Efficient Self-Collision Avoidance based on Focus of Interest for Humanoid Robots

Cheng Fang, Alessio Rocchi, Enrico Mingo Hoffman, Nikos Tsagarakis, Darwin Caldwell

Efficient Self-Collision Avoidance based on Focus of Interest for Humanoid Robots

Cheng Fang, Alessio Rocchi, Enrico Mingo Hoffman, Nikos G. Tsagarakis and Darwin G. Caldwell

***Abstract*— This paper deals with the self-collision avoidance problem for humanoid robots in an efficient way. Self-collision avoidance is introduced as a constraint for each task in a hierarchical Inverse Kinematic (IK) problem. Since the number of link pairs which needs to be updated and checked for self-collision, in every control loop, is large, the novel concept of Self-Collision Avoidance Focus of Interest (SCAFoI) is proposed. SCAFoIs permits to predict and dynamically select the necessary link pairs to be checked on-line to improve the computation efficiency. For each of the several SCAFoIs, which corresponds to the related pairs of kinematic chains of the whole body, the status of the relative positional relationship is predicted. The prediction is done using a Support Vector Machine (SVM) which is a widely used classifier from the machine learning field. Moreover, techniques are proposed to guarantee and improve the prediction performance of the trained classifier. The effectiveness of the framework is verified using the whole-body motion control library OpenSoT by simulation on the model of the recently developed humanoid robot WALK-MAN.**

I. INTRODUCTION

Recently, as robots have moved from traditional factory applications to the potential use in service and domestic environments, researchers have increasingly been focusing on humanoid robots which have potential capabilities to effectively execute and complete the realistic tasks that are central to our daily lives. To do this, one of the most important capabilities for humanoid robots is to be able to detect and avoid obstacles and self-collisions, the latter is the subject considered and discussed in this paper.

Self-collision avoidance methods for humanoid robots can be roughly classified into two categories: global offline methods and local on-line methods. The most popular approach in the first category is the sampling-based method, such as Rapid-exploring Random Trees (RRT) [1], which tries to explore the feasible configurations (considering all of the constraints) as much as possible in the configuration space and search for the best path connecting the initial configuration and the target configuration. It is proved very successfully in complex motion planning in high-dimensional situations like humanoid robots. However, when it comes to the practical application on real robots, some drawbacks are pointed out in [2]: complexity of transforming the planned path to practicable trajectory and discontinuity of path in C^1.

*The work is supported by European Community's 7th Framework Programme SAPHARI FP7-ICT-287513 and WALK-MAN FP7-ICT-2013-10.

The authors are with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163, Genova, Italy (e-mail: {cheng.fang, alessio.rocchi, enrico.mingo, nikos.tsagarakis, darwin.caldwell}@iit.it).

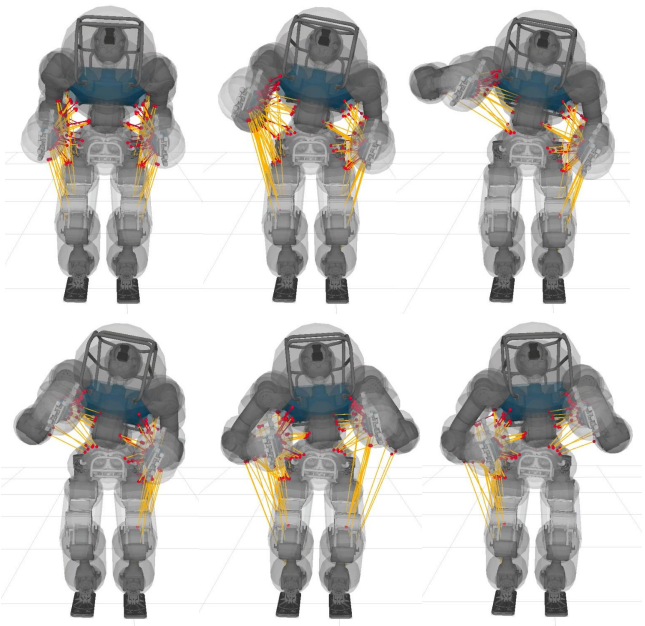

Fig. 1. SCAFoI activation while executing a valve turning task on the simulated humanoid robot WALK-MAN.

For the second category, earlier researchers [3], [4] employed some fast on-line collision detection methods to stop the motion of the robot when the next generated reference joint angle data is predicted to result in a possible collision. In this case, the stability of the robot would be heavily influenced due to the abrupt stop behaviour, which is not desirable during task execution. To react to the potential collision several steps ahead, researchers [5], [6] usually establish a potential field based on the minimum distance between the most closest link pairs of a humanoid robot and try to control the distance to be as large as possible. Specifically, the gradient of the minimum distance with respect to the joint vector is calculated and projected into the null space of a main task with higher priority by using the *task priority framework*. In this case, the main shortcoming is that the priority of the self-collision avoidance is low and the corresponding task only works in the null space of a higher priority task. For example, the robot can only use self-motions to avoid self-collisions when the main task is set in Cartesian space. This implies that no effective self-collision avoidance motions will be performed if the main task is wrongly guided to a self-collision pose for some reason [5]. Another problem is that the robot might be stuck at a certain threshold distance because the self-collision avoidance task

could be only activated when the minimum distance is smaller than the threshold [7]. To overcome these limitations, Mansard et al., [8], [9] proposed a framework called "Stack of Tasks (SoT)", which is a general and hierarchical framework in nature for fast humanoid robot motion generation. Compared to the traditional task priority framework, the tasks and the constraints are able to be described by equalities or inequalities flexibly [10]. The strict satisfaction of the tasks in the stack is replaced by a sequence of optimization problems where tasks are achieved as much as possible while subject to some constraints, which means that some task errors would be allowed properly. In this framework, the self-collision avoidance could be deployed as an inequality constraint for every task, and the optimization of minimizing the task error is only conducted inside the feasible scope of the self-collision avoidance constraint. In this way, the inequality constraint should be always obeyed in any situations as a global constraint, even if this implies corresponding tasks could not be fully achieved because of it. In the meantime, this kind of constraint would never influence other tasks if the constraint is not violated, on the contrary, the generated virtual repulsive force based on the potential field could always take effect to actively change the configuration of robot in the traditional framework. Obviously, all these new features make the self-collision avoidance functionality more reasonable. The SoT framework is considered to realize the self-collision avoidance in this paper.

