndices

B. Visual servoing controller using the estimated dense image Jacobian

Here, we describe a markerless visual servoing controller utilizing the dense image Jacobian, that can control robots

[2]https://scipy.org/

with unknown body and tool kinematic structures. A self-body point $\boldsymbol{p}_{self}$ and target point $\boldsymbol{p}_{target}$ is defined in the image coordinates, and the controller generates joint angle update commands for the robot. We use the letter $\boldsymbol{p}$ rather than $\boldsymbol{x}$ to describe these points, to distinguish that these are time-varying coordinates that are updated according to the actual motion, rather than the fixed coordinates of each pixel (i.e. Lagrangian vs. Eulerian frame of reference).

$\boldsymbol{p}_{self}$ and $\boldsymbol{p}_{target}$ are initially defined by the experimenter clicking on the image. $\boldsymbol{p}_{self}$ is updated to follow the same point on the robot, by adding the flow interpolated at that point from the dense optical flow U_k^D for each timestep:

$$\boldsymbol{p_{self}} \leftarrow \boldsymbol{p_{self}} + \text{interpolate}(U_k^D, \boldsymbol{p_{self}}) \tag{15}$$

As long as the camera maintains a clear view of the robot body around the point $\boldsymbol{p}_{self}$, this was found to robustly track the same point on the robot. Then, the image Jacobian at $\boldsymbol{p}_{self}$ can be calculated:

$$J_{self} = \text{interpolate}(J_k^D, \boldsymbol{p_{self}}) \tag{16}$$

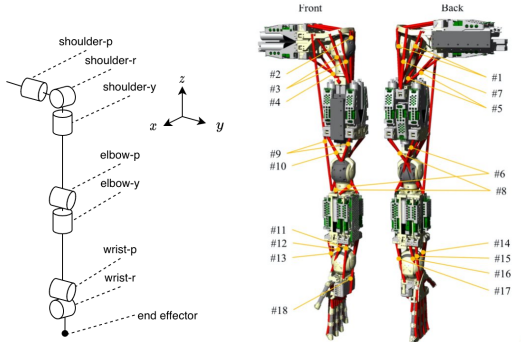
The pseudoinverse of the image Jacobian, $J_{self}^{\#}$, can be used to calculate a joint angle update command $\Delta\boldsymbol{q}$ that brings $\boldsymbol{p}_{self}$ closer to $\boldsymbol{p}_{target}$:

$$\Delta\boldsymbol{q} = k_p J_{self}^{\#}(\boldsymbol{p_{target}} - \boldsymbol{p_{self}}) \tag{17}$$

where k_p is the feedback gain.

V. Experiments

The experiments were conducted on a tendon-driven musculoskeletal humanoid robot Musashi [13]. Such musculoskeletal robots are actuated by flexible muscles, making them ideal for cases where the robot must adapt to holding unknown tools that cause unexpected forces [14]. Here, controllers which map joint angle commands to muscle length commands were used as the low-level controller [15], [16]. Thus, the recognition and control programs can treat Musashi as a conventional joint axis-driven robot, and we can expect that our methods work just as effectively for such axis-driven robots. The Astra S RGBD camera mounted on the head was used for input to DIJE, with a FOV of $60^\circ \times 49.5^\circ$ and captures a 640×480 image at 30fps, and the image size was compressed in half to shorten computation time. Before each experiment, the camera angle was manually servoed to keep the point of interest in the center of the view. The depth channel was disregarded in the experiment. The Gunnar-Farneback optical flow calculation algorithm was used to calculate the dense optical flow [17]. Fig. 3 shows the kinematic and muscle structure of Musashi. Only some of the joints were moved in each experiment, and the other joints were controlled to keep a constant angle. DIJE and the self-recognition and visual servo control programs were implemented in Python, and could be run in real time (30fps) on a conventional notebook computer. At the beginning of the algorithm, the value for the image Jacobian J_0^D was initialized with zeros, and the covariance $\boldsymbol{p}_0$ was initialized with a value of 1. Each experiment is also shown in the video attached to this paper.



