INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

ASSEMBLY SIMULATION FOR TOLERANCED PARTS: AN ADAPTED PATH PLANNING APPROACH

Stefan Wittmann[1], Marco Winter[1] and Kristin Paetzold[1]
(1) University of Erlangen-Nuremberg

ABSTRACT
One crucial task of product development is the definition of tolerances to assure assemblability and the fulfillment of functional and aesthetic requirements. It is extremely difficult to specify the necessary tolerance types and values of products that consist of several manufactured, imperfect components. The defined tolerances interact and can add up their negative effects. This results in a final product that differs from the original geometry and may lead to problems when assembling.
The contribution of this paper is a simulation for the assembly of toleranced parts, based on algorithms for automatic path planning. Goal of the simulation is to check whether the defined tolerances will lead to problems. Therefore, non-nominal part variants are virtually generated and assembled. Contrary to common methods, the used modeling allows the representation of measure, form and position errors.
After introducing basic path planning, several modifications are presented that enable statistic analysis and visualization of the assembly process. With the presented approach, an intuitive examination of toleranced assemblies is made possible in an early stage of the design process.

Keywords: Path planning, assembly simulation, computer aided tolerancing, visualization

1 INTRODUCTION
When modeling the detailed geometry of a product in a CAD system, the developer defines the desired ideal shape. This ideal shape can never be built, because every real manufacturing method causes smaller or larger deviations from the desired shape. Therefore, the product developer defines tolerances which limit the acceptable deviation of the manufactured part from the nominal geometry. Tolerances do not only limit dimensional deviation (like the length of an edge) but also form deviations (like cylindricity of a shaft) and position deviations (like parallelism of two surfaces). The correct definition of tolerances assures that the product fulfills its function and can be manufactured and inspected with acceptable costs.
The specification of reasonable tolerance types and tolerance values requires skill and experience of the product developer. Tolerances influence each other, and as soon as several tolerances on different parts interact, it becomes difficult or even impossible for the product developer to imagine the resulting effects. Therefore, tolerances are often copied from previous, similar constructions, or tolerance values are set unnecessarily precise, which causes high costs. To check the resulting non-nominal assembly, physical tests can be performed, but these late revisions also lead to very high costs and one can only test some combinations of part variants.
The contribution of this paper is a method that assists the proper tolerance synthesis by analyzing the assembly process of virtual variants of toleranced parts. From a given geometric design and tolerance definition, we want to give answers to the following questions: Will it be possible to move (assemble) the part from a start position to the desired position within the assembly, or do collisions occur on the way? If the part cannot be assembled, where (geometrically) does the problem occur? How high is the probability that the assembly process will fail? Because the shapes of the parts are not nominal, where will they finally be positioned relative to their environment?
To answer these questions, we use techniques from different domains to address issues of assemblability (existence of collisions) as well as of functional fulfillment (compliance of all functionally relevant measures of the resulting nonideal product). To handle the infinite number of possible, acceptable part variants defined by the tolerance specification, we use a Monte-Carlo approach, like many classical tolerance analysis tools do. To search for collision-free paths of parts,

we apply algorithms previously developed for robot motion planning. To find the final part positions, heuristic optimization is applied.
The paper is structured as follows: First, an overview of related work on computer aided tolerancing and motion planning is given. In Section 3, the intended workflow of the assembly simulation is presented, listing the needed input and output data of each step. The assembly simulation is described in detail in Section 4, showing how the needed steps can be computed in acceptable time. In Section 5, experimental results of the implementation are shown using an example of the assembly of a 3D CAD model. Finally, Section 6 concludes our work and gives some hints for possible future directions.

