Int J Adv Manuf Technol (2006) 28: 370–378
DOI 10.1007/s00170-004-2374-2

ORIGINAL ARTICLE

C.L. Li · K.M. Yu · Y.H. Lee

Automatic datum dimensioning for plastic injection mould design and manufacturing

Received: 7 May 2004 / Accepted: 10 August 2004 / Published online: 20 April 2005

Abstract Datum dimensioning (or ordinate dimensioning) technique is very popular in plastic injection mould drawings where the location dimensions of a large number of hole features must be specified in the drawings of the mould plates. Although commercial CAD/CAM systems provide semi-automatic tools to assist the designer in the dimensioning process, it is still a very tedious process, as the user has to specify the location of each dimension tag. This paper reports a completely automatic method where optimal placements of the dimension tags can be determined. The method employs dynamic programming technique to optimize the dimension process with respect to several criteria that can be selected by the user. The method has been implemented and incorporated into a commercial CAD/CAM system, and examples are given to illustrate the important features of the program.

Keywords Automatic dimensioning · Datum dimensioning · Dynamic programming · Optimal dimensioning · Ordinate dimensioning

1 Introduction

CAD/CAM systems are now widely used in the plastic injection mould-making industry. Many companies are using a solid modeling system to design the injection mould. They use a CAD system to model not only the core and cavity inserts of the mould (which are the most important components that form the impression of the mould), but also all other components in the entire mould assembly. With the advance in Internet technology and the recent development of Internet-enabled CAD, the design information of the injection mould can be communicated electronically between the product engineer (who designs the plastic part) and the tooling engineer (who designs the injection mould), even though they may be located in different geographic regions of the world. While flow of design information between product design and tooling design are communicated effectively through an electronic means, the communication of manufacturing information to the shop floor is done by both electronic and traditional techniques. Computer Numerical Control (CNC) machining toolpath or inspection instructions can be generated directly from the CAD/CAM system and downloaded through a network to the CNC controller for the machining or inspection operations. However, set-up instructions for a particular machining job may be specified in an engineering drawing. Moreover, not all machining tasks are done using CNC machine tools. Some traditional machining processes, such as drilling and grinding, are done using conventional machine tools because of cost consideration. Conventional engineering drawings are thus still playing an important role in communicating engineering information to the shop floor. The orthographic projections in engineering drawings can be generated automatically from the CAD model of the parts. Automatic tools for dimensioning of the parts are also provided by many commercial CAD systems. However, as pointed out by Chen et al. [1], those automatic dimensioning tools are not able to generate dimensions according to the drawing standards and engineering practices adopted in the shop floor.

In the specific application of injection mould design, datum dimensioning (or ordinate dimensioning) of hole features are used extensively. Figure 1 shows a typical detail drawing that can be found on the shop floor of a mould making company. Shown in the figure are the hole features and datum dimensions which are used to specify the locations of the holes. It can be seen that the dimensions are very crowded and it is a tedious task to manually adjust the placement of all the datum dimensions. The quality of the final fully-dimensioned drawing thus depends very much on the experience of the draftsman who produces the drawing. The purpose of this research is to develop a tool that can

C.L. Li (✉) · Y.H. Lee
Department of Manufacturing Engineering and Engineering Management,
City University of Hong Kong,
Tat Chee Avenue, Kowloon, Hong Kong
E-mail: meclli@cityu.edu.hk
Tel.: +8-52-27888432
Fax: +8-52-27888423

K.M. Yu
Department of Industrial and Systems Engineering,
The Hong Kong Polytechnic University

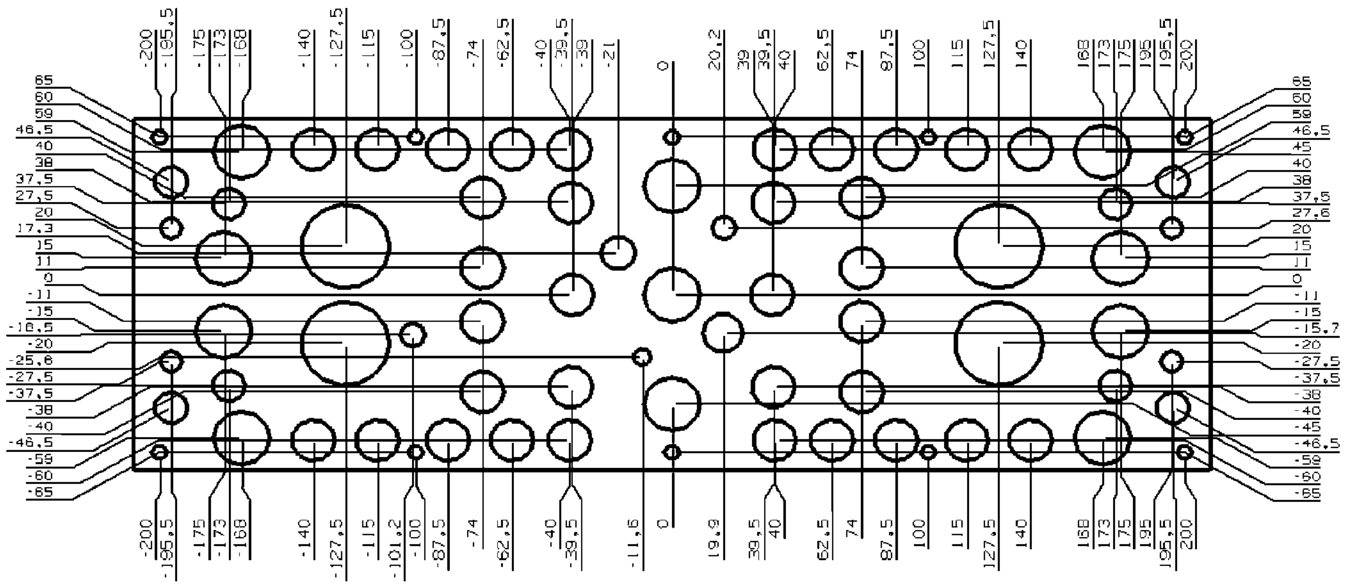


Fig. 1. Use of datum dimensioning in a drawing of a plastic injection mould part

generate the datum dimensions automatically from a given part of the injection mould. The resulting dimensions must satisfy two obvious requirements: first, that no two dimension tags may overlap; and second, that a dimension tag be placed as close as possible to the feature being dimensioned. The key issue in this research is to develop a method that can optimize the placement of the datum dimensions.