In order to realize the self-collision avoidance constraint, the position information of every link pair of a humanoid robot is usually updated to calculate the minimum distance and further to find out several closest link pairs to be constrained in every control loop. Since humanoid robots are usually composed of many links, the number of all the link pairs to be inspected would be excessively large for a control cycle, during which much time is spent completing the corresponding computation [2], [11]. There are two common approaches to reduce this computation effort. One is to decrease the number of the necessary link pairs to be checked according to an thorough analysis of the kinematic structure or/and the practical experience. Through heuristic and exhaustive search or sampling methods, a table of link pairs which would never collide with each other can be precomputed and removed from the list of pairs for which to compute the minimum distance information [1], [2], [3]. However, the number of remaining necessary link pairs to be checked is still large. Another approach is to try to use Exhaustive Attack method to find out the collision-free joint angle motion range for a certain joint to form a look-up table for on-line use. The problem of this approach would be that the data-set for the complete table is too large to fully apply in practice, so this look-up table method can be only used for some compound joints or some specified link pairs of interest [4], [12].

In this paper, we propose a novel concept called Self-Collision Avoidance Focus of Interest (SCAFoI) to indicate the body segments where we should focus to check for potential self-collisions when the robot is at a certain posture. The basic idea of this new method is that since the mechanical structure and kinematic chains of a robot are invariant, it is possible for the robot to obtain some experience/knowledge about potential self-collisions after observing a large number of its configurations. That implies that the robot will have the capability of locating the SCAFoIs immediately when a set of joint angles is given. We implement this concept using machine learning techniques with the purpose to speed up the computation of the on-line link pair checking.

The overall method and the details of its implementation together with results from simulation verification on WALK-MAN humanoid robot are introduced in the following sections.

II. SELF-COLLISION AVOIDANCE FRAMEWORK

In our work, OpenSoT [13], a robotics library which is inspired and derived from SoT, is used to deal with the kinematic inversion problem. The concept of the library is to maximally decouple the tasks/constraints description and the solvers implementation. OpenSoT provides base classes and standard interfaces to specify tasks, constraints and solvers and is distributed with a library of already implemented components that allows to write and solve IK problems. The architecture of OpenSoT encourages collaboration and helps integration and code maintenance.

In the default IK solver of OpenSoT, based on QP Optimization, a generic IK Problem is described as follows [13]:

$$\begin{aligned} \dot{\boldsymbol{q}}_i = \underset{\dot{\boldsymbol{q}}}{\operatorname{argmin}} \ & \|\boldsymbol{J}_i\dot{\boldsymbol{q}} - \dot{\boldsymbol{e}}_i\| \\ s.t. \ & \boldsymbol{A}_i\dot{\boldsymbol{q}} \leq \boldsymbol{b}_i \end{aligned} \tag{1}$$

where $\boldsymbol{J}_i$ denotes the Jacobian matrix of the link frame which we want to control with respect to a certain base frame for task $i(i = 1, 2, ...n)$. $\dot{\boldsymbol{q}}$ is the joint velocity vector which is the optimization variable of the problem. $\dot{\boldsymbol{e}}_i$ is the derivative of the task error with respect to time. At last, the optimal joint velocity should satisfy the inequality constraint $\boldsymbol{A}_i\dot{\boldsymbol{q}} \leq \boldsymbol{b}_i$. As a result of the aforementioned formulation, a series of QP problems are going to be solved in cascade to generate the (locally) optimal whole-body motions according to the set of tasks with the specified priorities [14]. Finally, the solution to the last task in the stack can be written as follows:

$$\begin{aligned} \dot{\boldsymbol{q}}_d = \underset{\dot{\boldsymbol{q}}}{\operatorname{argmin}} \ & \|\boldsymbol{J}_n\dot{\boldsymbol{q}} - \dot{\boldsymbol{e}}_n\| \\ s.t. \ & \boldsymbol{J}_1\dot{\boldsymbol{q}} = \boldsymbol{J}_1\dot{\boldsymbol{q}}_1 \\ & \vdots \\ & \boldsymbol{J}_{n-1}\dot{\boldsymbol{q}} = \boldsymbol{J}_{n-1}\dot{\boldsymbol{q}}_{n-1} \\ & \boldsymbol{A}_1\dot{\boldsymbol{q}} \leq \boldsymbol{b}_1 \\ & \vdots \\ & \boldsymbol{A}_n\dot{\boldsymbol{q}} \leq \boldsymbol{b}_n \end{aligned} \tag{2}$$

where $\dot{\boldsymbol{q}}_d$ is the final joint velocity to be commanded. From the formulation of (2), it is clear that the final solution to task n is only optimized inside the intersection of null spaces of all the tasks with higher priorities, i.e., equalities $\boldsymbol{J}_i\dot{\boldsymbol{q}} =$

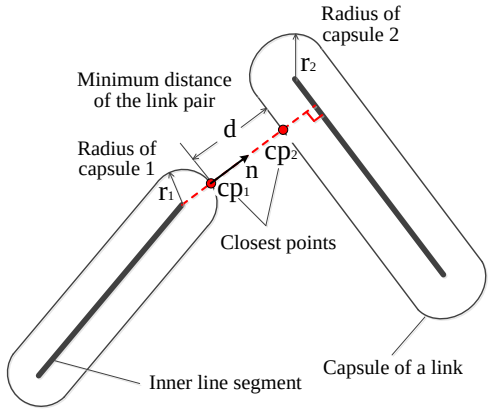

Fig. 2. Schematic diagram of the collision avoidance constraint on a capsule pair representing the collision model for a link pair.

