DNN-based visual perception for high-precision motion control

Vibhor Jain*, Sajid Mohamed†, Dip Goswami*, Sander Stuijk*
*Eindhoven University of Technology, †ITEC B.V.
{v.jain, d.goswami, s.stuijk}@tue.nl, sajid.mohamed@itecequipment.com

***Abstract*—The high-speed, high-precision positioning of objects is a critical component in various industrial manufacturing processes. The semiconductor die packaging, for instance, requires the precise pickup and placement of semiconductor dies on substrates. This is done by coupling the silicon wafer which contains thousands of semiconductor dies, with a motion control platform equipped with linear motors and encoders. The motion controller relies on linear motors and encoders to accurately position the silicon wafer at reference positions, which are determined through the relative positions of the dies on the wafer. However, the challenge arises when neighboring dies get misaligned during the pickup process, making it impossible to read the position of the die through encoders. This paper addresses the challenge of precise alignment in high-speed, micro-scale manufacturing environments, where traditional methods struggle due to the disconnect between the point-of-interest (dies) and point-of-control (motor/silicon wafer). To overcome these challenges, we propose a Deep Neural Network (DNN) based perception that allows for robust sensing of die positions. We also propose a fusion mechanism to incorporate this vision feedback with the encoder to accurately detect the misalignment and compensate for it before periodic pickups of the dies. We use a software-in-the-loop validation framework to demonstrate that our proposed method could successfully eliminate the misalignment before the pickup in the range under consideration.**

***Index Terms*—DNN based perception, vision-in-the-loop, high-precision control, semiconductor manufacturing**

I. Introduction

The high-speed, high-precision positioning system represents a fundamental component within numerous industrial manufacturing systems. Typically, these systems are responsible for transporting objects to predefined pick-up or placement locations with the help of motion control systems, where specialized mechanisms retrieve or deposit the objects. For example, in the die-bonding process of semiconductor manufacturing a fundamental task is picking up the semiconductor die from the silicon wafer and placing it onto the substrate. This is done by mounting the wafer on top of a physical platform (e.g. wafer stage) with linear motors and encoders to apply force on the platform to move the wafer (which is the point-of-control) at predefined *reference positions* and accurately read the position of the platform at high rates. The *reference positions* corresponds to the thousands of dies (point-of-interest) on the wafer, and are defined through the initial layout of all the dies with respect to the wafer ensuring that a particular die gets aligned with the pickup position when the wafer is moved there, as shown in Fig. 1. Once the dies reach the *reference position*, they are periodically picked up for further processes like quality inspection and ultimately placed onto the substrates. While picking up the die, the neighboring dies get misaligned from their initial positions.

The performance of these systems depends on the precise alignment of the object with the pick-up or placement location. This in-turn requires accurately sensing the position of the object, which is a challenging task especially in domains like semiconductor manufacturing, not only due to the micro-scale precision (product size of 200 µm to 10 mm) and extremely high throughput (production rate of up to 72,000 units/hour $\approx 50ms$ per product [2]) required in such domains, but also a fundamental disconnect between the point-of-interest (dies) and point-of-control (motor/wafer). This means that any noise added to the point-of-interest is hard to detect, and therefore compensated through the motion controller. Therefore, multi-sensor approaches with vision-based perception to read the position of the point-of-interest are being researched. The vision-based sensing methods are capable of reading the true position (with disturbances) of the point-of-interest which when incorporated efficiently with the motion controller can precisely position the point-of-interest to the reference. Although classical computer vision algorithms have been around for a while, they are less robust [3] and require a lot of manual configurations to locate a particular object in an image. Therefore, Deep Neural Network (DNN) based solutions are more favorable as they are robust and can be used to detect the positions of the objects in an image with high-precision. This comes at a cost of higher computation and processing delays, making it challenging to incorporate them with motion control systems.

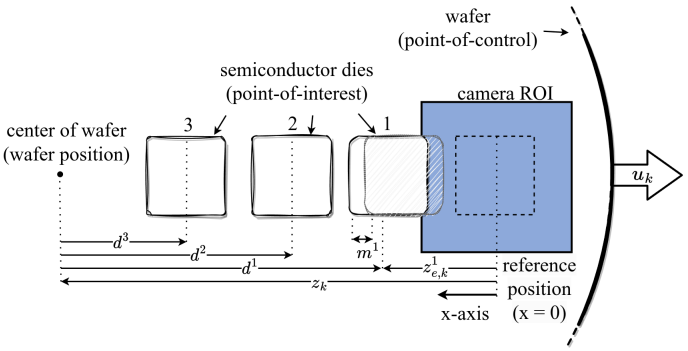

Fig. 1. Set of semiconductor dies located on a silicon wafer. The encoder reads the position of the wafer and the camera captures an image of the (partial) die(s) in the ROI. The dies are positioned at the reference by moving the wafer.

