A Vision-Based Tactile Sensor with In-Sensor Computing Paradigm for Seamless Tactile Sensing and Perception

I. Supplementary Notes

A. Supplementary Note 1

Compared with the challenges faced by traditional VBTSs, biology offers a compelling reference model. Human skin spans approximately 1.5-2 m^2 [1] and contains millions of mechanoreceptors, yet tactile response times average around 155 ms, comparable to auditory stimuli (140-160 ms) and even faster than visual stimuli (180-200 ms) [2], [3]. About 30,000 mechanoreceptors are distributed across the hand (241/cm^2 in the fingertips versus 58/cm^2 in the palm) [4]. Among them, slow-adapting receptors (SA-I Merkel cells and SA-II Ruffini endings) encode spatial features such as shape, texture, and stretch, whereas fast-adapting receptors (FA-I Meissner and FA-II Pacinian corpuscles) capture temporal features such as vibration and slip. Mechanoreceptors transduce spatiotemporal stimuli into action potentials, in a manner analogous to an ADC stage, and these signals are then transmitted via the dorsal-column-medial-lemniscal (DCML) pathway to the spinal cord and central nervous system (CNS). Crucially, tactile information is not processed from scratch at the CNS level: primary sensory neurons in the terminal phalanx already extract key features such as contact curvature and force direction from spike patterns [5], [6]. This early-stage neural encoding drastically reduces the CNS computational burden by sending only a small set of high-quality, pre-processed features for higher-level inference. In grip-force regulation, for example, humans respond to load perturbations within 70 ms [7], [8], which is faster than voluntary cortical responses (140 ms) and comparable to reflex withdrawal triggered by mechanical stress [9], [10]. Together, these findings suggest that raw tactile signals undergo efficient local pre-processing before reaching the CNS. Inspired by this biological model, we explore how such tactile pre-processing can be transferred to VBTSs via *in-sensor computing*, rather than conventional far- or near-sensor approaches. In-sensor computing seamlessly integrates sensing and perception rather than the simple hardware integration, thereby eliminating the traditional separation between data acquisition and processing.

Indeed, in other tactile sensing modalities, near- or even in-sensor approaches have already been explored. Examples include capacitive arrays [11]–[13], mechanoluminescent devices [14], and resistive sensors with adaptive signal compression [15]. These studies demonstrate the feasibility of performing low-level operations, such as edge detection, compression, or multimodal pattern recognition, directly at or near the sensing site. Nevertheless, practical tactile systems also require higher-level sensing and perception capabilities with broader multifunctionality, and existing MEMS-based tactile sensors [11]–[15] still face challenges in this regard because of their limited computational resources. By comparison, VBTSs offer stronger potential for high-level in-sensor tactile processing, owing to recent advances in vision chips. For instance, Eye-RIS [16], [17], Aistorm Mantis2 [18], and high-speed vision chips [19] all demonstrate pixel-parallel processing and on-chip intelligence. Material-based devices such as memristor photodetectors further extend this concept by combining optical detection with adaptive synaptic weights for in-situ learning [20]. These emerging vision chips function not only as image sensors, but also as computational substrates whose processing capability can exceed that of current MEMS-based in-sensor tactile devices. This development makes in-sensor computing for VBTSs increasingly feasible.

Within VBTS research itself, most existing systems still follow the far-sensor computing paradigm. Classical dense-image VBTSs, such as GelSight-mini [21], DIGIT [22], OmniTact [23], GelSight Wedge [24], ViTacTip [25], 9DTac [26], Tac3D [27], DelTact [28], and VisTac [29], rely on conventional RGB cameras to capture full tactile images and then transmit them to an external host for perception and inference. More recently, another important far-sensor branch has emerged based on sparse event outputs. Representative examples include the event-based tactile systems reported by Kumagai *et al.* [30], Naeini *et al.* [31], Huang *et al.* [32], NeuroTac [33], Evetac [34], E-bts [35], GelEvent [36], EVBTS [37], and the recent design by Khairi *et al.* [38]. Compared with frame-based VBTSs, these neuromorphic systems reduce temporal redundancy by emitting only asynchronous brightness-change events, thereby improving responsiveness and lowering transmission burden for highly dynamic contacts such as vibration, slip, or transient impact. However, despite this more efficient sensing format, their perception pipeline still remains externally hosted: the event streams must be transmitted off-sensor and interpreted by non-mainstream models on a separate computing unit, such as spiking neural networks (SNNs), specifically designed for spike data. Therefore, these methods alleviate the far-sensor bottleneck, but do not eliminate the physical separation between tactile sensing and tactile perception.

Only a limited number of studies have moved beyond far-sensor computing towards near-sensor computing in VBTSs. DIGIT360 [39] represents an early attempt in this direction by integrating an embedded AI processing module into the tactile sensor unit, enabling local extraction of tactile features rather than continuous transmission of full-resolution image streams.

Feng *et al.* [40] further proposed a wireless VBTS that combines a conventional camera with a low-power embedded processor, allowing real-time multifunctional contact detection through an on-board ResNet without relying on an external computer. Xu *et al.* [41] pushed this direction further by introducing a tactile sensor based on a neuromorphic vision chip and spiking neural network, which can exploit rich temporal dynamics for texture discrimination directly on board. These studies collectively demonstrate that part of the tactile perception pipeline can indeed be moved close to the sensing front-end. Nevertheless, they still fundamentally follow the near-sensor computing paradigm: sensing and computation are packaged within one physical module, but the imager and the processor remain algorithmically and architecturally separated. In DIGIT360 [39] and Feng *et al.* [40], tactile sensing is still first performed by a conventional RGB camera and only subsequently processed by a separate embedded accelerator. In Xu *et al.* [41], although the sensing hardware is more tightly coupled to temporal processing, the spike-based output format remains less compatible with the mainstream image- and feature-based algorithms commonly used in VBTS pipelines, which may constrain functional generalisation and broader adoption. More recently, FasTac [42] implemented depth reconstruction and force estimation on an FPGA located close to the vision sensor, enabling high-speed tactile processing without relying on host-side computation. Similarly, Zhu et al. [43] mapped photometric-stereo and Poisson-based depth reconstruction onto an FPGA for rapid visuotactile perception, while the optical imager and computing hardware still remain physically distinct. These studies collectively demonstrate that part of the tactile perception pipeline can indeed be moved close to the sensing front-end. Nevertheless, they still fundamentally follow the near-sensor computing paradigm: sensing and computation are brought into close physical proximity, but the imager and the processor remain architecturally distinct and communicate through an explicit data interface.

To address the above gap, here we present **PixelTac**, a VBTS designed explicitly under the *in-sensor computing* paradigm. Leveraging an in-situ vision processor (IVP) that unifies on-chip sensing, memory, and computation, PixelTac performs real-time tactile processing directly inside the sensing substrate at pixel-, feature-, event-, and inference-levels. This includes feature enhancement, data denoising, marker tracking, spatiotemporal contact feature inference, and contact force estimation, all without the latency of off-chip computation. Compared with dense-image far-sensor VBTSs, PixelTac avoids continuous transmission of high-bandwidth tactile image streams. Compared with sparse-event far-sensor VBTSs, it does not merely reduce data generation, but further removes the need for an external perception stage. Compared with existing near-sensor VBTSs, PixelTac does not simply place a separate processor next to a camera, but instead collapses sensing and computation into the seamless hardware-software backbone. Moreover, unlike spike-native neuromorphic tactile outputs, PixelTac directly produces practical, pre-processed tactile features that remain compatible with mainstream VBTS algorithms and downstream robotic pipelines. In this sense, PixelTac more closely mirrors the organisational principle of biological touch, where local transduction and early feature extraction occur before central decision-making. It therefore establishes a scalable route towards high-speed, low-latency tactile sensing and perception, and defines a new paradigm for VBTSs grounded in in-sensor computing.

B. Supplementary Note 2

The interaction between the analogue and digital domains in the in-situ vision processor (IVP) can be understood as a staged dataflow coordinated by the arithmetic logic unit (ALU). In a conventional CMOS imaging pipeline, the pixel array mainly performs photoelectric conversion, and the sensed frame is then digitised and transferred to an external processor for subsequent computation. In contrast, the IVP used in PixelTac integrates sensing, local storage, and computation inside the computational pixel matrix (CPM). Each computing unit (CU) contains photosensitive pixels, analogue computing units (ACUs), digital computing units (DCUs), and an ALU. The ACUs store grey-level analogue responses for early image enhancement, whereas the DCUs store one-bit binary states for subsequent logical and morphological operations. The ALU provides the instruction-driven operation path between these register domains.

At the beginning of the pixel-level task, the optical marker pattern is converted by the photosensitive pixels into analogue responses and stored in ACUs after gain adjustment. At this stage, the tactile image is not yet converted into a full digital frame. Instead, the MCU issues local parallel computing to the CPM, and the ALU operates directly on the values stored in ACUs. For example, filtering operations such as sharpening, Sobel edge enhancement, and Gaussian smoothing can be applied to the raw tactile image, and the processed responses are stored back into ACUs. In this way, low-level marker structures are enhanced close to the sensing site before full-frame readout or off-chip processing is required.

The analogue-to-digital interaction occurs when the enhanced analogue response needs to be converted into a binary decision. After analogue-domain enhancement, the ALU applies a thresholding operation by comparing the value held in an ACU with a predefined threshold or threshold mask. The output of this comparison is a one-bit result, which is written into a DCU. Therefore, the thresholding step acts as an implicit one-bit analogue-to-digital conversion (ADC). It does not first digitise the entire raw tactile image through a conventional frame-level or column-parallel ADC pipeline. Instead, it converts only the task-relevant analogue response into the binary representation required by the next processing stage.

Once the data are written into DCUs, the processing proceeds in the digital domain. The ALU then executes global parallel computing operations on binary maps, including dilation, erosion, masking, propagation-related operations, and Boolean logic such as AND and NOT. These digital operations are well suited to marker extraction and denoising because they operate on full-frame binary states in parallel and can suppress background artefacts while preserving high-contrast marker geometry. Thus, the same CPM first uses ACUs and the ALU for analogue-domain feature enhancement, then uses ALU-driven thresholding to transfer selected information into DCUs, and finally uses ALU-supported binary operations for digital-domain denoising and feature extraction.

The event-level task follows the same principle. Consecutive tactile frames are stored in ACUs, and the ALU performs inter-frame subtraction in the analogue domain to obtain a temporal change response. A trigger threshold is then applied so that only pixels with sufficiently large changes are converted into binary event states and stored in DCUs. The MCU subsequently scans this binary event map to obtain sparse spatiotemporal contact events. This example shows that the ALU is not only used for isolated arithmetic operations; rather, it coordinates the transition from analogue sensing and local enhancement to binary decision-making and digital post-processing.

This staged interaction also explains why PixelTac combines analogue and digital processing instead of relying exclusively on either domain. Analogue-domain operations in ACUs allow early tactile-image enhancement to be performed close to the photosensitive pixels with low data movement and high parallelism. However, analogue storage and computation may be affected by noise, temporal decay, and device non-idealities if too many operations are cascaded. PixelTac therefore performs only the necessary early enhancement in the analogue domain and then converts the relevant responses into DCUs for robust binary processing. Through this ACU-ALU-DCU dataflow, the IVP avoids continuous raw-image transmission while preserving the marker and event features required for higher-level tactile perception.

II. Supplementary Figures and Captions

A. Supplementary Figure 1

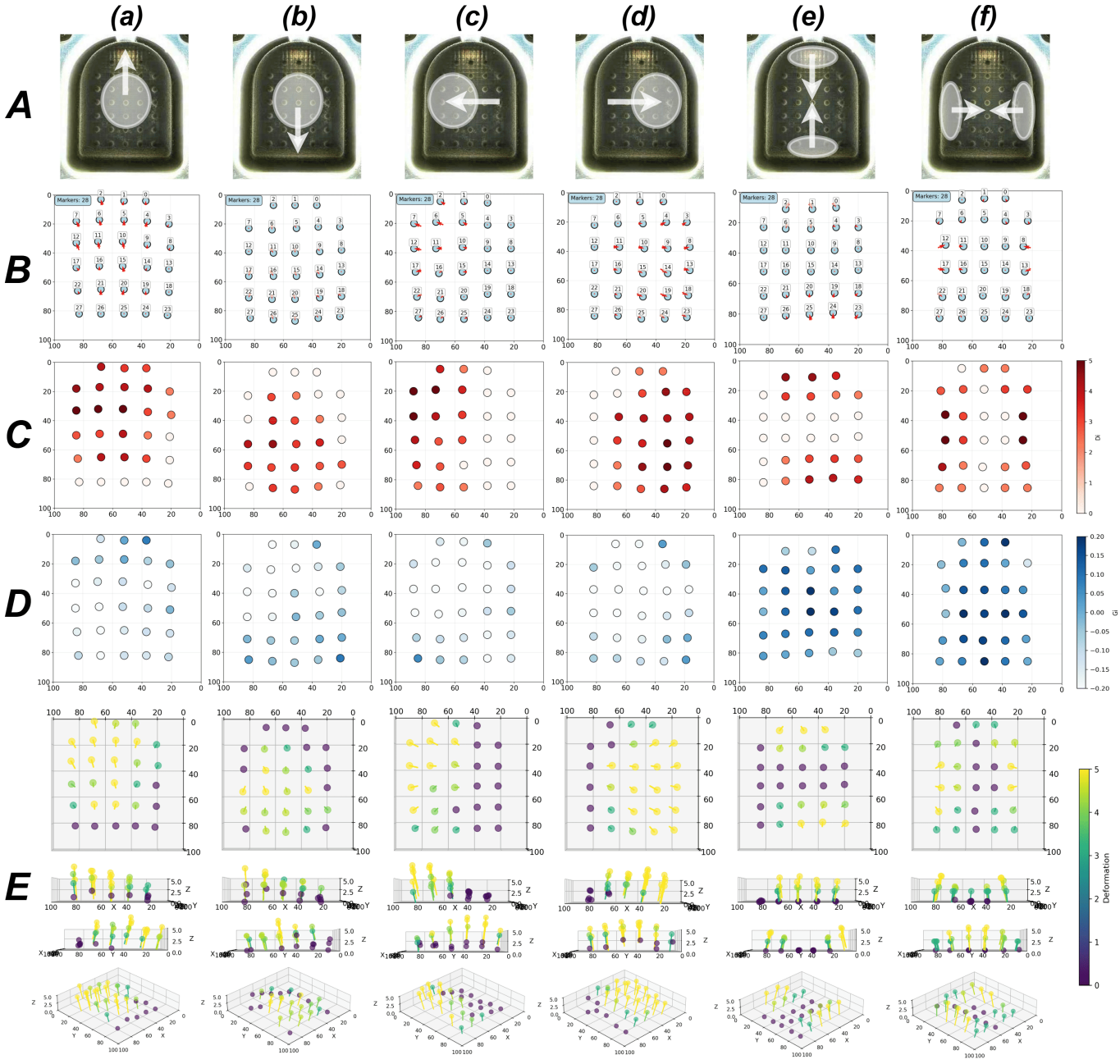

Fig. 1: In-sensor computing spatial contact response of PixelTac. A: Representative contact patterns, including four single-contact directions (a–d) and two opposing-contact configurations (e,f). B: Corresponding marker displacement vectors under each contact pattern. C: Marker displacement magnitude (D_i). D: Marker Gaussian density (G_i). E: Estimated contact deformation visualized from multiple viewing angles, with color indicating the deformation magnitude.

