Planar feature-based motion control for near-repetitive structures

Citation for published version (APA):

DOI:
10.1016/j.conengprac.2014.03.012

Document status and date:
Published: 01/01/2014

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 15. Aug. 2019
Planar feature-based motion control for near-repetitive structures

J.J.T.H. de Best*, M.J.G. van de Molengraft, M. Steinbuch

Dynamics and Control Technology Group, Department of Mechanical Engineering, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

**Article info**

Article history:
Received 3 May 2013
Accepted 29 March 2014
Available online 22 May 2014

Keywords:
Visual servoing
Motion control
Repetitive structures

**Abstract**

This paper focuses on the motion control for machines used for the production of products that inherently consist of equal features placed in a repetitive pattern. In many cases the repetitiveness of these structures is prone to imperfections, for example due to thermal expansion, such that the distance between successive features deviates. As a consequence the metric positions of the features of such near-repetitive structures are unknown a priori such that setpoints cannot be created a priori. The considered motion task in this paper is to position a tool relative to the features of a near-repetitive structure with an accuracy of < 10 μm. Instead of metric positions novel two-dimensional feature-based positions will be used that are obtained from a camera capturing images at 1 kHz for feedback, resulting in a direct visual servoing control approach. The robustness with respect to imperfections in the repetitiveness is investigated and the design is validated on an experimental setup.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Many production processes take place on repetitive structures. In each of these processes one or more consecutive steps are carried out on the particular features of the repetitive structure to create the final product. Such production machines often consist of a tool and a stage or carrier on which the repetitive structure is to be processed. The considered control task is therefore to position the tool relative to the features of the repetitive structure. In current industrial practice, local position sensors such as motor encoders are used to measure the position of the tool and the stage separately. Often the absolute reference points of these measurements do not coincide, such that the final accuracy of the alignment of the tool directly relies on properties such as thermal stability, mechanical stiffness and assumptions on the pitch between successive features of the repetitive structure. Any falsification of these assumptions results in a poor alignment.

Possible solutions for the posed problem can be found in the field of visual servoing (Hill & Park, 1979) or visual servo control (Chaumette & Hutchinson, 2006; Hutchinson, Hager, & Corke, 1996), in which machine vision data is used in the servo loop to control the motion of a system. Extensive overviews on the topic of visual servoing can be found in Kragic and Christensen (2002), Malis (2002), Hutchinson et al. (1996), Corke (2001), and Hashimoto (2003). Many classifications are known within visual servoing.

We will now briefly discuss these and position our work in the field of visual servo control. The first classification makes a distinction between indirect and direct visual servoing (Sanderson & Weiss, 1980). Indirect visual servoing has a hierarchical or cascaded control architecture in which the vision system provides (velocity) setpoints to low level joint controllers. Indirect visual servoing is often split up into static look-and-move and dynamic look-and-move approaches. In static look-and-move three steps are taken consecutively: (1) the system “looks” at the scene and measures the relative position between the tool and the feature, (2) based on the difference between the current position and the desired position a trajectory is planned and (3) the system “moves” to the desired position. In the dynamic look-and-move approach the above steps are executed in parallel. By far, most literature adopt the dynamic look-and-move approach (Chaumette & Hutchinson, 2006, 2007; Corke & Hutchinson, 2001; Créteil & Chaumette, 1997; Espiau, Chaumette, & Rives, 1992). In direct visual servo control the visual controller computes the input (typically torques and/or forces) to the plant directly (Ishii, Nakabo, & Ishikawa, 1996; Ishikawa, Morita, & Takayanagi, 1992; Nakabo, Ishikawa, Toyoda, & Mizuno, 2000). The second classification is the eye-in-hand versus the eye-to-hand visual servoing. The first configuration has the camera mounted to the tool. In this case it is often assumed that there is a known kinematic relation between the tool and the camera in order to position the tool relative to the feature. The second configuration has the camera mounted in the workspace. The eye-in-hand configuration has a precise sight of the scene relative to the camera, whereas the eye-to-hand configuration often has a more global sight which might be less precise. Blocking of the field of view is more likely to happen in the latter configuration. Position based or...
PBVS versus image-based visual servoing or IBVS is the third classification. In both concepts features are extracted from the image. However, in PBVS a cartesian position is estimated from these features and the control law is executed in the cartesian domain (Martinet & Gallice, 1999; Thuilot, Martinet, Cordesses, & Gallice, 2002; Wilson, Williams Hulls, & Bell, 1996). On the other hand, in IBVS the control law is computed directly on the basis of the image coordinates of the features (Espiau et al., 1992; Weiss et al., 1987). The last classification is concerned with endpoint open-loop or EOL versus endpoint closed-loop visual servoing or ECL. In EOL only the target feature is within the field of view, whereas in ECL both the tool and the target feature are within the field of view. In the latter the relative position between the target feature and the tool can be computed, whereas in the first this relies on how well the relation between tool position and camera position is known (see also eye-in-hand versus eye-to-hand). Note that EOL is often less computational expensive since only the target feature is to be detected and not the tool as is the case in ECL.

