Estimating Pose Statistics for Robotic Part Feeders

Brian Mirtich* UCB

Yan Zhuang† UCB

Ken Goldberg‡ UCB

John Craig§ Adept Technology, Inc.

Rob Zanutta Adept Technology, Inc.

Brian Carlisle Adept Technology, Inc.

John Canny¶ UCB

Abstract

In automated assembly lines, part feeders often impose a bottleneck that restricts throughput. To facilitate the design of parts and assembly lines, we'd like to estimate feedrates based on CAD models of parts. A previous paper [8] described how to predict throughput for a vision-based robotic part feeder given the distribution of part poses when parts are randomly dropped on a conveyor belt. Estimating this distribution is also useful for the design of traditional feeders such as vibratory bowls.

In this paper, we describe three algorithms for estimating pose distributions. We review the quasi-static estimate reported in [21] and introduce a refinement that takes into account some measure of dynamic stability. The perturbed quasi-static *estimate can be computed very rapidly and is more accurate than the quasi-static. Still more accurate are estimates based on Monte Carlo simulation using* Impulse *[12; 13], although the latter comes at the penalty of increased computation time. We compare estimates from all three algorithms with physical experiments. We believe this is the first paper to systematically compare estimators with experiments using real industrial parts.*

1 Introduction

In contrast to fixed assembly lines, flexible assembly lines can be rapidly reconfigured to handle new parts. This can dramatically reduce the time needed to bring new products to market and permit the cost of assembly lines to be amortized over multiple products. *Part feeders*, which singulate and orient the parts prior to packing and insertion, are critical components of the assembly line and one of the biggest obstacles to flexible assembly. Currently, the design of parts feeders is a black art that is responsible for up to 30% of the cost and 50% of workcell failures [14; 2].

Carlisle *et. al.* [3] proposed a flexible part feeding system that combines machine vision with a high-speed robot arm. In contrast to custom-designed hardware such as the bowl feeder, only software is changed when a new part is to be fed. The idea is that a collection of like parts are randomly scattered on a flat worktable where they are subject to the force of gravity. An overhead vision system determines the pose (position and orientation) of each part. The robot arm then picks up each part and moves it into a desired final pose as illustrated in Figure 1. To facilitate rapid programming, we are developing a simulator that models feeder mechanics. Goldberg *et. al.* [8] outline how feeder throughput can be estimated based on estimates of pose statistics, conveyor speed, and arm cycle time. In this paper we describe several algorithms for estimating pose statistics.

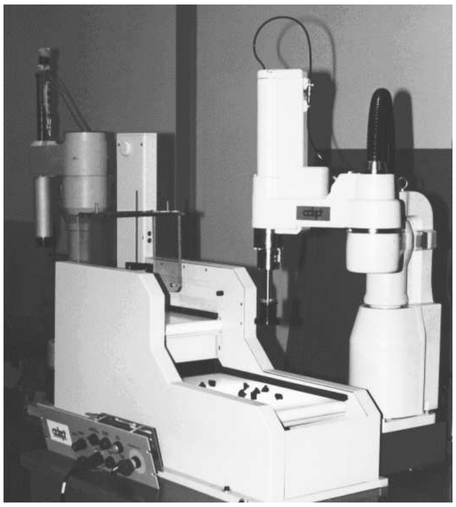

Figure 1: A flexible parts feeder using machine vision, a high-speed robot arm, and pivoting gripper. This illustration shows the system feeding car-stereo pushbuttons.

An excellent introduction to mechanical parts feeders can be found in [2], which describes vibratory bowl feeders in detail as well as non-vibratory feeders such as the magnetic and revolving hook feeders. The authors note that the feedrate for a parts feeder is related to the probability that parts are aligned correctly when they encounter a mechanical filter and give a quasi-static estimate for the orientation of rectangular and cylindrical parts dropped at random.

Similarly, the throughput analysis in [8] requires knowl-

*Computer Science Department, University of California, Berkeley, CA 94720–1776, (510) 642–8149, mirtich@cs.berkeley.edu.

Mirtich is supported in part by NSF grant #FD93-19412.

†Computer Science Department, University of California, Berkeley, CA 94720–1776, yzhuang@cs.berkeley.edu.

‡Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720, goldberg@iris.usc.edu.

Goldberg is supported by Adept Technology, Inc. and a grant from the State of California's Office of Competitive Technology.

§jjc@silma.com

¶Computer Science Department, University of California, Berkeley, CA 94720-1776, (510) 642-9955, jfc@cs.berkeley.edu.

Canny is supported in part by NSF grant #FD93-19412.

edge about the distribution of poses, particularly orientation, when the parts are scattered on the flat worktable. [21] generalized and improved on the Boothroyd estimate by treating the convex hull of any polyhedral part and propagating probability from unstable faces. The resulting estimate is a good first approximation to experimental distributions but do not take into account effects such as bouncing, vibrations, collisions, and friction. An ad-hoc estimator based on face area and height of the cg was reported in [15] but only tested with regularly-shaped parts.

