A Path Planning Algorithm for Multi Manipulators

A Path Planning Algorithm for Multi Manipulators

Shinsuke Tamura, Tomonari Murata, Md. Nazrul Islam, Tatsuro Yanase and Shuji Taniguchi
Graduate School of Engineering, University of Fukui
tamura@fuis.fuis.fukui-u.ac.jp

***Abstract*-This paper discusses an approach to the development of a path planning algorithm for cooperating multi manipulators. In the approach, multiple manipulators are considered as a single composite one. Therefore, the composite manipulator possesses all arms of the manipulators included in the collaboration. A newly developed path planning algorithm BFA (Backtrack Free path planning Algorithm) enables the efficient generation of paths of this composite manipulator with extremely many arms. The algorithm is backtrack free and resolution complete. Computation volume of the algorithm is proportional to the total number of arms included in the composite manipulator. An additional advantage of this approach is that paths of individual manipulators can be calculated completely in parallel.**

I. INTRODUCTION

Recent manufacturing systems require highly sophisticated handlings of work-pieces that can be achieved only through cooperation among multiple manipulators. Therefore the development of efficient path planning algorithms for multiple manipulators is one of the key issues for enabling advanced manufacturing. This paper discusses an approach to the development of a path planning algorithm for cooperating multi manipulators, based on a newly developed path planning algorithm BFA [3] (Backtrack Free path planning Algorithm). In the discussed approach, multiple manipulators are considered as a single composite one, of which arms are coupled through work pieces that they are handling. Therefore the composite manipulator becomes to have extremely large number of arms, and BFA, of which computation volume is proportional to the number of arms, enables efficient path planning of this composite manipulator. The algorithm based on the discussed approach is backtrack free and resolution complete; times and memories required are proportional to the total number of arms included in composite manipulators. An additional advantage is that paths of individual manipulators can be calculated completely in parallel.

II. BACKTRACK FREE PATH PLANNING ALGORITHM

This section explains the behavior of BFA. Different from many existing path planning algorithms [1], [2], BFA searches paths in *2* or *3*-dimensional work spaces directly, in order to reduce the huge computation volume caused by the high dimensionality of configuration spaces (C-spaces). Compared with the C-space where the dimension increases with the number of arms, the dimension of the work space is fixed at 2 or 3. Therefore when loci of individual arms can be calculated sequentially without any backtrack, it is possible to construct an algorithm, of which computation volume is proportional to the number of arms. BFA achieves just this. It determines the existence of paths, and finds correct paths of individual arms sequentially from the base to the top arms without any backtrack when paths exist, provided that the size of the gird is small enough compared with that of arms and obstacles (in BFA, positions in *2* or *3*-dimensional spaces are approximated by the finite number of grid points). In this section, firstly a basic theorem is explained with definitions of terms, and then the algorithm and its performance are discussed. In the following, fulcrums and the other ends of arms are called joints and movable ends. It is assumed that a sequential number is assigned to each arm of a manipulator, so that *1* and *N* are assigned to the base and the top arms, respectively (*N* is the number of arms included in the manipulator), and the joint position of the base arm is fixed.

Location and attitude of an arm: A location of the *n*-th arm is represented by the grid point occupied by its joint. An attitude of the *n*-th arm is represented by a pair of grid points (X, Y). Here, X and Y are grid points occupied by the joint and the movable end of the *n*-th arm, respectively.

Feasible attitude set (FAS): Attitude (X, Y) of the *N*-th arm (*N* is the maximum arm number) is called feasible when the *N*-th arm does not collide with any obstacle. Also attitude (X, Y) of the *n*-th arm ($n < N$) is called feasible when the *n*-th arm does not collide with any obstacle and there exists at least one feasible attitude (Y, Z) of the ($n+1$)-th arm. A feasible attitude set (FAS) of the *n*-th arm at X is a set of grid points that are occupied by the movable end of feasible attitudes of the *n*-th arm located at *X* and denoted as A(X, *n*).

