Learning Kinematics from Direct Self-Observation Using Nearest-Neighbor Methods

Hannes Schulz, Lionel Ott, Jürgen Sturm, and Wolfram Burgard

Abstract. Commonly, the inverse kinematic function of robotic manipulators is derived analytically from the robot model. However, there are cases in which a model is not a priori available. In this paper, we propose an approach that enables an autonomous robot to estimate the inverse kinematic function on-the-fly directly from self-observation and without a given kinematic model. The robot executes randomly sampled joint configurations and observes the resulting world positions. To approximate the inverse kinematic function, we propose to use instance-based learning techniques such as Nearest Neighbor and Linear Weighted Regression. After learning, the robot can take advantage of the learned model to build roadmaps for motion planning. A further advantage of our approach is that the environment can implicitly be represented by the sample configurations. We analyze properties of this approach and present results obtained from experiments on a real 6-DOF robot and from simulation. We show that our approach allows us to accurately control robots with unknown kinematic models of various complexity and joint types.

1 Introduction

Robotic manipulators typically require a Cartesian controller that maps the three-dimensional world space coordinates to joint configurations. The equations for the inverse kinematics can analytically be derived in advance when a robot model is available. Careful parameter calibration then ensures high accuracy in positioning tasks.

In the emerging field of home robotics, robots are likely to be assembled without surveillance of an engineer, meaning that no accurate construction model is available at design time. Further, home robots need to operate for extended periods of

Hannes Schulz · Lionel Ott · Jürgen Sturm · Wolfram Burgard
Department of Computer Science, University of Freiburg,
Georges-Köhler-Allee 79, 79110 Freiburg, Germany,
e-mail: {schulzha,ottli,sturm,burgard}@informatik.uni-freiburg.de

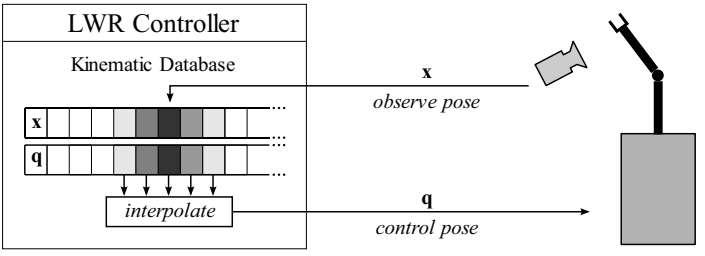


Fig. 1 Schematic overview of the proposed controller. Configurations are sent to the robot, the world-position is visually observed and recorded. Configurations for novel world positions are obtained by locally interpolating between known configurations.

time, during which they might be subject to damage, inaccurate repairs or other unforeseen modifications. This requires the development of new approaches to robust self-adaptation of the robot's internal kinematic models.

In this paper we introduce methods that can be used to approximate the inverse kinematic function without an a priori model. The robot visually observes the effects of its motor-commands on the position of its end-effector. The robot can then use these observations to estimate the kinematic function using nearest-neighbor methods (Figure 1). The remainder of this paper is organized as follows: Section 2 relates our paper to other approaches on kinematic learning. In Section 3 we introduce our approach on using nearest-neighbor methods for inverse kinematics estimation and planning. In Section 4, we analyze the performance and robustness in experiments on a real robot and in simulation.

2 Related Work

The problem of learning kinematics of robots has been investigated heavily in the past. In classical engineering, a typical solution starts out with a parameterized forward model, whose parameters are iteratively optimized in order to minimize the prediction error. However, only few approaches exist that do not rely on such a priori models of the robot.

Dearden and Demiris [2] present a method which estimates the parameters for the motion and observation model of a recursive Bayesian filter from visual observations. Whereas this approach has been applied successfully to a gripper with two separately controlled fingers, the sheer size of the parameter space prevents us from the direct application to highly complex robots. Sturm *et al.*[9] presented an alternative method that also uses Bayesian networks to estimate the robot model. In this work, the networks are learned by visually observing the individual parts of the robot while performing random movements. The authors estimate the global robot structure by learning the actuator models and finding the network topology that best explains the data in terms of data likelihood. The resulting model can then be used to predict the end-effector pose, so that gradient descent strategies can be applied

for solving inverse kinematics. Our method presented in this paper, being simpler in nature, requires only a single marker on the end-effector instead of a marker for every rigid body part.