In this paper, we examine the problem of estimating pose distribution in greater detail. We first summarize the quasi-static algorithm from [21]. Next we develop a refinement to this algorithm which incorporates a simple model of dynamic stability. Finally, we discuss results from full dynamic simulation of dropped parts to obtain stochastic data. We discuss impulse-based simulation, a paradigm that makes such simulation possible in a reasonable amount of time, and present its model for frictional collisions. All three algorithms for estimating pose distributions are compared to real data for a variety of test parts; we find that dynamic simulation provides the most accurate results, but requires significantly more computation time. The central problem is:

Estimating Pose Statistics (EPS) *For a rigid polyhedral part X with known center of mass and inertia tensor, denote the n faces of its convex hull H by $F_1, \ldots, F_n$. Assuming X is repeatedly dropped onto a flat surface from some known distribution of initial poses, compute the values $p_1, \ldots, p_n$, where p_i is the probability that X reaches a final resting state with F_i down (against the surface).*

In all three algorithms, we assume that the workspace is flat and much larger than X, and that X does not collide with other parts.

2 Quasi-static algorithm

Our first approach to the EPS problem was proposed by Wiegley, *et. al.* The algorithm, presented in detail in [21], is summarized here. Motion of the part is assumed to be quasi-static; inertia and velocity are neglected.

After computing the part's convex hull H, the idea is to project the facets of H onto a unit sphere centered at the center of mass c. If F_π is the projection of face F, the ratio of the area of F_π to the total surface area of the sphere gives the probability that the part will land on face F under quasi-static conditions, if its initial pose is uniformly distributed over $SO(3)$.

Assuming triangular faces, the ratio in question is given by

$$A = \frac{\beta_0 + \beta_1 + \beta_2 - \pi}{4\pi} \tag{1}$$

where the β_i are the interior angles of F_π (see Figure 2).

The β_i are computed as follows. Let $d_{c0} = dist(c, v_0)$, $d_{c1} = dist(c, v_1)$, and $d_{01} = dist(v_0, v_1)$. Using standard notation for triangles, let δ_2 be the arc that results from projecting the line from v_0 to v_1 onto the sphere (note that arcs

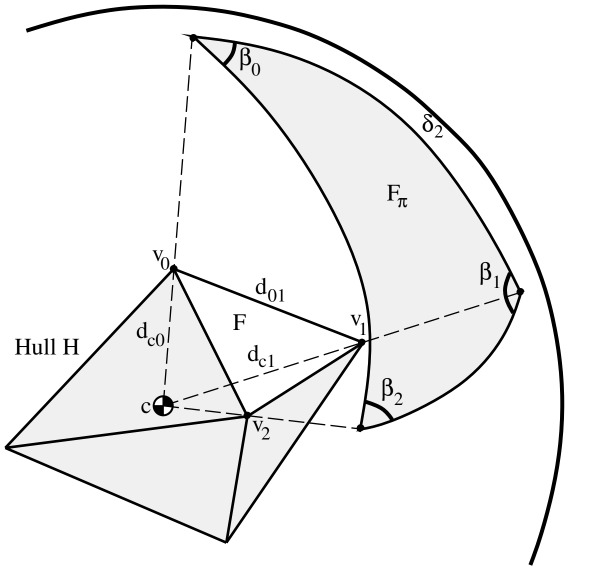

Figure 2: *Computing initial probabilities for each face.*

are measured by the angle subtended at the center of the sphere). One can solve for δ_2 using the law of cosines,

$$d_{01}^2 = d_{c0}^2 + d_{c1}^2 - 2d_{c0}d_{c2}\cos\delta_2, \tag{2}$$

and δ_0 and δ_1 are found similarly. Given all the δ_i, β_2 can be found using the spherical law of cosines,

$$\cos\delta_2 = \cos\delta_0\cos\delta_1 + \sin\delta_0\sin\delta_1\cos\beta_2, \tag{3}$$

and analogous computations give β_0 and β_1.

This procedure results in an initial estimate of each p_i. To treat faces of H that are statically unstable, we project the center of mass onto the plane of each face F_i. If the projected point lies outside face F_i, gravity will cause the part to topple over to adjacent face F_j. In this case p_i is added to p_j, and p_i is set to zero.

To facilitate this propagation, we define the *quasi-static graph* (**QSG**) to be a directed graph in which each node corresponds to one facet of the convex hull H. The QSG has a directed link from node i to node j if and only if facet F_i topples to facet F_j and they share one common edge. Clearly, QSG is acyclic. We propagate probability along the QSG using a breadth-first traverse. The algorithm is $O(n^2)$ in the number of polyhedral edges [21].