2 Related work

While dimensioning and tolerancing are two closely related processes in specifying the size and location information of the features in a mechanical part or an assembly, most of the past research work has focused on tolerancing. The major research issues in tolerancing are representation, analysis and synthesis. Tolerancing representation is concerned with the incorporation of tolerance information into a product modeling scheme. Examples include the solid offset approach developed by Requicha [2], the feasibility space approach proposed by Turner [3], and the TTRS by Desrochers and Clement [4]. More detailed review can be found in Roy et al. [5] and Yu et al. [6]. Tolerance analysis aims to determine the combined effect of part tolerances on the assembly tolerance. It can be used to verify the functionality of a design given known or assumed variations of individual part dimensions. Examples of technique in tolerance analysis include Monte Carlo simulation [7] and the direct linearization method [8]. The main objective of tolerance synthesis or tolerance allocation is to allocate part tolerances based on given functional requirements of the assembly. Recently, Islam [9] reported a concurrent engineering approach to address this problem. Based on a systemic analysis of the functional requirements from different customer requirements and the technical requirements from engineering considerations, a methodology for extracting dimensional requirements is developed. A software prototype FDT [10] is also developed for supporting the implementation of the methodology. FDT provides tools for representing the functional requirements, dimensions, tolerances and process capability into a functional requirement/dimensions matrix. The functional equations captured in the matrix are then separated into groups, and each group is then solved using a solution strategy specific to the functional requirement and the tolerancing problem involved. More detailed review in tolerance analysis and synthesis can be found in Roy et al. [5], Ngoi and Ong [11] and Hong and Chang [12].

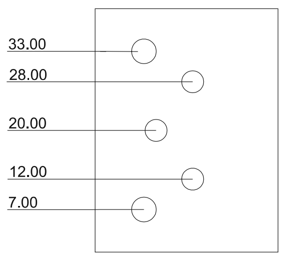
Several methods have been developed for generating dimensions automatically from the CAD model of a part. Yuen et al. [13] reported an early attempt in automatic dimensioning of parts represented in Constructive Solid Geometry (CSG) solid modeling technique. Points from planar faces and axes of cylinders are extracted from the solid model. The coordinates of the points are arranged in a tree structure to generate linear dimensions in the three principal directions. A simple technique for diametric and radial dimensions was also reported. Other early works in automatic dimensioning have been summarized by Yu et al. [6]. Recently, Chen et al. [1, 14] reported a more in-depth study of automatic dimensioning. Their method analyzed dimension redundancy, determined dimensioning schemes that are specific to feature patterns, selected appropriate views for specifying the dimension, and determined the appropriate location of the dimension using an expert system approach [15]. The expert system analyses the geometry and topology of the feature to be dimensioned, and determined a position for placing the dimension based on a set of rules that is relevant to the current dimensioning feature. With the placement of one dimension, a forbidden region is constructed so that all subsequent dimensions will not be placed in this region. This avoids overlap or intersection between two dimensions.

A limitation in the existing approach for the placement of the dimension is due to the sequential nature of the method. For example, in Chen's [1, 14] method the features to be dimensioned are prioritized, and the positions of the dimensions are determined one after another. The approach is not appropriate for determining the placement of datum dimensions, especially when the dimensions are very crowded, as in the case of injection mould plates. This is because the placement of one datum dimension may have an effect on the placement of another dimension that may be located far away from the current dimension. This paper reports our work in solving the placement problem in datum dimensioning. The major contribution of our work is the development of a new method that determines the optimal placement of each datum dimension. Using the dynamic programming approach to optimization, this new method overcomes the limitation of the sequential approach used in the existing method.

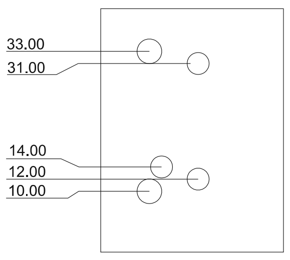
3 Basic characteristic of datum dimensioning

In datum dimensioning, the location of a feature is specified by the horizontal and vertical distances from the reference location of the feature and a reference datum. The default form of datum dimension is shown in Fig. 2a. When the vertical distance between two features to be dimensioned is less than the dimension tag size (i.e. the sum of the dimension text height and the minimum spacing between adjacent dimension texts), the alternative forms shown in Fig. 2b are required.[1] The dimension tags are shifted upward or downward from the default location to prevent overlap. As shown in Fig. 2c, the shifting of the dimension tag is achieved by breaking the single extension line of the dimension into three segments: two horizontal segments which are connected by one inclined segment. The extent to which a dimension tag can be shifted is governed by three parameters: (i) the dogleg angle α, which is the angle between the inclined segment and the horizontal segments of the dimension line; (ii) the margin distance m between the dimension text and the part boundary; and (iii) the location (x_i^f, y_i^f) of the feature f_i. The two extreme positions (i.e. the uppermost position $y_i^{\max}$ and lowermost position $y_i^{\min}$) of the dimension tag are given by:

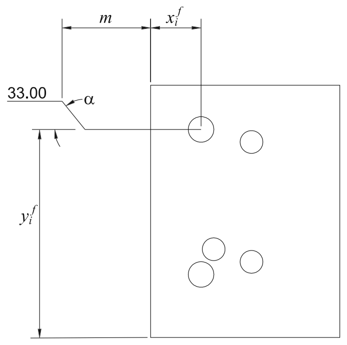
$$y_i^{\max} = y_i^f + (x_i^f + m)\tan\alpha$$
$$y_i^{\min} = y_i^f - (x_i^f + m)\tan\alpha \qquad (1)$$



(a) Default form of datum dimension.