This work uses a direct eye-in-hand endpoint open-loop visual servo control approach. Regarding the PBVS versus IBVS classification the authors introduced a new control design paradigm in de Best, van de Molengraft, and Steinbuch (2009, 2012) in which feature-based position measurements on the basis of camera images in combination with non-collocated visual feedback is used leading to feature based visual servoing or PBVS. As such, motion setpoints can be defined from feature to feature without knowing the exact absolute metric position of the features beforehand, while still achieving a high positioning accuracy. The proposed method was restricted to the one-dimensional case. In practical applications the repetitive structure in general will contain a two-dimensional grid pattern, like for example the repetitive structure depicted in Fig. 1(a), which shows diodes on a wafer. Therefore, in this paper the feature domain is extended towards two dimensions. Furthermore, in de Best et al. (2012) the feature-based position is constructed by piecewise linear interpolation between successive features. When passing a feature, a different pitch between the current features is considered. Due to the piecewise linear interpolation the feature-based position is continuous when passing a feature but the feature-based velocity is not and switches instantaneously. As a result, undesired transient responses are observed. In this paper, higher order interpolation will be implemented to reduce these undesired transient responses. The introduction of feature-based positions results in a straightforward setpoint creation from one feature to another target feature, referred to as feature-to-feature movements, without having to know the absolute metric position of the target feature. However, besides these feature-to-feature movements, many production processes require metric movements of the tool with respect to the feature, like for example engraving text on each feature. These movements are referred to as relative feature movements. Typical movements in such applications are therefore constructed by repeatedly alternating between (1) feature-to-feature movements from the current feature to the target feature and (2) metric relative feature movements with respect to the target feature. These relative feature movements will be implemented in the feature-based control approach, so the contributions of this paper are fourfold: (1) the feature-based position measurement is extended towards two dimensions, (2) the piecewise linear interpolation is extended to higher order interpolation to reduce the transient responses when passing features, (3) next to feature-to-feature movements, relative feature movements are implemented, to increase the versatility of programmable movements and (4) a stability analysis is presented to prove robust stability of the closed-loop system.

The rest of the paper is organized as follows. In Section 2 the notation with respect to the repetitive structure and the different coordinate representations will be presented. Section 3 will first introduce two-dimensional feature-based positions, followed by higher order feature interpolation. At the end of Section 3 the implementation of relative feature movements will be discussed. The experimental setup that will be used for validation will be given in Section 5. The control design and stability analysis will be given in Section 6. Finally, conclusions will be given.

2. Notation

Throughout this paper a repetitive structure will be used that consists of equal features ordered in a near-rectangular repetitive pattern. A practical example is depicted in Fig. 1(a) which shows diodes on a wafer. A schematic representation of such a repetitive structure is given in Fig. 1(b) where the features are circular black dots on a white background.

The image captured by the camera, denoted as $I$, has a height $I_h$ and width $I_w$ pixels and captures only a part of the repetitive structure. The features have a diameter of $D$ pixels and are placed in a rectangular repetitive pattern. The nominal pitch between features is $P$ pixels in both horizontal and vertical directions. In this work pitch imperfections will be considered, which can occur for example due to inaccurate preceding process steps, local stretching of the structure when flexible plastic or metal foil is used as product carrier or thermal expansion of the structure.

![Fig. 1. A part of a two-dimensional repetitive structure. (a) Diodes on a wafer. (b) Schematic representation of a repetitive structure.](image-url)
The pitch imperfection is denoted by $\Delta P$, with $0 < \Delta P \ll P$, such that each pitch $P$ satisfies

$$\mathcal{P} - \Delta P \leq P \leq \mathcal{P} + \Delta P. \quad (1)$$

More specifically, once a feature is found, the other features are expected in the shaded areas in Fig. 1(b). The position and the size of the shaded area are related to the nominal pitch $P$ and the pitch imperfection $\Delta P$. These pitch imperfections are allowed and will be taken into account such that precision requirements of the repetitiveness of the pattern can be less strict, while at the same time being able to position accurately with respect to the features.

Throughout this paper two different coordinate representations will be used, which are metric pixel coordinates and feature coordinates. A point $p$ within the image that is expressed in pixel coordinates is represented as $p^i_p$. Similarly, $p^o_p$ represents the point $p$ in feature coordinates. In the remainder of this paper subscripts given in Table 1 will be used. Using this notation the pixel coordinates of the center of the top left feature for example are denoted as $p^i_{tl} = (x^i_{tl}, y^i_{tl})^T$. Similar notations are used for the top right feature $p^i_{tr}$, the bottom left feature $p^i_{bl}$ and the bottom right feature $p^i_{br}$. The point of interest, which initially is taken as the center of the image sensor, is denoted by $p_r$.

### Table 1

<table>
<thead>
<tr>
<th>Subscript</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>Left</td>
</tr>
<tr>
<td>$r$</td>
<td>Right</td>
</tr>
<tr>
<td>$t$</td>
<td>Top</td>
</tr>
<tr>
<td>$b$</td>
<td>Bottom</td>
</tr>
<tr>
<td>$i$</td>
<td>Interest</td>
</tr>
</tbody>
</table>

3. Feature-based positions

This section will first introduce the feature domain. Next, the detailed steps in obtaining feature-based positions in the feature domain will be discussed, which are the feature detection, bilinear feature interpolation and the high order feature interpolation. Finally, relative feature movements will be presented by combining the metric domain with the feature domain.

#### 3.1. The feature domain

In this work we adopt the end-point open-loop visual servoing topology (Hutchinson et al., 1996), which states that only the feature is registered with the camera as opposed to end-point closed-loop visual servoing, in which both the tool and the feature are registered. For the purpose of explaining the feature-based position measurement the tool is assumed to be located at the center of the image. For laser cutting/engraving applications in which the image can be captured through the same lens as used for directing the laser this can be realized. In many other applications there might be an offset between the tool position and the center of the image, i.e., a different point of interest as this will be called later on.