3 Perturbed-Quasi-Static (PQS) Estimate

Since the quasi-static analysis does not model dynamic disturbances, it often overestimates the probability of landing on a facet that is stable but easily dislodged by small vibration. In this section we describe a modification to the quasi-static estimate that considers a "perturbation region" around each edge of a stable face. Consider two facets of the part's convex hull, F_i and F_j, and the bounding edge e between them. Let g be the downward vector from the part's center of mass. In the quasi-static estimate, we assume that if g intersects F_i when in contact, the part will remain on facet F_i. However, dynamic energy may cause the part to rotate across edge e when g points inside facet F_i but close to edge e (Figure 3). The spherical projection of the perturbation region that falls inside of the face yields a heuristic estimate of how likely the part is to topple from F_i to F_j across their shared edge. We call Δp_{ij} the "perturbation probability" for each pair of adjacent faces (each edge). We use these to compute a "perturbed quasi-static" (PQS) estimate that can be computed in time $O(n^2)$ and agrees better with experiments.

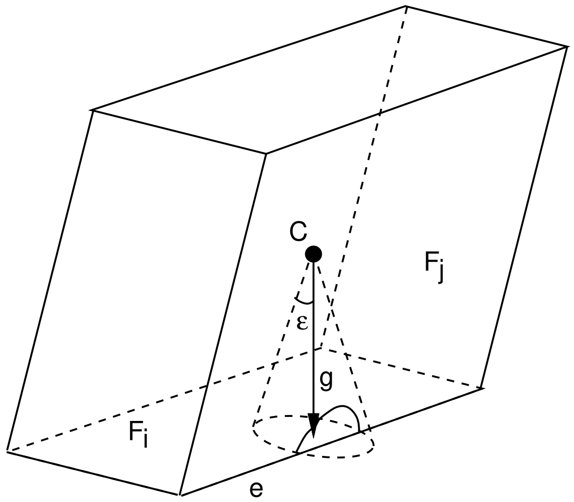

Figure 3: Under the perturbed-quasi-static model, the part remains on facet F_i, since g intersects F_i. However dynamic effects make it likely that the part will topple onto facet F_j. To model this, we consider a "perturbation region" around the common edge using a cone of disturbance vectors.

We consider perturbations to the gravity vector that form a right cone of half-angle ϵ with apex at the part's center of mass. The value of ϵ depends on how far dynamic forces can tilt the gravity vector; we used a value of 20° in the tables. If we sweep the gravity vector g along the part edge e, the perturbation cone sweeps out a perturbation region around the edge.

To compute the perturbation probability, we consider the triangle formed by edge e and two edges from its endpoints to the projected center of mass on facet F_i. Call this T. When we project T onto the unit sphere, we denote the arc corresponding to e with a. If we translate the plane defined by the center of mass c and the arc a until it intersects the sphere with a new arc a' such that the spherical distance between a' and a is ϵ, the spherical region Σ between a' and a is the spherical projection of the perturbation region (Figure 4(c)).

If we denote any point on the unit sphere by a vector $r = (x, y, z)$, which is parameterized by:

$$\left\{ \begin{array}{l} x = \sin\varphi\cos\theta \\ y = \sin\varphi\sin\theta \\ z = \cos\varphi \end{array} \right. \tag{4}$$

then the area of Σ is:

$$\sigma = \int\int_{\Omega} ||N(\varphi,\theta)|| d\varphi d\theta, \tag{5}$$

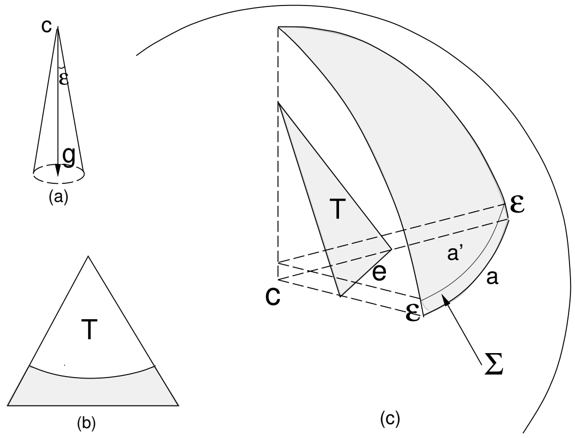

Figure 4: (a) Perturbations to the gravity vector form a cone. (b) Perturbation region for one part edge. (c) Perturbation probability for each edge, using the PQS model.

where Ω is the corresponding region in terms of φ and θ, and $N(\varphi,\theta)$ is the fundamental vector product of the surface, $r'_\varphi \times r'_\theta$. As we cannot solve this integral in closed form, we approximate the projected perturbation region by the area of a rectangle of length $|a|$ and width ϵ.

We transfer perturbation probability between adjacent facets and then propagate down the QSG as in the quasi-static estimate. We transfer Δp_{ij} from facet F_i to adjacent facet F_j if F_j is unstable or if F_i has a lower initial probability under the quasi-static estimate. The first condition insures that the perturbation probability will wind up at a stable facet after propagating through the QSG. Both conditions reflect the intuition that parts will tend to roll toward more stable states. The only extra computation is finding the probabilities of $O(n^2)$ perturbation regions and transferring the perturbation probabilities. Therefore the PQS estimate has the same time complexity as the quasi-static estimate: $O(n^2)$.