(b)Alternative form of datum dimension.



(c) Parameters that define the shift of a dimension tag.

Fig. 2. Basic characteristics of datum dimensioning

4 Automatic datum dimension

The objective of the automatic datum dimensioning system is to find an optimal position for each datum dimension. The process consists of two phases of operation: the preparation phase and the optimization phase. In the preparation phase, major parameters that facilitate the optimization process will be established. Feasibility for placing the dimensions for all the features using the given dogleg angle, margin offset and dimension tag size will also be tested. In the optimization phase, a dynamic programming approach is used. The dimension tag locations can be optimized with respect to different sets of criteria, including the minimization of the shift of every dimension from their default locations, or maximization of the use of the default form as much as possible.

4.1 The preparation phase

The features to be dimensioned are first grouped into one or more feature sets. For each feature in a feature set, there exist at least one other feature in the set such that the vertical distance between them is less than the dimension tag size. In other words, the features in a feature set cannot be dimensioned using the default form exclusively without overlap between adjacent dimension tags. Instead, at most one feature can use the default form while all others require the use of the alternative form. The set of dimension tags associated with a feature set is called a dimension block. The *configuration* of a dimension block refers to the forms and locations of each datum dimension within the dimension block. For each position of a dimension block, its configuration is uniquely defined. Figure 3 shows two feature sets and their dimension blocks at two different configurations.

[1] To simplify the explanation of the technique, only vertical dimensions placed on the left hand side of the part are discussed. The method developed is general and can be applied to the other sides of the part.

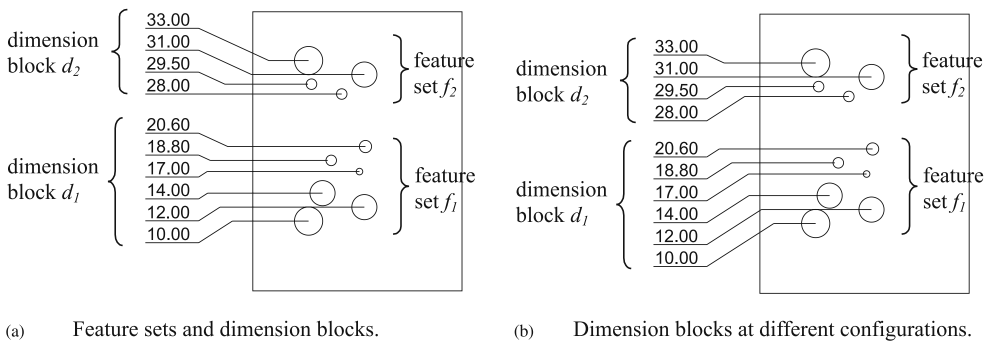


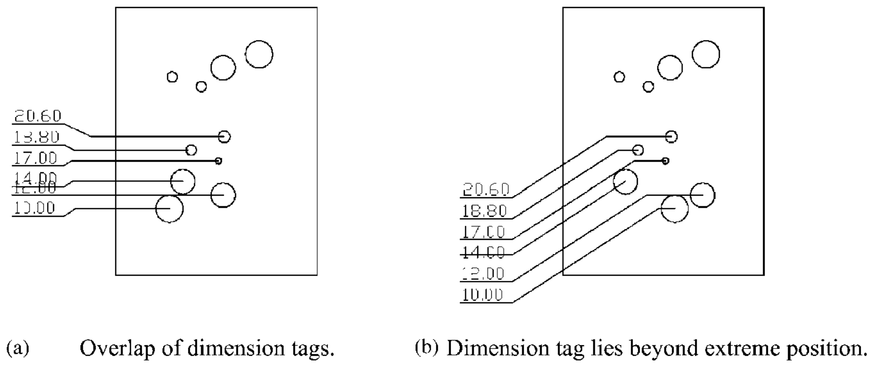
Fig. 3. Feature sets and different configurations of dimension blocks

***Definition 1**: Validity of a configuration.* A configuration of a dimension block is valid if there is no overlap between any dimension tags in the dimension block, and each dimension tag lies within its extreme positions.

The configurations of the dimension blocks shown in Fig. 3b are valid. Two examples of invalid configuration are shown in Fig. 4. The configuration shown in Fig. 4a is invalid because two of the dimension tags overlap. For the configuration shown in Fig. 4b, the extension line of the dimension tag 14.00 is at its lowermost position, while the required position for the dimension tag is beyond this lowermost position.