$\boldsymbol{J}_i\dot{\boldsymbol{q}}_i(i = 1, 2, ..., n-1)$ and inequalities $\boldsymbol{A}_i\dot{\boldsymbol{q}} \leq \boldsymbol{b}_i(i = 1, 2, ..., n-1)$ must be satisfied.

In our framework, the self-collision avoidance constraint will be formulated as an inequality constraint for each task in the IK Problem, as shown in (1), to guarantee the safety of the robot. The basic idea is to calculate the minimum distance between every link pair in every control loop and obtain a list of several closest link pairs according to these minimum distances. For each of these link pairs of interest, we try to control the relative velocity of the one link with respect to the other in the direction connecting the closest points on the two links respectively, which is called velocity damping firstly introduced in [15] and reformulated inside the SoT [16], to avoid the potential collision between them.

So, to accelerate the computation of minimum distance of each link pair, we should employ some simpler collision model for each link first. In our paper, the capsule, sphere-swept line, is chosen among a lot of types of bounding volumes because of the convenience of the computation. As shown in Fig.2, the minimum distance of the link pair can be just considered as the minimum distance between the inner line segments minus the sum of the radii of the capsules for the two links respectively. The method proposed in [2] is employed to generate the minimum bounding capsule for each of exact link body geometries represented by mesh. Furthermore, the minimum distance between a capsule pair and the shortest points on them can be easily obtained by using the Flexible Collision Library (FCL) [17]. FCL is a fully templated library which aims at general proximity calculation and collision detection on various types of collision geometries such as axis-aligned bounding box (AABB) and oriented bounding box (OBB). As of version 3.0 of the library, the capsule collision geometry has been added [18].

Once the closest points are computed, for instance, $\boldsymbol{cp}_1$ and $\boldsymbol{cp}_2$ shown in Fig. 2, the relative motion of the capsule pair is going to be restricted in the direction:

$$\boldsymbol{n} = \frac{\boldsymbol{cp}_2 - \boldsymbol{cp}_1}{\|\boldsymbol{cp}_2 - \boldsymbol{cp}_1\|} \tag{3}$$

Where the distance $\|\boldsymbol{cp}_2 - \boldsymbol{cp}_1\|$ is calculated according to the Euclidean L2-norm. So, the relative velocity constraint in this direction can be formulated as follows:

$$\boldsymbol{n}^T[\boldsymbol{J}(\boldsymbol{cp}_1, \boldsymbol{q}) - \boldsymbol{J}(\boldsymbol{cp}_2, \boldsymbol{q})]\dot{\boldsymbol{q}} \leq \varepsilon\frac{d - d_s}{\Delta t} \tag{4}$$

In (4), $\boldsymbol{J}(\boldsymbol{cp}_1, \boldsymbol{q})$ and $\boldsymbol{J}(\boldsymbol{cp}_2, \boldsymbol{q})$ refer to the Jacobian matrices of the frames located at the closest points $\boldsymbol{cp}_1$ and $\boldsymbol{cp}_2$ respectively with respect to the base frame. Since the relative location of the capsule relative to the corresponding link frame is fixed, these two Jacobians can be obtained based on the Jacobians of their link frames by some simple transformation. It is worth noting that only the linear velocity component of the Jacobian (the first three rows) is taken into account in this case. Then, the formula $[\boldsymbol{J}(\boldsymbol{cp}_1, \boldsymbol{q}) - \boldsymbol{J}(\boldsymbol{cp}_2, \boldsymbol{q})]\dot{\boldsymbol{q}}$ can be considered as the relative velocity of the capsule pair. In addition, d_s is the safety distance which is the shortest distance allowed for the capsule pair, Δt denotes the control loop period, and ε indicates the gain value which is used to smoothly reduce the relative approaching velocity at which the threshold distance is reached. So, the inequality in (4) means the velocity projected from the original relative velocity of the capsule pair onto the direction connecting the closest points on the capsules would never make the distance between them smaller than the safety distance d_s, which is then used to avoid potential collision between the corresponding link pair. Furthermore, for multiple link pairs, the constraint formulation evolves to:

$$\begin{aligned} \begin{pmatrix} \boldsymbol{n}_1{}^T\bar{\boldsymbol{J}}_1 \\ \vdots \\ \boldsymbol{n}_k{}^T\bar{\boldsymbol{J}}_k \end{pmatrix}\dot{\boldsymbol{q}} &\leq \begin{pmatrix} \bar{d}_1 \\ \vdots \\ \bar{d}_k \end{pmatrix} \\ \boldsymbol{N}\,\dot{\boldsymbol{q}} &\leq \boldsymbol{D} \\ \text{where} \quad \bar{\boldsymbol{J}}_i &= \boldsymbol{J}(\boldsymbol{cp}_{1,i}, \boldsymbol{q}) - \boldsymbol{J}(\boldsymbol{cp}_{2,i}, \boldsymbol{q}) \\ \text{and} \quad \bar{d}_i &= \varepsilon_i\frac{d_i - d_{s,i}}{\Delta t} \end{aligned} \tag{5}$$

In this case, in order to add the self-collision avoidance constraint for k pairs of links in the IK Problem, the inequality constraint of every task in (1) should be replaced by $\boldsymbol{N}\dot{\boldsymbol{q}} \leq \boldsymbol{D}$ in (5).