The PQS data in the tables are obtained by setting $\epsilon = 20°$. The computation time on SPARC20 is less than 1 second for all 4 parts, among which about 90% of the computation time is on the construction of the convex hull. It is important to keep in mind that the PQS is a heuristic estimate. We do not make any claims that this captures the intricate physics of dynamic collisions. A better heuristic may be possible by more sophisticated propagation of perturbation probabilities which we are now exploring. A full treatment of dynamic effects, at the price of increased computation, is described in the next section.

4 Dynamic simulation

To obtain more accurate pose distribution predictions, one could perform full dynamic simulations of the dropped part over many trials. This seems prohibitive for two reasons. First, the interaction between the part and environment is very collision intensive, and it is notoriously difficult to model the dynamics of collisions with friction [11]. Second, dynamic simulation is much slower than the previous described prediction algorithms, and so obtaining a statistically significant number of trials may take too long.

Mirtich and Canny have studied *impulse-based* simulation, a paradigm for dynamic simulation that addresses these problems. The method handles frictional collisions in a natural way, and for general 3D rigid body simulation, the simulator *Impulse* has the fastest execution times reported in the literature [13].

4.1 Computing frictional collisions

Details about *Impulse*, and a comparison of constraint- and impulse-based simulation are in [12; 13]. In the latter paradigm, *all* interactions between simulated bodies are affected through frictional collisions, thus a good collision model is crucial to physical accuracy. Our model is similar to that of Routh [17], although we derive equations which are more amenable to numerical integration. Keller also gives an excellent treatment [9], and Bhatt and Koechling give a classification of frictional collisions, based on the flow patterns of tangential contact velocity [1]. Finally, Wang and Mason have studied two-dimensional impact dynamics for robotic applications, based on Routh's approach [20].

Consider two rigid bodies coming into contact as shown in Figure 5. Each body i has a known mass m_i, inertia tensor

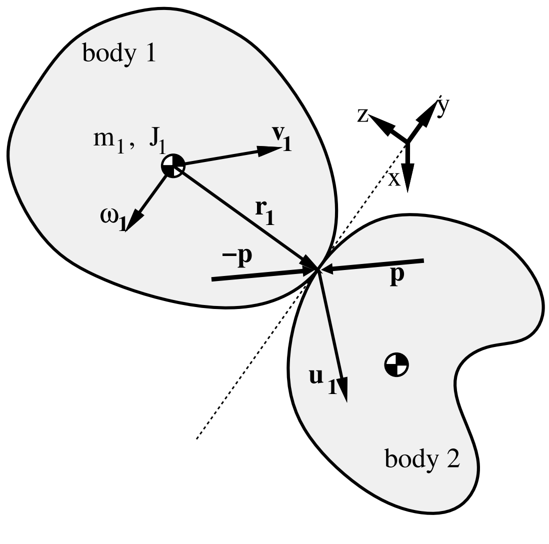

Figure 5: *A collision between two rigid bodies.*

$\mathbf{J}_i$, linear center of mass velocity $\mathbf{v}_i$, and an angular velocity $\boldsymbol{\omega}_i$. If $\mathbf{r}_i$ is the offset vector of the contact point relative to body i's center of mass, then the absolute velocity $\mathbf{u}_i$ of the contact point on body i is given by

$$\mathbf{u}_i = \mathbf{v}_i + \boldsymbol{\omega}_i \times \mathbf{r}_i \tag{6}$$

and the relative contact velocity $\mathbf{u}$ at the contact point is given by

$$\mathbf{u} = \mathbf{u}_1 - \mathbf{u}_2 \tag{7}$$

Under the coordinate system in Figure 5, the objects are colliding if if $\mathbf{u}$ has negative z (i.e. normal) component. In this case, a pair of collision impulses ($\mathbf{p}$ and $-\mathbf{p}$) must be applied to prevent interpenetration; the goal is to compute $\mathbf{p}$. We assume: infinitesimal collision time, the Coulomb friction model, and Poisson's hypothesis for restitution.

Infinitesimal collision time implies the positions of the two bodies may be treated as constant during the collision. Since $\mathbf{p}$ is an impulsive force, the velocities of the bodies change during the course of the collision. Because the frictional forces depend on the relative sliding velocity, the velocity profile *during* the collision must be analyzed.