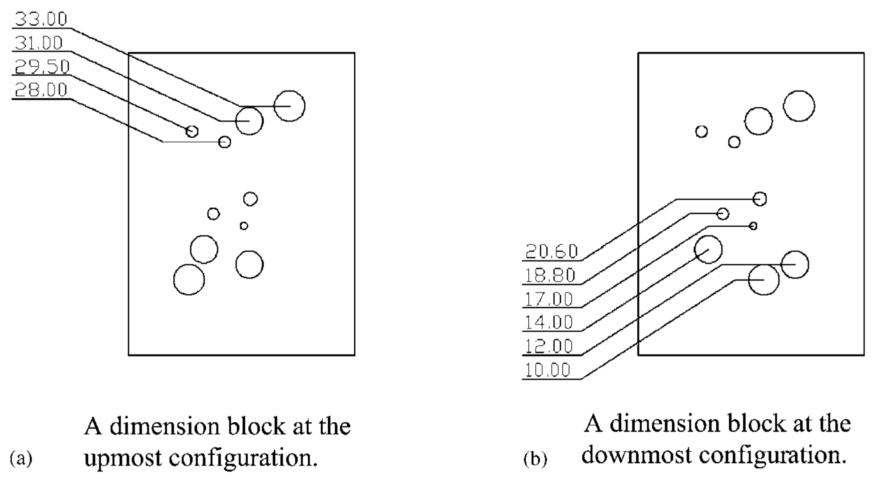
***Definition 2**: Extreme configurations.* There are two extreme configurations: the uppermost and lowermost configurations. A dimension block is at its uppermost (lowermost) configuration if the dimension block is valid and is at a position such that any other higher (lower) position results in an invalid configuration. The extreme configurations of a dimension block d_i are denoted by $Y_i^{\max}$ and $Y_i^{\min}$.

Figure 5a shows a dimension block at its uppermost configuration. It cannot move further upward because the dimension tag 29.5 is at its highest position. Figure 5b shows a dimension block at its lowermost configuration. It cannot move fur-



20.60
18.80
17.00
14.00
12.00
10.00
(a) Overlap of dimension tags.
(b) Dimension tag lies beyond extreme position.

Fig. 4. Invalid configurations of a dimension block



33.00
31.00
29.50
28.00
20.60
18.80
17.00
14.00
12.00
10.00
(a) A dimension block at the upmost configuration.
(b) A dimension block at the downmost configuration.

Fig. 5. Dimension block at extreme configurations

ther downward because the dimension tag 14.00 is at its lowest position.

The extreme configurations of a dimension block are the two important parameters that will be used by the optimization process. They are also useful in testing whether it is feasible to dimension all the features without any overlap between the dimension tags. It is observed that two properties are useful in developing a method to determine the extreme configurations.

Property 1:. For a dimension block at its uppermost (lowermost) configuration, at least one of its dimension tags is at its uppermost (lowermost) position.

Property 2:. A dimension block has a valid configuration if and only if it has extreme configurations.

Property 1 can be proved by contradiction. Assume that a dimension block is at its uppermost (lowermost) configuration, and none of its dimension tags are at their uppermost (lowermost) positions. Since all the dimension tags are not at their uppermost (lowermost) positions, they can all be moved upwards (downwards) simultaneously by the same amount until any one of them reaches its uppermost (lowermost) position. As all dimension tags are moved simultaneously by the same amount, the dimension tags do not overlap, and thus the resulting configuration is still valid and at a higher (lower) position than its original configuration. This violates the assumption that the original configuration is the uppermost (lowermost) configuration.

Property 2 can be verified directly. Given a valid configuration, the dimension block is moved upward (downward) until one or more of its dimension tags reach its uppermost (lowermost) position. Since all the dimension tags are moved simultaneously by the same amount, overlap does not occur. Moreover, the dimension block cannot be moved upwards (downwards) any further without invalidating the configuration because at least one of its dimension tags is at its uppermost (lowermost) position. According to Definition 2, the resulting configuration is thus the uppermost (lowermost) configuration. On the other hand, it is obvious that if a dimension block has extreme configurations, then it has a valid configuration because the extreme configurations are, by definition, valid.

Property 1 indicates that the extreme configurations of a dimension block can be obtained by investigating the extreme positions of the dimension tags in the block. The configuration of a dimension block can be specified by $\{y_i\}$, $i = 1, 2, \ldots, n$, where y_i is the location of the dimension tag of the ith feature in the feature set $\{f_i\}$. This assumes that $\{f_i\}$ are arranged in ascending order by their vertical positions (i.e. $y_i^f > y_j^f$ if $i > j$). Then, to avoid overlap between dimension tags, the location of the ith dimension tag is given by:

$$y_i = (i-1) \times SIZE + y_1; \quad n \geq i \geq 2 \tag{2}$$

where *SIZE* is the dimension tag size and y_1 is the location of the dimension tag for the first feature (f_1) of the set. y_1 is also used as the reference location of the dimension block.

For a configuration to be valid, all dimension tags must lie below its own uppermost position given by Eq. 1. That is:

$$y_i^{\max} \geq y_i$$

and thus

$$y_i^{\max} \geq (i-1) \times SIZE + y_1$$

The above relationship must be satisfied by all i. Therefore, the highest allowable value for y_1 is given by:

$$\underset{i}{Min}\{y_i^{\max} - (i-1) \times SIZE\} \tag{3}$$

with the y_1 value given by Eq. 2, and one or more y_i equal to $y_i^{\max}$. All other y_i are less than its $y_i^{\max}$. Since no other larger value of y_1 results in a configuration that satisfies $y_i^{\max} > y_i$, the resulting configuration, if valid, is the uppermost configuration. However, it is possible that at this configuration some of the y_i given by Eq. 2 is less than $y_i^{\min}$. Therefore, a check is performed for each y_i. If $y_i \geq y_i^{\min}$ for all i, then the uppermost configuration is found. If $y_i < y_i^{\min}$ for some i, then the configuration is invalid and no uppermost configuration can be found. By Property 2, the feature set $\{f_i\}$ does not have any valid configuration.

To find the lowermost configuration, $y_i \geq y_i^{\min}$ for all i and the lowest allowable value for y_i is given by:

$$\underset{i}{Max}\{y_i^{\min} - (i-1) \times SIZE\}$$

For a part with multiple dimension blocks, if the two extreme configurations of all the dimension blocks can be found successfully, then it is feasible to place the dimension tags within each dimension block without any overlap. However, it is still possible that a dimension tag of one dimension block overlaps with a dimension tag of another dimension block. The next step is thus to test whether it is feasible to place all the dimension blocks without any overlap. Property 3, stated below, will be useful in explaining the procedure that performs this test.