In the approach presented by Cheah *et al.*[1], the robot learns, instead of the end-effector position, only the Jacobian, which describes the first derivative of the robot's kinematics and dynamics, using the visually observed world position of the end-effector, the joint velocities and configurations. The Jacobian can then be applied for calculating the inverse kinematics. The approach is evaluated on 2-DOF (degrees of freedom) robots in a 2D workspace only. In our experiments we demonstrate, that our approach also works in the full 3D space and also with up to 6-DOF.

In the area of visual servoing (see the work by Malis for an overview [8]) the robotic arm is typically controlled locally by observing the effect of motor commands on the end-effector in the image space. As visual servoing is a pure online-technique that relies on visual feedback, it cannot be used for (offline) planning. We could however apply visual servoing to improve the accuracy of our method after a plan has been generated.

To sum up, compared to previous approaches our method does not rely on prior knowledge of the robot model, it scales better even with an increasing number of DOF, and allows for global planning. Additionally, our approach is able to implicitly represent the environment and to handle self-collisions.

3 The Model-Free Approach

We collect data by sending a configuration request $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ to a robot with n joints and observing the resulting end-effector position $\mathbf{x} = (x_1, x_2, x_3)$ in 3D space. After m trials, this results in a kinematic database

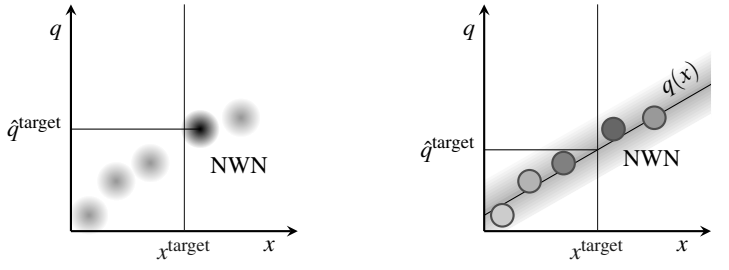
$$S = \left(\langle \mathbf{q}^1, \mathbf{x}^1 \rangle, \langle \mathbf{q}^2, \mathbf{x}^2 \rangle, \ldots, \langle \mathbf{q}^m, \mathbf{x}^m \rangle \right).$$

If a collision is encountered, the corresponding sample is rejected. We describe two online learning methods for estimating the inverse kinematics from the acquired data. This allows us to generate a valid configuration $\mathbf{q}$ for a given target position $\mathbf{x}$. We furthermore present an approach for planning based on the kinematic database.

3.1 Nearest Neighbor Method

Using a probabilistic formalization, we assume that the collected samples in S are drawn from a joint probability distribution $p(\mathbf{q}, \mathbf{x})$ defined by the unknown underlying mechanical model. Estimating a configuration $\hat{\mathbf{q}}^{\text{target}}$ that is supposed to yield $\mathbf{x}^{\text{target}}$ then corresponds to finding a configuration that maximizes the conditional probability

$$\mathbf{q}^{\text{target}} = \arg\max_{\mathbf{q}} p(\mathbf{q} | \mathbf{x}^{\text{target}}).$$



(a) Nearest Neighbor (b) Linear Weighted Regression

Fig. 2 Visualization of our approach for a 1-DOF robot. (a) Shaded areas: Collected samples, NWN: Nearest world-neighbor of x^{target}, Horizontal line: Joint position $\hat{q}^{\text{target}}$ associated with NWN. (b) Circles: Subset of collected samples which are close to NWN in configuration space, Shaded area: $p(q,x)$, with an assumed linear relationship between x and q. Evaluation of fitted linear function $q(x)$ at the desired world position x^{target} yields desired configuration $\hat{q}^{\text{target}}$. Lighter shading of samples indicates higher world space distance to x^{target} and thus less influence on the slope of $q(x)$.