This work was supported by ECSEL Joint Undertaking in H2020 project IMOCO4.E [1], grant agreement No.101007311

Contributions: The paper proposes the use of Deep Neural Network (DNN) based perception for vision-in-the-loop (VIL) systems in high-precision motion control. In particular, we consider the alignment problem in semiconductor die packaging. To efficiently use the vision feedback with the encoder readings, a fusion mechanism is introduced. For validation of the VIL system with the proposed fusion mechanism, a software-in-the-loop (SIL) framework is developed. With the help of this SIL framework, we evaluate the performance of the closed-loop system with different vision sampling rates. We further present a worst-case analysis to eliminate the misalignment in a die positioning system.

II. Motivation and Problem Statement

In this section, we introduce the application of a high-precision motion control system for die positioning in semiconductor die packaging (Fig. 1). We introduce the overall system architecture and relevant design elements as shown in Fig. 2. Next, we formulate the system dynamics and introduce the multi-sensor sensing setup and the relevant challenges in designing a multi-sensor estimator and controller for these kinds of systems.

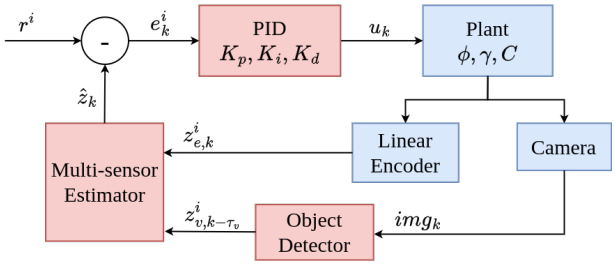

Fig. 2. Block diagram of the system, with the linear encoder and camera as sensors, multi-sensor estimator, and PID controller.

A. System description

For the scope of the paper, for simplicity, we consider a one-dimensional system, though the same approach applies to the actual system, which is two-dimensional, without any significant modifications. The system is composed of a linear motor which is used to control the position of the platform which has the silicon wafer with thousands of dies systematically placed onto it. This pattern is defined by the coordinates representing the ideal position of the dies d^i with respect to the center of the wafer. For example, in Fig. 1 since the dies 1, 2, and 3 are on the right side of the center of the wafer d^1, d^2 and d^3 are negative constants. In practice, as the dies are distributed uniformly on the wafer, these values can be defined by just two parameters, the die size and the center-to-center distance between the neighboring dies. The primary goal of the motion control system is to accurately position these semiconductor dies one after the other at the *reference position* where they can be picked up. To achieve this task the output of linear motors is sensed by encoders, which accurately read the movements of the motor and therefore the wafer (point-of-control). The positioning task is achieved by defining a set of control references with the help of the ideal positions of the semiconductor dies with respect to the wafer i.e. d^i.For example, to maintain the throughput of more than 70,000 units per hour a die is picked up every 50 ms. During pickup, the neighboring dies get misaligned from their initial positions. This misalignment is denoted by m^i_k with respect to the ideal position of the die. To sense this misalignment vision feedback is used by capturing the images of the die in the camera's Region of Interest (ROI) and using an object detector to get the true position of the die and thus the misalignment of the die from its ideal position.

B. System dynamics

The discretized dynamics of the motion control platform are represented by the following equations:

$$x_{k+1} = \phi x_k + \gamma u_k \tag{1}$$

$$z_k = C x_k \tag{2}$$

The system is an eighth-order single input single output system, where x_k is the state vector, u_k is the input (force applied by the motor) and z_k is the output (position of the wafer) at an instance k. The matrices ϕ and γ are the discrete-time state transition, input matrices for the base sampling period of $h_{encoder}$ obtained by:

$$\phi = e^{Ah_{encoder}}, \tag{3}$$

$$\gamma = \int_0^{h_{encoder}} e^{At} dt B, \tag{4}$$

where $A \in \mathbb{R}^{8\times8}$, $B \in \mathbb{R}^{8\times1}$ and $C \in \mathbb{R}^{1\times8}$ represent the system matrices obtained by system identification using the data collected from the machine.