In Fig. 1, six representative spatial contact patterns were employed as case studies to cover different loading directions and contact configurations, including four single-direction contacts and two opposing-contact configurations (Fig. 1(A.a-f)). The marker displacement vectors extracted by the feature-level processing (Fig. 1(B)) provide the directional basis for subsequent spatial contact characterization. The marker displacement magnitude D_i (Fig. 1(C)) further quantifies the intensity and spatial extent of the contact response. The four single-direction contacts produce asymmetric displacement distributions that vary with the loading direction, whereas the two opposing-contact configurations generate broader bilateral responses. The Gaussian density G_i (Fig. 1(D)) provides an additional spatial descriptor associated with the deformation distribution across the sensor surface, distinguishing localized responses under single-direction loading from the broader compressive distributions induced by opposing contacts. Based on these spatial descriptors, the estimated contact deformation field $\vec{V}_i$ can be further reconstructed and visualized from multiple viewing angles (Fig. 1(E)), providing a more intuitive three-dimensional representation of the different contact patterns.

B. Supplementary Figure 2

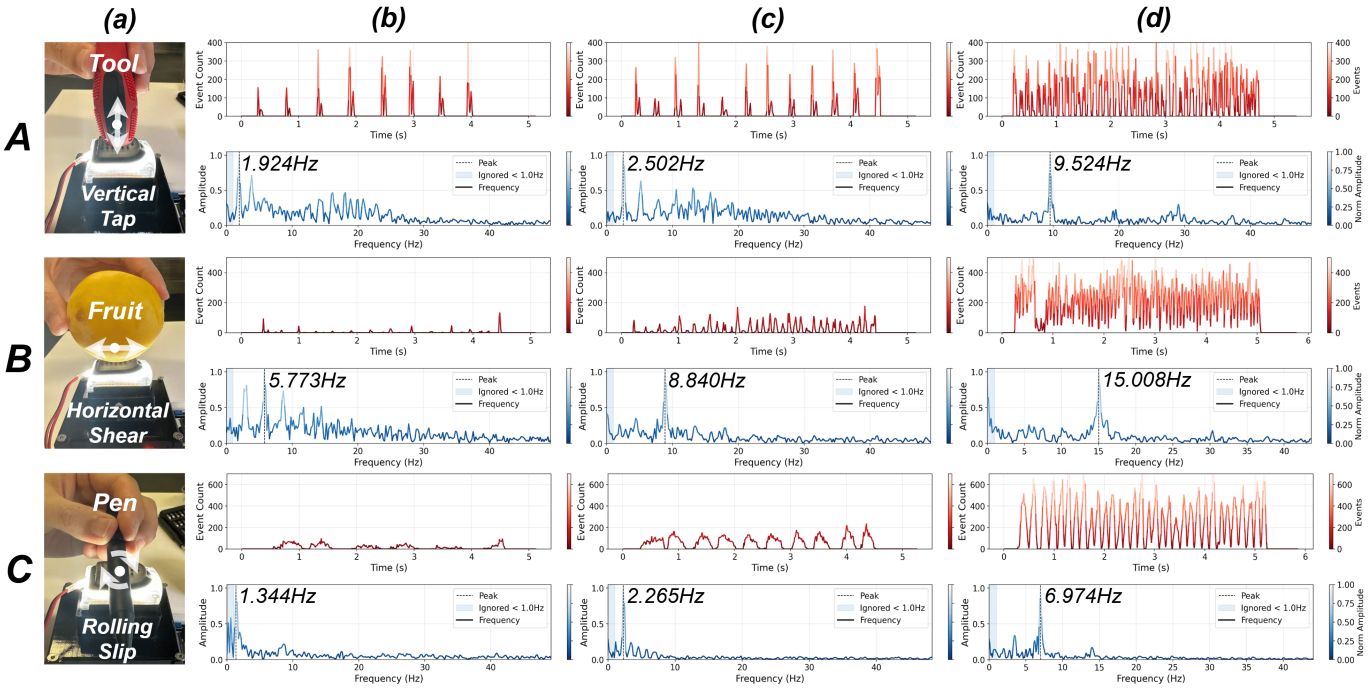

Fig. 2: In-sensor computing temporal contact response of PixelTac. A–C, Temporal responses of PixelTac to three representative dynamic stimuli. A: vertical tap using a tool. B: horizontal shear using a fruit. C: rolling slip using a pen. (a) Photographs of the corresponding stimulation configurations. (b–d) Representative responses under increasing input frequencies from low to high. For each condition, the upper plot shows the temporal variation of the event count, ($n_E(t)$), and the lower plot shows the corresponding frequency spectrum, ($N_E(f)$), with the detected peak frequency annotated.

In Fig. 2, three representative dynamic interactions were applied to PixelTac at increasing input frequencies: vertical tapping using a rigid tool (Fig. 2(A)), horizontal shearing using a soft fruit (Fig. 2(B)), and rolling slip using a cylindrical pen (Fig. 2(C)). The corresponding stimulation configurations are shown in Fig. 2(a), while Fig. 2(b-d) present the responses from low to high input frequencies. For each condition, the temporal event count $n_E(t)$ and its corresponding frequency spectrum $N_E(f)$ are used to characterize the temporal contact response. For vertical tapping, each discrete contact produces a sharp increase in $n_E(t)$, and the number of peaks increases with the tapping rate. Correspondingly, the detected dominant frequency f_{dominant} increases from 1.924,Hz to 2.502,Hz and finally to 9.524,Hz (Fig. 2(A.b-d)). For horizontal shearing of the fruit, relatively lower and more distributed event responses are observed at low and intermediate frequencies, reflecting the softer and more compliant contact interface. As the shearing rate increases, the temporal response becomes progressively denser, with f_{dominant} increasing from 5.773,Hz to 8.840,Hz and 15.008,Hz (Fig. 2(B.b-d)). Rolling and slipping the pen across the sensor surface produces more prolonged event responses, arising from the continuous alternation between frictional contact and local slippage along the curved surface. The corresponding f_{dominant} increases from 1.344,Hz to 2.265,Hz and 6.974,Hz (Fig. 2(C.b-d)), remaining lower than those of the other two dynamic stimuli across the tested conditions.

C. Supplementary Figure 3

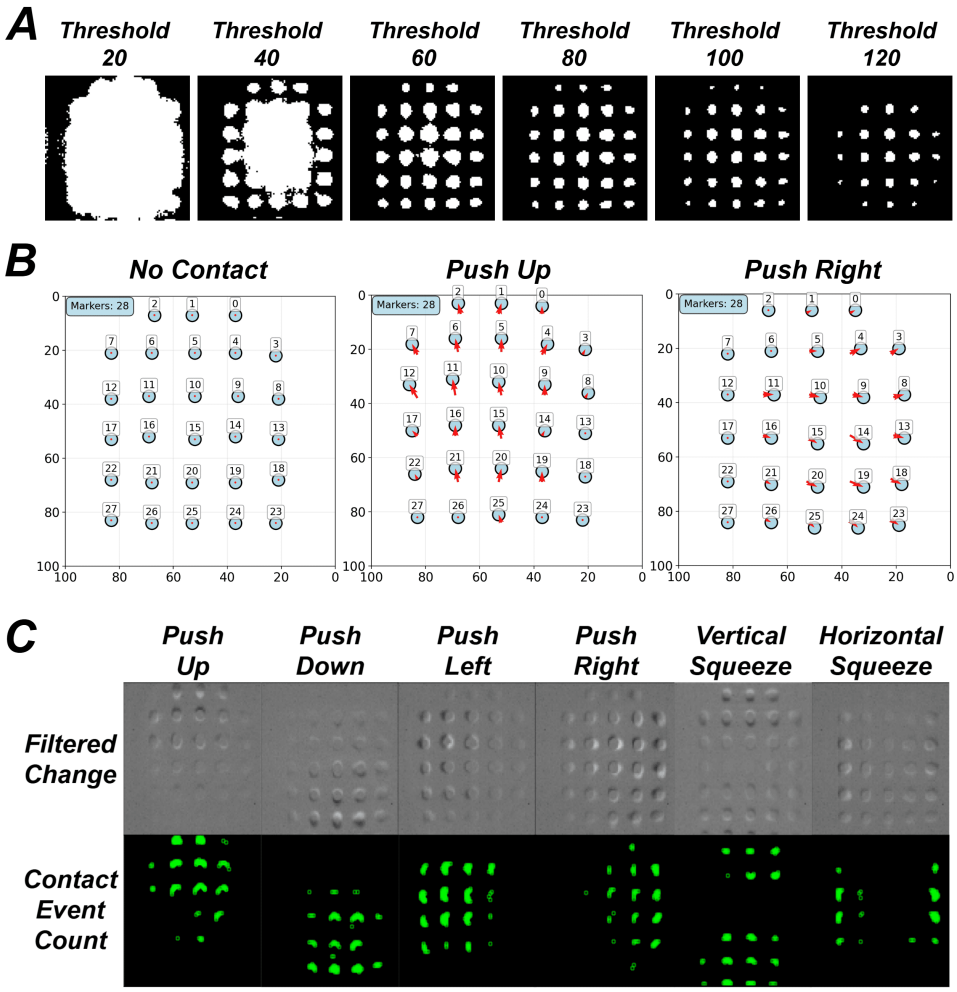

Fig. 3: In-sensor computing characterization of data preprocessing, marker tracking, and contact-event detection. A: Digital-threshold optimisation showing $k_{\text{thred}}^{\text{digital}}$=80 as the best compromise between marker visibility and denoising. B: Marker-tracking results of PixelTac under no contact, vertical stimuli and horizontal stimuli. C: Contact-event distribution consistent with temporal changes in contact stimuli.

In Fig.3(A), we optimised the digital threshold $k_{\text{thred}}^{\text{digital}}$: thresholds below 60 fail to extract central markers, whereas values above 100 over-erode edges and can even eliminate peripheral markers. Thereby, $k_{\text{thred}}^{\text{digital}} = 80$ yields the best compromise between retention and noise suppression, providing a high-quality input for subsequent feature- and inference-level tasks.

In Fig.3(B), the feature-level output of PixelTac can be transferred from the MCU to the host for verification, which maintains stable marker IDs and trajectories under various stimuli relying on in-sensor computing only, producing coherent contact-event distributions. These explicit marker features form the basis for downstream perception algorithms such as spatial contact inference and resultant force estimation.

As shown in Fig.3(C), the resulting contact-event distribution successfully tracks the location and direction of physical stimuli, confirming the effectiveness of event-level task.

D. Supplementary Figure 4

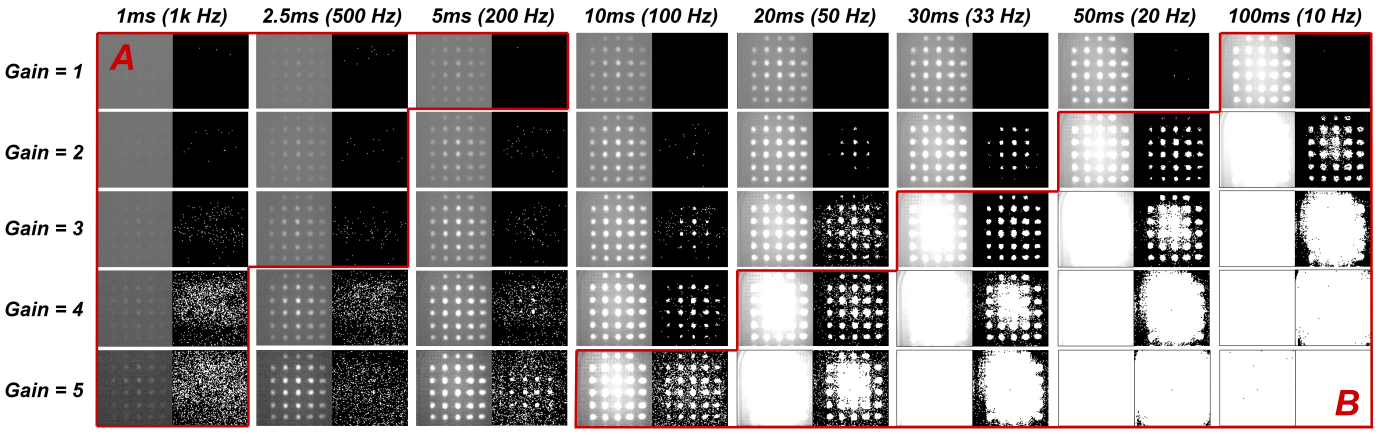

Fig. 4: Raw images and extracted contact events under different temporal parameters. A: too high fps with limited gain causes underexposure, B: whereas too low fps or excessive gain causes overexposure.

E. Supplementary Figure 5

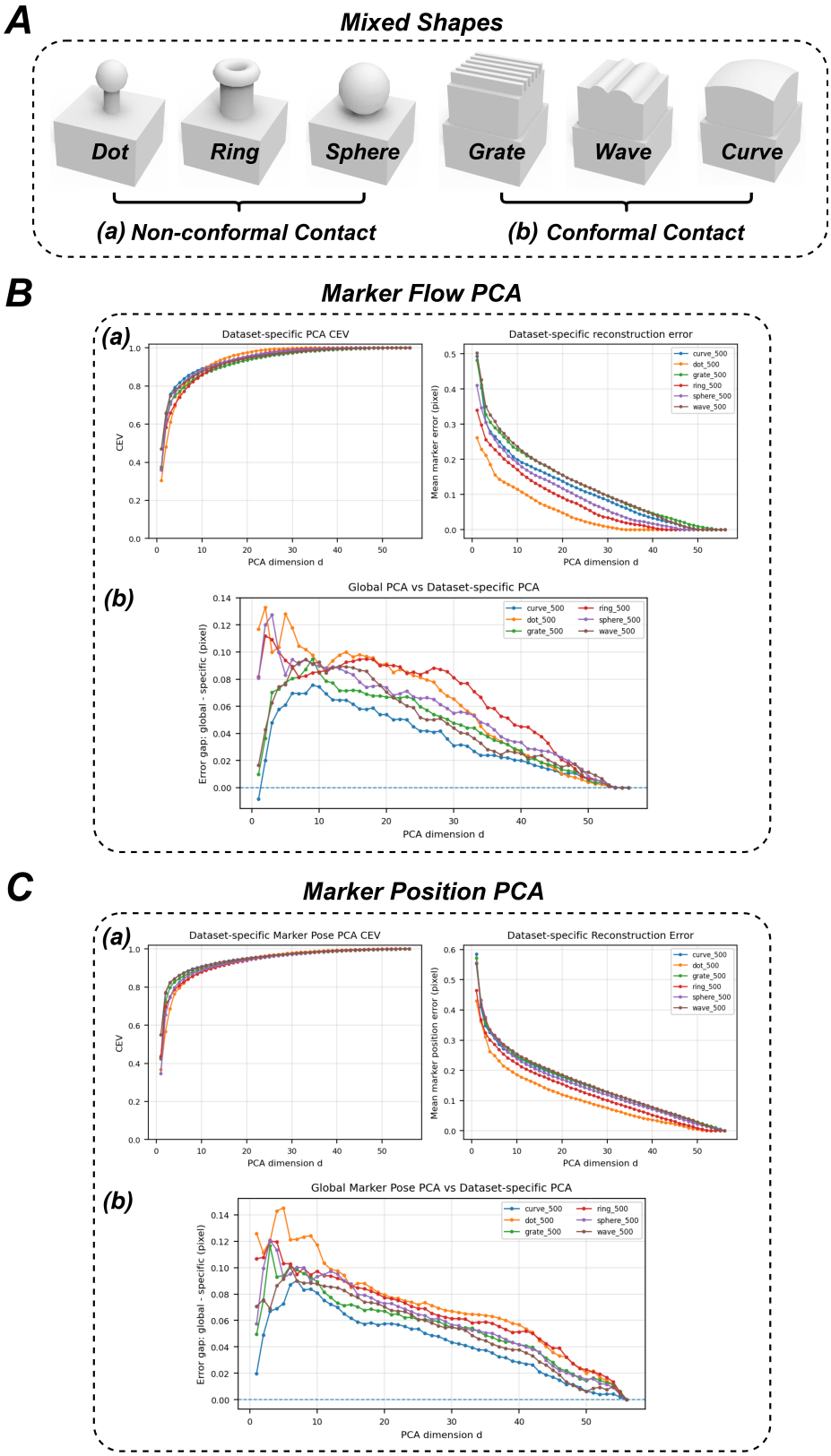

Fig. 5: PCA analysis of marker feature encoding and reconstruction across mixed contact geometries. A: Six 3D-printed indenters used for supplementary force-data collection, covering (a) non-conformal contacts generated by dot, ring and sphere geometries, and (b) conformal contacts generated by curve, grate and wave geometries. B: PCA analysis of marker-flow features, including (a) cumulative explained variance, (b) dataset-specific reconstruction error, and (c) the reconstruction-error gap between global PCA and dataset-specific PCA. C: PCA analysis of marker-position features using the same evaluation protocol.