Adjacent attitudes: A pair of attitudes of the *n*-th arm (X_1, Y_1) and (X_2, Y_2) is called adjacent, when X_1 and Y_1 are equal or adjacent to X_2 and Y_2, respectively.

Connecting point pair: Any grid point pair P and Q included in A(X, *n*) and A(Y, *n*) is called a connecting point pair of a FAS pair A(X, *n*) and A(Y, *n*), when attitudes of the *n*-th arm (X, P) and (Y, Q) are adjacent.

n-connectivity: Adjacent grid points X_1 and X_2 are called *N*-connective (*N* is the maximum arm number), when they are not occupied by any obstacle. Adjacent grid points X_1 and X_2 are called ($n-1$)-connective, when a FAS pair A(X_1, *n*) and A(X_2, *n*) has at least one connecting point pair (Y_1, Y_2) such that Y_1 and Y_2 are *n*-connective.

n-reachable set: *n*-reachable set R(*n*) is a set of grid points that are reachable from H_n, the start position of the

movable end of the *n*-th arm, by chaining grid points, which are mutually *n*-connective.

The following theorem ensures that BFA determines the existence of paths and finds paths when they exist without any backtrack.

[Theorem] [3] Under the assumptions that individual FAS A(X, *n*) are connective sets in terms of *n*-connectivity and collisions of arms themselves are allowable, the necessary and sufficient condition for the existence of collision free paths of a manipulator from its start attitude H = {H_0, H_1, H_2, ----, H_N} to the goal attitude D= {$D_0 = H_0$, D_1, D_2, ----, D_N} is that R(*n*) and A(D_{n-1}, *n*) includes D_n for each *n* (= *1, 2*, ---, *N*). Also when collision free paths from H to D exist, L_0, the locus of the joint of the *1*st arm is the single fixed point {H_0}, and L_n the locus of the movable end of the *n*-th arm can be determined without backtracks based on L_{n-1}. Here H and D, start and goal attitudes of the manipulator, are represented as sets of start and goal positions of movable ends of individual arms, i.e. H = {H_0, H_1, H_2, ----, H_N} and D= {D_0, D_1, D_2, ----, D_N}. $D_n \in A(D_{n-1}, n)$ means that the attitude of the *n*-th arm, of which joint and movable end are located at its goal positions, is feasible.

```
/* off-line part */
 calculate R(N), a set of grid points to which the movable end of
 the N-th arm can reach from the start position as a single point
 n=N
 while (n > 0) {
   find feasible attitude set A(X, n) of the n-th arm at
       each point X in the workspace
   determines the (n-1)-connectivity of individual neighboring
   point pairs
   calculate R(n-1), a set of points, which are reachable by the
     joint of the n-th arm from its start position, based on
     (n-1)-connectivity
   n=n-1
 }
 /* real-time part */
 if ( Dn∈R(n), and Dn∈A(Dn-1, n) for all n) {
   n=1
   while (n=<N) {
    find the locus of the movable end of the n-th arm
       that connects its start position to its goal position
    n=n+1
    }
 }
 else {there is no collision free path}
```

Fig. 1. BFA

Based on the above theorem, a backtrack free path planning algorithm can be constructed easily. Fig. 1 shows the overall structure of the algorithm. The algorithm consists of two parts, off-line and real-time parts. The off-line part is executed only when locations of the manipulator or obstacles are changed and the real-time part is executed every time when the goal attitude is given to the manipulator. For each *n* beginning from *n = N* to *1*, the off-line part finds feasible attitude sets of the *n*-th arm at individual points, and based on them, it determines the *n*-connectivity of individual neighboring point pairs, and calculates *n*-reachable set R(*n*). The real-time part generates loci of individual arms sequentially from the *1*st (base) to the top (*N*-th) arm based on *n*-connectivity when goal attitudes are given. Here existence of paths is ensured when $D_n \in R(n)$ and $D_n \in A(D_{n-1}, n)$ are satisfied for all *n* at the beginning of the real-time part, and loci of individual arms are calculated without any backtrack.

Computation time and memory space required for the algorithm execution is the order of *NVR*. Here, *V* and *R* represents the total number of grid points in the workspace and the maximum number of grid points included in individual FASs, respectively. *N* is the number of arms.