C. Multi-sensor sensing setup

As mentioned earlier, the system has two sensors i.e. linear encoders and the camera. Encoder has a sampling period of $h_{encoder}$ and its readings are available almost instantly. It measures the linear displacement of the wafer in one sample as z^w_k and the coordinate of the wafer at any instance k with respect to the reference is obtained by:

$$z_k = \sum_{n=0}^{k} z^w_k,$$

where z^w_0 is the initial coordinate of the center of the wafer, which is a known constant. The encoder feedback for the die with index i (as shown in Fig. 1 for die 1) which is about to be picked up is determined using

$$z^i_{e,k} = z_k + d^i \tag{5}$$

The camera has a sampling rate of h_{camera} and the center of its region of interest (ROI) is aligned with the origin. The images are captured after every $h_v = \left\lceil \frac{h_{camera}}{h_{encoder}} \right\rceil$ number of encoder samples, i.e. one image is captured after every h_v encoder samples. To obtain the vision feedback, the image img_k, captured by the camera at an instance k is passed to a DNN-based object detector (Section IV) which has a worst-case execution time of τ_{vision}, so the vision feedback is obtained $\tau_v = \left\lceil \frac{\tau_{vision}}{h_{encoder}} \right\rceil$ samples after the image is captured. The object

detector returns the true coordinate of the die with respect to the origin. The vision feedback, denoted by $z^i_{v,k+\tau_v}$ is also able to determine the misalignment m^i_k of the die from its ideal position.

D. Problem statement

In the context of the multi-sensor multi-die positioning system, the control problem is to design a controller with output u_k that can accurately position the die at the references. To sense the misalignment with the help of above mentioned two sensors with different sampling rates and delays, an estimator is required that can accurately estimate the actual position of the die, i.e. ideal position and the misalignment from the ideal position.

III. Related Work

Visual perception is emerging as a crucial approach for sensing in high-precision motion control systems, especially in applications like industrial manufacturing such as semiconductor die packaging as these systems are becoming more sophisticated and performance-demanding. Although sensors such as encoders, potentiometric, and hall effect sensors have been a go-to choice for these systems, they are not enough to deal with the dynamically changing environments of these systems.

In recent years, there have been significant advancements in the field of computer vision, particularly in the development of deep learning algorithms such as Deep Neural Networks (DNNs). These networks have shown promising results in several computer vision tasks, like [4] and [5] for image classification, [3], [6] and [7] for object detection, and [8] for segmentation. They provide a significant performance improvement over their classical computer vision counterparts like [9] and [10] to identify and locate an object in an image.

Visual perception has been studied extensively, especially in semiconductor manufacturing for tasks such as defect detection in [11], [12] and [13]. Where [11] uses classical computer vision, [12] and [13] use DNN-based approaches to detect semiconductor dies with faults. Even though these methods have proven to be effective in detecting and locating the dies with defects on the wafer, they are only used without any real-time constraints as seen in the die positioning applications. Therefore, using visual perception for object detection in high-precision die positioning systems presents challenges, such as the need for computational resources and significant delays in processing times. Strategies to use vision feedback in closed-loop system have been proposed in [14] and [15] which uses classical computer vision algorithm, which although requires less processing, but requires a significant manual configuration for each product and are generally less robust to the dynamic nature of these systems. Therefore incorporating DNNs as a means of visual perception in performance-demanding motion control systems remains a challenging task.

IV. DNN-Based Object Detection pipeline

To develop a robust object detection model specifically designed for locating semiconductor dies within a region of interest (ROI), we leverage the Faster R-CNN [6] architecture with a ResNet-50 [5] feature pyramid network (FPN) [16], known for its precision and speed in computer vision tasks. The Faster R-CNN model detects the object by drawing a bounding box around the object, represented by coordinates of the top-left and bottom-right corners of the box. Our approach begins with transfer learning, fine-tuning a model pre-trained on the COCO [17] dataset with our dataset generated with the help of the SIL framework described in Section VI. The dataset is composed of around 2,000 top-down images of semiconductor dies, which are used for fine-tuning and testing the model performance. This fine-tuning process allows the network to adapt its feature representations to the unique characteristics of our data, a crucial step for achieving high accuracy in object detection. The object detection pipeline shown in Fig. 3 comprises several key components each performing a crucial task to obtain the accurate position of the die. The components with their functions are discussed below.

A. Feature extractor

The ResNet-50 FPN model takes the input image and generates feature maps from the image. These feature maps are spatially scaled down (1/4, 1/8, 1/16, and 1/32) dimensions of the input image with 256 channels, generated from the output of different layers of ResNet-50 model and they contain high-level information about the composition of the input image and allow the model to identify the objects at various scales. These scaled-down feature maps are then passed to the subsequent layers to locate and classify the object that might be present.

B. Region Proposal Network

Faster R-CNN incorporates a region proposal network (RPN) to generate potential bounding box proposals around the semiconductor die. The RPN scans the feature maps produced by the backbone and provides the regions that might contain the semiconductor die. RPN does this by utilizing anchor boxes, which are pre-defined bounding boxes of different scales and aspect ratios that serve as references. At different spatial positions in the feature map, the RPN generates multiple proposals using these anchor boxes. The output of the RPN is the set of regions in the image that might contain the die (Fig. 3, RPN block) with a rough estimation of the die's size. This along with their corresponding feature maps is passed to the ROI pooling layer.