III. SELF-COLLISION AVOIDANCE FOCUS OF INTEREST

As introduced in the previous section, in order to guarantee the safety of the robot, the natural idea is to check the relative position information of every link pairs, which has the potential to collide with each other, and then to add the collision avoidance constraint for each pair in every loop. However, for a humanoid robot with many links, the number of link pairs to be checked is usually very large as it grows quadratically with the number of links, resulting in a heavy computational burden for a very limited control period. The commonly used method is to remove some link pairs which would never collide with each other for any configurations or at least for the specified task, and some other link pairs which are connected by joints because this kind of collision can be avoided by adjusting the joint limits [1], [2], [3]. But,

generally speaking, the number of the remaining necessary link pairs is still large. For instance, in our case, the WALK-MAN humanoid robot with 31 DOFs has 383 link pairs which needs to be inspected on-line after removing some unnecessary link pairs by using the collision checking tools in Moveit! offline [19].

A. Concept

The concept of Self-Collision Avoidance Focus of Interest (SCAFoI) is proposed in this paper to describe and indicate the most likely place(s) where the self-collision of humanoid robot could happen. There are two reasons for introducing the SCAFoI notion: first, it is noticed in practice that the self-collision avoidance constraint would not be active most of the time during task execution because of the joint velocity constraint. That means the limited joint velocities would restrict the linear velocity in the direction of the shortest distance of link pair, which is not able to cause any collision in the feasible space of joint velocity vector when the relative distance of link pair is still quite far. Second, the kinematic structure of the humanoid robot is usually fixed and the robot would complete a large number of repetitive or similar tasks frequently. To make it more intelligent, a natural idea is to endow the robot with the capability of predicting the most dangerous parts of its body when a configuration is specified by using previous experience as a human being does. So, the SCAFoI model is introduced as an attention control tool to quickly identify and locate the most dangerous places around the body of the robot where the potential collision is likely to occur. In this way, we can selectively update the position information of some links and compute the shortest distances between these link pairs of interest and add the corresponding constraints to speed up the on-line calculation of self-collision avoidance. As far as we know, this is the first time that the concept of Self-Collision Avoidance Focus of Interest is proposed.

B. Design

The basic idea of SCAFoI is to design a classifier to identify the binary status of the robot about self-collision, safe or dangerous, given a certain joint angle vector. Since the whole joint space of humanoid robot is too complicated to classify, we suggest dividing the robot into six parts or kinematic chains: *head*, *torso*, *left arm*, *right arm*, *left leg* and *right leg*. In this way, for each two of the six parts, one classifier would be designed to recognize the status between the two parts. According to our practical experience, the success rate of prediction of the classifier would increase significantly due to the reduction of dimensionality of the relevant joint space. There might be some exceptions for several part pairs which would never collide with each other inside the feasible joint space of the whole body. In our case, the part pairs, *head - left arm*, *head - right arm*, *head - left leg*, *head - right leg*, *head - torso*, *torso - left leg* and *torso - right leg* (there is a protective frame around the head), are excluded from the checking list. So, we have 8 pairs of parts left which needs to be checked on-line in every control loop. Consider that only the collisions between different kinematic chains are taken into account in this paper because we assume that the collisions between the adjacent links and the links from the same kinematic chain could be avoided by suitable joint limits.

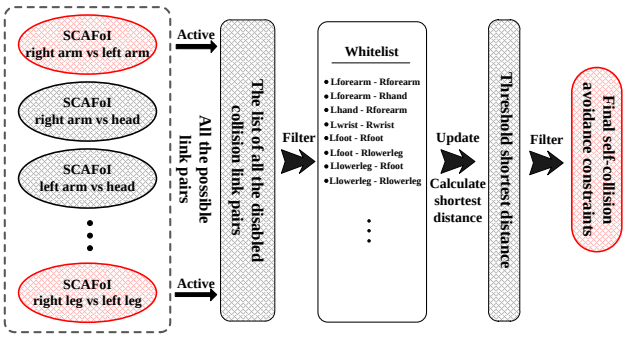

Fig. 3. Whole procedure of the SCAFoI model.

So, one classifier is responsible for one pair of kinematic chains, which is actually one SCAFoI. In every control loop, these classifiers would be run to indicate which pairs of parts should be paid attention to, and the necessary link pairs for each active SCAFoI would be retrieved. The link pairs to check are the remaining pairs after subtracting the link pairs which would never collide with each other, which is obtained by the offline collision checking tool in Moveit!, from all of the possible link pairs from the corresponding pair of kinematic chains. After that, a whitelist is created to include all the necessary link pairs from all the active SCAFoIs. The position information of all the links in this whitelist would be updated immediately and the shortest distance between each pair of links would be calculated subsequently. In this way, we can guarantee that only a minimum set of link pairs are updated. Finally, in order to further simplify the calculation, several self-collision avoidance constraints as the inequality in (4) would be computed and added only for the link pairs whose shortest distances are lower than a certain threshold distance set in advance. The whole procedure is presented in Fig. 3.