To further examine the operational boundary of the marker-based force-estimation pipeline proposed by PixelTac, we conducted an additional study using six 3D-printed indenters with distinct contact geometries. As shown in Fig. 5(A.a) and Fig. 5(A.b), these indenters cover both non-conformal contacts, including dot, ring and sphere geometries, and more conformal contacts, including curve, grate and wave geometries. Compared with the cylindrical indenters used for the proof-of-concept validation, these six new indenters cover a wider range of contact geometries and dimensions, with characteristic sizes ranging from 2 mm to 13 mm.

Following the PixelTac force-estimation pipeline, each indenter was used to collect 500 force-labelled contact samples, yielding a total of 3000 samples containing the measured force components, F_x, F_y and F_z, and the corresponding 28-marker trajectories. To introduce more realistic contact variability, the rigid robotic-arm-based loading setup was replaced by manual operation, where PixelTac was held by a human operator during contact. This protocol introduces additional perturbations in contact position, orientation, loading rate and shear components, thereby providing a more challenging dataset for evaluating the robustness of the lightweight inference pipeline. In this section, we focus on the representation-level generality of PCA-compressed marker features, while their influence on force-estimation performance is analysed in the following sections.

Based on this dataset, we compared two tactile feature formats for PCA pre-processing: marker flow, which represents reference-normalised marker displacement, and marker position, which directly uses the absolute marker coordinates. This comparison allows us to examine whether PCA compression remains effective for both displacement-based and coordinate-based tactile representations, and whether the resulting low-dimensional features remain consistent across different contact geometries.

As illustrated in Fig. 5(B.a) and Fig. 5(C.a), both the cumulative explained variance (CEV) and the mean reconstruction error demonstrate that the marker deformation patterns can be represented by a compact low-dimensional PCA basis across all six contact geometries. Reconstruction performance improves consistently as the retained PCA dimension increases. Notably, marker-position PCA shows smaller variation across different indenter geometries than marker-flow PCA, suggesting that the absolute marker-coordinate representation is less sensitive to contact-geometry changes during dimensionality reduction. The reconstruction error is generally lower for non-conformal contacts, such as dot and ring indenters, than for more conformal or spatially extended contacts, such as grate and wave indenters. This is consistent with the fact that non-conformal contacts induce more localised marker deformation, whereas conformal contacts generate broader and more distributed marker displacement fields.

Furthermore, a global PCA basis trained using the mixed-shape dataset was compared with shape-specific PCA bases trained separately for each indenter. As summarised in Fig. 5(B.b) and Fig. 5(C.b), the global PCA basis initially introduces a larger reconstruction-error gap, approximately 0.08-0.12 pixels, at low PCA dimensions, particularly when only 5-10 principal components are retained. However, this gap decreases as the number of retained components increases, especially beyond approximately 10 principal components. At a PCA dimension of 30, for example, the reconstruction-error gap is reduced to an average of approximately 0.06 pixels for both marker-flow and marker-position features. These results indicate that global PCA compression provides a controllable trade-off between reconstruction accuracy and computational efficiency, while retaining sufficient representation-level generality across different contact geometries and tactile feature definitions.

F. Supplementary Figure 6

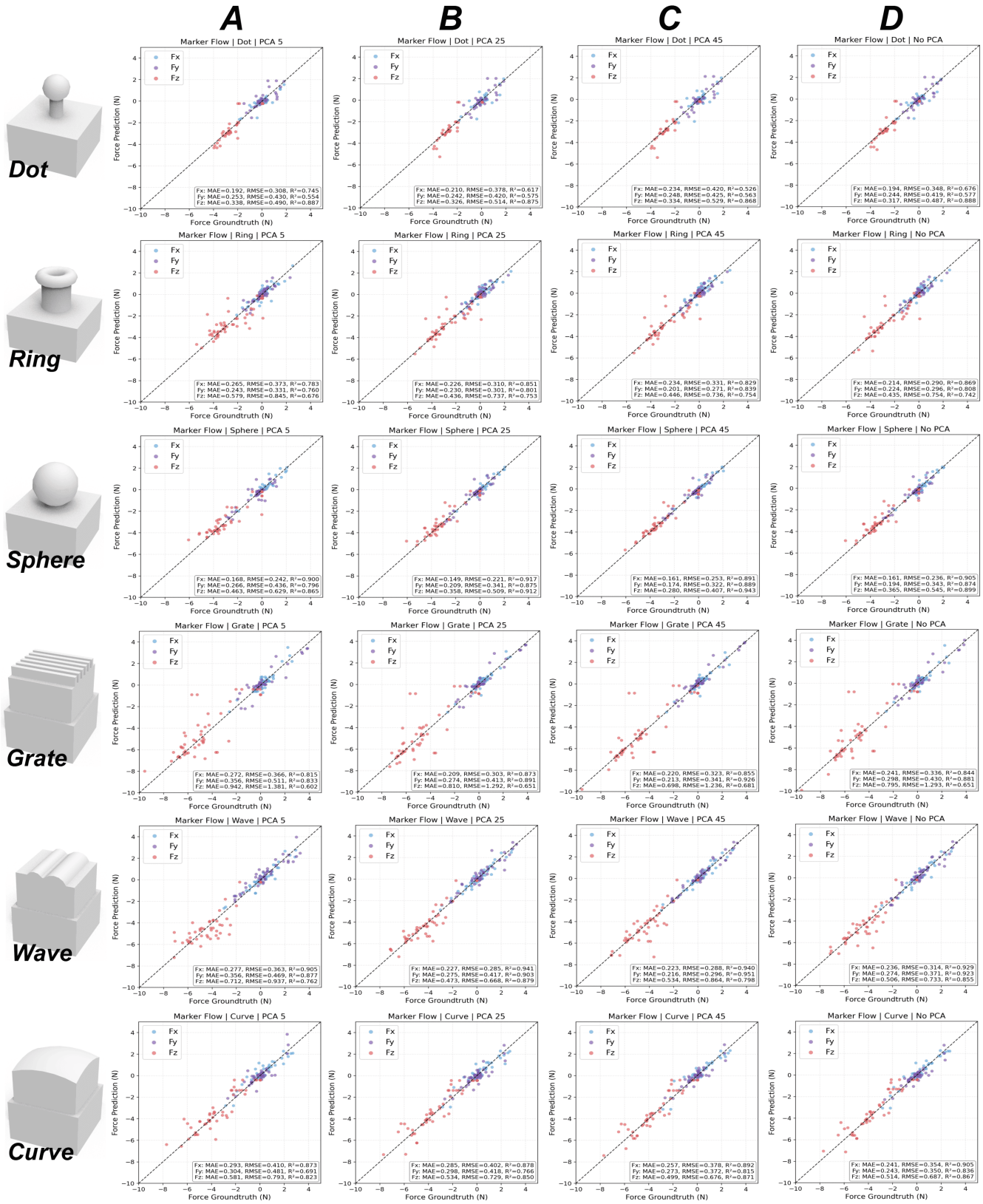

Fig. 6: Force-estimation performance of marker-flow features for individual indenter geometries under fixed MLP capacity. Rows show the six single-shape indenter datasets (Dot, Ring, Sphere, Grate, Wave, and Curve). A-C: Predicted-versus-ground-truth force scatter plots using PCA-compressed marker-flow features with 5, 25, and 45 dimensions. D: Results using raw marker-flow features without PCA compression.

G. Supplementary Figure 7

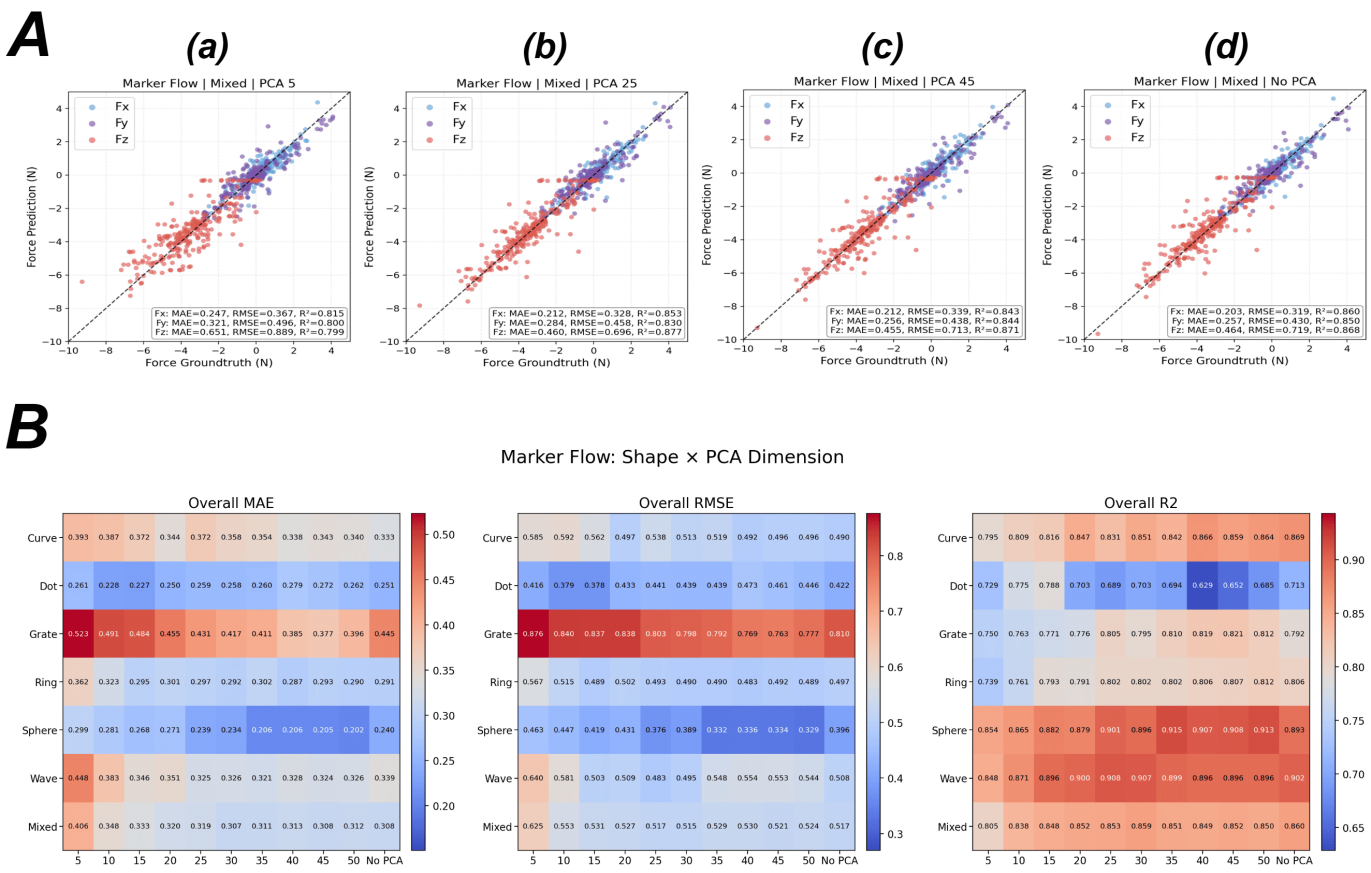

Fig. 7: Force-estimation performance of marker-flow features for the mixed-shape dataset and overall PCA comparison under fixed MLP capacity. A: Predicted-versus-ground-truth force scatter plots for the mixed-shape dataset using (a-c) PCA-compressed marker-flow features with 5, 25, and 45 dimensions and (d) raw marker-flow features without PCA compression. B: Heatmaps summarizing the overall MAE, RMSE, and R^2 across PCA dimensions from 5 to 50 and the no-PCA baseline for the mixed-shape and six single-shape indenter datasets.

H. Supplementary Figure 8

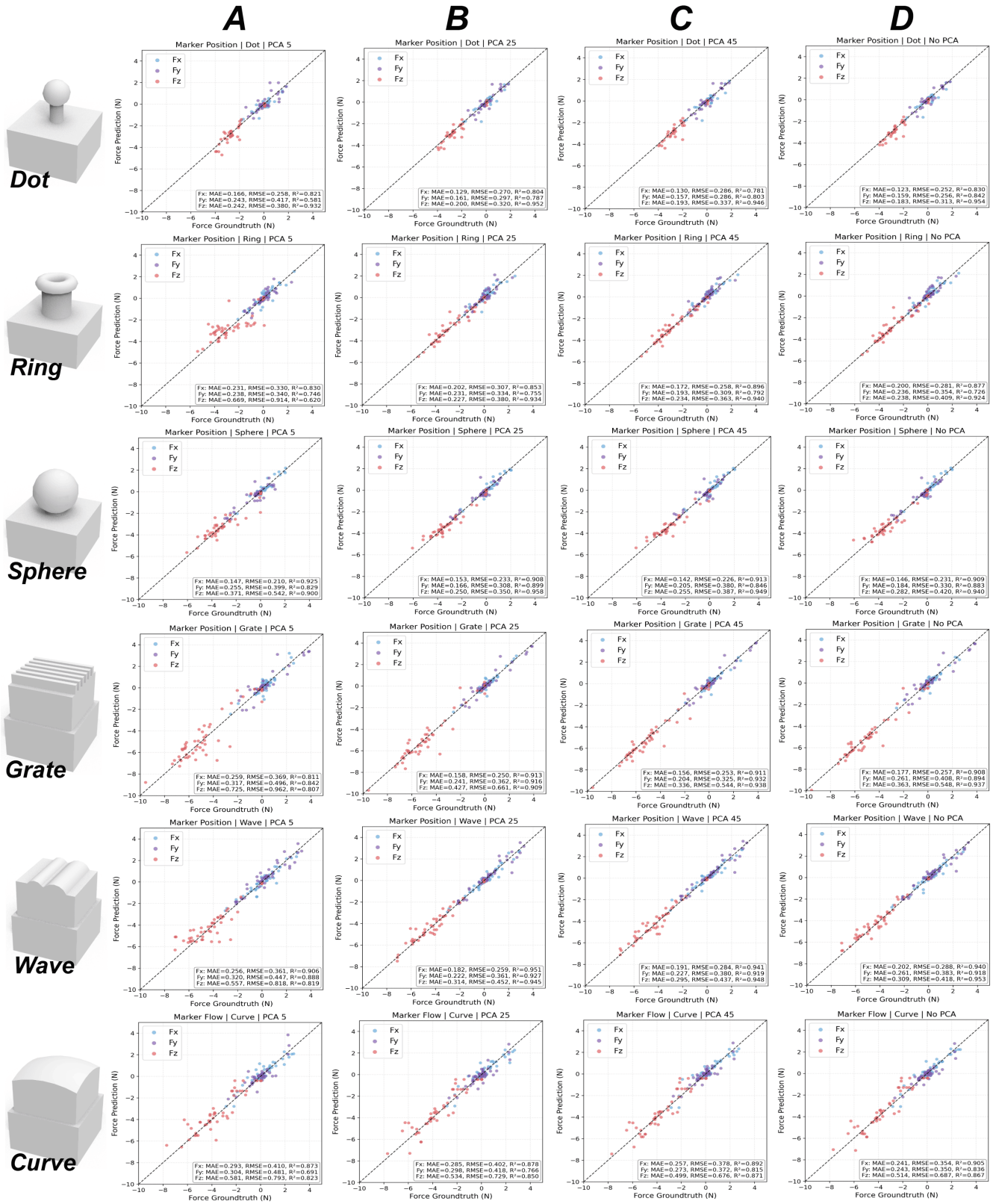

Fig. 8: Force-estimation performance of marker-position features for individual indenter geometries under fixed MLP capacity. Rows show the six single-shape indenter datasets (Dot, Ring, Sphere, Grate, Wave, and Curve). A–C, Predicted-versus-ground-truth force scatter plots using PCA-compressed marker-position features with 5, 25, and 45 dimensions. D, Results using raw marker-position features without PCA compression.