Property 3:. A valid configuration can always be constructed between the two extreme configurations of a dimension block.

Property 3 is easy to observe. Starting from the lowermost configuration, the dimension tags within the dimension block can be moved simultaneously by the same amount, such that none of its dimension tags reach their uppermost positions. According to Definition 1, the configuration at this position is valid. When one of the dimension tags reaches its uppermost position, the uppermost configuration of the dimension block is reached.

To test the feasibility of placing all the dimension blocks without overlap, the features within a feature set is first sorted in ascending order of their feature reference location y_i^f. The feature sets are then sorted in ascending order according to the location of their first feature y_1^f. The first dimension block d_1 is placed at its lowermost configuration $Y_1^{\min}$. For the second feature set, the dimension block d_2 is placed either at its lowermost configuration $Y_2^{\min}$ if $Y_2^{\min}$ is higher than the top of the d_1, or

d_2 is placed on top of d_1 if this lies within the range $Y_2^{\min}$ and $Y_2^{\max}$. According to Property 3, the latter is also a valid configuration for d_2. Otherwise, a valid configuration for d_2 cannot be found without overlapping dimension block d_1. This process is repeated for the next dimension block until valid configurations for all dimension blocks are found, or terminated wherever a valid configuration cannot be found for a dimension block.

4.2 The optimization phase

After the preparation phase, the extreme configuration of each dimension block is found and it is certain that both the overlap of dimension tags within a dimension block and between two adjacent dimension blocks can be avoided. To determine the optimum configuration of each dimension block, a dynamic programming method is used. The dimension blocks are sorted into ascending order according to their reference positions, as assumed above. The process of determining the configurations of the dimension blocks is divided into stages S_i, $i = 1, \dots, n$, where the ith stage decides the configuration of the dimension block b_i. Within each stage S_i, each possible configuration of b_i corresponds to a state $t_{i,j}$. In order words, state $t_{i,j}$ corresponds to the jth configuration of the ith dimension block b_i. The “cost” of selection of a state $t_{i,j}$ for a stage S_i is reflected by an overall cost function $S_i(t_{i,j})$, which is given by the recursive relation:

$$S_i(t_{i,j}) = \underset{k}{Min}\left\{C_i(t_{i,j}, t_{i-1,k}) + S_{i-1}(t_{i-1,k})\right\}$$

where $C_i(t_{i,j}, t_{i-1,k})$ is the cost function that reflects: (i) the interaction between dimension block b_i and b_{i-1} at states $t_{i,j}$ and $t_{i-1,k}$, respectively; (ii) the extent of deviation of the dimension block b_i from its default configuration; (iii) the overlap between the dimension blocks b_i, b_{i-1}; and (iv) the overlap between the dimension block b_i and a set of regions called the “forbidden regions”. The forbidden regions are specified by the user, and are usually the regions where other dimensions tags have been placed by the user and thus do not allow for any further placement of dimension tags. The optimal solution is obtained from $Min_j\{S_n(t_{n,j})\}$. The set of states $t_{i,j}$, $i = 1, \dots, n$ that gives rise to this optimum is the set of configurations that gives the optimal placements of the dimension tags.

4.3 State resolution

By Property 3, any configuration between the two extreme configurations of a dimension block b_i is a valid configuration, and thus there are an infinite number of choices for each stage. To enable the use of the dynamic programming method, discretization is necessary so that there is a finite number of states $t_{i,j}$ for each stage S_i. The simplest way of discretization is to extract a fixed number of positions uniformly from $Y_i^{\min}$ to $Y_i^{\max}$. However, this approach is not an efficient way to utilize computing resources. This is because the ranges (given by $Y_i^{\max} - Y_i^{\min}$) of the dimension blocks d_i can vary widely. It is obvious from Eq. 1 that a dimension block d_i, with all its features having large values of x_i^f, gives a large range, while that with small values of x_i^f gives a small range. With a fixed number of states, those dimension blocks having large ranges will have coarse resolutions, and those with small ranges will have fine resolutions. On the other hand, for a dimension block with a large range, it is likely that most of the valid configurations near the two extreme configurations will overlap with their neighbouring dimension blocks. This is best explained by an example: if the top position of a dimension block d_i at its uppermost configuration is higher than the lowest position of dimension block d_{i+1} at its uppermost configuration $Y_{i+1}^{\max}$, then dimension block d_i at configurations which are near the uppermost configuration must overlap with d_{i+1}. A similar argument applies to the lowermost configuration. Thus, for each dimension block d_i, a feasible range, characterized by the difference between $YF_i^{\max}$ and $YF_i^{\min}$, is defined:

$$YF_i^{\max} = Min(Y_i^{\max}, \underset{n \geq j > i}{Min}(Y_j^{\max} - SIZE \times n_i))$$
$$YF_i^{\min} = Max(Y_i^{\min}, \underset{i > j \geq 1}{Max}(Y_j^{\min} + SIZE \times n_j))$$

where n_i and n_j are the number of dimension tags in dimension blocks d_i and d_j, respectively. Using this definition of the feasible range, those configurations that always cause overlap with adjacent dimension blocks are excluded from the feasible range. A fixed resolution, say 0.5 mm, is specified and the number of states for a given stage is obtained by dividing the feasible range by the given resolution.

4.4 Cost functions

The overall cost of a stage and the cost function $C_i(t_{i,j}, t_{i-1,k})$ are vector and vector-valued functions, respectively. A cost vector consists of five components c_i, $i = 1, \dots, 5$ arranged in descending order of importance. That is, c_i is considered more important than c_j if $i < j$. The dynamic programming process requires the selection of the minimum cost, and thus comparisons between cost vectors are needed. Two cost vectors are compared by comparing their components. The comparison starts with the first component, which is considered the most important. If the first components of two cost vectors are equal, then the comparison continues to the next component, which is considered less important. The comparison stops once a corresponding pair of components are not equal. That is, the comparison between cost vectors is based on the first pair of components that are not equal.