C. Implementation

In this paper, we employ the Support Vector Machine (SVM) [20], [21], which is a very popular binary classifier in machine learning field (SVM can be also extended to apply to multi-class classification problems), to implement the classification. In the classical binary classification problem, given training vectors $\boldsymbol{x}_i \in R^n, i = 1, ..., l$, in two classes, and a target indicator vector $\boldsymbol{t} \in R^l$ such that $t_i \in \{1, -1\}$, a linear model is used as a hyperplane to try to separate these training vectors as much as possible:

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b \tag{6}$$

where $\phi(\boldsymbol{x})$ denotes a mapping function which is responsible for transforming the original data into a feature space which is supposed to classify these data easier, and b is the bias parameter. Then, the classification problem is

converted to an optimization problem where we optimize the parameters of this linear model, $\boldsymbol{w}$ and b, to maximize the shortest distance (margin) between the two classes of data:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi_i \\ s.t. \ \ & t_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, ..., l. \end{aligned} \tag{7}$$

where ξ_i indicates the slack variable for each training data to allow some of them to be misclassified, and parameter $C > 0$ is used to control the trade-off between the slack variable penalty and the margin. After the optimization (7) is solved, the decision function below can be used for prediction:

$$\text{sgn}(\boldsymbol{w}^T\phi(\boldsymbol{x}) + b) = \text{sgn}\left(\sum_{i=1}^{l} y_i\alpha_i\boldsymbol{k}(\boldsymbol{x}_i, \boldsymbol{x}) + b\right). \tag{8}$$

where $\boldsymbol{k}(\boldsymbol{x}_i, \boldsymbol{x}) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_i)$ is called kernel function in SVM, and α_i is the Lagrange multiplier term for the constrained optimization problem (7). The vectors whose corresponding α_i satisfy $\alpha_i \neq 0$ are named as Support Vectors which make actual contribution to the classification decision function above.

In our case, the training vectors $\boldsymbol{x}_i$ are from different degenerated joint spaces of the humanoid robot depending on the various SCAFoIs. For instance, if the SCAFoI refers to the pair of left arm and right arm, the $\boldsymbol{x}_i$ is the vector from the joint space spanned only by the 2×7 joint angles from the two arms because the other joint angles are not able to change the relative position of the arms. In addition, the target value for the training vector, $t_i = 1$, when the two parts are too close, otherwise, $t_i = -1$ meaning the current posture is safe for collision. Here, we employ the shortest distance between kinematic chains to measure the positional relationship of two parts, which can be defined as the minimum of all the minimum distances between any pairs of links from the two chains. In this way, a target value t_i is supposed to be set to be 1 when the shortest distance is lower than a lower threshold distance, d_{t_l}, as shown in Fig. 4, while $t_i = -1$ when the shortest distance is larger than another upper threshold distance, d_{t_u}. So, the task of the designed classifier is to predict the value of t_i given a configuration. It is worth noting that we employ two different threshold distances to classify the data in order to increase the success rates of classification and prediction. The data inside the transition area between the two threshold distances could be regarded as any one of the two classes. This design is going to be explained in detail in the next section.

As the rate of correct prediction for a classifier can never reach up to 100% in practice, it would cause serious damage to the robot if the designed classifier could not predict the dangerous status of the robot successfully. So, taking this situation into account, the SVM classifier is combined with a quadratic interpolation function based on three most recent actual shortest distances to predict the status of two parts more precisely in order to avoid misclassification in dangerous areas as much as possible. Specifically, the SVM classifier is only in charge of predicting when the status of the corresponding SCAFoI start changing from inactive to active, and then the actual shortest distance can be updated normally when the robot enters into the area where the target value is

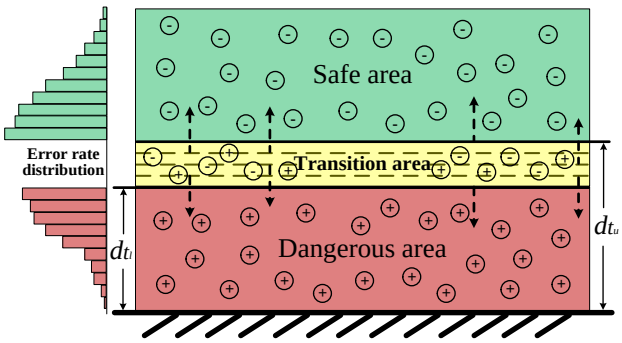

Fig. 4. Schematic diagram of the training data distribution and the error rate distribution of prediction for the SVM classifer.

Algorithm 1 SCAFoI prediction algorithm

Initialize $d_1 = d_2 = d_3 = 0$
/* d_1, d_2 and d_3 are the three most recent real shortest distances, which would be set to be 0 when they are larger than the upper threshold distance d_{t_u} */
while $(!timeout())$ **do**
  if $(d_1 \neq 0$ && $d_2 \neq 0$ && $d_3 \neq 0)$ **then**
    $d_p \leftarrow predict\_QLI(d_1, d_2, d_3)$;
    if $(d_p < d_{t_u})$ **then**
      $d_r \leftarrow update\_shortest\_distance(\boldsymbol{q})$;
      if $d_r < d_{t_u}$ **then**
        $add\_constraints(d_{t_l})$;
        $store(d_r)$;
      else
        $store(0)$;
      end if
    else
      $store(0)$;
    end if
  else
    $t \leftarrow predict\_SVM(\boldsymbol{q})$;
    if $(t = 1)$ **then**
      $d_r \leftarrow update\_shortest\_distance(\boldsymbol{q})$;
      if $(d_r < d_{t_u})$ **then**
        $add\_constraints(d_{t_l})$;
        $store(d_r)$;
      else
        $store(0)$;
      end if
    else
      $store(0)$;
    end if
  end if
end while

1, then the quadratic interpolation function can predict the next shortest distance by using previous distance data until the robot goes back to the safe area again. The idea is that the real shortest distances for the previous configurations in the neighborhood of the current configuration could be used for more accurate prediction if these local information have already been updated before. That means the continuity of the joint trajectory could be utilized to compensate for the natural inaccuracy of the general classifier to guarantee the success rate of prediction especially for the dangerous area.