C. ROI Pooling

Since the feature maps have different scales the output of previous layers needs to be made consistent to be fed to the subsequent fully connected layers. Thus, ROI pooling needs to be applied to these regions to ensure that they all have a consistent size, the same as the dimension of the next fully connected layer.

D. Classification and Regression

The fixed-size feature maps from the ROI pooling are passed through fully connected layers. These layers branch out into two output layers - one for object classification and the other

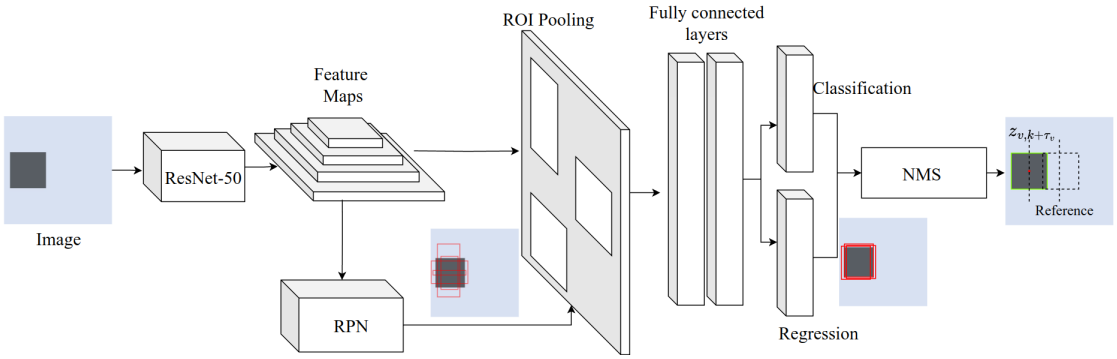

Fig. 3. DNN-based object detection pipeline for misalignment detection. Comprising of Faster R-CNN architecture with ResNet-50 FPN with NMS post-processing to obtain the vision feedback $z_{v,k+\tau_v}$ for img_k.

for bounding box regression. Object classification determines the chance of the object belonging to a certain pre-defined class which in our case is the semiconductor die. Bounding box regression refines the anchor boxes around the die and gives a precise estimation of the die position and its shape.

E. Post-processing

The output of fully connected layers results in one or more bounding boxes drawn around the semiconductor die. To refine these predictions and eliminate redundancy, Non-Maximum Suppression (NMS) is used. NMS is a crucial post-processing step that ensures only the most relevant bounding boxes are retained. NMS works by iteratively selecting the bounding box with the highest confidence score and suppressing (removing) any other boxes that have a significant overlap with it. This process continues until all boxes have been evaluated. The result is a set of non-overlapping bounding boxes, each corresponding to a distinct die in the image if multiple dies are present. The bounding box output is generated as the top-left and bottom-right corner of the die, the center of the die is the mid-point of these two. In cases where a partially occluded die is present in the image, no vision feedback is generated. The position of the relevant die is then converted to the world coordinates by using the camera calibration parameters. As these are obtained through visual feedback, they accurately tell the position of the die with respect to the reference and are thus used to compute the misalignment of the die from its ideal position.

V. DNN-BASED ESTIMATION AND CONTROL

The position estimates from the two sensors have different sampling rates and delays. Furthermore, in the case of DNN-based feedback, there can be situations when the die is not entirely present in the ROI and misalignment can not be estimated. To run the control loop at the rate of $1/h_{encoder}$ an estimator is required which can incorporate the feedback from the two sensors.

A. DNN-based estimation

The goal of the DNN-based estimation block is to give an accurate estimate of the position of the die to the motion controller. It updates the misalignment value after every successful detection of the die in the image as follows:

$$m_k^i = z_{v,k-\tau_v}^i - z_{e,k-\tau_v}^i \tag{6}$$

where m_0^i is the actual misalignment of the die, $z_{v,k-\tau_v}^i$ is the delayed vision feedback, i.e. the position dies at an instance $k-\tau_v$ obtained at an instance k and $z_{e,k-\tau_v}^i$ is the position of the die sensed by the encoder when the image was captured to obtain the vision feedback. With this, the true position of the die is estimated by:

$$\hat{z}_k = z_{e,k}^i + m_k^i \tag{7}$$

This position estimation is used by the controller to move the wafer ensuring precise positioning of the die at the reference.

B. Motion controller

The motion controller uses a PID controller with a set of references r^i. These set of references are target coordinates of the wafer that ensures i^{th} die to be at origin and are therefore defined as $r^i = -d^i$.