As a first approximation, we search for the closest world-neighbor to the target location $\mathbf{x}^{\text{target}}$ in our database, and assign the associated joint value to $\hat{\mathbf{q}}^{\text{target}}$:

$$\hat{\mathbf{q}}^{\text{target}} = \underset{\mathbf{q}^i \in S}{\arg\min} \left\| \mathbf{x}^{\text{target}} - \mathbf{x}^i \right\|_2 \tag{1}$$

This idea is also visualized for a 1-DOF robot in Figure 2a.

There are two obvious disadvantages of this method: First, it is sensitive to the noise in the observations. Second, the accuracy of the method is directly related to the sample density in S. Interestingly, it does not rely on the dimensionality of $\mathbf{q}$ or the effective complexity of the mapping between configuration space and world space; in particular, the database lookups can be implemented very efficiently for example by using KD-trees [4]. Also, in the noise-free case, the nearest neighbor (NN) method will converge to the true function in the limit.

3.2 Linear Weighted Regression

To improve the NN method one can interpolate between the k-nearest neighbors in configuration space. However, this interpolation is not straightforward since the configurations resulting in a single world position are ambiguous, as a robot may have multiple ways to approach it. We solve this problem by interpolating only between configurations that are "close" in configuration space. We start by selecting the nearest neighbor $\langle \mathbf{q}^w, \mathbf{x}^w \rangle$ to $\mathbf{x}^{\text{target}}$ in world space and from there select the set of k nearest neighbors in configuration space:

$$\bar{S} = \underset{\bar{S}\subset S, |\bar{S}|=k}{\arg\min} \sum_{\langle \mathbf{q},\mathbf{x}\rangle \in S} \|\mathbf{q} - \mathbf{q}^w\| \tag{2}$$

This locally selected subset $\bar{S}$ of kinematic samples can now be used for interpolation, because all samples in $\bar{S}$ are neighbors in configuration space.

Note that rotational joints, for example, move the end-effector on a circular path. Even though it is possible to interpolate between the joint angles, the information how a joint affects the position of the end-effector is part of the model, which is not available. We assume that the actuators are locally linear, allowing us to fit a linear model to the samples in $\bar{S}$ such that for a 1-DOF robot whe have

$$\forall \langle \mathbf{q}, \mathbf{x}\rangle \in \bar{S} : q = w_3 x_3 + w_2 x_2 + w_1 x_1 + w_0, \tag{3}$$

where $\mathbf{w} = (w_3, w_2, w_1, w_0)$ is the weight vector defining a plane.

Since the data contains observation noise and the assumption of local linearity is only an approximation, the linear model will not fit perfectly. Therefore, we estimate $\widehat{\mathbf{w}}$ from the least-squares solution

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\arg\min} \sum_{s^i \in \bar{S}} \left(q^i - (w_3 x_3^i + w_2 x_2^i + w_1 x_1^i + w_0)\right)^2 = \underset{\mathbf{w}}{\arg\min} \|X\mathbf{w} - Q\|_2^2, \tag{4}$$

where $X \in \mathbb{R}^{N\times 4}$ with $N = |\bar{S}|$ is the number of joint neighbors used. $Q \in \mathbb{R}^{N\times 1}$ are the corresponding joint values and $\mathbf{w} \in \mathbb{R}^{4\times 1}$ are the weights we want to estimate. Using SVD [5], Equation (4) can be solved efficiently. To reduce the effects of non-linearity, we weight the error introduced by a sample and thus its influence on the slope of the function by the inverse of its squared world space distance to $\mathbf{x}^{\text{target}}$. Given the weights $\widehat{\mathbf{w}}$, we can determine the configuration $\hat{q}^{\text{target}}$ for a novel world position $\mathbf{x}^{\text{target}}$ as

$$\underset{q}{\arg\max}\, p(q|\mathbf{x}) \approx \hat{q}^{\text{target}} = \widehat{w}_3 \mathbf{x}_3^{\text{target}} + \widehat{w}_2 \mathbf{x}_2^{\text{target}} + \widehat{w}_1 \mathbf{x}_1^{\text{target}} + \widehat{w}_0$$