The first component equation $c_1(t_{i,j}, t_{i-1,k})$ of the cost function $C_i(t_{i,j}, t_{i-1,k})$ is devised to penalize overlap between adjacent dimension blocks d_i and d_{i-1} and overlap between the dimension block d_i and a set of forbidden region r_i. It is given by:

$$c_1(t_{i,j}, t_{i-1,k}) = O_A(t_{i,j}, t_{i-1,k}) + \sum_m O_R^m(t_{i,j})$$

where $O_A(t_{i,j}, t_{i-1,k}) = 0$, if there is no overlap between d_i and d_{i-1}. $O_A(t_{i,j}, t_{i-1,k})$ is set to a very large value if overlap occurs. $O_R^m(t_{i,j})$ is the cost function used to penalize overlap between dimension block d_i at the configuration corresponding to state $t_{i,j}$ and the forbidden region R_m. The summation of $O_R^m(t_{i,j})$ over all m is used to consider overlap with all forbidden regions

R_m. $O_R^m(t_{i,j}) = 0$, if there is no overlap between d_i and R_m. And $O_R^m(t_{i,j})$ is set to a very large value if there is overlap.

The next four component equations – c_2, c_3, c_4 and c_5 – are assigned to four optional sub-cost functions which return a value to reflect the configuration with respect to four different criteria. The mapping between c_2, c_3, c_4, c_5 and the sub-cost functions are determined by the users. This gives the flexibility to the user to decide the relative importance of the criteria.

It is desirable that a dimension block can be placed such that the feature in the middle of the feature set is dimensioned with the default configuration. This is considered to be the default configuration of the dimension block. The sub-cost function $D_V(t_{i,j})$ is devised to measure the extent of deviation of a dimension block d_i at the jth configuration from its default configuration. The deviation can be estimated from the difference between the average location of the features to be dimensioned, and the average location of the dimension tags at the jth configuration.

$$D_V(t_{i,j}) = \frac{\left|\sum y_{i,j} - \sum y_i^f\right|}{n_i}$$

where n_i is the number of dimension tags in the dimension block d_i, $y_{i,j}$ are the locations of the dimension tags at the jth configuration, and y_i^f are the locations of the features. The summations are taken over all the dimension tags and features of the dimension block.

Using $D_V(t_{i,j})$ alone as the only sub-cost function, the cost computed by the cost function $C_i(t_{i,j}, t_{i1,k})$ is not able to distinguish one state that involves deviation of a dimension block d_i with an amount a, from a state that involves deviations of d_i for an amount a_1 and deviation of d_{i-1} for an amount a_2, where $a_1 + a_2 = a$. Because the total amount of deviation is the same in both states, the cost function is based only on the optional cost function $D_V(t_{i,j})$ that will give both states the same cost. The sub-cost function $D_N(t_{i,j})$, which counts the number of dimension blocks that are not at their default configurations, can be used to overcome the problem. It can be assigned to another component equation to give a lower cost to the former state. $D_N(t_{i,j})$ is set to one if the dimension block d_i at state $t_{i,j}$ is not at its default configuration. Otherwise, $D_N(t_{i,j})$ is set to zero.

It may be desirable to limit the extent a dimension block deviates from its default location by a user specified amount as a certain percentage p of its feasible range. The sub-cost function $D_P(t_{i,j})$ is devised to penalize excessive deviation. $D_P(t_{i,j})$ is set to one if $D_V(t_{i,j}) > p \times \left|YF_i^{\max} - YF_i^{\min}\right|$, and is set to zero otherwise.

When two adjacent dimension blocks at their default locations overlap for a certain amount a, the overlap can be removed by shifting d_i upwards by a_i, and shifting d_{i-1} downwards by a_{i-1}, such that $a_i + a_{i-1} = a$. It may be desirable that $a_i = a_{i-1}$. That is, the required total shift is being shared equally between two dimension blocks. The sub-cost function $D_V(t_{i,j})$ is not able to achieve this purpose because it only measures the total shift amount and not the distribution of the amount between adjacent dimension blocks. $D_E(t_{i,j})$ is devised to equalize the amount of shift between adjacent dimension blocks. It is set to the difference between the extent of deviations of the adjacent dimension blocks.

$$D_E(t_{i,j}, t_{i-1,k}) = |D_V(t_{i,j}) - D_V(t_{i-1,k})|.$$

The four optional sub-cost functions are freely selected and assigned to the second to fifth cost components by the user to achieve different objectives. In the next section, examples are given to illustrate the different results obtained by choosing different optional sub-cost functions.

5 Illustrative examples

The proposed optimization method has been implemented using C++ and incorporated into the Unigraphics II CAD/CAM system through its Application Program Interface (API). To illustrate the effects of the sub-cost functions, consider the example illustrated in Fig. 6. There are five dimension blocks in the figure. All the dimension blocks are at their default configurations and overlap occurs between d_1 and d_2, and between d_4 and d_5. These dimensions are generated automatically by the optimization program, with the cost function contributed only by the sub-cost function D_V. As a result, the dynamic programming procedure selects a set of states where the lowest cost corresponds to the minimum deviation from the default location of each dimension block.