The error for the PID controller at any given instance is thus computed as:

$$e_k^i = r^i - \hat{z}_k \tag{8}$$

and the controller output u_k is given by:

$$u_k = K_p e_k^i + K_i \sum_{i=0}^{k} e_k^i + K_d (e_k^i - e_{k-1}^i)$$

where K_p, K_i, and K_d are proportional, integral, and derivative constants respectively.

C. Control with DNN-based estimation

The logic of the die positioning system for the multi-die scenario with DNN-based estimation is shown in Algorithm 1. The images are captured after every h_v sample (line 7) and sent to the DNN-based object detection pipeline to detect the die's true position. Depending on the position of the die in the captured image, there can be two possible outputs of the vision feedback :

1) Die is not detected: When the die is partially present in the camera ROI estimating the center of the die is not possible since extrapolation is not possible due to the irregular shape of the dies, therefore a complete die needs to be in the ROI to be able to detect misalignment. In that case, m_k^i remains 0 and the controller relies on the encoder feedback to move the die toward the reference and inside the ROI. While that happens the vision pipeline keeps running.

2) Die is detected: When the first vision feedback is obtained,

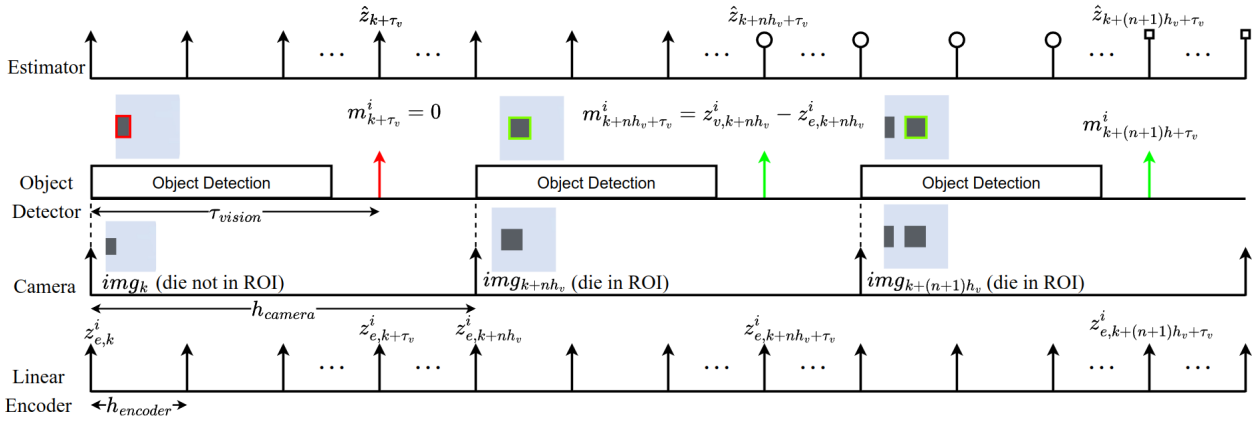

Fig. 4. Timing diagram of the encoder, camera, object detection and estimator blocks

Algorithm 1 control with DNN-based estimation

1: $k \leftarrow 0, i \leftarrow 1$
2: **for** $i \leftarrow 1$ to N **do** ▷ N = number of dies
3: $r^i \leftarrow -d^i$
4: $m^i_k \leftarrow 0$
5: **while** $k \neq i(PickupPeriod/h_{encoder})$ **do**
6: **if** k mod h_v is 0 **then**
7: $imCapture \leftarrow k$ ▷ $img_k \rightarrow$ Object Detector
8: **else if** k is $(imCapture + \tau_v)$ **then**
9: **if** $z^i_{v,k-\tau_v} \neq null$ **then**
10: $m^i_k \leftarrow z^i_{v,k-\tau_v} - z^i_{e,k-\tau_v}$
11: **end if**
12: **end if**
13: $\hat{z}_k \leftarrow z^i_{e,k} + m^i_k$
14: $e^i_k \leftarrow r^i - \hat{z}_k$ ▷ $e^i_k \rightarrow$ PID controller
15: $k \leftarrow k+1$
16: **end while**
17: **end for**

and the misalignment is known, the controller gets the updated $\hat{z}_k$ from the estimator. After every successful detection of the die $\hat{z}_k$ is updated and the detected misalignment is used by the estimator till the next update. Thus the misalignment used in equation 7 is same for $k \in [nh_v + \tau_v, (n+1)h_v + \tau_v)$, where $n \in \mathbb{W}$. The inner loop (line 5) ensures that the die is picked up at a time corresponding to its pickup period, and the outer loop (line 2) updates the die index and the reference associated with it. Fig. 4 shows how these two conditions affect the estimator output for handling vision feedback for three images img_k, img_{k+n} and $img_{k+(n+1)h_v}$. Since img_k doesn't have the die inside the ROI, the estimator output $\hat{z}_{k+\tau_v}$ is the same as encoder feedback at that instance i.e. $z^i_{e,k+\tau_v}$. Whereas for img_{k+nh_v}, since the die is present in ROI, the misalignment $m^i_{k+nh_v+\tau_v}$ is computed as per equation 6, which is used till next vision feedback is available at instance $k + (n+1)h_v + \tau_v$.