According to the theorem, the algorithm is effective only under assumptions that collisions of arms themselves are allowable, and FAS of the *n*-th arm at each position is *n*-connective, i.e. arms can rotate without collision from one attitude to any other attitude within the same FAS at every point. The former assumption is not serious in *3*-dimensional cases. Collisions among arms can be removed easily by the local adjustments of arm positions, because usually there are enough free spaces around collision free attitudes.

It is possible to apply BFA also to cases where the latter assumption is not satisfied by defining copies of points corresponding to individual connected components of FASs. Namely, when a FAS has *m* disjoint connected components at point P, *m* copies of P are generated including the original point P, and different connected components of the FAS are assigned to different copy points as their FASs. Therefore, path-planning problems for these cases can be converted to the one, in which the assumption is satisfied, i.e. all arms have FASs with single connected components at each point.

Advantages of BFA are, 1) its computation volume is proportional to the number of arms, 2) its computation time is not sensitive to environments, 3) it is easy to generate locus or attitude constrained paths, and 4) it is a resolution complete algorithm.

Although the complete BFA program is available only for *2* dimensional work spaces until now, experimental results have exhibited the above advantages [8].

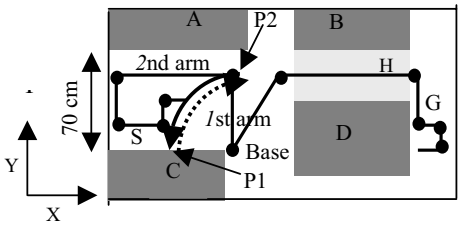

Fig. 2. Free spaces connected by a single point

Fig. 2 is one of environments, in which BFA performance was evaluated. In the figure, *4* obstacles A, B, C and D are allocated, and the gap between A and C is set just as the same size as the length of the *2*nd arm (in the figure, line segments and small circles represent arms and their joints, and lengths of the *1*st and the *2*nd arms are set to *50* cm and *70* cm, respectively). Therefore, the movable end of the *1*st arm must be located at single point P1 in order to change the direction of the *2*nd arm. In other words, *2* areas that include start and goal attitudes of the manipulator S and G are connected by just a single point P1, and finding collision free paths that

connect attitudes S and G is very difficult for conventional heuristics based algorithms [4], [5], [6]. BFA found paths successfully and efficiently even in these cases. The manipulator had firstly moved the movable end of the *1*st arm from P2 to P1 in order to change the attitude of the *2*nd arm as shown by the solid arc, and then moved it back to P2 as shown by the dashed arc. By using a PC with *1.53* GHz CPU and *224* Mbyte RAM, BFA found these paths within *2.5* sec. for *2* to *6* arms manipulators that behave in 80 x 80 grids (grid size is *5*cm).

Figs. 3 and 4 show the BFA performance for the environment shown in Fig. 2, they are relations between the computation time and the number of arms, and that between the computation time and path length. The off-line part computation time is proportional to the number of arms. In spite of the fact that BFA is resolution complete, the off-line part computation time does not increase exponentially. Regarding to the real time part computation time and the total computation time, they are not proportional to the number of arms. However, this is because that path length increases not linearly with the number of arms. It is obvious that any algorithm has parts that require the computation volume at least proportional to path length, and according to Fig. 4 the real-time part computation time of BFA increases just linearly with the path length. Consequently, different from existing resolution complete algorithms, the total computation time of BFA can be suppressed at linear order of the number of arms provided that path length increases also linearly with the number of arms.