When this method is extended to multiple joints, the weight vector $\mathbf{w}$ needs to to be replaced by a weight matrix W. This would treat the individual joints as being independent during the maximization of $p(\mathbf{q}|\mathbf{x}^{\text{target}})$. Of course, this is not the case, choosing the value of one joint limits the possible values of subsequent joints. Actually, finding the configuration with highest likelihood of leading to a pose $\mathbf{x}^{\text{target}}$ involves finding the global maximum, i.e., computing $\arg\max_{q_1\ldots q_n} p(q_1,\ldots,q_n|\mathbf{x}^{\text{target}})$. We suggest to find this maximum greedily by factorizing this probability distribution according to the chain rule as

$$\begin{aligned}\mathbf{q}^{\text{target}} &= \underset{q_1\cdots q_n}{\arg\max}\, p(q_1,\ldots,q_n|\mathbf{x}^{\text{target}}) \\ &= \underset{q_1\cdots q_n}{\arg\max}\, p(q_1|\mathbf{x}^{\text{target}}) p(q_2|\mathbf{x}^{\text{target}}, q_1)\cdots p(q_n|\mathbf{x}^{\text{target}}, q_1,\ldots,q_{n-1})\end{aligned}$$

and then maximizing the factors one by one, extending the linear least-squares model from Equation (4) for each joint:

$$\hat{\mathbf{q}}^{\text{target}} = \begin{pmatrix} \arg\max_{q_1} p(q_1|\mathbf{x}^{\text{target}}) \\ \arg\max_{q_2} p(q_2|\mathbf{x}^{\text{target}}, q_1) \\ \dots \\ \arg\max_{q_n} p(q_n|\mathbf{x}^{\text{target}}, q_1, \dots, q_{n-1}) \end{pmatrix} \tag{5}$$

In this way, the linearization of the kinematic model for each joint takes into account all previous joints. Note that the factorization in Equation (5) is valid even for an arbitrary order of the joints.

3.3 Path Planning

With the methods described above, we can infer joint configurations that move the robot arm to previously un-encountered target positions. The data stored in the kinematic database S implicitly encodes information about the robot itself and the surroundings, as it contains only samples from accessible, obstacle-free regions both in configuration space and world space. In the following, we present a planning method based on probabilistic roadmaps [7]. Planning is performed directly in the kinematic database, by using a cost metric combining world space and configuration space metrics.

From our kinematic database, we create a undirected graph with the nodes corresponding to the samples in the kinematic database. Two nodes $s^i, s^j \in S$ are assumed to be connected in the graph, when they are close in world space, i.e., $\left\|\mathbf{x}^i - \mathbf{x}^j\right\|_2 < d_{max}$.

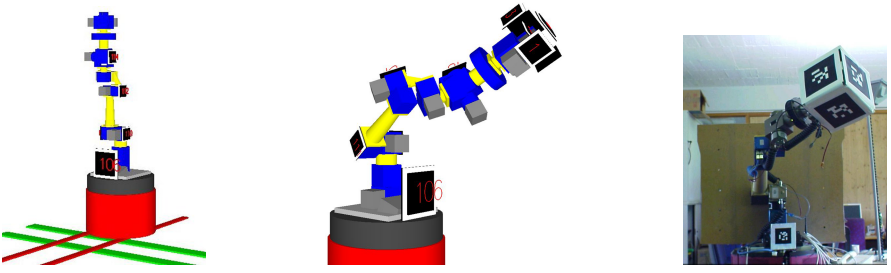
The planning problem then reduces to finding the shortest path $\langle \pi(0), \dots, \pi(n) \rangle$ through the graph [6]. To avoid long initial paths and to get as close as possible to the target, we add a starting node at the current configuration $\mathbf{q}^{\text{current}}$ of the robot, and an ending node close to $\mathbf{x}^{\text{target}}$:

$$s^{\pi(0)} = \arg\min_{s^i \in S} \left\|\mathbf{q}^i - \mathbf{q}^{\text{current}}\right\|_2, \qquad s^{\pi(n)} = \arg\min_{s^i \in S} \left\|\mathbf{x}^i - \mathbf{x}^{\text{target}}\right\|_2.$$

We find the shortest path using the A* algorithm. Our cost function takes into account differences in world space and in configuration space:

$$\sum_{i=0}^{n-1} \left(\alpha \left\|\mathbf{x}^{\pi(i)} - \mathbf{x}^{\pi(i+1)}\right\|_2 + \beta \left\|\mathbf{q}^{\pi(i)} - \mathbf{q}^{\pi(i+1)}\right\|_2^2 \right),$$

where $\alpha = 1/d_{max}$, and β can be set likewise to 1 divided by the maximum joint distance. The squared joint-distance causes the planner to prefer many small steps through the roadmap instead of one large one. This increases the resolution of the controlled path and simultaneously penalizes shortcuts through potential obstacles that appear as unpopulated regions in the roadmap.