2 RELATED WORK

There exist several commercial tools for Computer Aided Tolerancing (CAT), like CETOL, emTolMate or VisVSA. These complex CAT tools are usually employed by experts, who get the input (CAD geometry and tolerances) from the product developers. The expert has to "translate" the data into the description of the CAT program by abstracting the geometry, variable parameters (i.e., certain lengths or angles of the part), assemble operations and mating conditions. Finally, measurements have to be defined, like the distance between two surfaces or the height of the resulting assembly. Afterwards, the CAT program performs a simulation, for example by generating part variants with extreme (High, Low, Median) or random (Monte-Carlo) parameter values. The results are presented to the user as statistical data of the defined measurement (distribution curve, mean value, standard deviation, Cp and Cpk values). Söderberg et al. [1] give an overview of tolerancing tools available for the design, preproduction and production phase. For each phase, exemplary software tools and methods are described.
When generating part variants by changing geometric parameters (like certain lengths or angles), it is not possible to generate all possible shape deviations described by the tolerance definition. For example, in most of the CAT tools, flat surfaces remain flat, even if the defined tolerances would allow different deviation types. Parallelism e.g., would allow arbitrary dents and deflection within the tolerance range. Maxfield et al. [2] and Stoll [3] describe methods to deform nominal parts within the tolerance range. The presented deformation methods set up continuous functions to calculate deformations of the vertices of the part model.
The possibility of arbitrary shape variants for every part of the assembly (compliant to the tolerances) introduces an additional problem. The part surfaces are no longer ideal (i.e., flat) and therefore the assembly of parts is no longer trivial. The process of positioning a deviating part relative to another part in an assembly is called relative positioning [4]. The authors describe a way to position variant parts in three dimensions with varying angles and lengths, which prevents part intersection. In this work the faces remain flat to simplify mating relations. In [5], a method for worst-case analysis of parts consisting of line and arc segments and a self-developed, not standardized tolerance specification is presented. In [6], Latombe et al. describe the use of non-directional blocking graphs to calculate the collision-free assembly of parts consisting of line segments. The approach has a high computation time complexity even for simple parts and is not able to include arbitrary part imperfections. Pierce and Rosen [7] describe an approach to position arbitrary NURBS-based solid models using a mathematical programming method, which avoids collisions between the parts and minimizes the distance between two mating faces. The authors present an example of positioning measured parts manufactured by end-milling. In [8], a framework for positioning parts modeled as triangle meshes is described. The framework uses heuristic optimization (simulated annealing, evolutionary algorithm and particle swarm optimization) to position the parts. User-defined objective functions like collision detection and squared distance computation are used to describe the desired optimization criteria for the positioning.
Relative positioning aims to find the end position of a part within the assembly, but does not assure that a collision-free way to the end position exists (if the part can be assembled). In contrast, motion planning algorithms try to automatically find collision-free paths of objects. These general purpose algorithms are used in a large spectrum of applications, ranging from the steering of mobile robots over surgery assistance to steering of spacecrafts [9]. The task of motion planning algorithms is to find a continuous, collision-free path from a given start position to an end position. There is strong evidence that any complete planner will have a complexity that grows exponentially in the degrees of freedom (DOF) of the robot (the moved part), see [10]. Randomized planners are used to be able to find solutions in acceptable time [11]. These planners generate random configurations in solution

space and try to link these configurations. The linking of two configurations is done by a local planner, which interpolates the configurations and performs collision-checks for every intermediate step. If it is possible to connect start and end position, a collision-free path has been found. Popular randomized planning algorithms are Probabilistic Roadmap algorithms (PRM, see [12], [14]) and Rapidly-Exploring Random Trees (RRT, see [9] and [13]). The RRT algorithm will be described in detail in Section 4.

A path planning algorithm can only show that there exists a path between start and goal, but cannot prove that it is impossible to connect them. This problem is still unsolved for the general case; however [15] shows how to build up disconnection proofs for some geometric primitives. Unfortunately, the approach described there cannot be applied to arbitrary CAD geometry. Exept for [15] path planning research always deals with one certain variant of the analyzed part. For statistical tolerance analysis, many possible variants of the parts have to be analyzed, which necessitates new approaches for the definition of start and end-position and the path planning progression.

In this paper, we present a combination of Monte-Carlo simulation, relative positioning and path planning to simulate the assembly process of non-ideal, toleranced parts. The following Section describes the intended workflow of the simulation.

3 WORKFLOW OF THE ASSEMBLY SIMULATION

Aim of the proposed method is to show the consequences of the defined tolerances to the product developer. Figure 1 presents the workflow of the complete simulation process. Input data is depicted by ellipses, the executed applications are depicted as rectangles. The right column is a symbolic illustration of exemplary assembled parts.

As a first step, a variant simulation generates random “mutations” of the assembled parts. The model representation of the part variants must be able to completely describe the closed surface of the parts. Also, it must be possible to apply arbitrary geometric changes to the part, to represent manufacturing defects. For the path planning, a fast collision detection between parts is essential. Therfore, a triangle mesh representation has been chosen. Nurbs and other splines would make collision detection much more difficult (see [17]).

A meshing tool (which is available for all common CAD applications) is used to generate a fine mesh out of the CAD parts. Large triangles are subdivided to be able to apply deformations on any point of the surface. The variant simulation deforms the meshes by repositioning the vertices of the nominal mesh. At any time, the defined tolerances are preserved. Generation of non-ideal parts is beyond the focus of this paper, details on the used deformations can be found in [3]. The contribution of this paper is to find assembly problems caused by tolerancing. So it is not intended to simulate shape variants of a particular manufacturing process like casting or deep-drawing, but to generate random variants of the parts, compliant to the restrictions defined by the product developer (which are the defined tolerances). As soon as position tolerances are defined, it is not possible to generate *the* worst case part or *the* variance hull.

After generating random part variants, the next step is to simulate the assembly process. Therefore, start and goal position of the part have to be specified. The start position can be an arbitrary position outside of the assembly. The goal position is defined in the CAD data of the assembly, but has to be adjusted by a relative positioning process to guarantee meaningful, collision-free positioning of the now deformed part. When start and goal position are defined, the path planning algorithm tries to create a collision-free connection of these states. If the planning algorithm succeeds, the path can be visualized or saved for further use. If the assembly fails due to the shape deviations of the input parts, a visualization shows *where* it fails (see black circles in Figure 1, down right).

To be able to produce statistically relevant data, a huge number of random part variants has to be generated and virtually assembled. Finding an assembly path for *one* part variant is already a challenging and computationally expensive task, especially when the assembled parts fit very tight, because this leaves a very narrow passage in the solution space, which has to be found by the path planner.