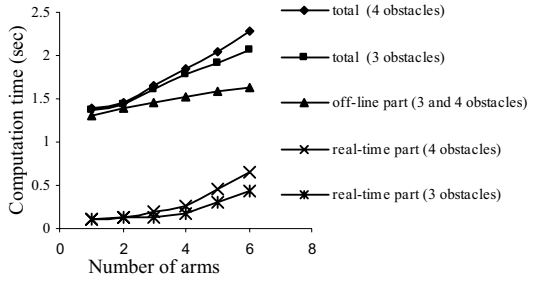

Fig. 3. Computation time and number of arms in Fig.2 cases

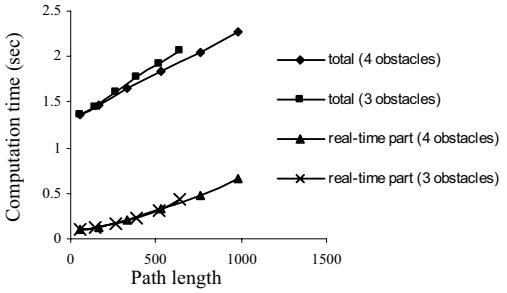

Fig. 4. Computation time and path length in Fig.2 cases

Fig. 3 also shows that BFA performance is not sensitive to environments. When obstacle A is removed from Fig. 2, the gap that divides the free space of the *2*nd arm disappears. Therefore for heuristics based algorithms, computation times necessary for these *2* environments, i.e. the one where obstacle A is allocated (*4* obstacles case in the figure) and the other where A is removed (*3* obstacles case in the figure) differ extremely. In contrast, BFA can find paths with almost the same time regardless of environments. Namely, the off-line part computation time is constant even environments change, and although the real-time part computation time changes with environments, it is because that the path length becomes long when arms avoid many obstacles.

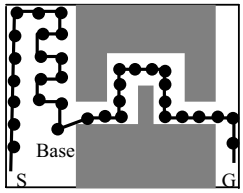

Fig. 5 An environment with a narrow corridor

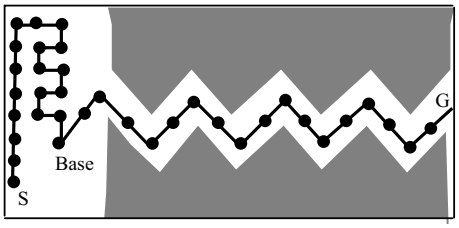

Fig.6 An environment with a long narrow corridor

Table 1 Performance comparisons between BFA and PRM

Number of arms	Width of gap (cm)	Computation time of BFA (sec.)	Computation time of PRM (sec.)
1 (disk robot)	60	-	0.87
1 (disk robot)	40	-	1.32
1 (disk robot)	20	-	346.00
18	60	4.16	-
18	40	4.16	-
18	25	4.96	-
18	20	5.15	-
18	15	7.16	-

Performances of BFA and PRM (Probabilistic Roadmap Method) [6] were compared in the environment shown in Figs. 5 and 6. For finding paths of *8* arms manipulator from the initial attitude S to the goal attitude G in Fig.5, BFA and PRM require *3.07* and *98.2* seconds, respectively (BFA and PRM were executed by *1.53*GHz and *2.8*GHz CPUs respectively, and both were implemented by Java). Table 1 shows computation times for the environment shown in Fig. 6. BFA was implemented by Java on a PC with *1.53*GHz CPU, and PRM was implemented by C^{++} on a PC with 1GHz CPU. According to these results, apparently BFA finds paths with much shorter time than PRM. Moreover, different from PRM cases where the computation time increases rapidly with the decrease of the size of the gap in Fig.6, BFA can find paths within almost the same time even the gap size changes.

Although the computation time increases when the gap size becomes less than *20*cm, it is simply because arms should change their attitudes more frequently, i.e. the path length increases when the gap size is small compared with the arm lengths.

Regarding to the performance in *3* dimensional work spaces, although only the off-line part is implemented for *3* dimensional spaces until now, it has been confirmed that the off-line part computation volume is the linear order of the number of arms. Also several tens of seconds are enough for completing the off-line part calculation for manipulators with *5-6* arms that behave in *50* x *50* x *40* grids.