For each SCAFoI, the detailed algorithm for this is shown in Algorithm 1, where $predict\_QLI(d_1, d_2, d_3)$ is the Quadratic Lagrangian Interpolation Function which is used to predict the shortest distance for the next time step based on the three most recent actual distances, d_1, d_2 and d_3. Since the control period is a constant, this function would be simplified to:

$$d_p = l_3(t)d_3 + l_2(t)d_2 + l_1(t)d_1 = d_3 - 3d_2 + 3d_1 \quad (9)$$

and function $predict\_SVM(\boldsymbol{q})$ would be implemented by using LIBSVM which is currently one of the most widely used SVM libraries [22], [23]. $add\_constraints(d_{t_l})$ would compute and add the collision avoidance constraints for the link pairs of which the shortest distances are lower than d_{t_l}.

IV. IMPROVEMENT TO THE PREDICTION ACCURACY OF SVM

Since it is found in practice that it is very hard for the designed classifier to classify the two-class data set which is divided by one exact shortest distance in our case, we finally choose to employ a transition area as a "soft" partition instead of one single distance to train the classifier to separate the data as much as possible. As shown in Fig. 4, the transition area is defined as an area between the dangerous area where the shortest distances of the robot's configurations inside are lower than d_{t_l}, and the safe area where the corresponding shortest distances for the postures inside are larger than d_{t_u}. Please note that the term area is actually a concept of joint space here, but we use a rectangular area instead of the complicated high-dimensional space for the convenience of visualization in Fig. 4. In this way, we just focus on the performance of the designed classifier in the safe area and dangerous area, and do not care about the transition area. This area could be regarded as a buffer area or protective area before the dangerous area to give the classifier more opportunities to successfully judge if the robot is entering a non-safe area.

To train the classifier, a set of training data needs to be generated in some way. For the design of the transition area, we have more choices for the strategy of generating training data. Any two distances between d_{t_u} and d_{t_l}, which are visualized by the dashed dividing lines in Fig. 4 for instance, could be selected to be used to generate the training data. Specifically, a joint angle vector $\boldsymbol{q}$ is generated randomly from the related joint space, and then, the shortest distance of this configuration is going to be calculated. the target value of vector $\boldsymbol{q}$ would be considered to be -1 if the shortest distance is larger than the bigger one of the two chosen distances, while the target value equals to 1 if the distance is lower than the smaller distance. In this way, we can get a large number of sample training vectors in two classes. The amounts of training data in the two classes is specified manually to make sure the balance of the two classes of data and to guarantee the number of random samples are big enough to describe or represent the spatial characteristics of safe area and dangerous area well. We are going to illustrate later that the number of training data is an important factor to affect the performance of the designed classifier.

After we obtain the generated training data set, a prediction model could be produced by using LIBSVM library. So, an important thing is to find a way to evaluate the trained model. In order to check the distribution of the misclassification rate in safe area and dangerous area, these two areas are divided into several subareas with the same interval in terms of the shortest distance and the same number of test data in each subarea would be collected for test. Finally, the misclassification rate for each subarea can be obtained after the prediction by the trained model. The profile of error rate is proved to tend to be a unimodal, bell-shaped curve, in which the error rates of the subareas close to the transition area are relatively high, while those of the subareas far away from the transition area are relatively low. Since the role of the designed classifier is to predict when the robot starts entering the non-safety area as precisely as possible, the average misclassification rate of the classifier is expected to be as low as possible and the shape of the error rate profile is required to be more peaked at the same time. So, a criterion, $I_{C_l}, (l = 1, 2, ..., k)$ for trained classifier C_l, is designed to measure the performance of classifier as follows:

$$\begin{aligned}
I_{C_l} &= w^l_{sa} I^l_{sa} + w^l_{sv} I^l_{sv} + w^l_{da} I^l_{da} + w^l_{dv} I^l_{dv}, \\
p^l_{sa} &= \frac{1}{m}\sum_{i=1}^{m} p^l_{s_i},\ I^l_{sa} = \frac{p^l_{sa} - \min\limits_l p^l_{sa}}{\max\limits_l p^l_{sa} - \min\limits_l p^l_{sa}}, \\
v^l_{sv} &= \sum_{i=1}^{m} i\frac{p^l_{s_i}}{m p^l_{sa}},\ I^l_{sv} = \frac{v^l_{sv} - \min\limits_l v^l_{sv}}{\max\limits_l v^l_{sv} - \min\limits_l v^l_{sv}}, \\
p^l_{da} &= \frac{1}{n}\sum_{j=1}^{n} p^l_{d_j},\ I^l_{da} = \frac{p^l_{da} - \min\limits_l p^l_{da}}{\max\limits_l p^l_{da} - \min\limits_l p^l_{da}}, \\
v^l_{dv} &= \sum_{j=1}^{n} j\frac{p^l_{d_j}}{n p^l_{da}},\ I^l_{dv} = \frac{v^l_{dv} - \min\limits_l v^l_{dv}}{\max\limits_l v^l_{dv} - \min\limits_l v^l_{dv}},
\end{aligned} \quad (10)$$

where $p^l_{s_i}$ is the error rate of trained classifier C_l in the subarea i from the safe area. The number i means the subarea is the i–th subareas in the safe area counting from the distance, d_{t_u}. And $p^l_{d_j}$ refers to the misclassification rate in the subarea j from the dangerous area counting from the distance, d_{t_l}. The whole criterion I_{C_l} consists of four indices. I^l_{sa} is used to measure the average error rate of the classifier in safe area, and I^l_{sv} is employed to measure the extent to how the distribution of the error rate in safe area is close to the borderline of the safe area, d_{t_u}, and indices I^l_{da} and I^l_{dv}