I. *Supplementary Figure 9*

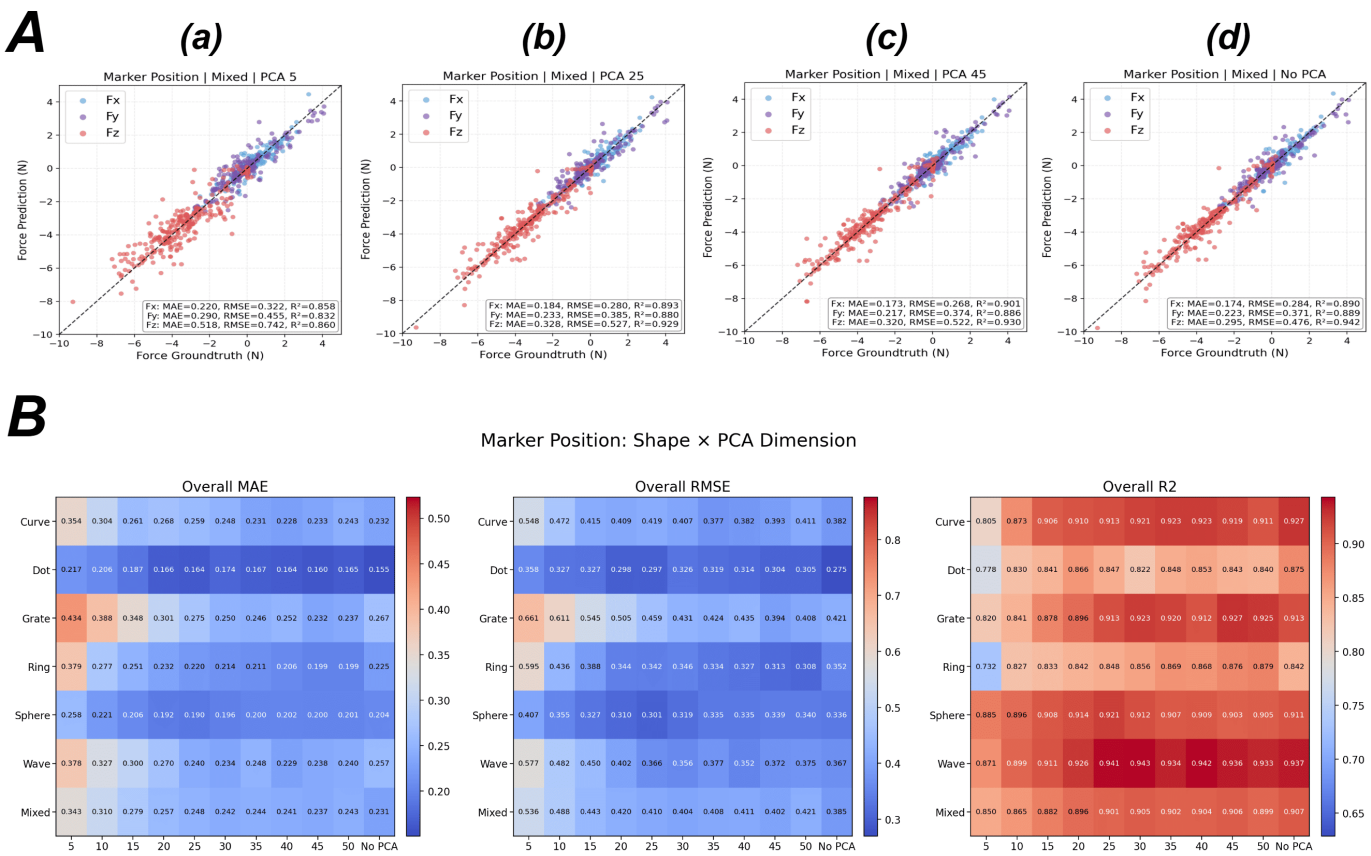

Fig. 9: Force-estimation performance of marker-position features for the mixed-shape dataset and overall PCA comparison under fixed MLP capacity. A, Predicted-versus-ground-truth force scatter plots for the mixed-shape dataset using (a–c) PCA-compressed marker-position features with 5, 25, and 45 dimensions and (d) raw marker-position features without PCA compression. B, Heatmaps summarizing the overall MAE, RMSE, and R^2 across PCA dimensions from 5 to 50 and the no-PCA baseline for the mixed-shape and six single-shape indenter datasets.

J. Supplementary Figure 10

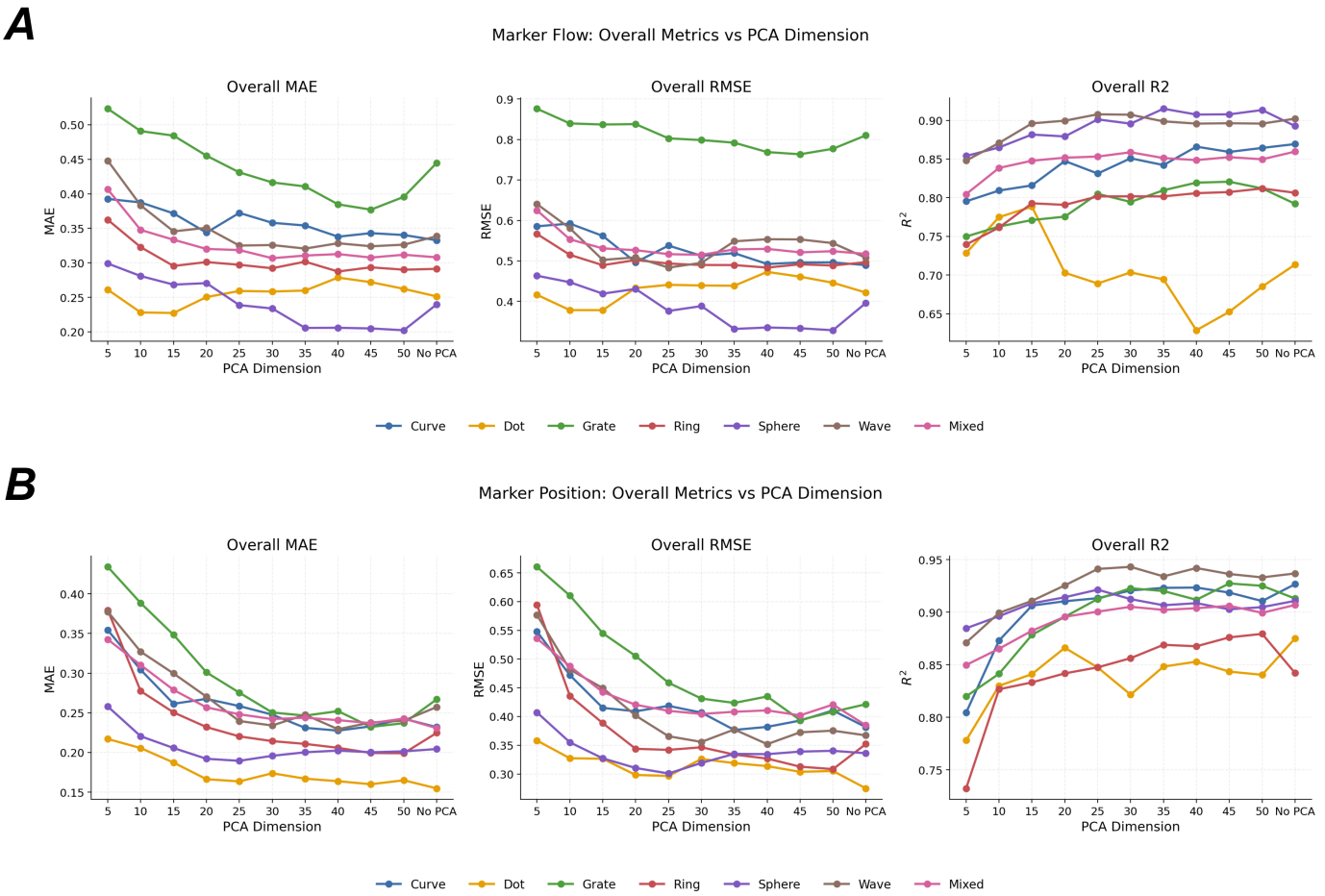

Fig. 10: Influence of PCA dimension on force-estimation performance under fixed MLP capacity. A: Overall MAE, RMSE and R^2 obtained using marker-flow features. B: Overall MAE, RMSE and R^2 obtained using marker-position features. Each curve corresponds to either the mixed-shape dataset or one of the six individual indenter datasets.

After verifying the representation-level robustness of PCA-compressed marker features across complex contact states, we further examined whether such compact representations remain effective for force estimation across different indenter geometries. In this analysis, the MLP capacity was fixed at $(H_1, H_2) = (128, 64)$, while the number of retained PCA components was varied from 5 to 50, with the no-PCA setting used as a reference baseline. This setting isolates the effect of PCA dimensionality from model-capacity variations and allows us to assess whether a low-dimensional marker representation is sufficient for estimating the three-axis contact force components, F_x, F_y, and F_z, across the mixed-shape and six individual indenter datasets.

For marker-flow features, the predicted-versus-ground-truth scatter plots in Supplementary Fig. 6 show the results for representative PCA dimensions of 5, 25, and 45, together with the no-PCA baseline. The rows correspond to the mixed-shape dataset and the six individual indenter datasets. Across these conditions, the lightweight MLP recovers the overall force trend for all three force components. The shear-force components, F_x and F_y, are mainly distributed within approximately ± 5 N, whereas the normal-force component, F_z, is primarily distributed between 0 and -10 N. Apart from a small number of outliers, most predictions remain close to the diagonal reference line, with the distributions generally becoming more compact as the PCA dimension increases. This indicates that the PixelTac force-estimation model can accommodate a broader range of contact geometries beyond the two-cylinder proof-of-concept setting.

The influence of PCA dimensionality is further quantified in Supplementary Fig. 7. Increasing the PCA dimension generally reduces MAE and RMSE and improves R^2, particularly when moving from PCA 5 to moderate PCA dimensions. The improvement becomes less pronounced at higher dimensions, indicating that the dominant force-relevant displacement modes can be captured by a compact PCA basis. For the mixed-shape marker-flow input, approximately 30 principal components already approach the no-PCA baseline, suggesting that retaining the full marker-flow representation is not strictly necessary for accurate force inference. The results also reveal geometry-dependent behaviour: localized contacts, such as the dot and sphere

indenters, generally yield lower absolute errors, whereas spatially extended contacts, such as the grate and wave indenters, tend to exhibit higher MAE and RMSE.

Compared with the proof-of-concept task reported in the main text, the force-estimation accuracy obtained using marker-flow features on the mixed-shape dataset is lower. For example, the MAE of the shear-force components increases from approximately 0.03–0.04 N to 0.20–0.25 N, while the MAE of the normal-force component increases from approximately 0.3 N to around 0.45 N. This decrease is expected because the new dataset is substantially more challenging: the six indenter geometries generate more diverse marker-flow patterns, the force range is broader, with F_x/F_y increasing from approximately ± 2 N to ± 4 N and F_z extending from approximately -6 N to -10 N, and additional contact and acquisition variability is present during data collection. These factors increase the difficulty of fitting a lightweight MLP model, leading to a moderate reduction in absolute force-prediction accuracy.

Marker-position features exhibit a similar but generally more favourable trend. Supplementary Fig. 8 presents the predicted-versus-ground-truth results for PCA dimensions of 5, 25, and 45 and the no-PCA baseline across the same mixed-shape and individual indenter datasets. Compared with marker-flow inputs, marker-position features generally produce more compact prediction distributions around the diagonal reference line. This trend is quantitatively confirmed by Supplementary Fig. 9, where marker-position inputs achieve lower MAE and RMSE and higher R^2 across most contact geometries and PCA dimensions. For the mixed-shape dataset, the overall MAE decreases from 0.343 at PCA 5 to 0.231 in the no-PCA setting, while the overall RMSE decreases from 0.536 to 0.385 and R^2 increases from 0.850 to 0.907. Moderate PCA dimensions, particularly PCA 30–45, already approach the no-PCA baseline, suggesting that absolute marker-coordinate features preserve useful global spatial information for handling geometry-dependent contact variations. Geometry-dependent differences nevertheless remain, with spatially extended contacts such as the grate and wave indenters generally producing higher absolute errors than more localized contacts.

To provide a direct comparison of contact geometry, PCA dimensionality, and feature type, Supplementary Fig. 10 plots the overall MAE, RMSE, and R^2 as functions of PCA dimension for both marker-flow and marker-position features. For both feature types, MAE and RMSE generally decrease rapidly as the PCA dimension increases from 5 to a moderate range and then gradually saturate. Increasing the dimensionality further, or using the raw no-PCA representation, provides only limited additional benefit and can slightly reduce performance for certain contact geometries, potentially due to redundant or noise-sensitive feature components. Similarly, R^2 generally improves with PCA dimensionality and reaches a relatively stable level once approximately 20–30 principal components are retained. Marker-position features show lower overall errors and more consistent performance across most datasets than marker-flow features, particularly at moderate and high PCA dimensions.

Overall, these results demonstrate that PCA compression does not substantially compromise force-estimation performance when a sufficient number of principal components is retained. Moderate PCA dimensions can approach the performance of the no-PCA baseline, indicating that the lightweight MLP does not require the full high-dimensional marker representation for accurate force inference. These findings support the use of compact tactile representations in computationally constrained PixelTac force-estimation pipelines and further show that marker-position features provide a generally more robust and geometry-stable representation than marker-flow features across the multi-geometry dataset.