(a) (b) (c)

Fig. 3 Robots used for testing. (a) Robots with rotational and prismatic ("rails") joints, (b) Simulated 6-DOF robot, and (c) Real-world 6-DOF counterpart.

4 Results

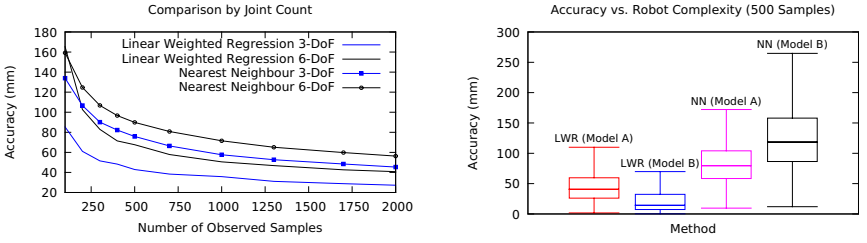
We first investigate the performance of both methods for a varying number of joints. Then, the influence of different joint types is evaluated. Finally we applied the methods to both a simulated version of our 6-DOF robot and the real robot composed of rotatory Schunk PowerCube modules.

4.1 Positioning Accuracy

We compared the accuracy of the a simulated robot with 3 versus 6 active joints. The results are displayed in Figure 4a. For an arm length of 100 cm operating in a hemisphere. The Nearest Neighbor method achieves an accuracy of 9.0 cm (3-DOF) and 10.7 cm (6-DOF) with 300 samples, while Linear Weighted Regression obtains an accuracy of 5.2 cm and 8.3 cm, respectively. Note that the decrease in accuracy of Nearest Neighbor can only be attributed to the increasing workspace volume. Linear Weighted Regression, however, additionally suffers from the increasing dimensionality in configuration space, nevertheless it outperforms Nearest Neighbor in both cases.

Another property we investigated was the ability to cope with different types of joints. For this purpose, we constructed two robots. The first robot (similar to Figure 3b) has five rotational joints in the arm. For the second robot (Figure 3a) we replaced three rotational joints with prismatic joints.

Analyzing the accuracy of both methods, we find that Linear Weighted Regression outperforms Nearest Neighbor on both robots (Figure 4b). As expected, Nearest Neighbor performance drops when the workspace volume increases. The accuracy of Linear Weighted Regression increases since prismatic joints conform to the linearity assumption and are thus better suited for interpolation.



(a) (b)

Fig. 4 (a) Accuracy of our methods on 3 and 6-DOF robots with arm length of 1 m. (b) Prismatic joints (conform to linearity assumption) increase accuracy of LWR, although workspace volume increases. Contrarily, Nearest Neighbor (NN) performance decreases. Model A contains only rotational joints, while Model B contains prismatic joints as well.

4.2 Real Robot Evaluation

The following experiments aim at evaluating the general performance of the approaches with a complex robot, with six rotational joints. Our robot "Zora", depicted in Figure 3c, consists of a B21 base and a robotic arm composed of Schunk Powercubes on top with a total length of 100 cm.

In order to observe the 3D position of the end-effector, we placed a 20cm $\times$ 20cm $\times$ 20cm AR-Toolkit [3] marker cube on it (Figure 3c). We inferred the end-effector position from the 6D pose of the observed markers, with an average observation accuracy of 7.4 cm. Alternatively, motion capturing systems or a robot-mounted camera observing its position with respect to the environment could be used.

The database S was filled with 250 samples by sending a joint-configuration to the robot and observing the resulting world position of the end-effector. The recording took approximately two hours. These joint-configurations were generated by choosing a random value for each joint.

We evaluated our methods in simulation by determining the accuracy of our simulated robot with respect to the samples gathered on the real robot for noise-free ground truth comparison.