The control task is to position the center of the image with respect to the target feature. The target feature however might be outside the field of view, since for resolution purposes only a small part of the repetitive structure is observed. Due to a priori unknown pitch imperfections, the absolute position of the target feature in the pixel or metric domain is therefore not known on beforehand. Hence, metric setpoint creation cannot be done offline. To solve this problem, position measurements in the so-called feature domain are introduced. The advantage of feature-based positions over metric positions is that the positions of each feature in the feature domain are known a priori, such that there is no need for online trajectory generation. Therefore, instead of using metric position measurements, novel feature-based position measurements will be used for feedback, see Fig. 2. The system captures images $I$ of the scene, which together with the pixel coordinates of the point of interest $p^i_p$ are fed into the image processing block $\text{IP}$ that gives a two-dimensional feature-based position $p^o_p$. The $\text{IP}$ image processing block consists of finding the pixel positions of the features surrounding the center of the image. The main algorithm for determining these pixel positions is by calculating the center of gravity of the feature within a search area. This algorithm is stated in pseudocode in Fig. 3. At this point, the point of interest is taken as the center of the image $p^o_p = (h_i/2 \ l_w/2)^T$, since this point is to be positioned with respect to the features. The feature-based position $p^o_p$ is compared to the feature-based reference $p^o_i$ and fed to a controller $K$ that generates the input $u$ to the system.

The two-dimensional measurement principle of the feature-based positions will be explained using Fig. 4. The figure shows

![Fig. 2. Feature-based control approach.](image-url)

![Fig. 3. Finding the pixel positions of the feature is done via the center of gravity.](image-url)

![Fig. 4. Feature-based positions. The frame indicates the field of view.](image-url)
a part of a near-repetitive structure that is captured by the camera. The pitches between neighboring features are such that they satisfy (1) in both directions. For the proposed measurement method it is required that four features will be completely within the field of view, such that they enclose the point of interest \( p_r \), indicated by the cross, see Fig. 4. Each feature is assigned an integer feature-based position, irrespective of their mutual pixel distance, i.e., feature-based positions are determined by counting features starting from the top left corner of the image. So, \( p_{ri}^f = (x_{ri}^f, y_{ri}^f) \in \mathbb{Z}^2 \) is the feature-based position of the top left feature which in this example is \((1 \, 1)^T\). The feature-based positions of successive features can be obtained by simply incrementing the values of \( x_{ri}^f \) and/or \( y_{ri}^f \).

Since the goal is to position the center of the image with respect to the target feature, we want to express the inter-feature position of the center of the image, here denoted by the point of interest \( p_{si} \) in the feature-based coordinates. More specifically, \( p_{si}^f = (x_{si}^f, y_{si}^f) \in \mathbb{Z}^2 \) is to be found. To obtain a unique feature-based position it is required that the positions \( x_{si}^f \) and \( y_{si}^f \) increase monotonically between

\[
x_{si}^f \leq x_{si}^f \leq x_{si}^f + 1, \tag{2}
\]

and

\[
y_{si}^f \leq y_{si}^f \leq y_{si}^f + 1. \tag{3}
\]

Therefore, the idea of de Best et al. (2009) is extended towards two dimensions as follows. If the point of interest is perfectly aligned with one of the features, its position is the integer two-dimensional feature-based position. In between features an interpolation is used as is indicated by the gray dashed grid lines in Fig. 4. Here, the horizontal and vertical grid lines indicate lines of equal \( x \) and \( y \) feature-based positions, respectively. More specifically, they connect equal fractions of the horizontal and vertical lines between the features. The feature-based position of the point of interest in Fig. 4 is given by \( p_{si}^f = (1_5^T, 1_5^T)^T \).

To obtain the correct feature-based position it is essential to determine which features are in the field of view. A problem arises when the repetitive structure moves with a velocity larger than one pitch per sample. In that case it is desired to track which features are in the field of view. A steady state Kalman (1960) filter will be used for that, from which only a one step ahead prediction will be used to predict (1) which features will be in the field of view and (2) where these features will be located in the field of view. The first point assures that the feature-based position is incremented when new features enter the field of view, whereas the second point will generate the initial pixel position estimate \( \hat{p}_r \) of the feature closest to the point of interest, which is used in the feature detection, see Section 3.2.

In the next sections the consecutive steps needed for obtaining the feature-based position will be explained in more detail, which involve (1) feature detection, (2) bilinear feature interpolation and (3) higher order feature interpolation.

### 3.2. Feature detection

This section will describe the first step towards obtaining a feature-based position, which is the accurate detection of the pixel coordinates of the four features that enclose the point of interest. These coordinates will be used in the second step as described in the next section, which involves the bilinear feature interpolation.

Initially, assume that a pixel position estimate \( \hat{p}_r = (\hat{x}^r, \hat{y}^r)^T \) generated by the Kalman filter is available for the feature that is expected to be closest to the point of interest. In Fig. 5 the bottom right feature is closest to the point of interest. The pixel position estimate of this feature is given by the gray cross.