The other important advantage of BFA, which is exploitable in multi manipulator path planning, is that it is easy to generate locus or attitude constrained paths. Fig. 2 was used for evaluating BFA locus and attitude constrained path generation ability. As a locus constraint, the movable end of the top arm was enforced to follow the edge of obstacle B when its joint was inside of area H, and as an attitude constraint, the top arm was enforced to be parallel to the X-direction when its joint was in H. BFA can incorporate these constraints in a straightforward and intuitive way, i.e. they can be incorporated only by deleting attitudes that do not satisfy the constraints from feasible attitude sets. In Fig. 7, A(F, *N*), a feasible attitude set of the *N*-th arm (top arm) at point F is an arc (U, W) when there is no constraint, and the above locus constraint can be incorporated by only deleting points that are not on the edge of obstacle B from A(F, *N*). Then, A(F, *N*) is reduced to a single point U, and as a consequence, paths automatically follow the edge when the joint of the *N*-th arm moves within H, because BFA generates paths by connecting only points included in FASs.

An attitude constraint can be incorporated in the same way by deleting points that are not parallel to the X-direction from A(F, *N*). In this case A(F, *N*) is reduced to a single point V, and the *N*-th arm attitude on the path automatically becomes parallel to the X-direction when the joint of the *N*-th arm moves within H. As shown above, attitude constraints are easier to be incorporated than locus constraints. In cases of attitude constraints, points on FASs to be deleted can be determined without considering path positions to be followed, different from locus constraint cases. Different from existing algorithms, in which only the top arm loci or attitudes are constrained [7], it is apparent that loci and attitudes of general arms can be constrained in the same way as the top arm in BFA.

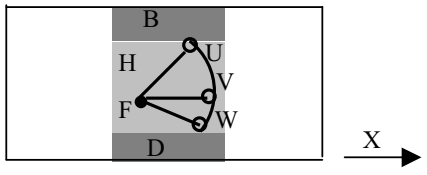

Fig.7 Locus and attitude constraints

Regarding to the advantage about the resolution completeness, BFA determines the existence of paths promptly, and it finds correct paths within estimated time when they exist, i.e. without any backtrack, provided that the size of grids is small enough. Actually different from heuristics based algorithms, BFA determines the existence of paths by only executing its off-line part. Because of this property, BFA becomes applicable to real time uses, in which times necessary for generating paths must be predetermined.

III. Path Planning for Multi Manipulators

In this section BFA is extended so that the path planning of multi manipulators that cooperate to achieve complicated tasks can be accomplished efficiently.

A. Simple cases

Firstly, *M* manipulators that are cooperating while holding the same position on a work-piece are considered. In Fig. 8, *3* manipulators, of which base arm joint positions are fixed, are cooperating while holding a work-piece at point P by their top arms. The extension is straightforward by considering *M* cooperating manipulators as a single composite manipulator consists of ΣN_m arms. In the followings, N_m represents the number of arms included in the *m*-th manipulator, and arms of the composite manipulator are numbered so that arms of the *m*-th manipulator have the smaller number than those of the *r*-th manipulator when $m < r$, and arms of the same manipulator are numbered in the reverse way from the top to the base (i.e. the *j*-th arm has the smaller number than the *k*-th arm when $j > k$). Fig. 8 shows an example, where arm numbers on the upper lines and those on the lower lines (included in parentheses) indicate arm numbers of individual manipulators and those of the composite manipulator, respectively. However to make explanations comprehensive, notation $n(m)$ is also used to represent the arm number of the composite manipulator that is assigned to the *n*-th arm of the *m*-th manipulator, e.g. $N_2(2)$-th arm of the composite manipulator represents N_2-th arm (top arm) of the *2*nd manipulator.

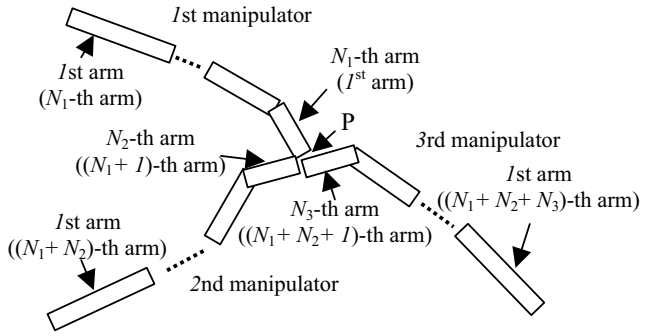

Fig.8 Composite manipulator consists of *3* manipulators

Then the slight modifications of definitions of feasible attitudes and the *n*-connectivity make BFA efficiently generate paths of the composite manipulator that avoid collision while bringing work-pieces from initial to goal attitudes. Definitions are modified by exchanging roles of