With Nearest Neighbor, we achieved a positioning accuracy of 8.2 cm ($\sigma =$ 6.0 cm), whereas we determined the accuracy of Linear Weighted Regression to be 5.7 cm ($\sigma = 5.8$ cm). The reported values were obtained on an arm with a length of 100 cm.

4.3 Path Planning

We qualitatively tested our planning system described in Section 3.3. To this end, we constructed a simple test environment containing a 2-DOF planar robot and two obstacles. The robot had to move to random positions in its workspace. We then

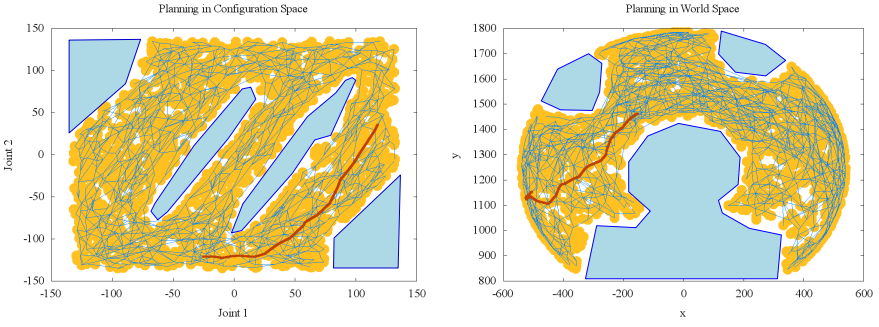


Fig. 5 Planning in configuration space (left) and world space (right). Our method is able to find a path short in both spaces. Blue regions indicate areas inaccessible to the robot (self-collisions, obstacles). Yellow: recorded samples. Blue lines: roadmap of Planner. Red lines: sample trajectory.

analyzed the resulting trajectories. The 2D visualization of world space and the corresponding configuration space with a sample trajectory is displayed in Figure 5.

Although the trajectory does not run close to obstacles in world space, the corresponding configuration space trajectory is avoiding such an obstacle by following a curved path. Moreover, the path is curved more than necessary in configuration space to allow an almost straight path in world space. It therefore seems that our approach can help to circumvent problems that would occur if planning was only performed in configuration space.

5 Conclusions

In this paper, we presented a general and efficient approach to learn the inverse kinematic function for robots with unknown mechanical design. In contrast to classical approaches, which require the knowlege of the model to determine the kinematic function, we collect data consisting of visually observed end-effector positions and the corresponding robot configurations. We then locally approximate the inverse kinematic function. Further, we showed how a planner can construct a probabilistic roadmap directly from the kinematic database. In experiments carried out with simulated and real robots with varying number of DOFs and joint types, we obtained accurate positioning results.

In future work, we would like to extend the world space representation to full 6D, i.e., control both the translational and rotational components. For the planning, we want to additionally include those kinematic samples that were previously rejected due to (self-)collisions. Thanks to the simplicity and efficiency of our approach, we plan to realize a real-time implementation on an embedded device in near future.

References

1. Cheah, C.C., Liu, C., Slotine, J.J.E.: Adaptive jacobian tracking control of robots based on visual task-space information. In: Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 3498–3503 (2005)
2. Dearden, A., Demiris, Y.: Learning Forward Models for Robots. In: International Joint Conference on Artificial Intelligence, vol. 19, p. 1440. Lawrence Erlbaum Associates LTD, Mahwah (2005)
3. Fiala, M.: ARTag, a Fiducial Marker System Using Digital Techniques. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005, vol. 2 (2005)
4. Freidman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw. 3(3), 209–226 (1977)
5. Golub, G., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix. Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis 2(2), 205–224 (1965)
6. Latombe, J.: Robot Motion Planning. Kluwer Academic Publishers, Norwell (1991)
7. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
8. Malis, E.: Survey of vision-based robot control. ENSIETA European Naval Ship Design Short Course, Brest, France (2002)
9. Sturm, J., Plagemann, C., Burgard, W.: Unsupervised body scheme learning through self-perception. In: IEEE International Conference on Robotics and Automation, 2008. ICRA 2008, pp. 3328–3333 (2008)