A rectangular search area is defined around the estimate \( \hat{p}_r \), with a width of \( S_w \) pixels and a height of \( S_h \) pixels. The search area should be such that it completely confines a single feature. The size of the search area is directly dependent on (1) the feature size, (2) the feature position variation (pitch imperfections) and (3) the quality of the estimate \( \hat{p}_r \). With the introduction of the search area it is possible to search for a single feature within the search area, such that labeling implementations to distinguish between multiple features, which cause a computational overhead, can be eliminated in the image processing steps. Stated otherwise, the image processing can be done in a smaller timespan by using a priori knowledge about the repetitiveness of the structure. Referring to Fig. 1(a), in our considered application (diodes on a wafer) the pitch between the diodes is approximately 300 \( \mu \)m. The size of the diode (the squares in Fig. 1(a)) is 250 \( \mu \)m. However, in the image processing we only detect the pad of the diode, i.e., the circular shape within the diode. This has a dimension of approximately 150 \( \mu \)m, which is much smaller than the pitch. By choosing a sensible search area it is a reasonable assumption that only a single feature is within this search area. Moreover, if more than one feature would be in the search area, additional image processing can be added to distinguish which is the feature of interest and which is not. This however might increase the image processing time. But still an improvement is expected compared to processing the whole image, where definitely labeling issues need to be solved.

In our case, the pixel position of the center of the feature \( p_{si} \) is determined by first thresholding the search region followed by a center of gravity calculation (van Assen, Egmont-Petersen, & Reiber, 2002). For more complex feature shapes more elaborated image processing techniques can be applied such as hough transforms or template matching. From Fig. 5 it can be seen that the measured \( p_{si} \) can be different from \( \hat{p}_r \) indicating the estimation error. In this case the bottom right feature is found, i.e., \( p_{si} = P_{si}^r \), since it is located at the bottom right of the point of interest. The pixel position of the three remaining features can be estimated as follows:

\[
\hat{p}_{si}^f = p_{si}^f - (0 \, F)^T, \tag{4}
\]

\[
\hat{p}_{si}^r = p_{si}^r - (F \, 0)^T. \tag{5}
\]
\[ p^p_{it} = p^p_{br} - (P P)^T, \]  

with \( P \) being the nominal pitch between features. Similar to finding the pixel coordinates of the feature \( p^p_{it} \), search areas can be defined around these estimates and the pixel coordinates of the remaining features \( p^p_{br}, p^p_{bl} \) and \( p^p_{tl} \) can be calculated. During these operations it is important to check that the four features are enclosing the point of interest, i.e.,

\[ p^p_i = \left\{ \sum_{t=1}^4 \alpha_t p^p_t | \forall \alpha_t \in \mathbb{R}, \alpha_t \geq 0, \sum \alpha_t = 1 \right\}, \]

with \( L = (tl, tr, bl, br) \). At the end of this step the pixel positions \( p^p_{it}, p^p_{bl}, p^p_{br} \) and \( p^p_{tl} \) are known. These coordinates will be used in the second step, the bilinear feature interpolation.

### 3.3. Bilinear feature interpolation

This section will describe the second step for obtaining the feature-based position, which is the bilinear feature interpolation. The bilinear feature interpolation will use the detected pixel coordinates of the four features that enclose the point of interest as explained in the previous section. The bilinear feature interpolation will be extended in the next section leading to second order feature interpolation.

The bilinear feature interpolation is based on the similar idea as in de Best et al. (2009) and will be explained using Figs. 4 and 6. In Fig. 6 two lines intersect the point of interest. The vertical line connects the point \( p^v_t = (x^v_t, y^v_t)^T \) with the point \( p^v_b = (x^v_b, y^v_b)^T \), whereas the horizontal line connects the point \( p^h_t = (x^h_t, y^h_t)^T \) with the point \( p^h_b = (x^h_b, y^h_b)^T \). This horizontal line is constructed such that

\[ A = \frac{x^h_t-x^h_b}{x^h_t-x^h_t} \frac{y^h-t-y^h_b}{y^h_t-y^h_t} = \frac{x^h_t-x^h_b}{x^h_t-x^h_b} \frac{y^h_t-y^h_b}{y^h_t-y^h_t}. \]

In the case of Fig. 6, the value of \( A \) is \( \frac{3}{2} \). For the vertical line the same reasoning holds. It connects equal \( B \) fractions of the top and bottom lines, so

\[ B = \frac{x^v_t-x^v_b}{x^v_t-x^v_t} \frac{y^v-t-y^v_b}{y^v_t-y^v_t} = \frac{x^v_t-x^v_b}{x^v_t-x^v_b} \frac{y^v_t-y^v_b}{y^v_t-y^v_t}. \]

The value for \( B \) in Fig. 6 is \( \frac{1}{4} \). Moreover, it can be shown that

\[ A = \frac{x^h_t-x^h_b}{x^h_t-x^h_t} \frac{y^h_t-y^h_b}{y^h_t-y^h_t}, \quad B = \frac{x^v_t-x^v_b}{x^v_t-x^v_t} \frac{y^v_t-y^v_b}{y^v_t-y^v_t}, \]

such that the values of \( A \) and \( B \) can be expressed as a function of four pixel coordinates of the enclosing features \( p^p_{it}, p^p_{br}, p^p_{bl}, p^p_{tl} \) and the pixel coordinates of the point of interest \( p^p_{it} \). The analytic expressions for \( A \) and \( B \) are not given here due to space limitations. The two-dimensional feature-based position can now be written as

\[ x'_f = x^f_t + A, \quad y'_f = y^f_t + B, \]

with \( A \in \mathbb{R}, 0 \leq A \leq 1, B \in \mathbb{R} \) and \( 0 \leq B \leq 1 \).