joints and movable ends of individual arms, i.e. in the following; it is considered that a joint of an arm rotates around the movable end of the arm. Therefore, provided that X and Y are points occupied by a movable end and a joint of the *n*-th arm, the location and an attitude of the *n*-th arm are represented by X and (X, Y), respectively. Modified definitions are followings.

Modified feasible attitude set: For each m ($1 \le m \le M$), attitude (X_m, Y_m) of the *1*(*m*)-th arm is called feasible when it does not collide with any obstacle (in this case, Y_m is the single fixed point where base of the *m*-th manipulator is located). Also attitude (X_m, Y_m) of the *n*(*m*)-th arm ($1 < n \le N_m$) is called feasible when it does not collide with any obstacle and there exists at least one feasible attitude (Y_m, Z_m) of the (*n-1*)(*m*)-th arm. As an exception, for $N_M(M)$-th arm, top arm of the *M*-th manipulator, its attitude (X_M, Y_M) is feasible when it does not collide with any obstacle and there exists at least a set of feasible attitudes (Y_M, Z_M), (X_M, Z_1), (X_M, Z_2), ---, (X_M, Z_{M-1}) of the (N_M-*1*)(*M*)-th, $N_1(1)$-th, $N_2(2)$-th, ---, $N_{M-1}(M-1)$-th arms. A modified feasible attitude set of the *n*-th arm at X is a set of grid points that are occupied by the joint of feasible attitudes of the *n*-th arm when its movable end is located at *X* and denoted as A(X, *n*).

Modified n-connectivity: For $n = 1(m)$ ($1 \le m \le M$), adjacent grid points X_1 and X_2 are called *n*-connective, when base arm attitudes (X_1, Y) and (X_2, Y) of the *m*-manipulator are feasible (Y is the base arm joint position of the *m*-th manipulator). For other *n* except $n = N_M(M)$, adjacent grid points X_1 and X_2 are called *n*-connective, when there is at least one connecting point pair (Y_1, Y_2) of FAS pair A(X_1, *n*) and A(X_2, *n*), and Y_1 and Y_2 are (*n+1*)-connective. For $n = N_M(M)$, adjacent grid points X_1 and X_2 are called *n*-connective, when they are $N_1(1)$, $N_2(2)$, ---, $N_{M-1}(M-1)$-connective, there is at least one connecting point pair (Y_1, Y_2) of FAS pair A(X_1, *n*) and A(X_2, *n*), and Y_1 and Y_2 are (*n+1*)-connective.

The corollary below ensures that BFA finds paths of the composite manipulator correctly with the computation time and memory space proportional to the total number of arms of the composite manipulator, i.e. ΣN_m.

[Corollary] Under the assumptions of the theorem in Sec. 2, the algorithm shown in Fig.1 determines existence of paths of the composite manipulator that connect start and goal attitudes, and finds the paths without any backtrack correctly when they exist.

(Proof)

The corollary can be proved when the condition of the theorem in Sec. 2 is also the necessary and sufficient one for the existence of paths of composite manipulators. The necessity is apparent. About the sufficiency firstly, when points X and Y in the work space of the composite manipulator are $N_M(M)$-connective (*M* is the number of manipulators), they are apparently $N_m(m)$-connective for all $m \le M$ from the definition of $N_M(M)$-connectivity, therefore from the theorem in Sec. 2 (in this case roles of joints should be replaced by those of movable ends), for all *m*, *m*-th manipulator has feasible attitude T(X, *m*) and T(Y, *m*), and there exists at least one path that brings T(X, *m*) to T(Y, *m*) without colliding with obstacles. Here, T(X, *m*) is an attitude of the *m*-th manipulator (i.e. a set of consistent attitudes of all arms in the *m*-th manipulator), in which the movable end of the N_m-th arm (top arm) of the *m*-th manipulator is located at point X. Then for any given locus *L* consists of a sequence of mutually $N_M(M)$-connective points, it is possible to constitute collision free paths of all manipulators, in which movable end positions of their top arms coincide with *L*. (Q.E.D.)