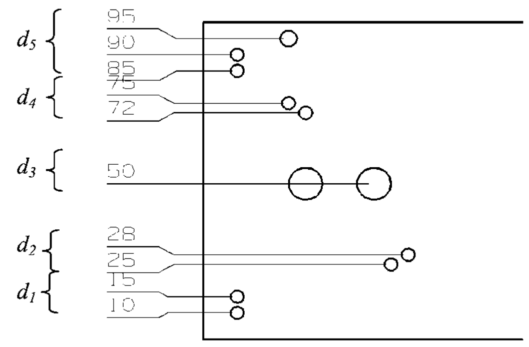


Fig. 6. Dimension blocks at their default configuration

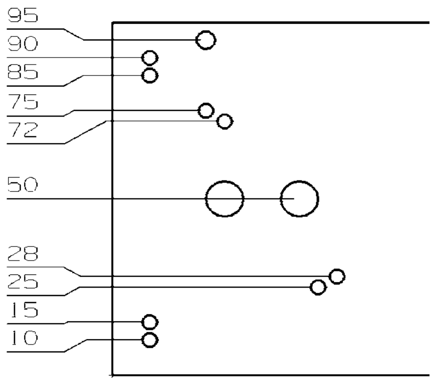
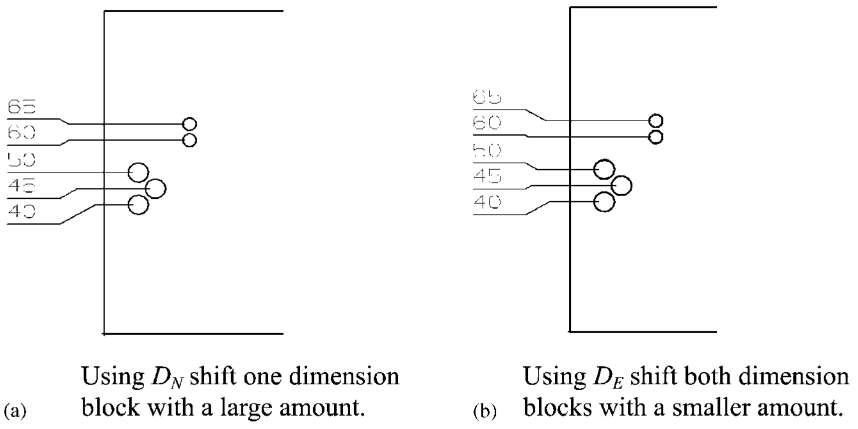
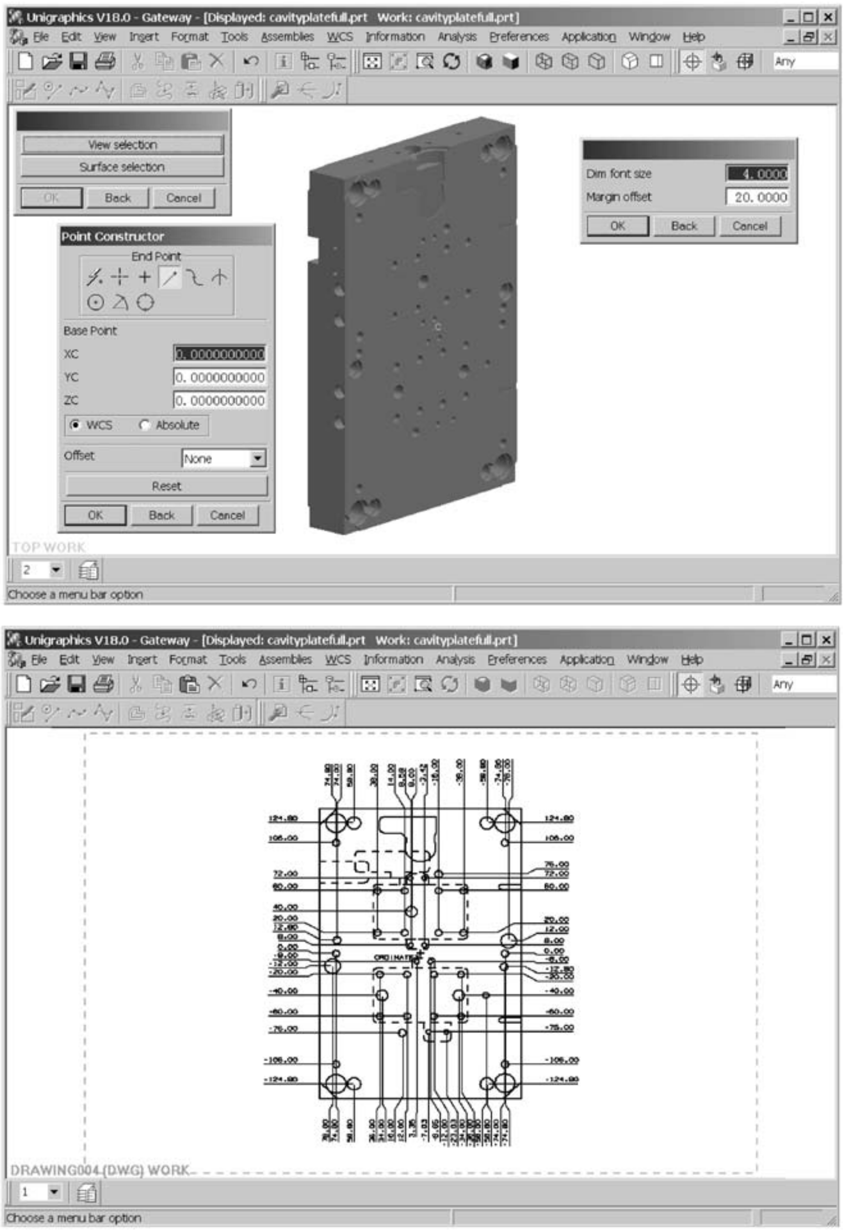


Fig. 7. Minimizing the amount of shift and the number of dimension blocks to be shifted to remove overlaps between dimension blocks



65
60
50
45
40
(a)
Using D_N shift one dimension block with a large amount.
65
60
50
45
40
(b)
Using D_E shift both dimension blocks with a smaller amount.