### 3.4. Second order feature interpolation

In this section the third step in obtaining the feature-based position measurement will be explained which involves a second order interpolation. When positioning the camera from one feature to another, the point of interest will leave the current area spanned by the four features and enter the next area spanned by four different features, which is referred to as feature frame transitions in the remainder of this work. Due to pitch imperfections the mutual distances between the features can be different after feature frame transitions. Using the bilinear interpolation of (11) the feature-based position is continuous during feature frame transitions, however the feature-based velocity is not and switches instantaneously. These switching velocities cannot be tracked by the controller and will result in transient position responses whenever feature frame transitions occur. These responses are comparable to the responses obtained when first order trajectories are applied to a closed-loop system. In the remainder of this section, the switching behavior will be explained in more detail and second order functions will be incorporated in the feature interpolation such as to prevent the feature-based velocity from switching and therefore to reduce the undesired transient position responses.

To explain the switching feature-based velocity, Fig. 7 will be used, where the current feature-based position is \( p^f_t = (2\frac{3}{4}, 3\frac{3}{4})^T \). Suppose the repetitive structure is moving with a constant pixel velocity \( v_\ell = (0, -P)^T \) pixels/s. As a result the feature-based position in the \( y \) direction is increasing. If \( y'_f < 4 \), the pitch in \( y \) direction is \( P = P \) and therefore the feature-based velocity in \( y'_f \) is \( 1 \) f/s. However if \( y'_f \geq 4 \), the pitch is \( P = 0.7P \) such that the feature-based velocity instantaneously becomes \( \frac{1}{2} f/s \). By introducing a different feature-based position measurement as

\[ x'_f = x^f_t + g(A), \quad y'_f = y^f_t + h(B), \]

Fig. 7. The features in the image are moving to the left with a velocity of \( P \) pixels/s. The feature-based velocity switches when \( y'_f \geq 4 \). This can be seen in the image by the interpolated grid lines that are closer to each other, when \( y'_f \geq 4 \).
we can design the functions \( g \) and \( h \) such that the feature-based velocity does not switch but changes smoothly. In the remainder the focus will be on the design of the function \( h \), while the design of \( g \) can be done in a similar way. The following constraints can be constructed for designing the function \( h \)

\[
h(0) = 0, \quad h(1) = 1, \quad \frac{dh(0)}{dB} = \frac{P}{P} = \frac{dh(1)}{dB} = \frac{P}{P} \tag{13}
\]

where \( P \) is the momentary pitch, i.e., the length of the horizontal grid line intersecting the point of interest, which is \( P \) if \( y_f < 4 \) and \( 0.7P \) in the case \( y_f \geq 4 \). The first two constraints imply that the feature-based position is continuous across feature frame transitions and is composed of two integer values when the point of interest \( p_i \) is aligned with a feature. The second two constraints imply that the feature-based velocity is constant, i.e., does not switch, across the feature frame transitions. Fig. 8 shows an example of the function \( h(B) \) for the cases where \( P = P \) and \( P = 0.7P \). It can be seen that if \( P = P \) the function is simply \( h(B) = B \). If \( P = 0.7P \) the function \( h(B) \) satisfies the aforementioned conditions. In this case the function \( h(B) \) is a piecewise quadratic function, i.e.,

\[
h(B) = \begin{cases} 
2 - \frac{2P}{P} B^2 + \frac{P}{P} B & \text{if } B < 0.5, \\
-2 + \frac{2P}{P} B^2 + \left(4 - \frac{3P}{P}\right) B - 1 + \frac{P}{P} & \text{if } B \geq 0.5.
\end{cases}
\tag{14}
\]

In the choice of the function \( h(B) \) a trade-off is made. The function \( h(B) \) in this case is chosen to have a non switching velocity when features are passed, so as to reduce the transient response caused by this switching. If for example higher order functions would be designed, even the feature-based acceleration and jerk can be made continuous. However, one should take into account that this interpolation has a high influence on the gain of the system, i.e., larger values of \( \frac{dvh(B)}{dB} \) lead to a higher momentary gain of the system and vice versa. As a result the control design needs to cope with high gain variation to prevent potential stability problems as will be explained in Section 6.

### 4. Relative feature movements

In the production of repetitive structures, the tool is typically moved from one feature to the next feature, where at every feature a processing step is executed. This processing step can for example be a pick and place action or jetting droplets using ink jet printing technology. For these processing steps it is sufficient to move from one feature to the next, i.e., no additional movements of the tool with respect to the feature have to be carried out. In processing steps, like for example engraving or cutting, additional movements of the tool are necessary before going to the next feature. This section will discuss how, next to feature-to-feature movements, these so-called relative feature movements can be incorporated in the feature-based control design approach.

From an operators point of view it would be preferable to design (1) a reference \( r_f^P(t) \) for performing feature-to-feature movements expressed in the feature domain and (2) a reference \( r_f^T \) for performing additional movements of the tool with respect to the feature expressed in the metric domain. Feature-based position measurements were introduced in order to handle feature-to-feature movements. Previously, the point of interest was taken static as the image center, \( p_f^P = (l_h/2 \ l_w/2)^T \). By prescribing the
metric position of the point of interest in time however, i.e., \( r_{p_{i}}(t) = p_{r_{i}}(t) \), relative feature movements can be induced. Hence, with the tool still assumed to be in the image center, the relative movement of the feature with respect to the tool is obtained.