To conclude, it is problematic to use a general purpose path planning algorithm to find a path for thousands of part variants, which is necessary for statistical analysis. The next Section describes how the similarity of the part variants can be exploited to reduce the complexity of the assembly simulation.

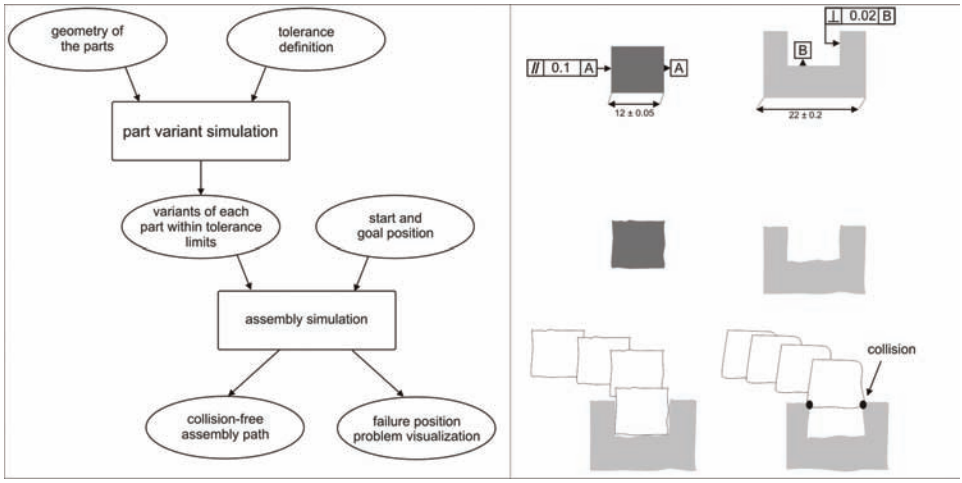

Figure 1: Workflow of the assembly simulation

4 ADAPTION OF THE RAPIDLY EXPLORING RANDOM TREE APPROACH

4.1 Standard Path Planning with RRT

The aim of a path planner is to find a continuous path of a moving object (often called "robot") from a start to a goal configuration, without colliding with the environment. In our case, a rigid part is moved from a start position to an assembled position by applying translations and rotations. This means, the solution space of the problem has six dimensions (3 translational DOF, 3 rotational DOF). The part moves "on its own", it is not moved by a tool or a human worker. This is because we want to know whether the part geometry itself could be assembled with the specified tolerances.

The used algorithm works as follows (see [9] for details): A random configuration is generated in the bounded search space (see Figure 2a). Based on a defined distance metric, the closest existing configuration is looked up (at the beginning, there exist only start and goal configuration). Hereupon, the algorithm tries to connect the random configuration with the closest found configuration.

It is checked whether the part can be moved from the closest configuration to the random configuration without collisions. For this purpose, a local planner generates many intermediate configurations by applying interpolation methods (see Figure 2b). To achieve smooth movement of the part, the translation values are interpolated linearly and the rotations are interpolated with spherical linear interpolation [16]. Each intermediate step is checked for collisions with the environment. For the collision detection in our implementation, RAPID was used for fast collision detection [18]. If no collisions occur on the way, the random configuration can be reached and is added to the set of connected position nodes. If only a small movement toward the random configuration was possible, the farthest reachable position is saved. In the shown example (Figure 2c), the moved part can only be slightly translated to the left and rotated counterclockwise before colliding.

This process of connecting random nodes iteratively grows two "trees" of configurations (see Figure 2d). Start- and goal configuration are the root nodes of the two trees. With a defined probability, the algorithm tries to connect a new random node to *both* trees. If the connection to both trees succeeds, a collision-free path has been found (Figure 2e). The shortest way through the graph is generated by applying A* search on the graph (see [19]).