(a) (b)

Fig. 3: Structure of musculoskeletal humanoid Musashi's arm. (a) joint and (b) muscle arrangement. [13].

A. Robot body labeling

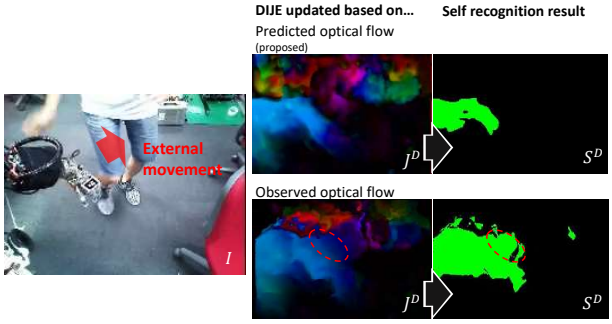


Fig. 4: Self body label "leaking out" of the actual robot body due to background movement for the observed optical flow-based update method, especially prominent in the dotted red circle area. The proposed update method does not suffer from this issue.

Here, the dense robot body labeling method introduced in Section IV-A was evaluated. $N_{\text{kmeans}} = 5$ was used for the experiments. In theory, $N_{\text{kmeans}} = 2$ should be enough to segment the scene into the robot and the background. However, by setting a larger value for N_{kmeans}, the scene can be oversegmented first by the kmeans clustering algorithm and becomes more robust to various false image Jacobian values generated by background movement.

In Fig. 4, the effect of the proposed dense image Jacobian update method used in (14), using the optical flow predicted from J^D, was compared to the update method in (13) which uses the measured optical flow. It is tested in a scenario where there is overlapping movement behind the robot by a human walking. From Fig. 4, it can be seen that with the observed optical flow-based update method, the self-body label leaks out onto the background movement due to the optical flow caused by the human being observed outwards from the robot body. By using the proposed update method of (14) which uses the theoretical optical flow, the background movement is not calculated as it doesn't result from the robot's movement, and thus the algorithm can maintain a consistent label of the robot's body.

B. Reaching experiments

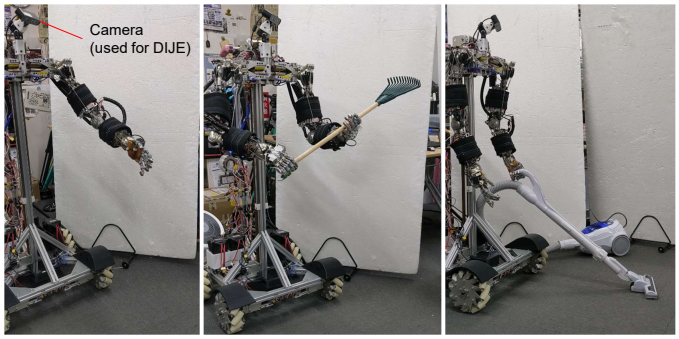


Fig. 5: Experiment setup for the dense image Jacobian estimation experiment. From left, tasks are: reaching, rake tip position control, vacuuming. The human experimenter placed the tools in the robot's hands in the beginning.

Here, the markerless visual servoing controller introduced in Section IV-B was run on Musashi. The feedback gain was set to $r = 0.034$, and the $\Delta \boldsymbol{q}$ joint angle update command was executed by the robot every $0.7\,\mathrm{s}$. In the first reaching

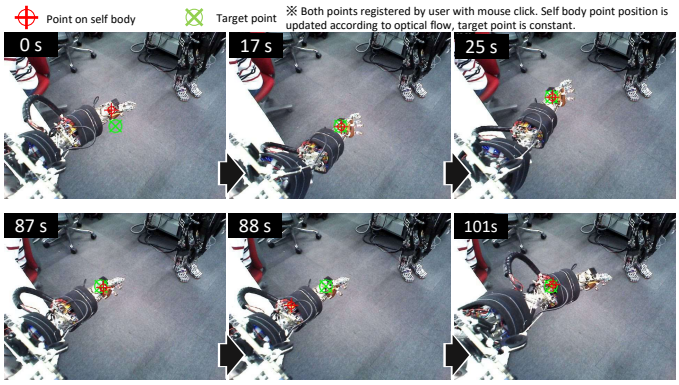


Fig. 6: Result overlayed on camera image for the reaching experiment using DIJE-based controller. Plotted results shown in Fig. 7.

experiment (left picture in Fig. 5), only shoulder-p, shoulder-r, shoulder-y, and elbow-p of the left arm were used.