An example of such a setpoint is given in Fig. 9. During the first 0.2 s a diagonal feature-based movement is performed from feature zero to feature one. After arriving at this feature, the relative feature movement will be carried out from \( t = 0.2 \) s to \( t = 1 \) s. When this movement is completed, the next feature-to-feature movement will be carried out. By repeating this sequence, the tool is moved along the features and each feature is processed.

The final control scheme is now given in Fig. 10. The feature-based reference \( r_{f_{i}} \) is applied to the closed-loop system. The controller \( K \) is connected to the system. The input of the system \( u \) is the applied forces, whereas the output of the system is the captured images \( I \). These images together with the metric reference \( r_{m_{i}} \) for relative feature movements are processed in the image processing block \( IP \) which performs the feature detection and the feature interpolation. The output of this block gives the feature-based position \( p_{f_{i}} \) that is used for feedback.

5. Experimental setup

The considered industrial application is an xy-wafer stage and is depicted in Fig. 11. On the stage a wafer is clamped which contains the small (250 × 250 \( \mu \)m) discrete semiconductor products, the so-called dies. A particular production process to obtain the final product is the picking and placement and wire bonding of each individual semiconductor. Therefore, the tool is to be positioned accurately with respect to each semiconductor. As mentioned, the tool is assumed to be positioned at the center of the image sensor, such that the problem at hand is transformed into controlling the xy-wafer stage such that the semiconductor is accurately positioned with respect to the center of the image, where the camera is used as position sensor instead of the onboard motor encoders. The feature-based position as explained previously is used for feedback. In the next sections the individual components of the experimental setup will be discussed.

5.1. Camera and optics

A frame is mounted above the stage which supports a Prosilica GC640M high-performance machine vision camera (Prosilica, 2009) with Gigabit Ethernet interface (GigE Vision). The camera generates 8 bits monochrome images and is capable of reaching a frame rate of 197 Hz full frame (near VGA, 659 × 493). To increase the frame rate of the camera to 1 kHz and to reduce the amount of data transport from the camera only a part of the sensor is read out as large as 90 × 90 pixels. Note that the framerate of the camera determines the maximum sampling frequency and not the amount of calculation time needed for the image processing steps. The camera supports jumbo frames of up to 9200 bytes, such that the entire image fits into a single packet. This reduces the CPU load due to less incoming data packets. The magnification of the MC1.00X lens (Opto Engineering, 2010) is one. The pixel size of the camera is 9.9 \( \mu \)m such that the field of view is approximately 0.9 × 0.9 mm which is large enough to view nine dies in a three by three formation. To further reduce the delay and to minimize image blur the exposure time is set as small as 60 \( \mu \)s.

5.2. Illumination

With a single power LED in combination with a half mirror placed under an angle of 45° with respect to the lens coaxial illumination is realized, see Fig. 12. Light from the power LED is deflected towards the wafer by the half mirror. The light is then reflected by the semiconductor products, and travels back through the half mirror again and the lens forming the image on the image sensor of the camera. A typical image of a part of the wafer is given in Fig. 1(a).

5.3. Host PC and data acquisition

The camera is connected to a host PC running a 2.6.28.3 low-latency Linux kernel on which the necessary image processing is done and the control law \( K \) is calculated. The real-time executable is built using the real-time workshop (RTW) of Matlab/Simulink. Furthermore, the data-acquisition is realized using an EtherCAT (Jansen & Buttner, 2004) data-acquisition system, where DAC, I/O,
and encoder modules are installed to drive the current amplifiers of the motors, to enable the amplifiers and to measure the position of the xy-wafer stage at the motor side via the on-board motor encoders. Hence, this motor encoder position is only used for evaluation purpose and is not used in the final control algorithm as such.

A schematic representation of the setup is shown in Fig. 13. The output of controllers as calculated on the host PC, i.e., control command (red), is two current setpoints for the amplifiers. These are translated into analog values via the DAC. These values are input to the current amplifiers together with encoder information, which is used for commutation purposes of the three phase motors. The encoders are also fed into the EtherCAT modules which provides us with safety information (blue), particular for detecting end of stroke. The captured image (green) is an 8-bit gray valued image, transferred via UDP.

6. Control design and stability analysis

In this section first the design of the controller $K$ will be presented. Due to the introduction of feature-based positions, the gain of the plant varies dependent on the momentary pitch between successive features. Therefore, a stability analysis will be given for investigating the closed-loop stability while robustness against pitch imperfections is guaranteed.

The transfer from input $u$ to the feature-based position output $p^f$ is denoted by $G$, see also Fig. 10. For this transfer a MIMO frequency response function (FRF) has been measured with a repetitive structure with a pitch of $P_\text{f}$. The two dimensional system identification is done via a three point identification in which the MIMO sensitivity and the MIMO process sensitivity are identified.

As a result of this movement a feature-based velocity $v^p$ is induced. The same feature-based velocity would occur if the pixel position of the point of interest would virtually move in the opposite direction

$$\hat{p}_\text{f}^p = -v^p,$$

while the pixel positions of the features are kept static. The relation between the pixel velocity $v^p$ and the feature-based velocity $v^p$

$$v^p = \frac{\partial p^f}{\partial x^p}.$$

Therefore, the pitch imperfection are written as a fraction of the nominal pitch, so \( \Delta P = \alpha P \), with \( 0 \leq \alpha < 0.25 \). The upper bound for \( \alpha \) indicates the validity of the bilinear feature-based interpolation. This is shown in Fig. 17. In this figure the pitch imperfection \( \Delta P = 0.25P \). It can be seen that three features are forming a straight line. If the imperfections are allowed to be larger, then the four features do not span a convex set any more. In that case we could not speak anymore of a repetitive structure. As was stated in the beginning the pitch imperfection is a lot smaller than the nominal pitch, i.e., \( \Delta P \ll P \), typically an order smaller or even less.