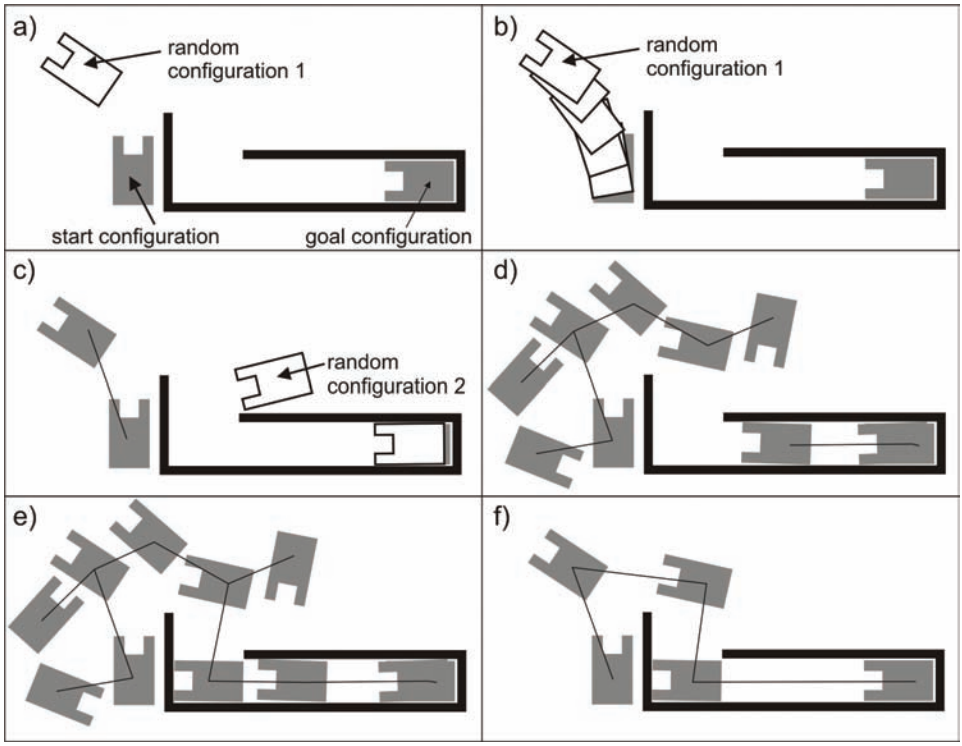

Figure 2: Basic steps of the path planning process

4.2 Adaptation of Path Planning to Statistical Tolerance Analysis

As mentioned in Section 3, it is necessary to find the paths for many variants and combinations of the moved and the environment parts within the tolerance limits. The search process described in Section 4.1 can take a long time, especially if the moved part has to perform complex movements or has to pass a narrow passage in the solution space. If the path planning algorithm would be re-run for every geometric variation, the computation would take too long to be applicable for statistical analysis.

One important fact can be noticed when examining the analyzed parts: The allowed deflections from the nominal shape are restricted by tolerances, and therefore are geometrically very small, which means all parts have quite similar shapes. This implicates that, once a path for one variant has been found, it should be possible to apply the path to all part variants with minor adjustments.

We will now describe how the path planning process can take advantage of this similarity property to run much faster. At first, the configurations that are sufficient to describe the whole movement from start to goal have to be extracted. As Figure 2e shows, the found path can contain unnecessary configurations of the moved part. The path can be simplified by deleting unnecessary nodes. This is done the following way: Starting at node 0 (the first node of the found path), it is checked whether a direct connection to node n (the last node of the path) is possible. If the connection is possible, the intermediate nodes 1 to n-1 are deleted, else it is tried to connect node 0 to node n-1, and so on. This loop is run for all remaining (not yet deleted) nodes. As a result, only “key configurations” of the path remain (Figure 2f).

The key configurations are now used as additional input data for all successive assembly simulations. They are used as automatically generated “hints” for the planning algorithm and guide the part through the solution space. A similar approach can be found in [20], where a haptic system was used to manually generate hints for an obstacle based probabilistic roadmap planner. In contrast to manual input of hints, our novel approach allows the automatic generation of hint configurations. To generate

the hints, it is reasonable to use the nominal parts, because the assembly of the ideal geometry should always be possible.
As can be observed, certain key configurations are essential for the success of the path planning, since they depict how to cross difficult, narrow regions in solution space. Nevertheless, because of the geometric variations of the parts, it is sometimes necessary to adjust these key positions. After the adjustment, it is tried to connect the key positions. The goal position must always be adjusted: all parts that differ from the nominal shape will reach different final assembly positions. To solve this problem, the framework presented in [8] is used. The relative positioning is formulated as an optimization problem. Two objective functions are formulated: One function adds a high constant value if collisions with the environment occur, the other function minimizes the distance to the original (ideal) position. Particle swarm optimization is used to find an acceptable alternative position close to the given goal position. If no collision-free alternative was found, the path planning fails, because it was not possible to assemble the part near enough to the wanted end position. The other key positions are also repositioned, where required (if they cause collisions). If no alternative was found, the concerned key position is deleted from the list of hint positions.