From the definition of feasible attitudes of arms in the composite manipulator, it is apparent that path calculations of individual manipulators can be executed completely in parallel except calculations for the $N_M(M)$-the arm.

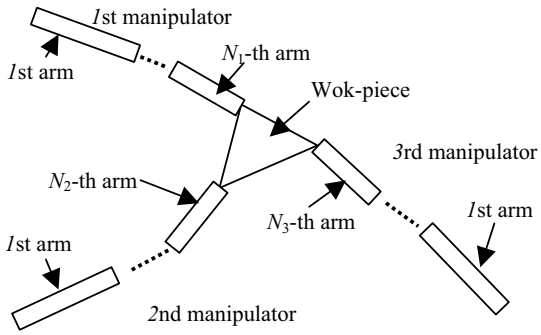

Fig.9 Multi manipulators holding 3 different points

B. General Cases

Generally, top arms of the multiple manipulators must hold different positions in order to hold work-pieces stably as shown in Fig.9. Path planning for these cases also can be accomplished efficiently based on BFA by defining virtual arms corresponding to the work-pieces. Path planning for *2* cooperating manipulators is trivial, i.e. only additions of virtual arms are sufficient. In Fig.10, work-piece W is considered as a virtual arm that is connected to the (original) top arm of the *2*nd manipulator at point Q. Then, when the movable end positions of the top arm of the *1*st and *2*nd manipulators share the same position P (the top arm of the *2*nd manipulator is the virtual arm W in this case), the original 2 manipulators constrained to collaborate while holding different positions P and Q of work-piece W.

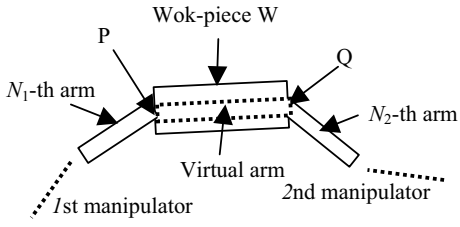

Fig.10 Cooperation between *2* manipulators

Here, an advantage of BFA in planning paths of cooperating multi manipulators is its ability for generating locus and attitude constrained paths. One of reasons to exploit multi manipulators is that it can convey large work-pieces while maintaining their desired attitudes. As discussed in the previous section, it is easy for BFA to constrain loci and attitudes of the virtual arm, i.e. the work-piece.

A virtual arm with *M-1* branches, of which attitudes are adjusted so that they can hold *M-1* different points on a work-piece, makes BFA applicable also to path planning for *M* (> *2*) cooperating manipulators. In an example of the cooperation among *3* manipulators shown in Fig. 11, the *3*rd manipulator has a virtual arm with *2* branches, and the end points of its *1*st and *2*nd branches and its joint are located at P, Q and R, respectively, so that top arms of the *1*st, *2*nd and *3*rd manipulators can hold points P, Q and R on the work-piece.

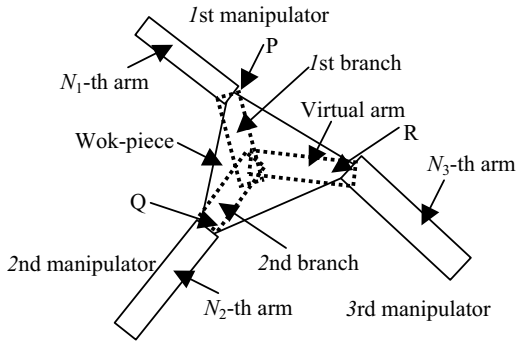

Fig.11 Cooperation among *3* manipulators

To enhance BFA applicable also to cooperation among *M* (*M* > *2*) manipulators, feasible attitudes of virtual arms with multiple branches must be defined. This is an extension of the modified feasible attitude and defined as follow (modifications of FASs and *n*-connectivity according to this extension are straightforward). Here, it is assumed that the virtual arm is connected to the top arm of the *M*-th manipulator, and the *m*-th branch of the virtual arm is located at the holding position of the *m*-th manipulator.