Let γ denote a collision parameter which starts at 0 and increases monotonically during the collision. All body velocities as well as the relative velocity at the contact point are functions of γ. Let $\mathbf{p}(\gamma)$ be the total impulse imparted up to point γ in the collision. From basic physics,

$$\Delta\mathbf{v}_1(\gamma) = \frac{1}{m_1}\mathbf{p}(\gamma) \tag{8}$$

$$\Delta\boldsymbol{\omega}_1(\gamma) = \mathbf{J}_1^{-1}\left[\mathbf{r}_1 \times \mathbf{p}(\gamma)\right]. \tag{9}$$

Applying (6) gives

$$\Delta\mathbf{u}_1 = \left[\frac{1}{m_1}\mathbf{I} - \tilde{r}_1\mathbf{J}_1^{-1}\tilde{r}_1\right]\mathbf{p}(\gamma) \tag{10}$$

where $\mathbf{I}$ is the 3×3 identity matrix and $\tilde{r}_1$ is the canonical skew-symmetric matrix corresponding to $\mathbf{r}_1$. Computing $\Delta\mathbf{u}_2$ analogously ($-\mathbf{p}$ is used instead of $\mathbf{p}$), and applying (7) gives

$$\Delta\mathbf{u} = \underbrace{\left[\left(\frac{1}{m_1} + \frac{1}{m_2}\right)\mathbf{I} - \tilde{r}_1\mathbf{J}_1^{-1}\tilde{r}_1 - \tilde{r}_2\mathbf{J}_2^{-1}\tilde{r}_2\right]}_{\mathbf{K}}\mathbf{p}(\gamma) \tag{11}$$

The 3×3 matrix $\mathbf{K}$ is symmetric. More importantly, the infinitesimal collision time assumption implies $\mathbf{r}_i$ and $\mathbf{J}_i$ are constant during a collision, hence $\mathbf{K}$ is also constant. We can differentiate (11) with respect to γ, obtaining

$$\mathbf{u}' = \mathbf{K}\mathbf{p}' \tag{12}$$

4.1.1 Sliding mode

While the tangential component of $\mathbf{u}$ is nonzero, the bodies are sliding relative to each other, and $\mathbf{p}'$ is completely constrained. Let $\theta(\gamma)$ be the relative direction of sliding during the collision, that is $\theta = \arg(u_x + iu_y)$. Also choose γ to be p_z, the accumulated normal component of impulse. Under Coulomb friction, one finds that

$$\mathbf{p}' = \begin{bmatrix} -\mu\cos\theta \\ -\mu\sin\theta \\ 1 \end{bmatrix}. \tag{13}$$

Expressing the right hand side of (13) in terms of $\mathbf{u}$ and substituting into (12) gives

$$\begin{bmatrix} u_x' \\ u_y' \\ u_z' \end{bmatrix} = \mathbf{K}\begin{bmatrix} -\mu\frac{u_x}{\sqrt{u_x^2+u_y^2}} \\ -\mu\frac{u_y}{\sqrt{u_x^2+u_y^2}} \\ 1 \end{bmatrix}. \tag{14}$$

This nonlinear differential equation for $\mathbf{u}$ is valid as long as the bodies are sliding relative to each other. By integrating the equation with respect to the collision parameter γ (i.e. p_z), one can track $\mathbf{u}$ during the course of the collision. Projections of the trajectories into the u_x-u_y plane are shown in Figure 6 for a particular $\mathbf{K}$.

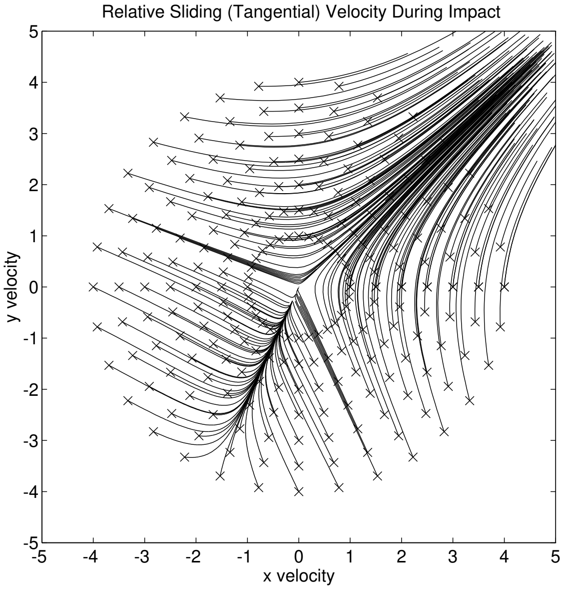

Figure 6: *Trajectories of the the tangential components of the system (14) for a particular* **K**. *The crosses indicate different initial sliding velocities.*

The basic impulse calculation algorithm proceeds as follows. After computing the initial **u** and verifying that u_z is negative, **u** is numerically integrated using (14) (p_z is the independent variable). During integration, u_z increases, reaching zero at the point of maximum compression. At this point, p_z is the normal impulse applied during compression, and multiplying it by $(1 + e)$ gives its terminating value, by Poisson's hypothesis for restitution. The integration continues to the terminating value, and **p** is recovered by inverting (11).