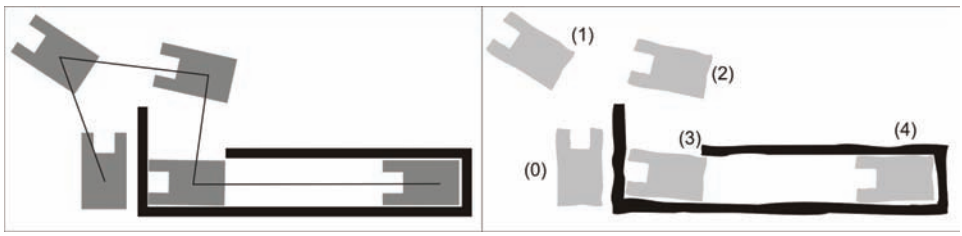

Figure 3: Adjustment of key positions

In Figure 3, an adjustment example is shown: Key configurations (0), (1) and (2) do not have to be re-adjusted. Because of the shape deviations of moved part and environment part, the original key configurations (3) and (4) would lead to collisions of the moved part with the environment geometry. Therefore, the configuration (3) is slightly moved upwards and rotated clockwise; configuration (4) is translated to the left to avoid collisions. This step is automatically done by heuristic optimization as described in [8].
To further speed up the successive path planning, the problem is subdivided into smaller sub-problems. Instead of searching for one path from start to goal position, multiple planning processes are started, which attempt to connect only two neighboring key positions. The example of Figure 3 would be subdivided into four tasks: the connection of configuration (0) and (1), (1) and (2), (2) and (3) and finally (3) and (4), respectively. For many subproblems, it is even possible to directly connect the configurations for all possible part variants without collisions, which greatly reduces computation time. Direct connection means that the given key positions are just interpolated to get intermediate positions. In the shown example, positions (0)-(1) and (1)-(2) can always be directly connected, regardless of the shape deviations. If the direct connection leads to a collision, an alternative path has to be found as explained in Section 4.1. The search space (in which random configurations are created) can be restricted around the two key configurations, as it is known that the solution path must be similar to the previously found path. This also simplifies the problem for the path planning algorithm.
The successful connection of all key positions means that a collision-free, continuous path from start to goal position was found for the examined combination of non-ideal parts. The path can be visualized, e.g., as an animation or as a presentation of piecewise movements. The found path can conveniently be saved as a sequence of transformation matrices.
A failure in the connection of two key positions means that the path planner was not able to find a connecting path for this segment. In these cases visualization can show where the assembly process actually failed. Section 5 shows exemplary visualizations of successful and failed path planning.

The approach can be concluded by the following pseudo-code:

```
Generate a hint Path
Part=idealPart;
Environment=idealEnvironment;
//use bidirectional Rapid Random Tree for path search
hintPath=findPath_RRT(assemblyPosition, disassemblyPosition);
hintPath.simplify(); //delete unnecessary positions

Loop over all Part and Environment Variants
for(i=0; i<nPartVariants; i++){
    part.loadVariant(i);
    for(j=0; j<nEnvVariants; j++){
      environment.loadVariant(j);
      findPath(part,environment);
    }
}

Find a Path for one combination of part variant and environment variant
bool findPath(part,environment){
      //find collision-free alternative for the original hint path
      adjustedPath=findAlternativePositions(hintPath);
      //find connection for all sections of the adjusted path
      for(i=0; i<adjustedPath.size()-1; i++){
        //use bidirectional Rapid Random Tree for path section search
        section=findPath_RRT(adjustedPath[i], adjustedPath[i+1]);
        if(sectionFound==TRUE){
          resultPath.add(section); //add found Path
        }else {
           //add direct connection
           resultPath.add(adjustedPath[i], adjustedPath[i+1]);
           sectionFailure=true;
          }
    }

    resultPath.simplify(); //delete unnecessary positions

    visualize(resultPath); //show path and collisions (if so)

    if(sectionFailure==true){
      return false; //one or more key positions were not connected
    }
    else{
      return true; //complete path from start to assembly-position found
    }
}
```

4.3 Discussion of the Proposed Method

To simulate a continuous movement of the part, an infinite number of checks would be necessary. Instead, the local planner performs discrete steps to check whether two positions can be connected. The number of collision tests is dependent on the distance of the configurations. If two positions have a large translational or rotational distance, many collision checks are necessary. The user of the assembly simulation is responsible for setting the step size of the local planner to an appropriate value. The RRT path planner is not complete, which means that if no path exists, it would run forever. For practical reasons, the user defines a maximum number of iterations. If this counter value is reached, the algorithm returns failure.

To speed up the computation, hint configurations are used. If the part deviations require a new path that completely differs from the hint configurations, this new solution will probably not be found, because the search space is restricted around the given hint positions.

Nevertheless, the use of path planning for the analysis of the assembly of toleranced parts has some major advantages. The user does not have to specify the exact, potentially complex spatial movement

of the parts. It is sufficient to define start and goal position. The suggested use of hint configurations considerably speeds up the simulation. Therefore the path planning becomes fast enough to be used for statistical analysis.
In commonly used CAT tools, the user already has to know where problems will occur within the assembly before the analysis starts. The problematic configuration has to be modelled in the CAT program and measurements reveal the properties of the assembly. In contrast, path planning is able to discover problems all along the assembly movement without a priori knowledge of the user. The results of the path planning (success and failure) are intuitively understandable because they can be visualized using the part geometry.
Compared to existing work on the geometric analysis of the consequences of part defects (i.e. [7], [8]) the presented method not only simulates the final relative position of the non-nominal parts, but also reveals problems during assembly.
The relative positioning enables our approach to accomplish path planning even if the given assembly position from the nominal part leads to collisions. Standard path planning (i.e. [9], [11]) would immediately report failure if start- or end-position have collisions. If the relative positioning finds an acceptable, close alternative, the adapted path planning approach can still proceed.

5 EXAMPLE: RAIL AND CARRIAGE

The path planning and adaption algorithms were implemented in C++ and run on an Intel Core2 CPU with 2.4 GHz. Practical use of the adapted path planning is shown with an assembly of two parts: A carriage assembled at the end of a rail. Both parts were modeled in Pro/Engineer, and then converted into fine triangle meshes of 45.666 and 20.476 triangles for rail and carriage respectively. The rail has an opening at one side. In the nominal case, there is enough clearance to insert the carriage into the opening and to slide it to the end of the carriage. Small shape errors are now introduced by applying deformations to the meshes as described in [3], taking into account the defined tolerances. The high mesh resolution is needed to generate arbitrary deformations of the geometry.