The results of the reaching experiment are shown in Fig. 6 and 7. In the visualized result in Fig. 6, the red $+$ mark indicates $\boldsymbol{p}_{self}$ and the green $\times$ mark indicates $\boldsymbol{p}_{target}$. In the graph of Fig. 7, the horizontal and vertical elements of $\boldsymbol{p}_{self}$ and $\boldsymbol{p}_{target}$ are plotted. Right after the control is enabled at around $t = 3\,\mathrm{s}$, $\boldsymbol{p}_{self}$ first moves away from $\boldsymbol{p}_{target}$, since the dense image Jacobian is initialized with incorrect values. However, DIJE manages to learn how to move the arm correctly, and it can follow $\boldsymbol{p}_{target}$ even as its position is updated. Further, at $t = 88\,\mathrm{s}$, the position of $\boldsymbol{p}_{self}$ is redefined (again, by a mouse click by the experimenter) from the wrist to the elbow. Since DIJE is run densely on the image, it already knows how to move the newly defined $\boldsymbol{p}_{self}$, and can keep following the target.

In the next experiment, we give the robot a tool such as a rake or a vacuum cleaner, and then set $\boldsymbol{p}_{self}$ on the tool to control it. We used exactly the same controller as that used in the previous arm reaching experiment, except that the joints included in the dense Jacobian (and consequently the joints to be controlled) were different. A joint angle posture that facilitates holding the tool was sent to the robot, and after the experimenter placed the tool in the robot's hands, the control process was initiated.

The robot held the rake with both hands as shown in Fig. 5. The rake is fixed to the right hand, while the left hand can freely slide along shaft of the rake. The joints shoulder-p and elbow-p of the left arm, and elbow-p of the right arm were used. In the experiment with the vacuum cleaner, the shoulder-p, elbow-p, and wrist-p of the left arm were used. As the head of the vacuum cleaner moves easily back and forth than in the other directions, the target points were also set along that line.

The experimental results are shown in Fig. 8 and Fig. 9. In both experiments, we can see that $\boldsymbol{p}_{target}$ converges to the target point $\boldsymbol{p}_{target}$ after the target is updated.

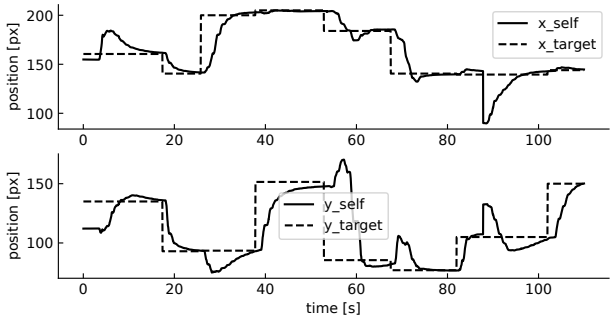


Fig. 7: Result of reaching experiment for the DIJE-based controller, shown in image coordinates. Visualized results shown in Fig. 6.

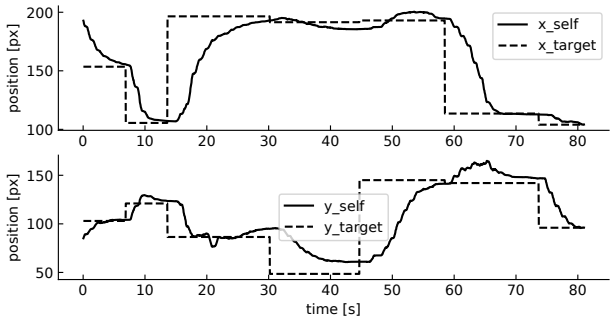


Fig. 8: Result of rake tool-tip control experiment control using dense image Jacobian estimation, shown in image coordinates.