For each value of \( \alpha \) maximum and minimum values of \( J \) can be found over all possible feature position configurations satisfying the bounds as defined in (1). In this case the maximum and minimum values of the elements of the Jacobian \( J \) are found to satisfy analytic functions of \( \alpha \) that are given by

\[
J_d = \frac{1}{P} \left( 0.125 \frac{0.25 - \alpha}{2} + 0.5 \right) \left( 2\sqrt{10} - 4 \alpha + 1 \right),
\]

\[
J_* = \frac{1}{P} (1 - 2\alpha) \left( 4 - 2\sqrt{10} \alpha + 1 \right),
\]

\[
J_{ud} = \frac{1}{P} \left( 0.125 \frac{0.25 - \alpha}{2} - 0.5 \right) \left( 2\sqrt{10} - 4 \alpha + 1 \right),
\]

\[
J_{ud} = \frac{1}{P} \left( -0.125 \frac{0.25 - \alpha}{2} + 0.5 \right) \left( 4 - 2\sqrt{10} \alpha + 1 \right).
\]

This result is graphically shown in Fig. 18. In this figure the four elements of the Jacobian \( J \) are shown. On the horizontal axis the value of \( \alpha \) is given, whereas on the vertical axis the possible values are indicated by the gray shaded areas. As can be seen in this figure an increase of \( \alpha \) leads to a larger set of possible values of \( J \). Using this figure, one can determine what possible values of the Jacobian can be present given a specific pitch imperfection. The nominal pitch \( P \) between consecutive features is 27 pixels. It can also be seen that if \( \alpha = 0 \), i.e., no pitch imperfections, the Jacobian satisfies (19).

For proving closed-loop stability the system is written as a linear differential inclusion (LDI) (Boyd, El Ghaoui, Feron, & Balakrishnan, 1994). In obtaining such an LDI first it will be showed that at a specific value of \( \alpha \) an arbitrary Jacobian can be written as the convex combination of all possible minimum and maximum value combinations of the Jacobian. The number of possible combinations is the number of elements of the Jacobian

\[
\begin{bmatrix}
\begin{array}{cc}
1/t & 0 \\
0 & 1/t
end{array}
\end{bmatrix}
\]

However, in the presence of pitch imperfections, the value of \( J \) depends on the pixel coordinates of the four features enclosing the point of interest. Hence, they depend on the momentary pitch \( P \) between successive features. Therefore, the Jacobian \( J \) becomes a non-linear mapping from \( \mathbf{v} \) to \( \mathbf{v}' \). At this point it is investigated how the Jacobian varies as a function of the pitch imperfection.

Fig. 16. Open-loop in x (black) and y (gray) directions.

Fig. 17. Validity of the feature interpolation concept.

Fig. 18. The values of the Jacobian \( J \) as a function of \( \alpha \).
squared, so $4^2 = 16$. Therefore an arbitrary Jacobian can be written as
\[ J = \left\{ 16 \sum_{m=1}^{16} \beta_m l_m L_m \in J, \forall \beta_m \in \mathbb{R}, \beta_m \geq 0, \sum_{m=1}^{16} \beta_m = 1 \right\}, \quad (24) \]
with
\[ J = \{ J_1, J_2, \ldots, J_{16} \} \]
\[ J = \left\{ \begin{bmatrix} j_{11} & j_{12} \\ j_{21} & j_{22} \end{bmatrix} | j_{11}, j_{12}, j_{21}, j_{22} \in \mathbb{U}_d, J_{1d} \in \mathbb{U}_d, J_{2d} \in \mathbb{U}_d \right\}. \quad (26) \]

To prove stability of the closed-loop system the system $G(z)$ is written as the multiplication of $R(z)$ and an integrator $I(z)$. The system $R(z)$ maps the inputs to the system $u$ to the pixel velocity of the features $v^p$. In $I(z)$, these pixel velocities $v^p$ are mapped to the feature-based velocities $p^p$ by the Jacobian $J$, which after integration lead to the feature-based position $p^f$.

The subsystems $R(z)$ and $I(z)$ are given by
\[ R : \left\{ \begin{align*}
\dot{x}_R(k+1) &= A_RW_R(k) + B_Ru(k), \\
v^p(k) &= C_Rx_R(k), \\
p^f(k) &= \dot{x}_R(k),
\end{align*} \right. \quad (27) \]
with $T$ being the sample time of the system. The total system $G$ is therefore
\[ G : \left\{ \begin{align*}
\dot{x}_G(k+1) &= A_Gx_G(k) + B_Gu(k), \\
p^f_G(k) &= C_Gx_G(k),
\end{align*} \right. \quad (29) \]
with $x_G = (x_0^T, x_1^T)^T$ and
\[ A_G = \begin{pmatrix} A_R & 0 \\ -JT_C & 1 \end{pmatrix}, \quad B_G = \begin{pmatrix} B_R \\ -JTD_R \end{pmatrix}, \quad C_G = (0 \ 1). \]