Fig. 8. Comparison of the effects obtained from D_N and D_E



Unigraphics V18.0 - Gateway - [Displayed: cavityplatefull.prt Work: cavityplatefull.prt]
File Edit View Insert Format Tools Assemblies WCS Information Analysis Preferences Application Window Help
Any
View selection
Surface selection
OK Back Cancel
Point Constructor
End Point
Reset
OK Back Cancel
OK Back Cancel
TOP WORK
DRAWING004 (DWG) WORK
ORDINATE

Dim font size 4.0000
Margin offset 20.0000

Base Point
XC 0.0000000000
YC 0.0000000000
ZC 0.0000000000
(• WCS) (Absolute)
Offset None

Choose a menu bar option

Fig. 9. Execution of the optimization program to generate datum dimensions for an example part

By activating the cost component equation c_1 which deals with overlaps, and setting cost component c_2 to D_V and c_3 to D_N, the optimization program shifts the dimension blocks from their default configurations by a minimum amount to remove the overlaps, and also the number of dimension blocks that have to be shifted are also minimized. The resulting dimensions are shown in Fig. 7. Only two dimension blocks d_1 and d_4 are shifted downwards to remove overlaps with d_2 and d_5.

Figure 8 shows another example which compares the effect of D_N and D_E. Figure 8a shows the result obtained by using D_V and D_N. The lower block is shifted while the upper block remains at its default configuration. Figure 8b shows the result obtained by using D_V and D_E. Both dimension blocks are shifted by a small amount to remove the overlap.

Figure 9 shows the execution of the optimization program in the Unigraphics II CAD/CAM system. After invoking the user function that contains the optimization program, a sequence of dialogue boxes appear to prompt user input. The user selects the face of a part where datum dimensions are to be generated and the reference datum to be used through the first and second dialogue boxes, respectively. Finally, the user is prompted for the font size of the dimension text and the margin offset to be used. A drawing that contains 120 datum dimensions for all hole features associated with the selected face is generated automatically within a second, as shown in the figure. Several typical example drawings in injection mould design with 50 to 250 hole features have been used to test the performance of the program. The tests show that datum dimensions can be generated within a second, which confirms that the performance of the program is suitable for interactive use.

A limitation of the current implementation is that the dogleg angle is set to a fixed value, and cannot be adjusted by the optimization program. When it is not feasible to place all the datum dimensions, the program execution terminates at the preparation phase and prompts the user for an alternative font size of the dimension text and the margin offset. It is desirable that the program attempts to solve the problem first by finding an optimal dogleg angle within a given range before prompting the user to change the two parameters. Further investigation is needed to incorporate the dogleg angle as an additional variable for optimization.

6 Conclusion

Automatic dimensioning is a useful tool in improving the productivity of the design process. Datum dimensions are frequently used in an injection mould workshop, and thus automatic datum dimensioning is particularly useful. A major problem in automatic datum dimensioning is to determine the proper placements for all the dimension tags such that they do not deviate too much from the features to be dimensioned, and so overlap between dimension tags can be avoided. An automatic method based on a dynamic programming technique has been developed in this research. The method allows the user to choose different criteria to control optimal placement generated by the automatic system. The method has been incorporated into a commercial CAD/CAM system and examples are given to demonstrate the various features of the method.

Acknowledgement The work described in this paper was fully supported by a Strategic Research Grant from the City University of Hong Kong (Project No. 7001633).

References

1. Chen KZ, Feng XA, Lu QS (2002) Intelligent location-dimensioning of cylindrical surfaces in mechanical parts. Comput Aided Des 32:185–194
2. Requicha AAG (1993) Mathematical definitions of tolerance specifications. Manuf Rev 6(4):269–274
3. Turner JU (1993) Feasibility space approach for automated tolerancing. Trans ASME J Eng Ind 115:341–346
4. Desrochers A, Clement A (1994) A dimensioning and tolerancing assistance model for CAD/CAM systems. Int J Adv Manuf Technol 9:352–361
5. Roy U, Liu CR, Woo TC (1991) Review of dimensioning and tolerancing: representation and processing. Comput Aided Des 23(7):466–483
6. Yu KM, Tan ST, Yuen MF (1994) A review of automatic dimensioning and tolerancing schemes. Eng Comput 10:63–80
7. Turner JU (1987) Tolerances in computer-aided geometric design. Dissertation, Rensselaer Polytechnic Institute, USA
8. Chase KW (1999) Tolerance analysis of 2-D and 3-D assemblies. ADCATS Report, Mechanical Engineering Department, Brigham Young University, available at http://adcats.et.byu.edu/WWW/Publication/index.html.
9. Islam MN (2004) A methodology for extracting dimensional requirements for a product from customer needs. Int J Adv Manuf Technol 23:489–494
10. Islam MN (2004) Functional dimensioning and tolerancing software for concurrent engineering applications. Comput Ind 54:169–190
11. Ngoi BKA, Ong CT (1998) Product and process dimensioning and tolerancing techniques. Int J Adv Manuf Technol 14:910–917
12. Hong YS, Chang TC (2002) A comprehensive review of tolerancing research. Int J Prod Res 40(11):2425–2459
13. Yuen MMF, Tan ST, Yu KM (1988) Scheme for automatic dimensioning of CSG defined parts. Comput Aided Des 20(3):151–159
14. Chen KZ, Feng XA, Lu QS (2001) Intelligent dimensioning for mechanical parts based on feature extraction. Comput Aided Des 22:949–965
15. Feng XA, Li XY (1987) Intelligent CAD of shafts, sleeves and cylindrical gears using micro personal computer. Ann CIRP 36(1):69–72