K. Supplementary Figure 11

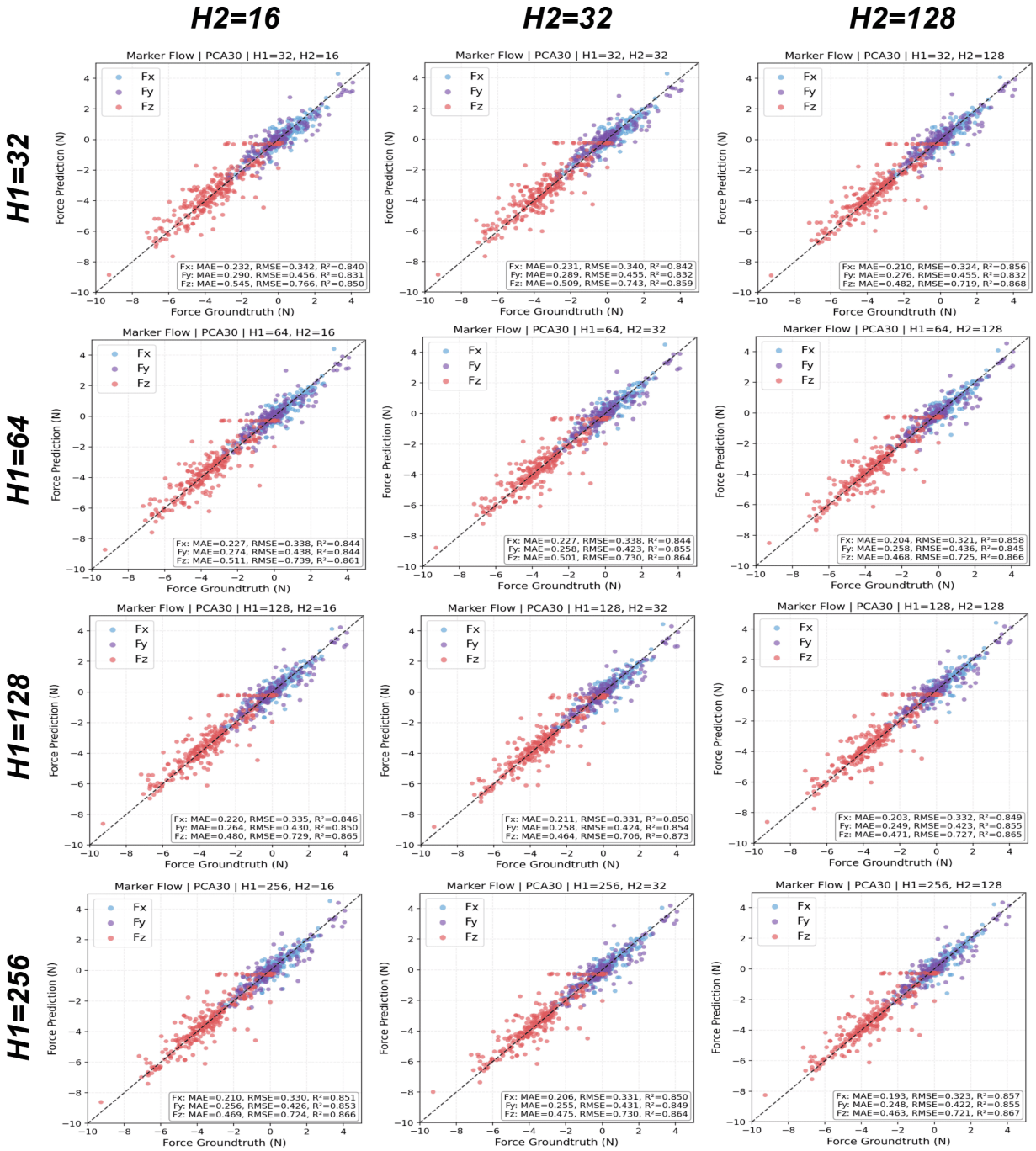

Fig. 11: Influence of MLP capacity on force-estimation performance using PCA-compressed marker-flow features for the mixed-shape dataset. Predicted-versus-ground-truth force scatter plots using PCA 30 features. Rows correspond to first-hidden-layer sizes $H_1 = 32, 64, 128$, and 256, while columns correspond to second-hidden-layer sizes $H_2 = 16, 32$, and 128.

L. Supplementary Figure 12

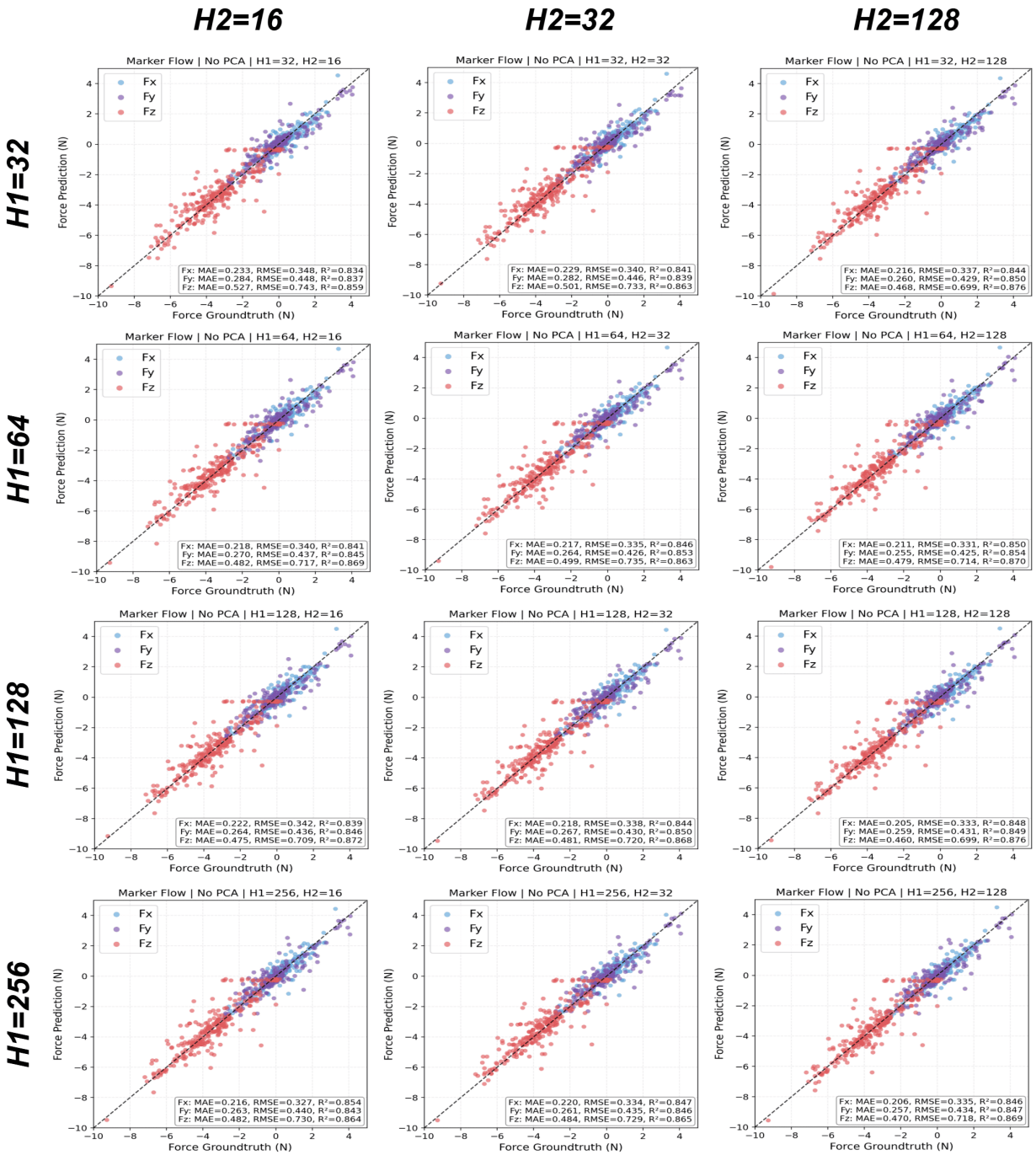

Fig. 12: Influence of MLP capacity on force-estimation performance using raw marker-flow features for the mixed-shape dataset. Predicted-versus-ground-truth force scatter plots using marker-flow features without PCA compression. Rows correspond to first-hidden-layer sizes $H_1 = 32, 64, 128$, and 256, while columns correspond to second-hidden-layer sizes $H_2 = 16, 32$, and 128.

M. Supplementary Figure 13

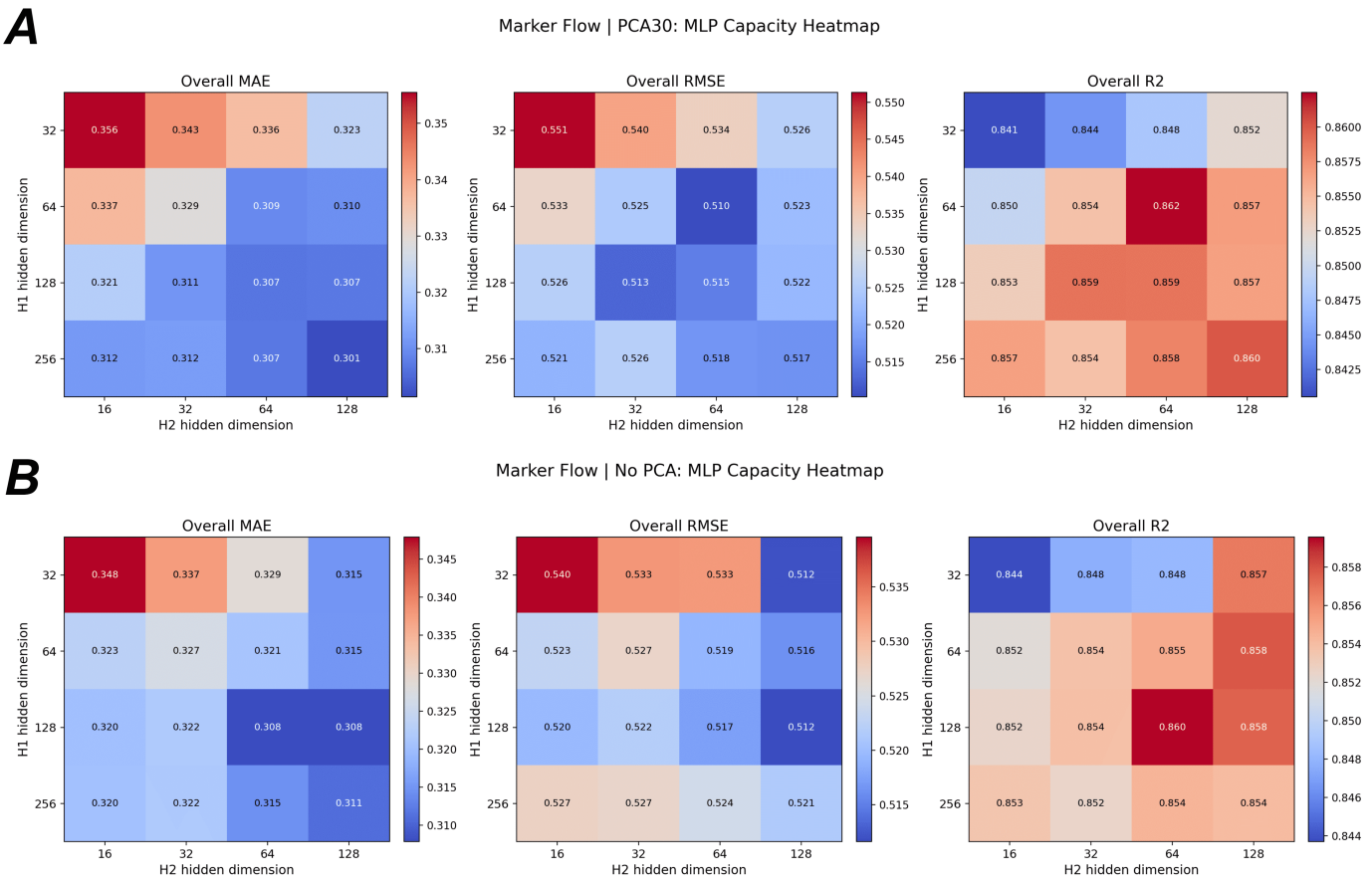

Fig. 13: Influence of MLP capacity on force-estimation performance using marker-flow features for the mixed-shape dataset. A, Heatmaps of overall MAE, RMSE, and R^2 across tested MLP capacities using PCA-compressed marker-flow features with 30 principal components. B, Corresponding heatmaps using raw marker-flow features without PCA compression. Rows indicate first-hidden-layer dimensions H_1, and columns indicate second-hidden-layer dimensions H_2.

N. Supplementary Figure 14

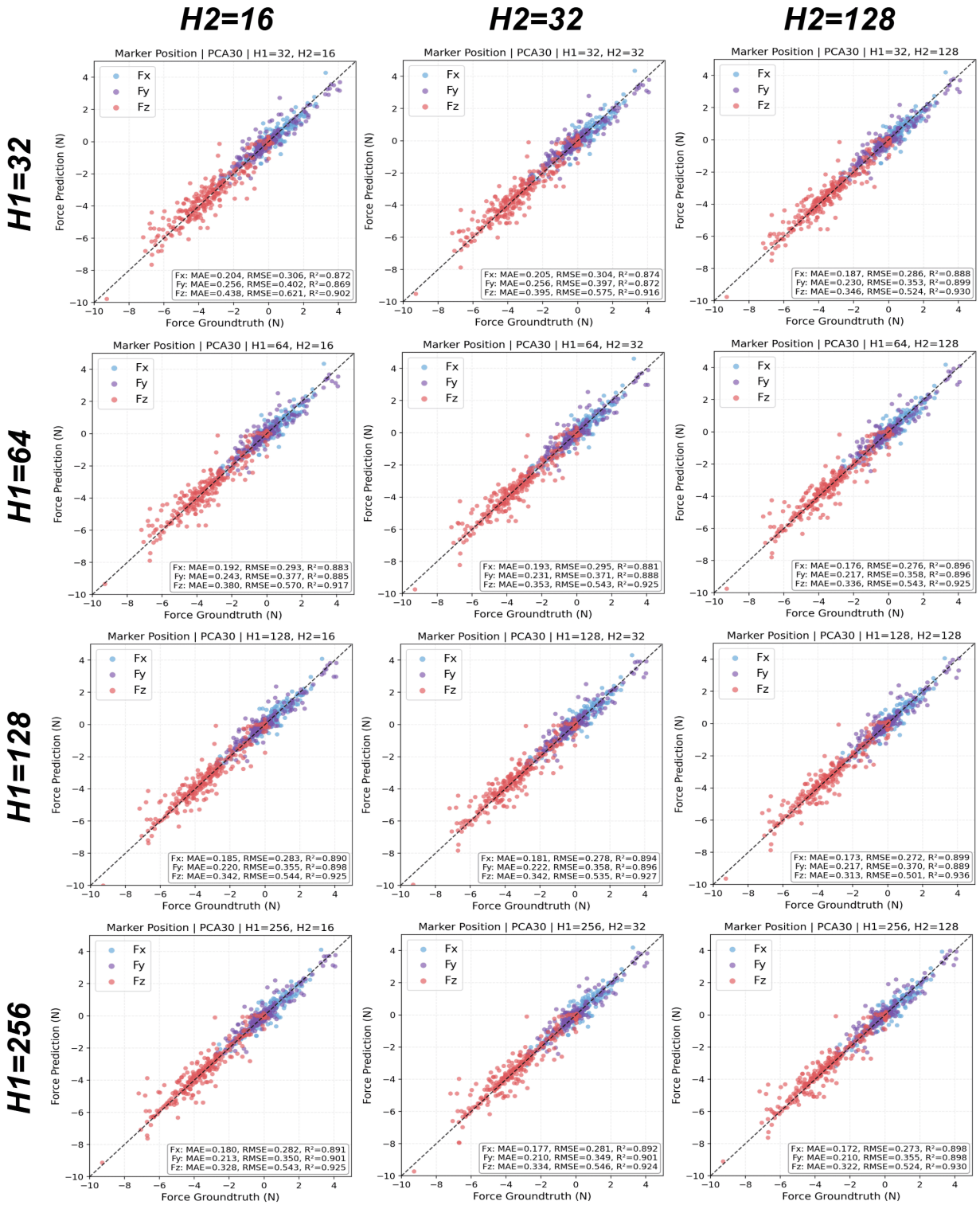

Fig. 14: Influence of MLP capacity on force-estimation performance using PCA-compressed marker-position features for the mixed-shape dataset. Predicted-versus-ground-truth force scatter plots using PCA 30 features. Rows correspond to first-hidden-layer sizes $H_1 = 32, 64, 128$, and 256, while columns correspond to second-hidden-layer sizes $H_2 = 16, 32$, and 128.