By defining the feature-based error as $e = p^f - p^f_G$, with $p^f_G$ being the feature-based position reference, the closed-loop system can be calculated as
\[ \dot{x}(k+1) = A_x(k) + B_y(k), \quad (30) \]
\[ p^f_G(k) = C_x(k), \quad (31) \]
with $x = (x_0^T, x_1^T)^T$ and
\[ A_x = \begin{pmatrix} A_G - BD_xC_G & B_GC_k \\ -B_kC_G & A_k \end{pmatrix}, \quad B_y = \begin{pmatrix} B_GC_k \\ B_k \end{pmatrix}, \quad C_x = (C_G \ 0). \]

To assess the stability of this closed-loop system it is noted that the system can now be written as an LDI (Boyd et al., 1994). That is,
\[ A_d \in \left\{ \sum_{m=1}^{16} \beta_m A_{cl,m} L_{cl,m} \mid \beta_m \in \mathbb{R}, A_{cl,m} \in A_{cl}, \sum_{m=1}^{16} \beta_m = 1 \right\}. \quad (32) \]
where the set $A_d$ is the set of closed-loop matrices evaluated for every Jacobian $J \in J$. Using this definition, a common quadratic Lyapunov function $V(x) = x^T \delta x$ is to be found with $E = E^T > 0$ such that $V(x(k)) - V(x(k+1)) > 0$, $\forall x(k+1) = A_dx(k)$ or similarly by simultaneously checking the following linear matrix inequalities (LMIs):
\[ E - A^T_{cl,i}EA_{cl,i} > 0, \quad i \in \{1, \ldots, 16\} \quad (33) \]
\[ E > 0. \quad (34) \]

For a given value of $\alpha$ however all the possible values of the Jacobian will not be on the extreme boundaries as indicated by the black lines in Fig. 18. Therefore the conditions for stability given above are sufficient but conservative. Given the closed-loop system it can be investigated up to which value of $\alpha$ the closed-loop system is stable using a bisection algorithm as depicted in Fig. 19. The LMIs (33) and (34) can be solved efficiently using commercially available software (Gahinet, Nemirovski, Laub, & Chilali, 1994). The found value is $\alpha = 0.095$. Therefore, it can be concluded that the closed-loop system is guaranteed stable for pitches that satisfy
\[ 0.905 \leq P \leq 1.095 \leq 0 \]. \quad (35) \]

Note that this is an a posteriori stability analysis. The considered imperfections in this work are within this boundary. If however the imperfections are outside this boundary, a redesign of the controller $K$ is necessary to be robustly stable against the pitch imperfections.

7. Results

The use of the proposed feature-based position measurement is validated in practice on the experimental setup to demonstrate the effectiveness. Therefore, two experiments have been carried out. In the first experiment the improvement of using second order interpolation will be shown. The control task during the experiment is to move the wafer in one direction with a constant feature-based velocity of 36.8 f/s (approximately 0.01 m/s) while pitch imperfections are present. The feature-based velocity obtained from numerical differentiation of the feature-based position is given in Fig. 20. The vertical dashed lines indicate when a feature is passed, such that a different pitch is considered. By the gray curve in the figure, it can be seen that the feature-based velocity switches when using the feature-based position of (11). This is especially the case around $t=8.18$ s and $t=8.21$ s and is emphasized in red in the figure. This instantaneous switching feature-based velocity is not present when the second order interpolation (12) is used, the black curve. The power spectrum of the feature-based velocity is shown in Fig. 21. It shows that around 36.8 Hz the power content is increased. This is expected since this frequency corresponds to the applied reference velocity
of 36.8 f/s. Furthermore, the power content for frequencies up to approximately 200 Hz is reduced. For frequencies above 200 Hz, there is approximately no difference. The feature-based error is given in Fig. 22. Using the second order interpolation the error is reduced by approximately 40%.

In the second experiment the reference given in Fig. 9 is applied to the closed-loop system, such that the wafer is moved diagonally over the semiconductors (a feature to feature movement), while at each semiconductor the contour is tracked (a cartesian relative movement). The goal is to validate the use of the two dimensional feature based position measurement in combination with relative feature movements under closed loop control. The schematic movement is given in Fig. 23. In the figure part 1 (from $t=0$ to $t=0.2$ s) is a diagonal feature to feature movement, whereas parts 2 until 5 (from $t=0.2$ to $t=1$ s) are the relative cartesian movements. After that the sequence repeats itself over and over starting with the feature to feature movement, part 6. Since the reference is repetitive in time, the controllers $K_x$ and $K_y$ are expanded with a standard add-on repetitive controller (Hara, Yamamoto, Omata, & Nakano, 1988) to improve the performance. The final positioning error is shown in Fig. 24. After a small learning transient the final feature-based error is less than 0.05 f. The nominal pitch between the semiconductors is 27 pixels, which with a pixel size of 9.9 μm is 267 μm. In Fig. 24 the metric position error of ±10 μm is indicated.
8. Conclusions

In this paper a novel feature-based motion control approach is presented, which uses two-dimensional feature-based positions for feedback. The advantage of using these feature-based positions is that online trajectory generation has become redundant in case the metric target position is unknown a priori due to the pitch imperfections between successive features. A stability analysis proves robust stability of the closed-loop system while pitch imperfections up to approximately 10% are considered. A second order feature interpolation is implemented to reduce transient responses caused by instantaneously switching feature-based velocities. Experimental validation showed that this leads to an error reduction of 40%. Next to feature-to-feature movements, relative feature movements have been incorporated in the feature-based control approach; operators can easily specify how the tool is to be aligned with respect to the feature as a function of time. During experiments full two-dimensional feature-to-feature movements as well as relative feature movements are applied resulting in position errors less than 10 μm.

References