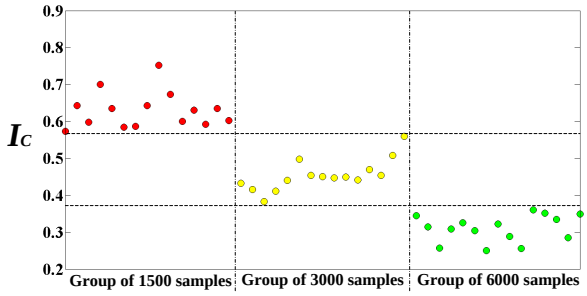

Fig. 5. The influence of the number of training data on the performance of trained classifier.

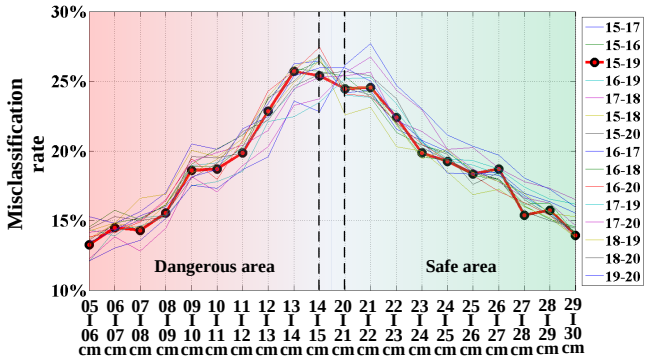

Fig. 6. Distribution profiles of misclassification rate for various trained classifiers of 6000 samples.

are the counterparts in the dangerous area. It is worth noting that all the four indices are relative indices which are already normalized to the range $[0, 1]$ to eliminate the influence of different magnitudes of various indices. Variables w^l_{sa}, w^l_{sv}, w^l_{da} and w^l_{dv} are the corresponding weighting factors for the four indices respectively, which holds the relationship $w^l_{sa} + w^l_{sv} + w^l_{da} + w^l_{dv} = 1$. Finally, the value of whole criterion I_{C_l} would be in the closed interval $[0, 1]$, and the smaller the value is, the better the performance of the classifier is considered.

We are going to use the SCAFoI for the pair of left arm and right arm as an example to present how to select the best trained classifier. In this SCAFoI, d_{t_u} and d_{t_l} are set to be 20cm and 15cm respectively, and then 15 pairs of borderlines which are used to generate the training data can be chosen: 15-16, 16-17, 17-18, 18-19, 19-20, 15-17, 16-18, 17-19, 18-20, 15-18, 16-19, 17-20, 15-19, 16-20, 15-20 (unit:cm). The numbers of subareas for classifier testing in the safe area and the dangerous area, m and n, are both equal to 10. The distance interval for each subarea is 1cm and the number of test data in each subarea is 1000. We set the total number of training data to be 1500, 3000, and 6000 for three groups of tests, and the ratio of training data of class +1 to those of class -1 is always 1 to 2 to try to make the training data to some extent uniformly distributed throughout the whole space. For the weighting factors of the criterion, $w^l_{sa} = w^l_{da} = 0.3$ and $w^l_{sv} = w^l_{dv} = 0.2$. The overall performance of the 15 trained classifier models are presented in Fig. 5:

In Fig. 5, each point represents one classifier trained by different training data set. It can be easily found that the number of training data would directly affect the performance of the classifier in terms of the criterion, I_C. So, we choose the group of trained classifiers of 6000 samples for further selection. The misclassification rate distribution profiles for this group of classifiers are shown in Fig. 6:

Based on the criterion, I_C, the best classifier is the one which is trained by the training data generated by the pair of borderlines, 15cm and 19cm. In Fig. 6, we can see that its shape of distribution profile of misclassification rate is relatively peaked and the performance of the classifier in dangerous area and safe area are balanced and relatively good, and all of the features would be very beneficial to identify the moment more precisely when the robot starts moving from the safe area to dangerous area. Since there are some distances between subareas, we can assume that the error rates in the subareas are independent of each other. So, the probability of the classifier failing in prediction to lead to a collision is at most $\prod_{j=1}^{n} p^l_{d_j}$, which is very small. Once the classifier succeeds in predicting the status of the robot in the dangerous area, the quadratic Lagrangian interpolation function would take over the prediction task which should be more accurate and stable. Therefore, in this way, the misclassification rate of the selected classifier model is acceptable. In the next simulation section, we are going to show that this framework would work well in practice.

V. SIMULATION

In this section, we are going to show some simulation results by using the model of Self-Collision Avoidance Focus of Interest on our humanoid robot, WALK-MAN. A sequence of screenshots are shown in Fig. 1 when the robot executes a valve turning task in the air, more details can be found in the attached video:

where the red points mean the shortest distance points between link pairs, the yellow lines indicate the link pairs which are from the activated SCAFoIs according to the prediction results, and the red lines denote the link pairs the shortest distance of which are smaller than lower threshold distance, which are constrained by the self-collision avoidance constraint in (4). Please note that the SCAFoI of left leg and right leg is removed because walking is not considered in this case. The prediction accuracy and the computation efficiency for this task are presented in Fig. 7 and Fig. 8.