VI. Discussion

In the experiments, DIJE was able to estimate the dense image Jacobian J^D appropriately and applied to the self recognition and visual servoing tasks. As can be seen in the visualization of the processing pipeline in Fig. 2, when there is background movement optical flow is detected that is of

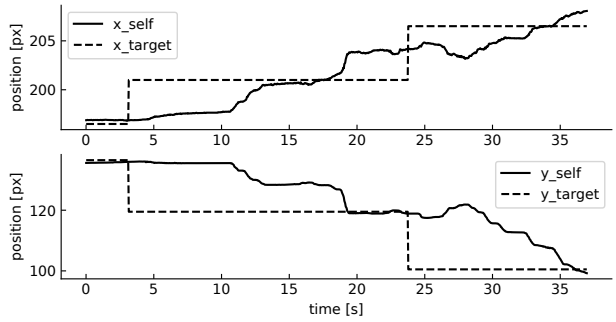


Fig. 9: Result of vacuum tool-tip control experiment using dense image Jacobian estimation, shown in image coordinates.

similar or greater magnitude than the optical flow from the robot's movement. The DIJE process itself does not try to distinguish between robot-induced and external movement, and as visible in Figs. 2 and 4, the dense image Jacobian is assigned to the external movement as well. Nonetheless, since the self recognition algorithm evaluates each cluster of the dense image Jacobian based on how consistent the value is across time, it can correctly identify which part of the image belongs to the robot. We note however that this method of self recognition does not allow for recognition of non-moving parts of the robot, and it can be seen that the robot's shoulder at the lower left, which is static, is not identified as part of the robot.

The self recognition algorithm based on DIJE can also keep a consistent label even when the background movement overlaps with the robot, as seen in Fig. 4. As previous dense self recognition algorithms were based on motion magnitude detection and did not consider the direction of motion as DIJE does, they could not distinguish when the external motion overlaps with the robot. As the representation of DIJE is higher dimension than the optical flow, it is more feature-rich and enables improved segmentation.

The markerless visual servoing controller using DIJE was also verified in an arm manipulation and tool-tip control experiment. The robot started with DIJE initialized with zero elements, and was able to quickly learn how to move the arm to the target point. Interestingly, in the rake tip control experiment we saw an emergent bimanual movement, in which by looking at only the tool tip, the robot recognizes how each joint movement relates to its movement. Although not the objective of this study, the resulting bimanual movement looks very natural and humanlike.

VII. Conclusions

In this research, we propose the concept of a dense image Jacobian, to acquire a global representation of the robot's visuomotor coordination. It is based on the dense optical flow estimation and a simplified Kalman Filter formulation, which does not rely on markers nor *a priori* knowledge of the robot's structure and can be computed in real time. The resulting high-dimensional feature has the potential to be used in a variety of recognition and control tasks, for which in this paper we propose a self-recognition algorithm and markerless visual servoing controller.

We believe that this dense visuomotor policy representation can be extended to be applied to dexterous manipulation tasks, as they can encode the relationship between the robot hand and the manipulated object in real time. One drawback of the current approach is the use of optical flow, which is purely a 2-frame image processing algorithm and does not have object constancy, making the currently method not applicable when the robot's body goes out of frame, is rotated, deformed excessively, or is occluded. Thus, we consider combining this method with a dense visual representation which can consistently generate representations invariant to deformations [18], [19], to create a more general formulation of the visuomotor policy.

Further, the proposed method does not cosider depth at any point in the algorithm, and cannot distinguish errors in the depth direction. Thus, all the presented reaching experiments work in 2D, and no depth target is given. This may be resolved by calculating the optical flow in the depth direction as well, by integrating depth camera measurements or using DL-based optical flow calculations that calculate the 3D optical flow [20], or by introducing an additional camera slightly offset from the first camera that can create a parallax effect encoding depth differences.