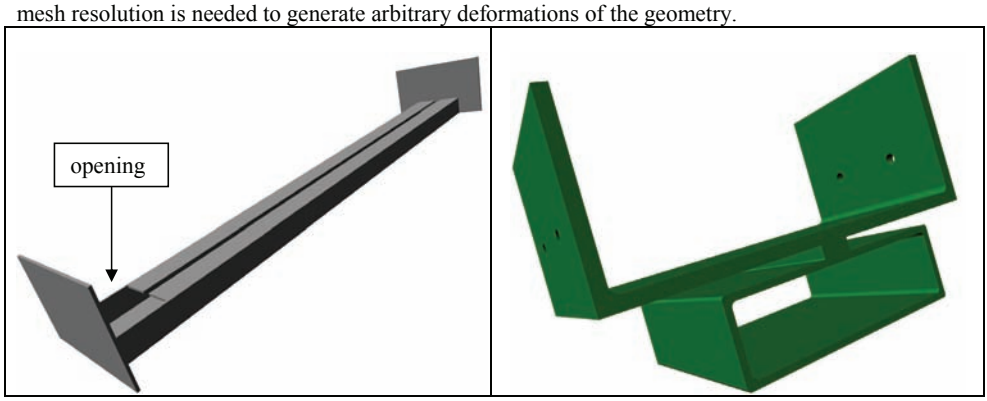

Figure 4: The modeled rail (left) and carriage (right, enlarged)

The task of the assembly simulation is to find a path from a user-defined start position to a goal position (end of the rail). In contrast to standard CAT modeling, the user does not a priori have to know *if* and *where* problems will occur. It is not specified which detail of the assembly is critical, e.g., if the carriage fits through all variants of the opening, if the clearance within the rail is big enough, if the bar between upper and lower part of the carriage is high enough, etc. .
Different geometric variants of rail and carriage are generated within the tolerance range. A hint file containing key positions is created by assembling the nominal parts. Using the hint configurations, the non-ideal variants are automatically assembled. On average, one assembly simulation has a runtime of 90 seconds. Using the aforementioned CPU, two simulations can be run in parallel. An output file concludes the examined results (success or failure) of the simulation runs.
When an assembly path has been found, the key configurations of the moved part are saved as transformation matrices and can be visualized. Figure 5 shows the movements of a successfully generated path, displaying several intermediate steps. At first, the carriage is slowly lifted and rotated, and then it is translated, lowered and inserted into the opening of the rail and slid to the end of the rail.

The visualization uses color-coding for the moved carriage, beginning with a bright color (green - start configuration) and ending with a dark color (purple - goal position). Note that the number of intermediate steps shown in the figure is not equal to the number of used intermediate steps that the local planner uses to detect collisions. The collision checks are performed with a much smaller step-size, to avoid skipping of critical regions.

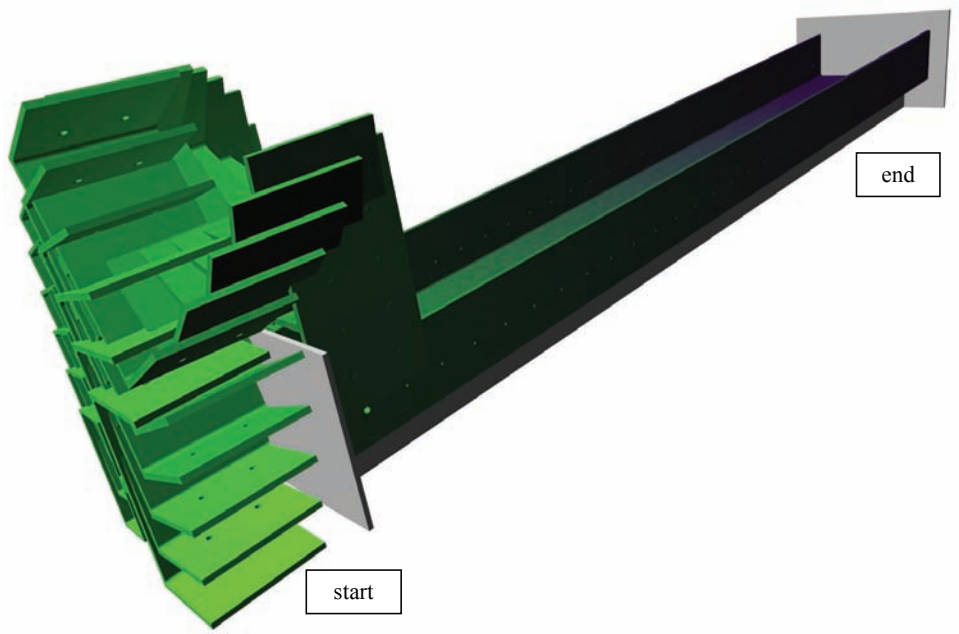

Figure 5: Visualization of a successfully assembled carriage

If the deviation of the parts is large and no assembly path has been found, the failure and the concerned part variants are saved. Failure means that the path planner was not able to generate a collision-free way for one or several path segments. To visualize the failure, the involved key positions are connected linearly and the colliding position is colored dark (red). Figure 6 shows a failure of the path planner to insert the carriage into the opening of the rail. The user now knows that the size of the opening is a critical dimension. Therefore the width of the opening can be increased or the width of the carriage can be decreased. The movement of the carriage within the rail does not cause collisions (the successful linear movement of the carriage in the rail is blanked out in Figure 6 to reveal the collision).