4.1.2 Sticking mode

Sticking occurs if the relative tangential velocity ever vanishes during integration of (14). In this case, Coulomb friction requires that the frictional force lie within the friction cone, although its direction is not specified. When sticking is detected, the system first checks whether it is a stable sticking condition by setting $\mathbf{u} = (0, 0, \lambda)^T$ in (12), and solving for $\mathbf{p}'$. One can choose λ such that $\mathbf{p}'$ is of the form $\mathbf{p}' = (\alpha, \beta, 1)^T$. If

$$\alpha^2 + \beta^2 \leq \mu^2, \tag{15}$$

a frictional force lying within the friction cone can maintain sticking, and so $u_x = u_y = 0$ and $\mathbf{p}' = (\alpha, \beta, 1)^T$ for the duration of the collision.

If $\alpha^2 + \beta^2 > \mu^2$, the friction is not sufficient to maintain sticking, and sliding immediately resumes in a direction θ_e of the ray emanating from the origin in the tangential velocity plot (In Figure 6, $\theta_e \approx 45°$). This ray always exists and is unique in cases of instable sticking.

4.2 Additional dynamic considerations

For dynamic simulation, many parameters not used quasi-static and perturbed quasi-static algorithms are important.

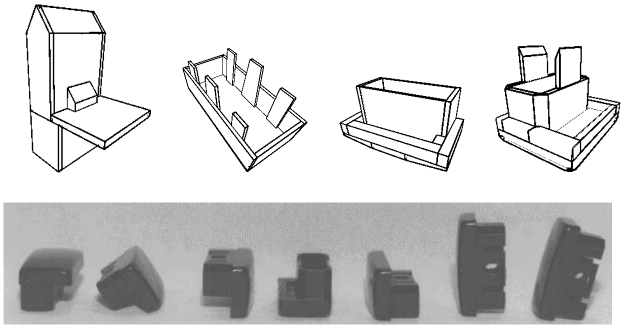

Figure 7: *Top: CAD models of the four parts used in the experiments. From left to right: insulator cap, large white, rectangular black, and square black stereo buttons. Bottom: Photographs of the rectangular black stereo button in its seven stable states.*

The coefficients of friction and restitution were both estimated to be 0.3.[1] The feeder configuration is also important. Adept's flexible feeder system dumps part from an upper belt onto a lower belt, where the parts are examined by the vision system, and then picked and placed by one or more manipulators. For dynamic simulation, the height of the drop was estimated at 12.0 cm. The horizontal velocity of the parts as they leave the upper belt was estimated at 5.0 cm/s.

The initial orientation of the parts poses a problem since the parts are in a stable resting state on the upper belt, before being dropped onto the lower one. Thus, the initial distribution of orientation is not uniform, but similar to the final (initially unknown) distribution. To circumvent this problem, the initial orientations for the first 20 drops are chosen randomly, assuming a uniform distribution over $SO(3)$. This bootstraps the process with a preliminary final pose distribution. For all remaining drops, the initial (upper belt) poses are chosen from the current distribution of final (lower belt) poses, so that the results of the drop tests are continually fed back to determine initial conditions. A slight perturbation (a rotation of up to 1.5 degrees about a randomly chosen axis) is also applied to the initial pose to introduce noise into the system due to belt vibration.

5 Experimental results

All of the algorithms described in this paper were applied to four test parts. The test parts were all small, plastic, rigid pieces, of the type typically used in automated assembly (Figure 7). Part #1 is an insulator cap purchased at a local hardware store. Parts #2, #3, and #4 are pushbuttons designed for a commercial car stereo system. Geometric models of each part were constructed by measuring the parts with a ruler. Centers of mass and moments of inertia for the parts were computed automatically by *Impulse*. As a control, physical experiments were performed by repeatedly running several samples of each test part through Adept's flexible feeder system (Figure 1); the final resting poses were recorded by

[1] Recently, Issa Nessnas of Adept has experimentally measured these coefficients; we plan to perform more tests using his numbers.

a human observer. Tables 1 through 4 show the results. All quantities in the tables are percentages.

The error percentages included in the tables indicate the overall performance of each algorithm for each sample part. They are computed as the average deviation of the algorithm's predictions from the physical test percentage, weighted by the frequency with which that state actually occurs. Let $p_1, \ldots, p_n$ represent the probability of each of n states, as measured in the physical test. Let $a_1, \ldots, a_n$ represent the corresponding probabilities computed by one of the algorithms. The error percentage for that algorithm is given by

$$e = 100 \sum_{i=1}^{n} p_i \, |a_i - p_i|. \tag{16}$$

Pose	Quasi-Static	P'turbed Q–S	Dynamic Sim.	Physical Tests[a]
1	30.5	46.5	41.9	46.0
2	37.3	30.2	26.2	27.1
3	19.6	19.8	28.3	19.7
4	8.3	3.5	3.0	5.0
5	4.2	0.0	0.8	2.2
error	10.1	1.2	4.0	–

[a] 1036 trials

Table 1: Orange insulator cap data.

Pose	Quasi-Static	P'turbed Q–S	Dynamic Sim.	Physical Tests[a]
1	34.5	48.8	71.7	75.8
2	39.9	30.6	20.9	13.8
3	19.2	20.5	7.4	10.5
4	6.3	0.0	0.1	0.0
error	35.8	23.8	4.4	–

[a] 545 trials

Table 2: White stereo button data.