O. Supplementary Figure 15

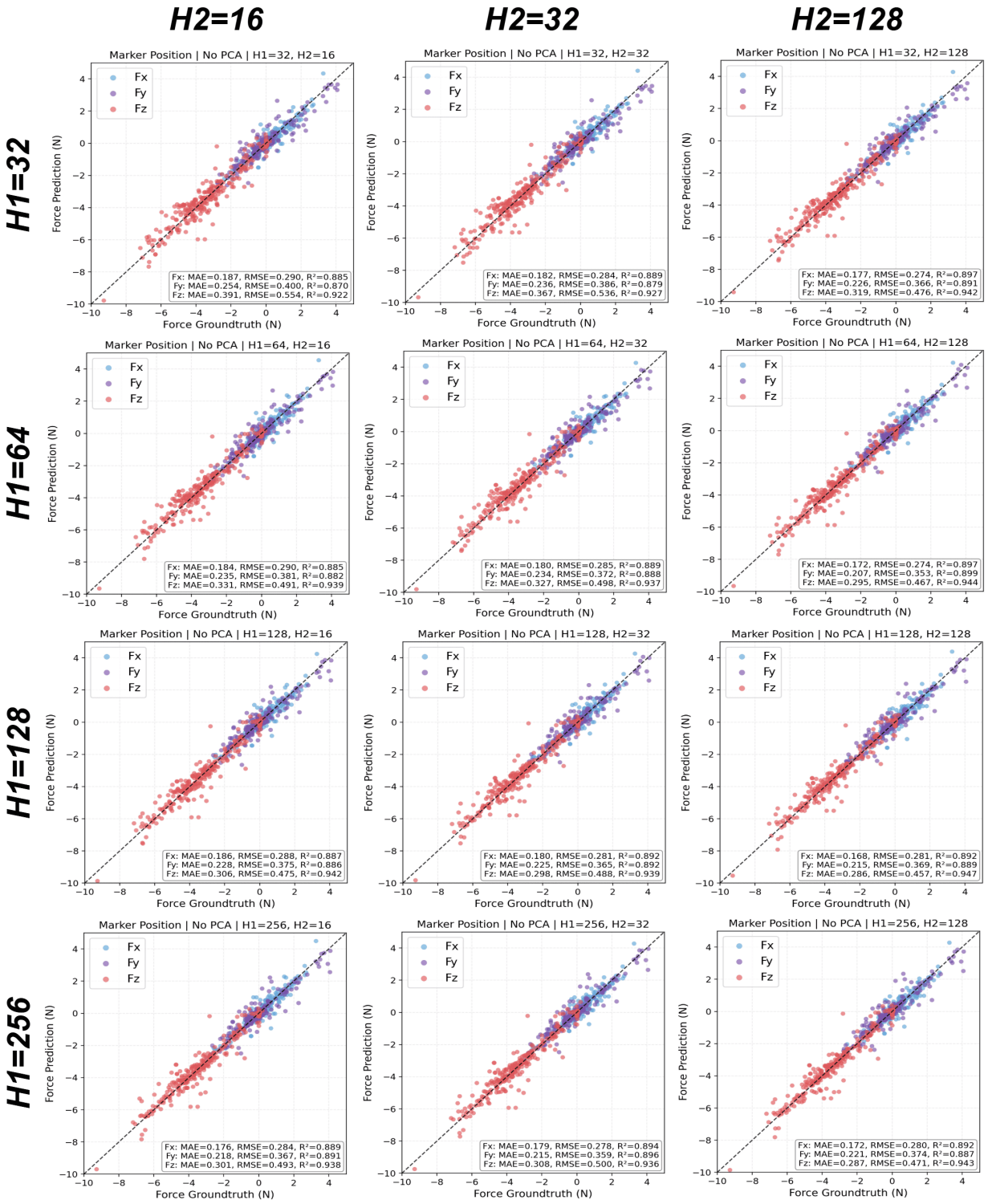

Fig. 15: Influence of MLP capacity on force-estimation performance using raw marker-position features for the mixed-shape dataset. Predicted-versus-ground-truth force scatter plots using marker-position features without PCA compression. Rows correspond to first-hidden-layer sizes $H_1 = 32, 64, 128$, and 256, while columns correspond to second-hidden-layer sizes $H_2 = 16, 32$, and 128.

P. Supplementary Figure 16

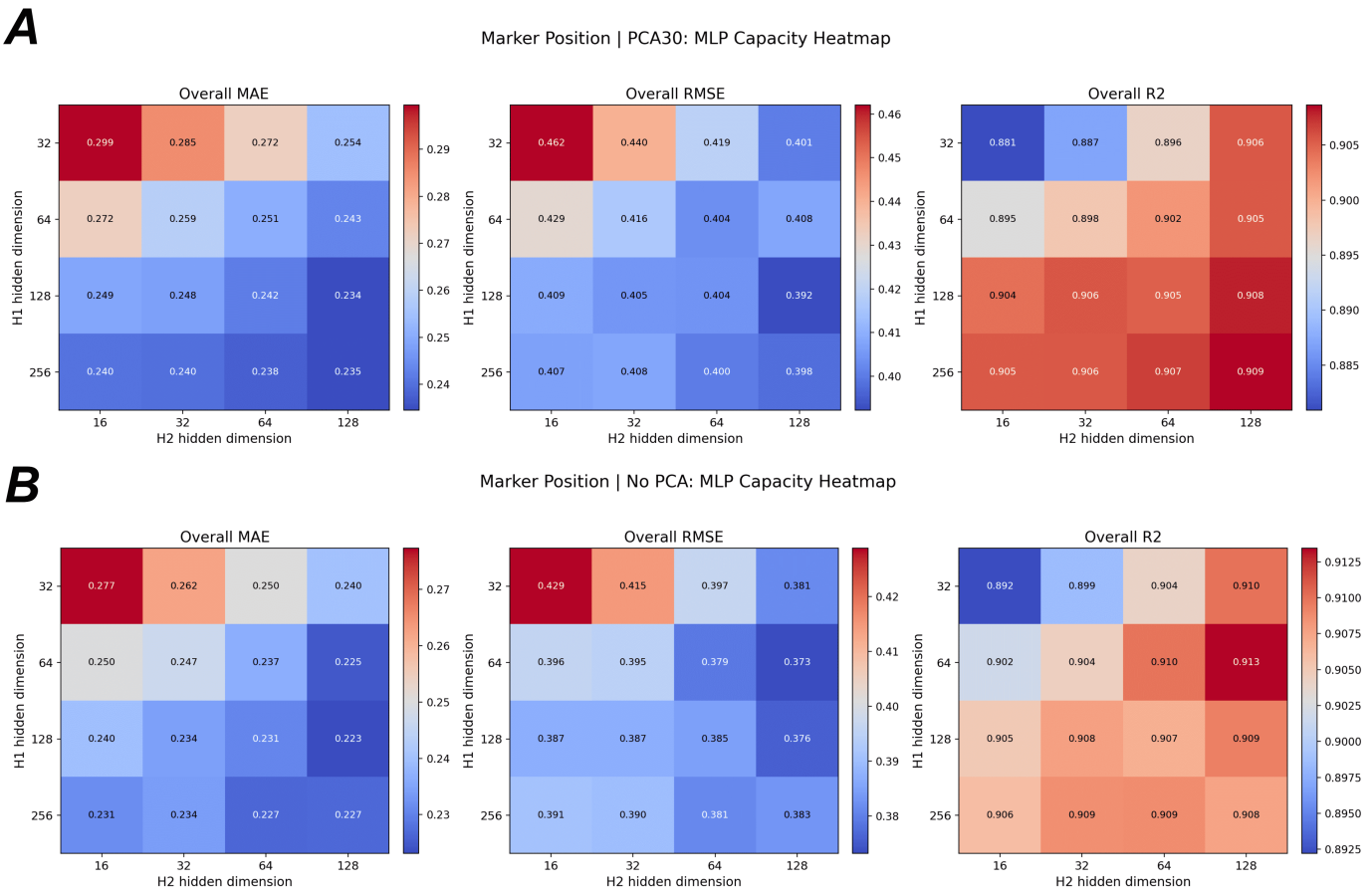

Fig. 16: Influence of MLP capacity on force-estimation performance using marker-position features for the mixed-shape dataset. A, Heatmaps of overall MAE, RMSE, and R^2 across tested MLP capacities using PCA-compressed marker-position features with 30 principal components. B, Corresponding heatmaps using raw marker-position features without PCA compression. Rows indicate first-hidden-layer dimensions H_1, and columns indicate second-hidden-layer dimensions H_2.

Q. *Supplementary Figure 17*

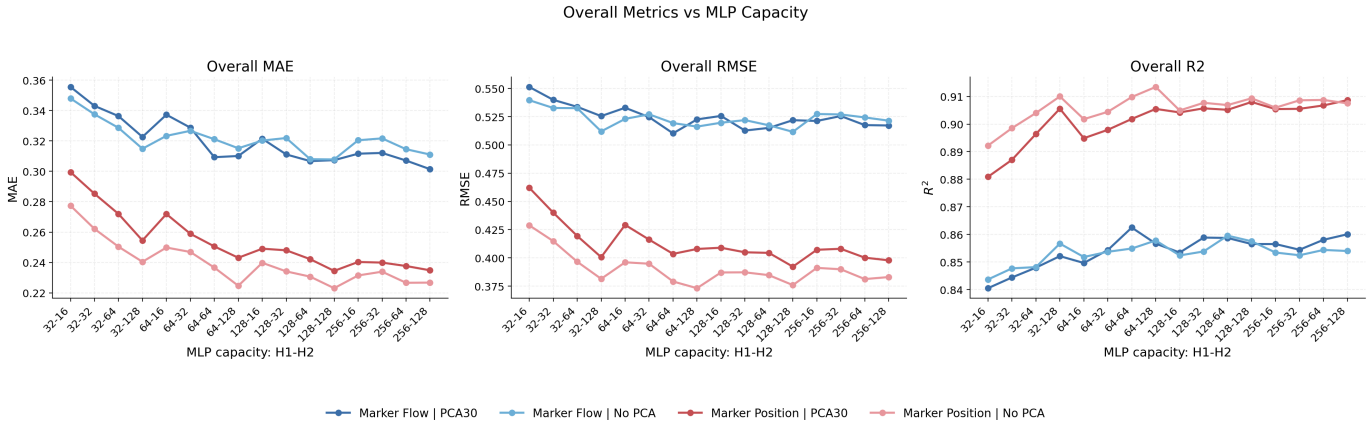

Fig. 17: Overall force-estimation metrics as functions of MLP capacity. The three panels compare overall MAE, RMSE, and R^2 across different hidden-layer configurations, denoted by H_1–H_2. Four input settings are compared: marker-flow with PCA 30, marker-flow without PCA, marker-position with PCA 30, and marker-position without PCA.

After examining the effect of PCA dimensionality under a fixed MLP architecture, we further investigated how model capacity influences force-estimation performance under complex contact geometries. In this analysis, 30 principal components were retained for the PCA-compressed inputs, as this setting provided a practical balance between compactness and prediction accuracy in the preceding analysis, while the no-PCA inputs were retained as reference baselines. The hidden-layer dimensions of the two-layer MLP were varied as $H_1 \in 32, 64, 128, 256$ and $H_2 \in 16, 32, 64, 128$ to assess whether lightweight MLP architectures remain sufficient for force estimation on the mixed-shape indenter dataset.

For marker-flow features, Supplementary Fig. 11 and Supplementary Fig. 12 show the predicted-versus-ground-truth results using PCA 30 and no-PCA inputs, respectively. The rows represent different first-hidden-layer dimensions H_1, while the columns show representative second-hidden-layer dimensions H_2. Across a broad range of model capacities, the predictions remain distributed around the diagonal reference line, indicating that marker-flow features contain sufficient force-relevant information for lightweight regression. Increasing the MLP capacity generally produces more compact prediction distributions, particularly when moving from the smallest architectures to moderate hidden-layer dimensions.

This trend is quantified by the heatmaps in Supplementary Fig. 13, which summarize all tested combinations of H_1 and H_2. For PCA-compressed marker-flow inputs (Supplementary Fig. 13A), increasing the hidden-layer dimensions reduces the overall MAE from 0.356 to approximately 0.301 and improves R^2 from 0.841 to approximately 0.860. A similar trend is observed for the no-PCA marker-flow inputs (Supplementary Fig. 13B), where the overall MAE decreases from 0.348 to approximately 0.308–0.315 for higher-capacity models, while R^2 increases from 0.844 to approximately 0.858–0.860. These results indicate that PCA-compressed marker-flow features retain most of the information required for force inference, with only a limited performance difference from the raw marker-flow representation.

Marker-position features exhibit a generally more favourable response. Supplementary Fig. 14 and Supplementary Fig. 15 show the corresponding predicted-versus-ground-truth results using PCA 30 and no-PCA marker-position inputs, respectively. Compared with marker-flow features, marker-position features produce tighter prediction distributions around the diagonal reference line across most tested MLP capacities, indicating that absolute marker-coordinate information provides a more stable representation under mixed contact geometries.

The corresponding heatmaps in Supplementary Fig. 16 further quantify this behaviour. For PCA-compressed marker-position inputs (Supplementary Fig. 16A), the overall MAE decreases from 0.299 to approximately 0.234–0.235 as model capacity increases, while R^2 improves from 0.881 to approximately 0.908–0.909. Without PCA compression (Supplementary Fig. 16B), marker-position inputs achieve the best overall performance, with MAE values approaching 0.223–0.227 and R^2 reaching approximately 0.908–0.913 for several moderate- and high-capacity architectures. These results further demonstrate the robustness of marker-position features for force estimation across complex contact geometries.

A direct comparison among feature type, PCA compression, and MLP capacity is provided in Supplementary Fig. 17. For both marker-flow and marker-position inputs, MAE and RMSE decrease rapidly as the model capacity increases from the smallest architectures to moderate configurations and then gradually plateau. This suggests that very small MLPs can underfit the nonlinear relationship between marker features and contact forces induced by multiple indenter geometries, whereas moderate hidden-layer dimensions are sufficient to capture the dominant mapping. Further increases in model capacity provide only marginal improvements and can occasionally introduce slight performance fluctuations.

Across most tested MLP capacities, marker-position features outperform marker-flow features, with lower MAE and RMSE and higher R^2. The no-PCA setting generally provides slightly better performance than PCA 30, particularly for marker-position

inputs; however, the performance gap becomes limited once a moderate MLP capacity is used. Overall, these results demonstrate that force-estimation performance benefits from increasing MLP capacity but gradually saturates beyond moderate hidden-layer dimensions. For the computationally constrained PixelTac pipeline, a moderate MLP capacity combined with PCA-compressed marker-position features therefore provides a practical trade-off among prediction accuracy, input dimensionality, and model complexity.

III. Supplementary Tables

A. Supplementary Table I

TABLE I: Structured comparison of representative VBTS computing paradigms.