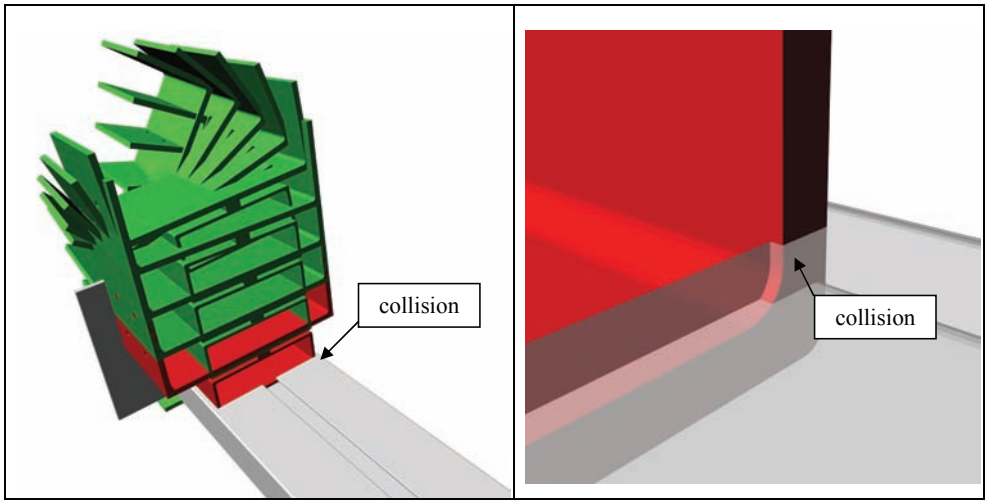

Figure 6: Visualization of a failed assembly process (right: transparent close up of the rail opening)

6 CONCLUSION AND FUTURE WORK

In this paper, the application of path planning methods for tolerance analysis was presented. Path planning algorithms automatically find a valid path of objects from a start to a goal position through a space filled with obstacles. In the presented work, path planning was used to simulate the assembly of many part variants within the tolerance limits to find potential assembly problems. With standard Computer Aided Tolerancing tools, it is difficult to investigate a complete assembly movement. Instead, several user-defined part configurations must be modeled and analyzed separately.

To be able to use path planning for statistical tolerance analysis, a triangle mesh representation of the CAD parts was applied. The meshes of the assembled parts were slightly deformed to generate part variants within the tolerance limits. For these variants, the path planning algorithm tries to generate collision-free assembly paths. Statistical analysis of the tolerancing scheme requires that many part variants are checked by the assembly simulation. Therefore, several acceleration methods, which take advantage of the similarity of the part variants, were implemented.

The presented approach has several advantages compared to known commercial tools and approaches found in the literature. It allows the integration of arbitrary geometric errors and is able to represent all possible imperfections of the manufactured parts allowed by the tolerancing scheme. The product developer can find potential assembly problems without the help of a tolerancing expert. There is no need to define an abstract model of the tolerancing and assembly scheme, all input data of the simulation is available to the product developer. Therefore, the simulation reveals errors in a much earlier stage of the design process and also gives understandable visual hints *why* and *where* problems occur.

Future work will include advanced visualization techniques that highlight the region where the assembly fails. To locate assembly failure more precisely, the insertion of additional hint positions between distant key configurations might be helpful. A haptic system will be applied for the manual review of found problems. To further speed up the calculation of paths, a modified convex hull approach (see [21]) that preserves geometric features like boreholes will be investigated for worst case estimations. As an alternative to the RRT path planner, an obstacle-based PRM planner [14] will be tested. Also, iterative search of paths by allowing some penetration within the obstacles [22] is promising and can be combined with a found hint path.

REFERENCES

[1] Söderberg R., Lindkvist L. and Carlson, J. Virtual geometry assurance for effective product realization. In: *NordPLM 2006*, Göteborg, 2006

[2] Maxfield J., Dew P., Zhao J., Juster N. and Fitchie M. A Virtual Environment for Aesthetic Quality Assessment of Flexible Assemblies in the Automotive Design Process. In *SAE 2002 World Congress*, Detroit, Michigan, 2002

[3] Stoll T. Generieren von nichtidealer Geometrie. In: *17. Symposium Design for X*, Neukirchen, Germany, 2006, pp. 143-150

[4] Li B. and Roy U. Relative positioning of toleranced polyhedral parts in an assembly. In *IIE Transactions*, 2001, Vol. 33, pp. 323-336.