VI. SIL Verification Framework

To validate the proposed method for correcting misalignment in the semiconductor die packaging, we create a model of discrete-time dynamics of the wafer for multi-die positioning scenarios within the Unity engine [18]. The object detector, controller, and estimator are developed in Python. The Python script acts as a server and the Unity model as a client, both operating in synchronous simulation for software-in-the-loop (SIL) verification as shown in Fig. 5. The base sampling period $h_{encoder}$ is $0.125ms$, which is also used to obtain ϕ and γ in equations 3 and 4. The framework allows image captures at a rate of $1/h_{encoder}$, which is not possible to achieve in the real system but allows us to gain important insights on controller performance by varying vision feedback rates.

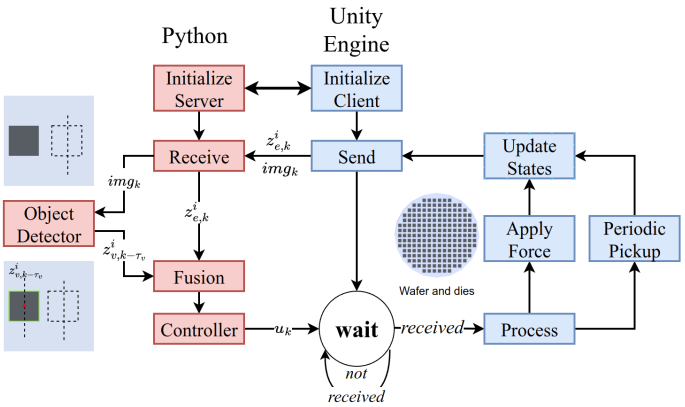

Fig. 5. Synchronous simulation framework for vision-in-the-loop system

A. Simulation flow

The simulation begins by initializing the Python server and Unity client. During the initialization process, references are generated for the controller based on the number of dies and their ideal positions. After the initialization, the main control loop starts where the Unity client sends the sensor data to the Python server which is then used by the object detector and fusion module to compute the control action as per the logic discussed in the previous section. The Unity model waits till the control action is received and processes the control action in the context of the current state. Based on the control action the force is applied to the wafer and the updated sensor readings are taken and sent to the Python server. To maintain the production throughput of more than 72,000 units per hour, a die has to be picked up every $50ms$ which is implemented in the periodic pickup block.

B. Multi-die scenario

The multi-die scenario involves 3 dies of size 200 µm × 200 µm with 300 µm center-to-center distance between neighbouring dies. With the initial wafer position, $z_0^w = 1200\,\mu m$, the ideal positions d^i of dies with respect to the wafer are $d^1 = -900\,\mu m$, $d^2 = -600\,\mu m$ and $d^3 = -300\,\mu m$ respectively. From equation 5 the ideal coordinates of the dies with respect to origin are 300 µm, 600 µm and 900 µm respectively. For the dies to be picked up every 50 ms they have to reach the reference with an error of less than ±20 µm (±10% of the die size) which corresponds to the physical limitation of the pickup mechanism. Although we consider three die scenarios, the framework can be used to generate different scenarios with different die features (like their sizes, and center-to-center distances) for many dies.

VII. Results

To analyze the behavior of the multi-die positioning system we use the above-mentioned SIL framework and the die configuration with the two misalignment profiles *m1* and *m2* described in Table I. Along with the two misalignment profiles, an ideal scenario is considered with no misalignment in die positions.

A. Performance metrics

To study the performance of the motion control system with vision feedback we consider two key metrics Steady State Value (SSV) i.e. the position of the dies at every 50 ms mark when they have to be picked up and Mean Absolute Error of the estimator during every 50 ms run.

TABLE I
Configuration of multi-die scenario

Die (i)	Initial die Position ($z^i_{e,0}$)	Control Reference (r^i)	$m1$	$m2$
1	300	900	-20	20
2	600	600	-25	25
3	900	300	-30	30

B. Closed-Loop Performance without vision feedback

The solid lines in Fig. 6 show the position of the individual dies during the motion control cycle for their pickup, without vision feedback. For the ideal scenario (shown in red) since there is no misalignment the dies reach the reference with close to zero steady-state value with the controller settling time of 49.75 ms. Whereas in cases with misalignment $m1$ (blue) and $m2$ (green), the pickup is not guaranteed as they reach the reference with the same misalignment since it is not detected by the encoder feedback.