Paradigm	Representative work	Output form	Tactile sensing	Tactile perception	Transmission burden
Far-sensor	Gelsight-mini [21]	Dense image	RGB camera	Host	2448×3264, 24 Hz
	DIGIT [22]	Dense image	RGB camera	Host	640×480, 60 Hz
	OmniTact [23]	Dense image	RGB camera	Host	400×400, 30 Hz
	Gelsight wedge [24]	Dense image	RGB camera	Host	640×480, 60 Hz
	ViTacTip [25]	Dense image	RGB camera	Host	1920×1080, 30 Hz
	9DTac [26]	Dense image	RGB camera	Host	2592×1944, 90 Hz
	Tac3D [27]	Dense image	RGB camera	Host	3280×2464, 24 Hz
	DelTact [28]	Dense image	RGB camera	Host	798×586, 40 Hz
	VisTac [29]	Dense image	RGB camera	Host	640×480, 60 Hz
	Kumagai et al. [30]	Sparse event	Event camera	Host	N/A, μs-level
	Naeini et al. [31]	Sparse event	Event camera	Host	$\ll$ 240×180, μs-level
	Huang et al. [32]	Sparse event	Event camera	Host	$\ll$ 240×180, μs-level
	NeuroTac [33]	Sparse event	Event camera	Host	$\ll$ 240×180, μs-level
	Evetac [34]	Sparse event	Event camera	Host	$\ll$ 640×480, 1000 Hz
	E-bts [35]	Sparse event	Event camera	Host	$\ll$ 640×480, μs-level
	GelEvent [36]	Sparse event	Event camera	Host	N/A, 180 Hz
	EVBTS [37]	Sparse event	Event camera	Host	N/A, ms-level
	Khairi et al. [38]	Sparse event	Event camera	Host	$\ll$ 640×480, μs-level
Near-sensor	DIGIT360 [39]	Tactile feature	RGB camera	Embedded AI accelerator	4k-level, 240 Hz
	Feng et al. [40]	Tactile feature	RGB camera	Embedded NPU	1920×1080, 60 Hz
	Xu et al. [41]	Tactile feature	Event camera	Embedded neuromorphic chip	$\ll$ 640×480, μs-level
	FasTac [42]	Tactile feature	RGB-NIR camera	Embedded FPGA	128×128, 240 Hz
	Zhu et al. [43]	Tactile feature	RGB camera	Embedded FPGA	128×128, $\leq$400 Hz
In-sensor	**PixelTac (ours)**	**Tactile feature**	**In-situ Vision Processor (IVP)**		**No physical separation**

Note: Bold values indicate our proposed design.

B. Supplementary Table II

TABLE II: Power Consumption of CPM under Different In-Sensor Computing Operations

Operation	1.5V			1.8V			3.3V			P_{CPM} (mW)		
	I_{ACU} (mA)			I_{DCU} (mA)			I_{IO} (mA)					
	10 fps	30 fps	60 fps	10 fps	30 fps	60 fps	10 fps	30 fps	60 fps	10 fps	30 fps	60 fps
Image Capture	6.01	6.06	6.04	9.53	9.58	9.61	0.0102	0.0298	0.0592	26.21	26.46	26.59
Edge Enhancement	32.91	30.97	28.04	9.51	9.56	9.65	0.0281	0.0832	0.1647	66.56	63.87	59.86
Inter-frame Change	75.70	74.70	73.60	9.47	10.13	11.01	0.0018	0.0044	0.0081	130.60	130.31	130.28
BoundingBox	0.24	0.48	0.87	9.36	9.62	10.03	0.0023	0.0052	0.0096	17.00	17.56	18.90
Average	**28.72**	**28.05**	**27.14**	**9.47**	**9.72**	**10.08**	**0.0106**	**0.0307**	**0.0604**	**60.09**	**59.55**	**58.91**

Note: Bold values indicate the averaged-performance results.

C. Supplementary Table III

TABLE III: Valid Contact Event Count under Surface Sliding Stimuli

Gain	FPS							
	1kHz	500Hz	200Hz	100Hz	50Hz	33Hz	20Hz	10Hz
1	-	-	-	10 (2)	200 (3)	300 (3)	**350 (2)**	300 (2)
2	-	-	-	10 (2)	100 (15)	300 (15)	**400 (15)**	300 (15)
3	-	Noise	Noise	10 (30)	100 (37)	100 (32)	**200 (36)**	100 (35)
4	-	Noise	Noise	10 (40)	100 (40)	**200 (40)**	100 (38)	10 (40)
5	-	Noise	Noise	10 (40)	**100 (40)**	10 (40)	10 (40)	10 (40)

Note: '-': no valid event; 'Noise': noisy events; '10 (2)': fewer than 10 valid events when applying the minimize trigger $k_{trigger}^{analog}$ value of 2.

D. Supplementary Table IV

TABLE IV: Resultant Force Estimation MSE across PCA Dimensions

Dataset	**PCA Dimension** d						
	5	10	15	20	25	30	noPCA
5mm	0.236 ± 0.077	0.186 ± 0.045	0.270 ± 0.110	0.212 ± 0.054	**0.179 ± 0.037**	0.195 ± 0.041	2.277 ± 2.400
10mm	0.141 ± 0.020	**0.124 ± 0.020**	0.202 ± 0.060	0.166 ± 0.034	0.170 ± 0.025	0.184 ± 0.026	2.027 ± 0.681
5+10mm	0.182 ± 0.039	**0.147 ± 0.028**	0.156 ± 0.032	0.182 ± 0.044	0.215 ± 0.060	0.254 ± 0.086	2.337 ± 1.943

Note: *mean* ± *variance* of MSE per PCA dimension is calculated upon 5-iteration train-test results in terms of each dataset.

IV. Supplementary Videos

A. Movie S1. Spatial contact inference using PixelTac

This video demonstrates the spatial-contact sensing capability of PixelTac under different contact patterns, including real-time marker-tracking performance, push-up, push-down, push-left, push-right, vertical squeeze, and horizontal squeeze stimuli. Marker displacement, Gaussian density distribution, and estimated contact deformation are visualised in real time.

B. Movie S2. Temporal contact sensing using PixelTac

This video demonstrates event-based temporal-contact sensing of PixelTac under different contact patterns, including tapping, shearing, and rolling-slip interactions with different objects. The generated event maps and corresponding temporal responses are visualised in real time.

C. Movie S3. Real-time force estimation using PixelTac

This video demonstrates the complete sensing-to-perception pipeline of PixelTac. Marker features extracted through in-sensor computing are used to estimate three-axis contact forces in real time under different contact patterns, including both 3D resultant force vector and force estimation curve.

V. Supplementary Algorithms

A. Supplementary Algorithm 1

Algorithm 1 Pixel-level Task for Data Preprocessing

Input:
Optical information $I_{optical}$, pixel value gain k_{gain},
Digital threshold value $k_{thred}^{digital}$
Output:
Processed tactile image of high SNR, $I_{processed}$
Hardware:
PIX, ACU, DCU, ALU in CPM
— Image Sensing —
$I_{pixel} = \mathrm{PIX}(I_{optical})$
— Feature Enhancement —
(1) Reading:
    $I_{unsign}[0, 127] = \mathrm{ALU}(k_{gain} \cdot I_{pixel}) \rightarrow$ ACU-1
    $I_{sign}[-128, 127] = \mathrm{ALU}(k_{gain} \cdot I_{pixel}) \rightarrow$ ACU-2
(2) Filtering:
    $I_{sharpen} = \mathrm{ALU}(I_{sign} * K_{sharpen}) \rightarrow$ ACU-3
    $I_{sobel} = \mathrm{ALU}(I_{sign} * K_{sobel}) \rightarrow$ ACU-4
    $I_{gaussian} = \mathrm{ALU}(I_{sign} * K_{gaussian}) \rightarrow$ ACU-5
— Data Denoising —
(1) Analog-Digital Conversion:
    $I_{threshold} = \text{Threshold}(I_{unsign}, k_{thred}^{digital}) \rightarrow$ DCU-1
(2) Morphological Operation:
    $I_{dilate} = \text{Dilate}(I_{threshold}) \rightarrow$ DCU-2
    $I_{erode} = \text{Erode}(I_{threshold}) \rightarrow$ DCU-3
(3) Logical Operation:
    $I_{mask} = \text{AND}(I_{threshold}, Mask) \rightarrow$ DCU-4
    $I_{not} = \text{NOT}(I_{threshold}) \rightarrow$ DCU-5
— Output —
$I_{\text{processed}} \in \{I_{sharpen}, \ldots I_{threshold}, \ldots I_{dilate}, \ldots I_{not}\}$

B. Supplementary Algorithm 2

Algorithm 2 Feature-level Task for Marker Tracking

Input:
Preprocessed tactile image frames $\{I_0, I_{i-1}, I_i\}$
Output:
Marker positions P_i, marker motion trajectories $\vec{T}_i$,
Explicit marker feature $\mathbf{F}_i^{marker} = [P_i|\vec{T}_i]$
Hardware:
DCU, ALU in CPM, MCU
— Marker Detection —
(1) Reading:
    $I_i \rightarrow$ DCU-1
    Clear(DCU-2)
(2) Multiple Marker Detection:
while e_j exists in I_i **do**
    $e_j = \mathrm{Event}(I_i) \rightarrow$ DCU-2
    $I_{prop} = \mathrm{Propagation}(I_i, e_j)$
    $B_j = \mathrm{Boundingbox}(I_{prop})$
    $p_j = \mathrm{Centroid}(B_j)$
    $I_{not} = \mathrm{NOT}(I_{prop})$
    $I_{and} = \mathrm{AND}(I_{not}, I_i)$
    $I_i = I_{and}$
end while
(3) Data Collection:
    $P_i = [p_0, ..., p_j]_{j=27} \rightarrow \mathrm{MCU}$
— Initialization —
if first frame $i = 0$ **then**
    $ID_{1\times28} = \mathrm{Sort}(P_0)$
    $N_{8\times28} =$ Neighbour Connection Matrix (ID)
end if
— Marker Registration —
(1) Reading:
    $P_{i-1}, P_i \rightarrow \mathrm{MCU}$
(2) Inter-frame Marker Matching:
    $P_i, ID_{lost} =$ Nearest Neighbor Association (P_{i-1}, P_i)
— Marker Motion Trajectory —
    $\vec{t}_i^{\,j} = p_i^j - p_0^j$
    $\vec{T}_i = \left[\vec{t}_i^{\,0}, \ldots, \vec{t}_i^{\,j}\right]_{j=27}$
— Missing Marker Prediction —
for each ID_{lost} **do**
    $\vec{T}_{ID_{lost}} =$ Neighbor Vector Interpolation$(N(ID_{lost}), \vec{T}_i)$
    $P_{ID_{lost}} = P_0 + \vec{T}_{ID_{lost}}$
end for
— Output —
$\mathbf{F}_i^{marker} = [P_i|\vec{T}_i]$

C. Supplementary Algorithm 3

Algorithm 3 Event-level Task for Contact Event Capturing

Input:
Current image I_i, last image I_{i-1}, frame interval Δt,
Pixel value gain k_{gain}, analog trigger value $k_{trigger}^{analog}$
Output:
Contact event set $S_E(i) = \{e_j(x, y, t),\ j \in n_E(i)\}$
Hardware:
ACU, DCU, ALU in CPM, MCU
— Inter-frame Processing —
(1) Reading:
    $I_{curr} = \mathrm{ALU}(k_{gain} \cdot I_i) \rightarrow$ ACU-1
    $I_{prev} = I_{i-1} \leftarrow$ ACU-5
    $I_{trigger} = \mathrm{Load}(k_{trigger}^{analog}) \rightarrow$ ACU-3
(2) Difference Computation:
    $I_{diff} = \mathrm{Sub}(I_{curr}, I_{prev}) \rightarrow$ ACU-2
(3) Trigger Filtering:
    $I_{filt} = \mathrm{Sub}(I_{diff}, I_{trigger}) \rightarrow$ ACU-4
(4) Frame Updating:
    $I_{i-1} = I_{curr} \rightarrow$ ACU-5
— Event Locating —
(1) Event Mask:
    $\mathrm{MASK} = \mathrm{Where}(I_{filt} > 0)$
(2) Event Map:
    $I_{event} = \mathrm{OVERLAP}(I_{filt}, \mathrm{MASK}) \rightarrow$ DCU-1
— Event Count —
(1) Coordinate Extraction:
while e_j exists in I_{event} **do**
    $e_j(x, y) = \mathrm{Event}(I_{event})$
    $n_E(i) += 1$
end while
(2) Event Set at $t = i \cdot \Delta t$:
    $S_E(i) = \{e_0(x, y, t), ..., e_j(x, y, t)\}_{j=n_E(i)}$
— Output —
$S_E(i) = \{e_j(x, y, t), j \in n_E(i)\}$

D. Supplementary Algorithm 4

Algorithm 4 Inference-level Task for Force Regression: Training

Input:
Dataset $D = \{(x^{(k)}, y^{(k)})\}_{k=1}^{N}$ where x is reduced tactile feature F_{reduced}, y is ground truth F_{label}, N is training dataset amount, hidden layer dimension H_1, H_2
Output:
MCU-ready quantized parameters of trained model $\mu, \sigma, S_{\text{in}}, S_{\text{hidden}^*}, S_{w*}, qW_*, qb_*$
Hardware:
Workstation PC
— Model Training —
(1) Dataset Preprocess:
    $X, Y \leftarrow \text{Load}(D)$
    $X_{\text{train}}, X_{\text{val}}, Y_{\text{train}}, Y_{\text{val}} \leftarrow \text{Split}(X, Y)$
(2) Input Standardization:
    $\mu, \sigma \leftarrow \text{StandardScaler.fit}(X_{\text{train}})$
    $\hat{X}_{\text{train}} \leftarrow (X_{\text{train}} - \mu)/\sigma$
    $\hat{X}_{\text{val}} \leftarrow (X_{\text{val}} - \mu)/\sigma$
(3) Model Setup:
    $\text{model} \leftarrow \text{MLP}(F_{\text{reduced}}, H_1, H_2, F_{\text{label}})$
    # activations: ReLU; loss: MSE; optimizer: Adam
(4) Train Floating-point Model:
    for each epoch:
        for each batch (x_b, y_b) from $(\hat{X}_{\text{train}}, Y_{\text{train}})$:
            $\hat{y} \leftarrow \text{model.forward}(x_b)$
            $\mathcal{L}_{train} \leftarrow \text{MSE}(\hat{y}, y_b)$
            $\text{model} \leftarrow \text{Adam}(\mathcal{L}_{train})$
            $\mathcal{L}_{valid} \leftarrow \text{model.evaluate}(\hat{X}_{\text{val}}, Y_{\text{val}})$
        end for
    end for
— Parameter Calibration —
(1) Collect Layer Activation Statistics:
    $AM_{\text{in}}, AM_1, AM_2 \leftarrow \text{Activation Maxima(model)}$
    $W_1, W_2, W_3 \leftarrow \text{Trained Weight(model)}$
    $S_{\text{in}} \leftarrow AM_{\text{in}}/(2^{15} - 1)$
    $S_{\text{hidden1}} \leftarrow AM_1/(2^{15} - 1)$
    $S_{\text{hidden2}} \leftarrow AM_2/(2^{15} - 1)$
    $S_{w1} \leftarrow \max(|W_1|)/(2^7 - 1)$
    $S_{w2} \leftarrow \max(|W_2|)/(2^7 - 1)$
    $S_{w3} \leftarrow \max(|W_3|)/(2^7 - 1)$
(2) Quantize Weights & Biases:
    $qW_1 \leftarrow \text{int8}(\text{Round}(W_1/S_{w1}))$
    $qW_2 \leftarrow \text{int8}(\text{Round}(W_2/S_{w2}))$
    $qW_3 \leftarrow \text{int8}(\text{Round}(W_3/S_{w3}))$
    $qb_1 \leftarrow \text{int32}(\text{Round}(b_1/(S_{\text{in}} \cdot S_{w1})))$
    $qb_2 \leftarrow \text{int32}(\text{Round}(b_2/(S_{\text{hidden1}} \cdot S_{w2})))$
    $qb_3 \leftarrow \text{int32}(\text{Round}(b_3/(S_{\text{hidden2}} \cdot S_{w3})))$
— Output —
$\mu, \sigma, S_{\text{in}}, S_{\text{hidden}^*}, S_{w*}, qW_*, qb_*$