[5] Ostrosky-Berman Y. and Joskowicz L. Tolerance envelopes of planar mechanical parts with parametric tolerances. *Computer Aided Design*, 2005, Vol. 37 (5), pp. 531-544.

[6] Latombe J.C. and Wilson R.H. Assembly Sequencing with toleranced parts. In *Solid Modeling '95*, Salt Lake City, Utah, 1995, pp. 83-94

[7] Pierce R.S. and Rosen D. Simulation of Mating Between Nonanalytic Surfaces Using a Mathematical Programming Formulation. *Journal of Computing and Information Science in Engineering*, 2007 Vol. 7, issue 4, pp. 314-321.

[8] Stoll T., Wittmann S., Helwig S. and Paetzold K. Registration of measured and simulated non-ideal geometry using optimization methods. In *Proceedings of the 10th CIRP International Seminar on Computer Aided Tolerancing*, CirpCAT '07 Vol. 10, Erlangen, Germany, March 2007.

[9] LaValle S.M. and Kuffner J.J. Rapidly-exploring random trees: Progress and prospects. In: *Algorithmic and Computational Robotics: New Directions*, pp. 293-308. A K Peters, Wellesley, MA, 2001.

[10] Reif J.H. Complexity of the mover's problem and generalizations. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, San Juan, Puerto Rico, October 1979, pp. 421-427.

[11] Tapia L., Thomas S. Boyd B. Amato N.M. An unsupervised adaptive strategy for constructing probabilistic roadmaps. Technical Report TR08-004, Parasol Lab. Department of Computer Science, Texas A&M University, September 2008.

[12] Kavraki L.E., Svestka P., Latombe J.C. and Overmars M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE Transactions on Robotics and Automation*, Aug 1996, Volume: 12, Issue: 4, pp. 566-580.

[13] LaValle S.M. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, 1998.

[14] Amato N. and Wu Y. A Randomized Roadmap Method for Path and Manipulation Planning. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, April 1996, Volume: 1, pp. 113-120.

[15] Basch J., Guibas L.J., Hsu D. and Nguyen A.T. Disconnection proofs for motion planning. In *IEEE International Conference on Robotics and Automation*, 2001, Proceedings 2001 ICRA. Volume: 2, pp. 1765- 1772

[16] Shoemake K. Animating rotation with quaternion curves. In *ACM SIGGRAPH Computer Graphics*, Volume 19, Issue 3, 1985, pp. 245-254

[17] LaValle S.M. Planning Algorithms. Cambridge, 2006, pp.74-75 (Cambridge University Press)

[18] Gottschalk S., Lin M.C. and Manocha D. OBB-Tree: A Hierarchical Structure for Rapid Interference Detection, In *Proceedings of ACM SIGGRAPH 1996*, 1996, pp. 171-180

[19] Hart P.E., Nilsson N.J. and Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths, In *IEEE Transactions on Systems Science and Cybernetics*, July 1968, Volume 4, issue 2, pp. 100-107.

[20] Bayazit O.B., Song G. and Amato N.M. Enhancing Randomized Motion Planners: Exploring with Haptic Hints. In *Autonomous Robots*, volume 10, issue 2, 2001, pp. 163-174

[21] Lööf J., Söderberg R. and Lindkvist L. Visualization of variation in early design phases: A convex hull approach. In: *DESIGN 2006*, Volume 2, Dubrovnik, 2006, pp. 905-912

[22] Ferre E., Laumond J.P. An Iterative Diffusion Algorithm for Part Disassembly. In IEEE International Conference on Robotics and Automation, 2004, Proceedings 2004 ICRA, pp. 3149-3154

Contact: Stefan Wittmann
University of Erlangen-Nuremberg
Department of Mechanical Engineering
Chair of Engineering Design
Martensstr. 9
91058, Erlangen
Germany
Tel: +49 9131 8523218
Fax: +49 9131 8523223
E-mail: wittmann@mfk.uni-erlangen.de
URL: http://www.mfk.uni-erlangen.de

Dipl.-Inf. Stefan Wittmann is scientific employee at the Chair of Engineering Design at the University of Erlangen-Nuremberg. He studied computer science and mechanical engineering as subsidiary subject from 2000 to 2005. Since then he works at the Chair of Engineering Design in Erlangen. His main research topics are simulation and visualization methods for Computer Aided Tolerance Analysis.

Dipl.-Inf. Marco Winter studied computer science from 1999 to 2004, with a central focus on computer graphics. Since then, he works as a scientific employee at the Chair of Computer Graphics at the University of Erlangen-Nuremberg. His current research topics include, amongst others, reconstruction and visualization of surfaces in medicine and construction design.

Prof. Dr.-Ing. Kristin Paetzold studied mechanical engineering. She received a doctor's degree in 2003. Afterwards she worked as chief engineer at the Chair of Engineering Design in Erlangen. Since April 2009, she is professor for Technical Product Development at the Universität der Bundeswehr in Munich. Her research areas are cognitive technical systems, workflow support and tolerancing.