C. Closed-loop performance with vision feedback

The dotted lines in Fig. 6 show the performance of the die positioning system with the vision feedback of $h_v = 128$ with $\tau_v = h_v - 1$ so that the vision feedback is made available to the controller as fast as possible. $h_v = 128$ means that an image is captured every $h_v \times h_{encoder} = 128 \times 0.125ms = 16ms$ and if the die is present in the camera's ROI, the vision feedback is used to estimate the true position of the die. So in both cases $m1$ and $m2$ the misalignment is detected for all the dies in time and they converge towards the reference.

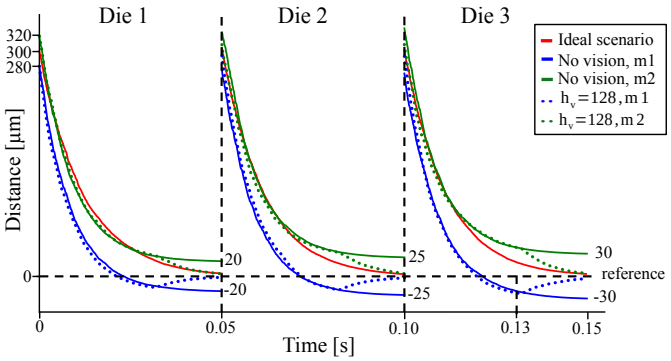

Fig. 6. Coordinates of dies for motion control with and without vision feedback for misalignment profiles m1 and m2 compared with the ideal scenario.

D. Design space exploration

The closed-loop performance depends heavily on how frequently the vision feedback can be provided to the controller i.e. vision sampling rate (h_v). To study the effect of h_v on the controller performance we consider three cases with $h_v = 8, 64, 128$. Fig. 7 and Fig. 8 show how the controller error (equation 8) is computed for different vision sampling rates. The spikes in the plots show when the first vision feedback is made available to the controller as per equation 7. The plots also show the point when the images corresponding to these corrections are captured, i.e. the first image with the die in ROI. Although the object detection pipeline keeps running, it is clear from the plots that the controller only needs the vision feedback at least once to correct the misalignment.

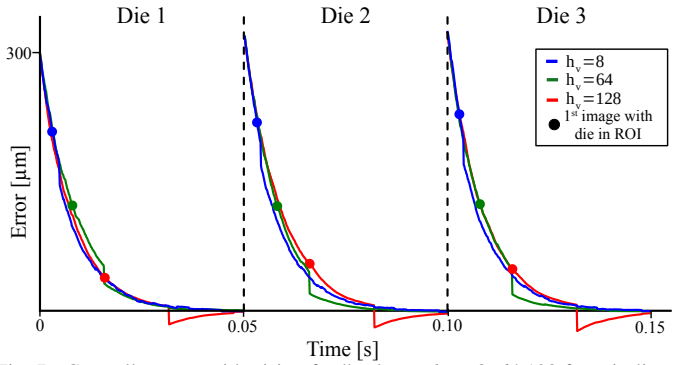

Fig. 7. Controller error with vision feedback rate $h_v = 8$, 64,128 for misalignment profile $m1$

Table II shows the performance (SSV and MAE) of the closed-loop system with different vision sampling rates. In the cases with an SSV in the range of [−20 µm,20 µm] the dies are within the acceptable misalignment range, and can be picked up. So in case of $m2$, for die 1 with a misalignment of 20 µm and the controller with no vision feedback, it reaches the reference with an SSV of 20.30 µm, which is out of the acceptable range of SSV.

E. Worst case analysis

From Fig. 7 and Fig. 8 it is clear that the controller needs at least one vision feedback to compensate for the misalignment. After receiving the vision feedback, the controller requires a

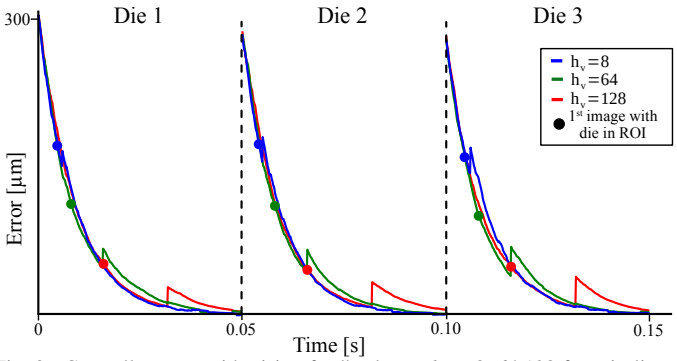

Fig. 8. Controller error with vision feedback rate $h_v = 8$, 64,128 for misalignment profile $m2$