E. Supplementary Algorithm 5

Algorithm 5 Inference-level Task for Force Regression: Deployment

Input:
MCU-ready quantized parameters of trained model $\mu, \sigma, S_{\text{in}}, S_{\text{hidden}^*}, S_{w*}, qW_*, qb_*$,
d-dimensional tactile feature F_{reduced}, hidden layer dimension H_1, H_2
Output:
Contact force estimation (F_x, F_y, F_z)
Hardware:
MCU
— MCU Inference —
    model $\leftarrow$ Load($\mu, \sigma, S_{\text{in}}, S_{\text{hidden}^*}, S_{w*}, qW_*, qb_*$)
    – Input Layer –
    (1) Reading:
    $x_{\text{raw}} \leftarrow F_{\text{reduced}}$
    (2) Standardize Input:
    $\hat{x}_{standard} \leftarrow (x_{\text{raw}} - \mu)/\sigma$
    (3) Quantize Input:
    $qx_{\text{in}} \leftarrow$ int16(Round($\hat{x}_{standard}/S_{\text{in}}$))
    $qx_{\text{in}} \leftarrow$ int8(Clamp($qx_{\text{in}}, \pm(2^7 - 1)$))
    – Hidden Layer 1 –
    for $i \in [0, (H_1 - 1)]$:
        $acc \leftarrow 0$ (wide integer accumulator)
        for $j \in [0, (d - 1)]$:
            $acc \leftarrow acc + qW_1[i, j] \cdot qx_{\text{in}}[j] + qb_1[i]$
        end for
        $act_f \leftarrow$ ReLU($0, acc \cdot (S_{\text{in}} \cdot S_{w1})$)
        $q_{h1}[i] \leftarrow$ int16(Round(act_f/S_{hidden1}))
        $q_{h1}[i] \leftarrow$ int8(Clamp($q_{h1}[i], \pm(2^7 - 1)$))
    end for
    – Hidden Layer 2 –
    for $i \in [0, (H_2 - 1)]$:
        $acc \leftarrow 0$ (wide integer accumulator)
        for $i \in [0, (H_1 - 1)]$:
            $acc \leftarrow acc + qW_2[i, j] \cdot q_{h1}[j] + qb_2[i]$
        end for
        $act_f \leftarrow$ ReLU($0, acc \cdot (S_{\text{hidden1}} \cdot S_{w2})$)
        $q_{h2}[i] \leftarrow$ int16(Round(act_f/S_{hidden2}))
        $q_{h2}[i] \leftarrow$ int8(Clamp($q_{h2}[i], \pm(2^7 - 1)$))
    end for
    – Output Layer –
    for $i \in [0, 1, 2]$:
        $acc \leftarrow 0$ (wide integer accumulator)
        for $i \in [0, (H_2 - 1)]$:
            $acc \leftarrow acc + qW_3[i, j] \cdot q_{h2}[j] + qb_3[i]$
        end for
        $out[i] \leftarrow acc \cdot (S_{\text{hidden2}} \cdot S_{w3})$
    end for
— Output —
$(F_x, F_y, F_z) = (out[0], out[1], out[2])$

F. Supplementary Algorithm 6

Algorithm 6 Spatial Contact Feature Perception

Input:
Explicit marker feature $\mathbf{F}_i = [P_i|\vec{T}_i]$, initial position P_0,
Neighbor connection matrix N, standard deviation σ
Output:
Marker displacement magnitude D_i, Gaussian density G_i
Spatial contact feature $\mathbf{F}_i^{contact} = [P_i|\vec{T}_i|D_i|G_i|\vec{V}_i]$
Hardware:
MCU
— Initialization —
if $i = 0$ **then**
  for marker $j \in [0, 27]$ **do**
    $d_0^{j,k} = \|p_0^k - p_0^j\|, \quad k \in N(j)$
    $w_0^{j,k} = \exp\left(-\frac{d_0^{j,k\,2}}{2\sigma^2}\right)$
    $g_0^j = \sum_{k \in N(j)} w_0^{j,k}$
  end for
  $G_0 = [g_0^0, ..., g_0^j]_{j=27}$
end if
— Spatial Contact Feature —
for frame i **do**
  for marker $j \in [0, 27]$ **do**
    (1) Marker Displacement Magnitude:
    $|\vec{t}_i^j| = \left\|p_i^j - p_0^j\right\|$
    (2) Marker Gaussian Density:
    $d_i^{j,k} = \|p_i^k - p_i^j\|, \quad k \in N(j)$
    $w_i^{j,k} = \exp\left(-\frac{d_i^{j,k\,2}}{2\sigma^2}\right)$
    $g_i^j = \sum_{k \in N(j)} w_i^{j,k}$
    $\Delta g_i^j = g_i^j - g_0^j$
    (3) Estimated Contact Deformation Field:
    if $\Delta g_i^j < 0$ **then**
      ${z_{start}}_i^j = -\Delta g_i^j$
    else
      ${z_{start}}_i^j = 0$
    end if
    ${z_{end}}_i^j = \sigma \cdot z_{start} + \left|\vec{t}_i^j\right|$
    $\vec{v}_i^j = ([x_0^j, y_0^j, {z_{start}}_i^j], [x_i^j, y_i^j, {z_{end}}_i^j])$
  end for
  $D_i = [|\vec{t}_i^{\,0}|, ..., |\vec{t}_i^{\,j}|]_{j=27}$
  $G_i = [\Delta g_i^0, ..., \Delta g_i^j]_{j=27}$
  $\vec{V}_i = [\vec{v}_i^0, ..., \vec{v}_i^j]_{j=27}$
end for
— Output —
$\mathbf{F}_i^{contact} = [P_i|\vec{T}_i|D_i|G_i|\vec{V}_i]$

REFERENCES

[1] L. Willemet, *The Biomechanics of the Tactile Perception of Friction*. Springer, 2022.

[2] C. E. Murchison, "A handbook of general experimental psychology." 1934.

[3] A. Welford, "Choice reaction time: Basic concepts," *Reaction times*, pp. 73–128, 1980.

[4] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile sensing—from humans to humanoids," *IEEE transactions on robotics*, vol. 26, no. 1, pp. 1–20, 2009.

[5] R. S. Johansson and I. Birznieks, "First spikes in ensembles of human tactile afferents code complex spatial fingertip events," *Nature neuroscience*, vol. 7, no. 2, pp. 170–177, 2004.

[6] R. S. Johansson and J. R. Flanagan, "Coding and use of tactile signals from the fingertips in object manipulation tasks," *Nature Reviews Neuroscience*, vol. 10, no. 5, pp. 345–359, 2009.

[7] R. S. Johansson and G. Westling, "Signals in tactile afferents from the fingers eliciting adaptive motor responses during precision grip," *Experimental brain research*, vol. 66, no. 1, pp. 141–154, 1987.

[8] K. J. Cole and J. H. Abbs, "Grip force adjustments evoked by load force perturbations of a grasped object," *Journal of neurophysiology*, vol. 60, no. 4, pp. 1513–1522, 1988.

[9] M. K. Floeter, C. Gerloff, J. Kouri, and M. Hallett, "Cutaneous withdrawal reflexes of the upper extremity," *Muscle & Nerve: Official Journal of the American Association of Electrodiagnostic Medicine*, vol. 21, no. 5, pp. 591–598, 1998.

[10] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. Siegelbaum, A. J. Hudspeth, S. Mack *et al.*, *Principles of neural science*. McGraw-hill New York, 2000, vol. 4.

[11] F. Zhou and Y. Chai, "Near-sensor and in-sensor computing," *Nature Electronics*, vol. 3, no. 11, pp. 664–671, 2020.

[12] Y. Chen, J. Cao, J. Qiu, D. Yang, M. Liu, M. Zhang, C. Li, Z. Wu, J. Yu, X. Zhang *et al.*, "Capacitive in-sensor tactile computing," *Nature Communications*, vol. 16, no. 1, p. 5691, 2025.

[13] Y. Li, Y. Yang, C. Wang, Y. Dai, X. Yan, D. Kong, Z. Liao, S. Wang, G.-J. Ruan, P. Wang *et al.*, "Massively parallel in-sensor skinomorphic computing," *Nature Communications*, 2026.

[14] J. Guo, F. Guo, H. Zhao, H. Yang, X. Du, F. Fan, W. Liu, Y. Zhang, D. Tu, and J. Hao, "In-sensor computing with visual-tactile perception enabled by mechano-optical artificial synapse," *Advanced Materials*, vol. 37, no. 14, p. 2419405, 2025.

[15] C. Wei, S. Yu, Y. Meng, Y. Xu, Y. Hu, Z. Cao, Z. Huang, L. Liu, Y. Luo, H. Chen *et al.*, "Octopus tentacle-inspired in-sensor adaptive integral for edge-intelligent touch intention recognition," *Advanced Materials*, p. 2420501, 2025.

[16] J. Fernández-Berni and R. Carmona-Galán, "Eye-ris cmos vision system-on-chip," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 10, pp. 2767–2778, 2009.

[17] Á. Rodríguez-Vázquez, R. Domínguez-Castro, F. Jiménez-Garrido, S. Morillas, J. Listán, L. Alba, C. Utrera, S. Espejo, and R. Romay, "The eye-ris cmos vision system," in *Analog circuit design*. New York City: Springer, 2008, pp. 15–32.

[18] Aistorm Inc., "Aistorm mantis2: Event-driven charge-domain vision processor," https://www.aistorm.com/mantis2, 2020.

[19] T. Yamazaki, H. Katayama, S. Uehara, A. Nose, M. Kobayashi, S. Shida, M. Odahara, K. Takamiya, S. Matsumoto, L. Miyashita *et al.*, "A 1ms high-speed vision chip with 3d-stacked 140gops column-parallel pes for spatio-temporal image processing," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. San Francisco, CA, USA: IEEE, February 2017, pp. 82–84.

[20] J. Meng, F. Wen, Z. Wang *et al.*, "Photonic memristor for artificial visual perception and in-sensor computing," *Advanced Functional Materials*, vol. 29, no. 52, p. 1902220, 2019.

[21] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, 2017.

[22] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer *et al.*, "Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3838–3845, 2020.

[23] A. Padmanabha, F. Ebert, S. Tian, R. Calandra, C. Finn, and S. Levine, "Omnitact: A multi-directional high-resolution touch sensor," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 618–624.

[24] S. Wang, Y. She, B. Romero, and E. Adelson, "Gelsight wedge: Measuring high-resolution 3d contact geometry with a compact robot finger," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 6468–6475.

[25] W. Fan, H. Li, W. Si, S. Luo, N. Lepora, and D. Zhang, "Vitactip: Design and verification of a novel biomimetic physical vision-tactile fusion sensor," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 1056–1062.

[26] C. Lin, H. Zhang, J. Xu, L. Wu, and H. Xu, "9dtact: A compact vision-based tactile sensor for accurate 3d shape reconstruction and generalizable 6d force estimation," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 923–930, 2023.

[27] L. Zhang, Y. Wang, and Y. Jiang, "Tac3d: A novel vision-based tactile sensor for measuring forces distribution and estimating friction coefficient distribution," *arXiv preprint arXiv:2202.06211*, 2022.

[28] G. Zhang, Y. Du, H. Yu, and M. Y. Wang, "Deltact: A vision-based tactile sensor using a dense color pattern," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 778–10 785, 2022.

[29] S. Athar, G. Patel, Z. Xu, Q. Qiu, and Y. She, "Vistac toward a unified multimodal sensing finger for robotic manipulation," *IEEE Sensors Journal*, vol. 23, no. 20, pp. 25 440–25 450, 2023.

[30] K. Kumagai and K. Shimonomura, "Event-based tactile image sensor for detecting spatio-temporal fast phenomena in contacts," in *2019 IEEE World Haptics Conference (WHC)*. IEEE, 2019, pp. 343–348.

[31] F. B. Naeini, A. M. AlAli, R. Al-Husari, A. Rigi, M. K. Al-Sharman, D. Makris, and Y. Zweiri, "A novel dynamic-vision-based approach for tactile sensing applications," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 5, pp. 1881–1893, 2019.

[32] X. Huang, R. Muthusamy, E. Hassan, Z. Niu, L. Seneviratne, D. Gan, and Y. Zweiri, "Neuromorphic vision based contact-level classification in robotic grasping applications," *Sensors*, vol. 20, no. 17, p. 4724, 2020.

[33] B. Ward-Cherrier, N. Pestell, and N. F. Lepora, "Neurotac: A neuromorphic optical tactile sensor applied to texture recognition," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2654–2660.

[34] N. Funk, E. Helmut, G. Chalvatzaki, R. Calandra, and J. Peters, "Evetac: An event-based optical tactile sensor for robotic manipulation," *IEEE Transactions on Robotics*, 2024.

[35] D. Mukashev, S. Seitzhan, J. Chumakov, S. Khajikhanov, M. Yergibay, N. Zhaniyar, R. Chibar, A. Mazhitov, M. Rubagotti, and Z. Kappassov, "E-bts: Event-based tactile sensor for haptic teleoperation in augmented reality," *IEEE Transactions on Robotics*, vol. 41, pp. 450–463, 2024.

[36] D. Yin, S. Lu, J. Yang, Y. Zhang, Z. Dai, D. Nan, B. Cai, S. He, and X. Chen, "Gelevent—a novel high-speed tactile sensor with event camera," *IEEE Transactions on Instrumentation and Measurement*, 2025.

[37] E. Sherif, A. Khairi, H. Sajwani, A. Solayman, A. M. Alkilany, A. Awadalla, M. Halwani, L. AbuAssi, D. Swart, A. Ayyad *et al.*, "Real-time fault detection in robotic manufacturing using high-bandwidth event vision-based tactile sensing," *Sensors International*, p. 100355, 2025.

[38] A. Khairi, H. Sajwani, A. M. Alkilany, L. AbuAssi, M. Halwani, I. M. Zaid, A. Awadalla, D. Swart, A. Ayyad, and Y. Zweiri, "They see me rolling: High-speed event vision-based tactile roller sensor for large surface inspection," *IEEE Transactions on Robotics*, 2026.

[39] M. Lambeta, T. Wu, A. Sengul, V. R. Most, N. Black, K. Sawyer, R. Mercado, H. Qi, A. Sohn, B. Taylor *et al.*, "Digitizing touch with an artificial multimodal fingertip," *arXiv preprint arXiv:2411.02479*, 2024.

[40] Y. Feng, Y. Jin, M. Chen, J. Yang, G. Cao, C. Yang, and Z. Lu, "Wireless tactile sensor with embedded neural network for multifunctional contact detection," in *2025 30th International Conference on Automation and Computing (ICAC)*. IEEE, 2025, pp. 1–6.
[41] X. Xu, A. Li, and B. Ward-Cherrier, "Exploratory movement strategies for texture discrimination with a neuromorphic tactile sensor," *arXiv preprint arXiv:2509.14954*, 2025.
[42] X. Lu, K. Huang, J. Chen, Y. Lin, H. Yang, and Z. Yin, "Fastac: A curved multispectral vision-based tactile sensor for high-speed high-precision 3d shape and force perception," *arXiv preprint arXiv:2607.28416*, 2026.
[43] Z. Zhu, R. Zhang, R. Hu, and C. Xiao, "Near-sensor computing for rapid visuotactile perception," *arXiv preprint arXiv:2608.05725*, 2026.