Feasible attitudes of virtual arms: Attitude (Y_1 ,Y_2, ---, Y_{M-1}, X) of the virtual arm with *M-1* branches connected to the $N_M(M)$-th arm (top arm of the *M*-th manipulator) is called feasible when it does not collide with any obstacle, and there exists at least a set of feasible attitude (X, Z_M) of $N_M(M)$-th arm and (Y_m, Z_m) of the $N_m(m)$-th arm ($m = 1, 2,$ ---, *M-1*)). Here, (Y_1 ,Y_2, ---, Y_{M-1}, X) is the attitude of the virtual arm, in which its joint and its *m*-th branch are located at X and Y_m.

The above definition makes calculations of feasible attitudes and determinations of *n*-connectivity complicated, because locations of the virtual arm are represented by sets of (*M-1*) points that are occupied by (*M-1*) braches of the virtual arm (not single points), i.e. the number of possible locations of the virtual arm increases exponentially with the number of cooperating manipulators. However, because locations of the end points of these (*M-1*) braches are constrained to the holding positions of the work-piece, usually the number of possible combinations is limited. Especially when attitudes of work-pieces are constrained in order to convey work-pieces stably, e.g. to maintain attitudes of the work-pieces parallel to the ground, the number of possible combinations decreases further.

The one of the most important advantages of this approach is that paths of individual manipulators except the locus of the virtual arm can be calculated completely in parallel, as discussed at the end of Sec. 3.A. Therefore when numbers of CPUs are available, loci of multiple manipulators can be calculated within almost the time that is required for a single manipulator.

IV. Conclusion

An approach to the development of a path planning algorithm for cooperating multi manipulators has been discussed. Based on a newly proposed algorithm BFA, the approach enables path planning in complicated environments within practical time, i.e. required computation time and memory space are proportional to the total number of arms included in individual manipulators. Moreover when many CPUs are available, path planning can be accomplished within the duration required for a single manipulator.

As future works, firstly BFA must be implemented for *3*-dimensional path planning. Also, occurrence of collisions among arm themselves must be evaluated. Different from path planning for single manipulators, collisions among multi manipulators may not be easy to remove by local adjustments of paths.

References

[1] Y. K. Hwang and N. Ahuja, "Gross motion planning-A survey," *ACM Computing Surveys*, Vol. 24, No.3, pp. 219-291, 1992.

[2] K. K. Gupta, "Motion planning for flexible shapes (systems with many degrees of freedom): a survey," *The Visual Computer*, Vol.14, No.5-6, pp. 288-302, 1998.

[3] S. Tamura, T. Yanase, Md. N. Islam, T. Ito and H. Miyashita, "A new path planning algorithm for manipulators," *IEEE Intl. Conf. on Systems, Man and Cybernetics*, pp. 2242-2247, Oct. 2005.

[4] Z. Sun, D. Hus, T. Jiang, H. Kurniawati and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Trans. on Robotics*, Vol. 21, No. 6, pp. 1105-1115, 2005.

[5] M. Saha and J. C. Latombe, "Finding narrow passages with probabilistic roadmaps: the small-step retraction method," *Autonomous Robots*, Vol. 19, pp. 301-319, 2005.

[6] D. Hsu, J. C. Latombe and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *The International Journal of Robotics Research*, Vol. 25, No. 7, pp. 627-643, 2006.

[7] Zhenwang Yao and Kamal Gupta: "Path planning with general end-effecter constraints," *Robotics and Autonomous Systems*, Vol. 55, pp. 316-327, 2007.

[8] Md. N. Islam, T. Murata, S. Tamura and T. Yanase, "Evaluation of a new backtrack free path planning algorithm for manipulators," *IEEJ, Trans. EIS*, Vol.128, No.8, pp. 1293-1302, 2008.