References

[1] K. Hosoda and M. Asada, "Versatile visual servoing without knowledge of true jacobian," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, vol. 1, Sept. 1994, pp. 186–193 vol.1.

[2] S. Mao, X. Huang, and M. Wang, "Image jacobian matrix estimation based on online support vector regression," *Int. J. Adv. Rob. Syst.*, vol. 9, no. 4, p. 111, Oct. 2012.

[3] X. Lv and X. Huang, "Fuzzy adaptive kalman filtering based estimation of image jacobian for uncalibrated visual servoing," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006, pp. 2167–2172.

[4] A. M. Farahmand, A. Shademan, and M. Jagersand, "Global visual-motor estimation for uncalibrated visual servoing," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2007, pp. 1969–1974.

[5] J. Qian and J. Su, "Online estimation of image jacobian matrix by Kalman-Bucy filter for uncalibrated stereo vision feedback," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, May 2002, pp. 562–567 vol.1.

[6] V. Florence, J. J. Corso, and B. Griffin, "Robot-Supervised learning for object segmentation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1343–1349.

[7] J. Lambrecht, "Robust Few-Shot pose estimation of articulated robots using monocular cameras and Deep-Learning-based keypoint detection," in *2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA)*, Nov. 2019, pp. 136–141.

[8] C. Rauch, T. Hospedales, J. Shotton, and M. Fallon, "Visual articulated tracking in the presence of occlusions," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 643–650.

[9] F. Widmaier, D. Kappler, S. Schaal, and J. Bohg, "Robot arm pose estimation by pixel-wise regression of joint angles," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 616–623.

[10] G. Metta and P. Fitzpatrick, "Early integration of vision and manipulation," in *Proceedings of the International Joint Conference on Neural Networks, 2003.* IEEE, 2004.

[11] P. Michel, K. Gold, and B. Scassellati, "Motion-based robotic self-recognition," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sept. 2004, pp. 2763–2768 vol.3.

[12] C. Kemp and A. Edsinger, "What can i control?: The development of visual categories for a robot's body and the world that it influences," *Proceedings of the Fifth International Conference*, 2006.

[13] K. Kawaharazuka, S. Makino, K. Tsuzuki, M. Onitsuka, Y. Nagamatsu, K. Shinjo, T. Makabe, Y. Asano, K. Okada, K. Kawasaki, and M. Inaba, "Component modularized design of musculoskeletal humanoid platform musashi to investigate learning control systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 7300–7307.

[14] Y. Asano, T. Kozuki, S. Ookubo, M. Kawamura, S. Nakashima, T. Katayama, I. Yanokura, T. Hirose, K. Kawaharazuka, S. Makino, Y. Kakiuchi, K. Okada, and M. Inaba, "Human mimetic musculoskeletal humanoid kengoro toward real world physically interactive actions," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov. 2016, pp. 876–883.

[15] K. Kawaharazuka, S. Makino, M. Kawamura, A. Fujii, Y. Asano, K. Okada, and M. Inaba, "Online self-body image acquisition considering changes in muscle routes caused by softness of body tissue for tendon-driven musculoskeletal humanoids," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1711–1717.

[16] K. Kawaharazuka, S. Makino, M. Kawamura, Y. Asano, K. Okada, and M. Inaba, "Online learning of Joint-Muscle mapping using vision in Tendon-Driven musculoskeletal humanoids," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 772–779, Apr. 2018.

[17] G. Farnebäck, "Two-Frame motion estimation based on polynomial expansion," in *Image Analysis*. Springer Berlin Heidelberg, 2003, pp. 363–370.

[18] P. O. O. Pinheiro, A. Almahairi, R. Benmalek, F. Golemo, and A. C. Courville, "Unsupervised learning of dense visual representations," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 4489–4500.

[19] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," June 2018.

[20] Z. Teed and J. Deng, "RAFT-3D: Scene flow using Rigid-Motion embeddings," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 8371–8380.