Pose	Quasi-Static	P'turbed Q–S	Dynamic Sim.	Physical Tests[a]
1	36.2	47.3	54.1	56.0
2	16.0	25.5	24.1	24.5
3	17.4	17.0	14.0	13.6
4	8.1	1.2	1.4	4.4
5	10.6	4.5	5.3	1.4
6	7.5	4.4	1.0	0.3
7	4.3	0.0	0.3	0.0
error	14.0	5.8	1.4	–

[a] 1099 trials

Table 3: Rectangular black stereo button data.

5.1 Discussion

The quasi-static and perturbed quasi-static algorithms are extremely fast, requiring less than a second of computation time for parts with 50 facets. Dynamic simulation is slower; for each part, 2000 drops were simulated, taking approximately two hours per part. The data presented in Tables 1 through 4 bring out several interesting points concerning the accuracy of the algorithms' predictions.

Pose	Quasi-Static	P'turbed Q–S	Dynamic Sim.	Physical Tests[a]
1	35.7	46.6	68.4	62.2
2	17.5	15.5	16.6	15.2
3	12.1	17.0	6.1	11.0
4	7.2	8.6	6.0	4.7
5	3.9	1.6	2.7	3.1
6	5.6	1.5	0.0	2.8
7	3.8	3.9	0.3	0.5
8	4.2	1.7	0.0	0.0
9	3.0	2.3	0.0	0.0
10	2.6	0.7	0.0	0.0
11	2.2	0.0	0.0	0.0
12	2.1	0.5	0.0	0.0
error	17.2	10.7	4.8	–

[a] 915 trials

Table 4: Square black stereo button data.

The perturbed quasi-static algorithm's predictions are consistently more accurate than those of the quasi-static algorithm, and the added computation time is negligible. Hence, the perturbed quasi-static algorithm should always be chosen over the quasi-static one.

The dynamic simulation algorithm is the most accurate for all sample parts, except the insulator cap (Table 1), for which the perturbed quasi-static algorithm slightly outperforms it. The dynamic simulation algorithm's prediction accuracy is also the most consistent; the composite error is less than 5% in all cases. Nonetheless, a penalty of three to four orders of magnitude in execution time must be paid for this added accuracy; whether this is appropriate or not depends on the situation.

In an interactive setting, where a designer is perhaps editing the CAD model of a part in order to improve feeder throughput, the perturbed quasi-static algorithm is clearly the best choice. The designer need only wait seconds to see how changing a part's CAD model alters the pose distribution and feeder throughput.

The dynamic simulation algorithm is useful for obtaining a more accurate estimate once the design has been determined, or for analyzing the effects of more subtle design changes. It models several factors, that aren't considered by the standard and perturbed quasi-static algorithms, including: friction, collisions with energy loss, mass moments of inertia, height of drop, and initial conditions of the part prior to drop. To study the effects of varying these parameters, dynamic simulation is appropriate.

Our simulation experiments involved 2000 drops tests. Often, fewer trials may be sufficient, reducing the computational cost of this method. Suppose the true (unknown) probability that a part lands in a particular pose is p. The number of times the part lands in this pose over n trials is a binomial random variable, which may often be approximated by a normal distribution[2]. A *confidence interval* statement is of the form: "p lies within the range $(\mu - \delta, \mu + \delta)$, with $100(1-\alpha)\%$ certainty." Here, μ is the probability estimate obtained from the n trials, δ is the allowable error tolerance, and α is the *level* of the statistical test. Given δ and α, one can bound the number of trials necessary by

$$n = \frac{\Phi^{-1}(1 - \frac{\alpha}{2})}{2\delta}, \tag{17}$$

where $\Phi(x)$ is the cumulative normal distribution function. For example, to pinpoint the probability of a particular final pose to within 5%, with 90% certainty, $\delta = 0.05$ and $\alpha = 0.10$. From (17), 385 trials are sufficient. See [5] for more information.

6 Summary

Predicting the pose distribution of rigid parts dropped onto a flat surface is important in evaluating part designs for assembly. These distributions are necessary to estimate feeder throughput, which can then be used to determine how many robots and assembly lines are required to meet specified production rates. This can greatly reduce the time required to set-up or changeover automated factories and hence allow new products to be more rapidly brought to market.

We have presented three algorithms for predicting the pose distributions of rigid parts dropped onto a flat surface. We have compared the predictions from these algorithms to physical test results, and believe that this is the first systematic comparison of pose estimators with experiments using real industrial parts.