In Fig. 7, the performance of the designed predictor in terms of prediction accuracy is pretty good because the configurations in this task are quite normal and are probably close to or even equal to some samples in the training set. The accuracy here is only considered and calculated in the safe area and the dangerous area. In Fig. 8, the full computation time refers to the time which were spent on updating all the link pairs and calculating their shortest distances. Computation was performed on a 64-bit Intel Xeon X5667 3.07GHz processor running Gazebo and Rviz together. It is obvious that the less SCAFoIs are activated, the

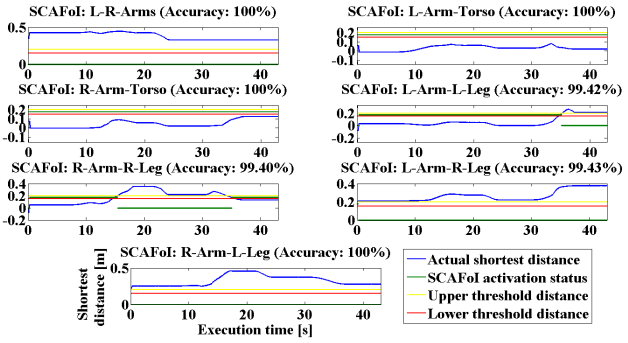

Fig. 7. Prediction accuracy for each SCAFoI in valve turning task.

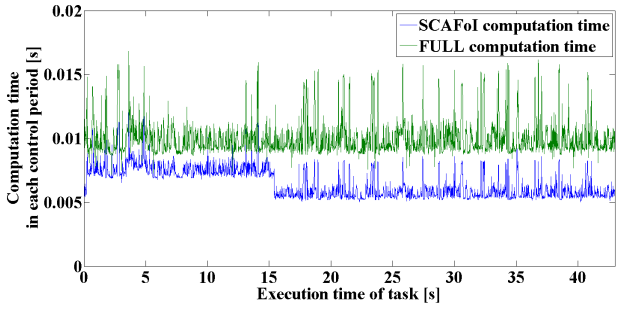

Fig. 8. Computation efficiency comparison in valve turning task.

less time it would take. So, the advantage of the framework in computation efficiency relies on the reduced number of the activated SCAFoIs, which would be the fact in most of normal tasks.

VI. CONCLUSIONS

In this paper, the concept of Self-Collision Avoidance Focus of Interest (SCAFoI) for humanoid robots was proposed to dynamically select the necessary link pairs to be checked online to accelerate the computation of a self-collision avoidance constraint. A predictor, designed by using Support Vector Machine combined with a Quadratic Lagrangian Interpolation function, was employed to estimate the status of any pair of kinematic chains of the humanoid robot. The prediction accuracy and the computational efficiency are verified by whole-body motions in simulation on the humanoid robot WALK-MAN. Future work will include the implementation of the method on the real robot and will also theoretically study the factors which are able to guarantee the accuracy and efficiency in any situations to make the whole predictor more robust and reliable.

REFERENCES

[1] J. Kuffner, K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue, "Self-collision detection and prevention for humanoid robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3. IEEE, 2002, pp. 2265–2270.

[2] A. El Khoury, F. Lamiraux, and M. Taix, "Optimal motion planning for humanoid robots," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3136–3141.

[3] K. Okada, M. Inaba, and H. Inoue, "Real-time and precise self collision detection system for humanoid robots," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 1060–1065.

[4] K. Okada and M. Inaba, "A hybrid approach to practical self collision detection system of humanoid robot," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 3952–3957.

[5] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick, "Real-time self collision avoidance for humanoids by means of nullspace criteria and task intervals," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. IEEE, 2006, pp. 575–580.

[6] H. Sugiura, M. Gienger, and H. Janssen, "Real-time collision avoidance with whole body motion control for humanoid robots," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 2053–2058.

[7] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar, "Real-time (self)-collision avoidance task on a hrp-2 humanoid robot," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 3200–3205.

[8] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*. IEEE, 2009, pp. 1–6.

[9] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, p. 0278364914521306, 2014.

[10] O. Kanoun, F. Lamiraux, P.-B. Wieber, F. Kanehiro, E. Yoshida, and J.-P. Laumond, "Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2939–2944.

[11] A. Dietrich, T. Wimböck, H. Taubig, A. Albu-Schäffer, and G. Hirzinger, "Extensions to reactive self-collision avoidance for torque and position controlled humanoids," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3455–3462.

[12] F. Kanehiro, N. Miyata, S. Kajita, K. Fujiwara, H. Hirukawa, Y. Nakamura, K. Yamane, I. Kohara, Y. Kawamura, and Y. Sanka, "Virtual humanoid robot platform to develop controllers of real humanoid robots without porting," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2001, pp. 1093–1099.

[13] A. Rocchi, E. M. Hoffman, D. G. Caldwell, and N. G. Tsagarakis, "Opensot: a whole-body control library for the compliant humanoid robot coman," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1093–1099.

[14] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *Robotics, IEEE Transactions on*, vol. 27, no. 4, pp. 785–792, 2011.

[15] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4, Mar 1987, pp. 1152–1159.

[16] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond, "A local collision avoidance method for non-strictly convex polyhedra," *Proceedings of robotics: science and systems IV*, 2008.

[17] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 3859–3866.

[18] K. Knese, "Realizing online (self-)collision avoidance based on inequality constraints with hierarchical inverse kinematics," Master's thesis, Technical University of Munich, Germany, July 2014.

[19] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 1, no. 19, pp. 18–19, 2012.

[20] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[21] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[22] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[23] C. J. Lin, C.-W. Hsu, and C.-C. Chang, "A practical guide to support vector classification," *National Taiwan U., www. csie. ntu. edu. tw/cjlin/papers/guide/guide. pdf*, 2003.