TABLE II
PERFORMANCE OF THE CLOSED-LOOP SYSTEM

		$m1$			$m2$		
h_v	Die	SSV µm	MAE µm	Pick-up	SSV µm	MAE µm	Pick-up
Without vision feedback	1	-19.61	20.0	Yes	20.30	20.0	No
	2	-24.70	25.0	No	25.47	25.0	No
	3	-29.34	30.0	No	30.32	30.0	No
8	1	0.31	2.61	Yes	0.36	3.03	Yes
	2	0.25	2.51	Yes	0.13	3.00	Yes
	3	0.23	2.89	Yes	0.18	4.14	Yes
64	1	0.53	6.68	Yes	1.02	6.90	Yes
	2	-0.29	8.2	Yes	1.02	8.47	Yes
	3	-0.20	9.91	Yes	0.99	9.93	Yes
128	1	-1.29	13.16	Yes	3.59	12.89	Yes
	2	-1.87	16.27	Yes	3.80	16.09	Yes
	3	-2.97	19.36	Yes	4.08	19.29	Yes

certain time to compensate for the misalignment using only the encoder feedback with period $h_{encoder}$. For example in Fig. 6, the vision feedback is available at 130 ms and the controller has around 20 ms to move the die 3 to the reference. We refer to this time as the time to reject disturbance, which depends on the magnitude of the disturbance and when the first vision feedback is made available to the controller, and for the range of misalignment considered in this work, it is around 20 ms. The delay in receiving the vision feedback after the image capture is τ_v ($\approx h_v$ in our case). In addition, a delay is imposed due to the time when the die enters the ROI to the time when the processing of the corresponding image starts. In the worst case, this delay can be as long as h_v i.e. in the scenario when the die enters the ROI just after an image is captured. In total the worst case time to correct the misalignment using vision feedback is $\approx 2 \times h_v +$ time to reject disturbance. In our work, for $h_v = 128$ the worst case correction time would be $2 \times 16 + 20 = 52$ ms. This further imposes the maximum effective vision sampling period for a given application scenario.

VIII. CONCLUSION AND FUTURE WORK

In the domain of vision-in-the-loop control systems, many works have been reported in the literature that use classical vision processing algorithms, which are manually configured to detect high-level features (like edges), of the objects in the images to generate the feedback for the system. To the best of our knowledge, this is the first work that exploits the robustness of DNN-based perception in closed-loop motion control systems. We demonstrated the effectiveness of the proposed method using an industrial case study. This opens up many future directions such as model optimization to reduce the compute time, model deployment on edge compute platforms, and fusion of the vision feedback with other sensing technologies and many more.

REFERENCES

[1] S. Mohamed *et al.*, "The IMOCO4.E reference framework for intelligent motion control systems," in *ETFA*, 2023.

[2] G. van der Veen, J. Stokkermans, N. Mooren, and T. Oomen, "How learning control supports industry 4.0 in semiconductor manufacturing," in *Proceedings of the 2020 ASPE Spring Topical Meeting on Design and Control of Precision Mechatronic Systems*, 2020.

[3] V. Jain, S. Mohamed, D. Goswami, and S. Stuijk, "Vision-based multi-size object positioning," in *DSD*, 2023.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," NIPS'12, p. 1097–1105, 2012.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[9] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, 1972.

[10] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, 1999.

[11] M. P.-L. Ooi, H. K. Sok, Y. C. Kuang, S. Demidenko, and C. Chan, "Defect cluster recognition system for fabricated semiconductor wafers," *Engineering Applications of Artificial Intelligence*, pp. 1029–1043, 2013.

[12] G. Wen, Z. Gao, Q. Cai, Y. Wang, and S. Mei, "A novel method based on deep convolutional neural networks for wafer semiconductor surface defect inspection," *IEEE Transactions on Instrumentation and Measurement*, pp. 9668–9680, 2020.

[13] D. Zhan, R. Huang, K. Yi, X. Yang, Z. Shi, R. Lin, J. Lin, and H. Wang, "Convolutional neural network defect detection algorithm for wire bonding x-ray images," *Micromachines*, 2023.

[14] C. Jugade, D. Hartgers, *et al.*, "Improved positioning precision using a multi-rate multi-sensor in industrial motion control systems," in *ECC*, 2023.

[15] S. Mohamed, A. U. Awan, D. Goswami, and T. Basten, "Designing image-based control systems considering workload variations," in *58th IEEE Conference on Decision and Control, CDC 2019*.

[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017.

[17] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[18] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," 2020.