Our results indicate that a perturbed quasi-static algorithm, based on a refinement of the quasi-static algorithm presented in [21], produces significantly more accurate results, with negligible added computation time. The perturbed quasi-static algorithm certainly has the highest accuracy to execution time ratio of all three algorithms studied. The third algorithm, based on dynamic simulation of the dropped parts using the simulator *Impulse*, generally gives the most accurate predictions, with averaged errors under 5% for all four test parts. This algorithm can also be used to study sensitivities to parameters not modeled by the other algorithms, such as the coefficient of friction or the initial part velocity. However, this algorithm takes one to two hours to generate predictions, as opposed to under a second required by the standard and perturbed quasi-static algorithms. In an interactive setting, the quasi-static algorithm is the method of choice, providing reasonably good predictions very quickly. The dynamic simulation algorithm might find application later in the design cycle, where more careful analysis is required.

This work in estimating pose statistics complements other ongoing work in automated assembly. Rao, Kriegman, and Goldberg have studied the use of a pivoting gripper for Adept's flexible feeder; they give an $O(m^2 n \log n)$ algorithm to generate pivot grasps for a part with n faces and m stable configurations [16]. Christiansen, Edwards, and Coello Coello give a genetic algorithm for designing efficient part feeders from component gates [4]. Their algorithm takes pose statistics such as the ones we compute as input.

[2] A common rule of thumb is that the normal approximation is valid if the numbers of successes and failures during the trial series both exceed five [5].

Acknowledgments. Versions of the quasi-static algorithm were implemented at USC: Wiegley in 1992, Zheng Yeh in 1993, and Yan Zhuang in 1994. The convex hull routine used in our current implementation is due to Ioannis Emiris and John Canny. Thanks to Randy Brost and Bruce Shimano for useful feedback on the initial draft of this paper.

References

[1] Vivek Bhatt and Jeff Koechling. Classifying dynamic behavior during three dimensional frictional rigid body impact. In *International Conference on Robotics and Automation*. IEEE, May 1994.

[2] Geoffrey Boothroyd, Corrado Poli, and Laurence E. Murch. *Automatic Assembly*. Marcel Dekker, Inc., 1982.

[3] Brian Carlisle, Ken Goldberg, Anil Rao, and Jeff Wiegley. A pivoting gripper for feeding industrial parts. In *International Conference on Robotics and Automation*. IEEE, May 1994. Also available as USC Techreport IRIS-93-316.

[4] Alan D. Christiansen, Andrea Dunham Edwards, and Carlos A. Coello Coello. Automated design of part feeders using a genetic algorithm. In *International Conference on Robotics and Automation*. IEEE, 1996.

[5] Jay L. Devore. *Probability & Statistics for Engineering and the Sciences*. Brooks/Cole Publishing Company, Monterey, California, 1982.

[6] Peter Dewhurst. Design for robotic assembly. *Manufacturing Engineering*, June 1991.

[7] Gregory T. Farnum and Bill Davis. Delivering the part. *Manufacturing Engineering*, March 1986.

[8] Ken Goldberg and John Craig. Estimating throughput for a flexible part feeder: Simulation and experiments. In *International Symposium of Experimental Robotics*. Stanford University, June 1995.

[9] J. B. Keller. Impact with friction. *Journal of Applied Mechanics*, 53, March 1986.

[10] David J. Kriegman. Let them fall where they may: Capture regions of curved objects and polyhedra. Technical Report 9508, Yale University, June 1995.

[11] Matthew T. Mason, Ken Goldberg, and Yu Wang. Progress in robotic manipulation. In *15th Grantees Conference on Production Research and Technology*. National Science Foundation, January 1989.

[12] Brian Mirtich and John Canny. Impulse-based dynamic simulation. In K. Goldberg, D. Halperin, J.C. Latombe, and R. Wilson, editors, *The Algorithmic Foundations of Robotics*. A. K. Peters, Boston, MA, 1995. Proceedings from the workshop held in February, 1994.

[13] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *Symposium on Interactive 3D Graphics*, New York, 1995. ACM Press.

[14] James L. Nevins and Daniel E. Whitney. Computer-controlled assembly. *Scientific American*, 1978.

[15] B. K. A. Ngoi, L. E. N. Lim, and S. S. G. Lee. Analyzing the probabilities of natural resting for a component with a virtual resting face. *ASME Journal of Engineering for Industry*, (to appear), 1995.

[16] A. Rao, D. Kriegman, and K. Goldberg. Complete algorithms for reorienting polyhedral parts using a pivoting gripper. In *International Conference on Robotics and Automation*. IEEE, May 1995.

[17] Edward J. Routh. *Elementary Rigid Dynamics*. Macmillan, London, 1905.

[18] Berhard J. Schroer. Electronic parts presentation using vibratory bowl feeders. *Robotics*, 3, 1987.

[19] Neil C. Singer. Utilizing dynamic and static stability to orient parts. Master's thesis, MIT, 1985.

[20] Yu Wang and Matthew T. Mason. Modeling impact dynamics for robotic operations. In *International Conference on Robotics and Automation*, pages 678–685. IEEE, May 1987.

[21] Jeff Wiegley, Anil Rao, and Ken Goldberg. Computing a statistical distribution of stable poses for a polyhedron. In *30th Annual Allerton Conference on Communications, Control, and Computing*, 1992.