A Constraint-Aware Motion Planning Algorithm for Robotic Folding of Clothes

Karthik Lakshmanan, Apoorva Sachdev, Ziang Xie, Dmitry Berenson, Ken Goldberg, and Pieter Abbeel

Abstract Motion planning for robotic manipulation of clothing is a challenging problem as clothing articles have high-dimensional configuration spaces and are computationally expensive to simulate. We present an algorithm for robotic cloth folding that, given a sequence of desired folds, outputs a motion plan consisting of a sequence of primitives for a robot to fold the cloth. Previous work on cloth folding does not take into account the constraints of the robot, and thus produces plans which are often infeasible given the kinematics of robots like the Willow Garage PR2. In this paper we introduce a class of motion primitives that start and end in a subset of the cloth's state space. To find a sequence of primitives that achieves all required folds, the algorithm takes advantage of the partial ordering inherent in folding, and produces a time-optimal motion plan (given the set of primitives) for the robot if one exists. We describe experiments with a general purpose mobile robotic platform, the PR2, folding articles that require dragging and base motion in addition to folding. Our experiments show that (1) many articles of clothing conform well enough to the assumptions made in our model and (2) this approach allows our robot to perform a wide variety of folds on articles of various sizes and shapes.

1 Introduction

Robotic manipulation of 2D deformable objects is a difficult problem largely because such objects typically have infinite-dimensional configuration spaces and are too computationally expensive to simulate in the inner-loop of a motion planner.

The problem we address is as follows: Given a robot model, the shape of a piece of cloth in a spread-out configuration on a horizontal table, and a final folded configuration specified by a sequence of g-folds, output a sequence of robot motions that achieves the final folded configuration or report that none exists.

The state-of-the-art approach is g-folding [24]. At the core of this approach is the definition of a cloth model that allows reasoning about the geometry rather than the physics of the cloth in relevant parts of the state space. Given the geometry of the cloth, their algorithm computes how many grippers are needed and prescribes

University of California, Berkeley
[karts,apoorvas,zxie,berenson,goldberg]@berkeley.edu, pabbeel@cs.berkeley.edu

motions for these grippers to achieve the final configuration, specified as a sequence of g-folds—folds that can be achieved while staying in the subset of the state space to which their geometric model applies. G-folding, however, has severe practical limitations: due to robot and environmental constraints, the gripper motions produced by g-folding are often infeasible. When that happens, the top-down approach followed by g-folding fails.

This paper presents a motion planning approach that plans directly at the level of robotic primitives rather than the gripper motions used in [24]. Given a sequence of g-folds that take the cloth from the initial to the final configuration, our approach determines whether a sequence of motion primitives exists that results in the successful execution of all specified g-folds. If so, the algorithm outputs the robot motion which brings the cloth to the final configuration.

To restrict the (otherwise unmanageably large) search space we restrict our search to a class of motion primitives, called *g-primitives*. This class of primitives consists of all motions that begin and end with the cloth in a *g-state* — a configuration of the cloth where all parts of the cloth are either horizontal or vertical. This is a broader class of motion than was allowed in [24] because the intermediate states of the cloth during execution of primitives are unrestricted. The primitives used in our experiments are of three types: performing a g-fold (including folds that allow the article to hang from the table), dragging the article along the table, and robot base motion.

To find a plan to fold a given article using a given set of g-folds, we associate a cost with each g-primitive and search for an optimal solution, returning failure if no solution exists. Our algorithm for searching over primitives consists of (1) generating a graph that captures dependencies between the folds specifed in a g-fold sequence and (2) using A* to search for a sequence of primitives that accomplishes all folds in the graph.

In our experiments the cost we associated with each g-primitive is the time it takes to execute. Hence not only does our approach allow us to find solutions in cases where g-folding would simply fail, it also enables finding better (i.e., time-optimal) solutions. Our experiments show that (1) many articles of clothing conform well enough to the assumptions made in our model and (2) this approach allows our robot to perform a wide variety of folds on articles of various sizes and shapes.

2 Related Work

Our work builds on g-folding [24], but does account for kinematic restrictions posed by real robots. Our work also draws from the work of Bell and Balkcom [4, 5], which deals with computing the grasp points needed to immobilize a polygonal non-stretchable piece of cloth. Relevant work in cloth folding includes dynamic towel folding [1], cloth perception [21], and strategies to bring the cloth into a spread-out configuration where folds can be applied [20, 10, 6].

Our work is also related to methods in motion planning that search over primitives to construct robot trajectories [13, 7, 17, 23, 9, 22, 15]. Similar search-based approaches have also been developed in the graphics literature [19, 18]. However, to our knowledge no previous work has applied a search over motion primitives to the cloth folding problem.

Other relevant work has been done in physical simulation of cloth [3, 8, 12], origami folding [2], carton folding [16, 12], and metal bending [14], which use similar material models to the one presented here.

3 Problem Statement

The problem we address is as follows: Given a robot model, the shape of a piece of cloth in a spread-out configuration on a horizontal table, and a final folded configuration specified by a sequence of g-folds, output a sequence of robot motions that achieve the final folded configuration or report that none exists.

We assume a coordinate frame in which gravity acts in the downward vertical ($-z$) direction and a rectangular table in the horizontal (xy) plane. We assume the article of clothing can be fully described by a 2D polygon (convex or non-convex) with n vertices, initially lying on the horizontal surface in a fully spread-out, known configuration. Except for the table surface, there are no obstacles in the environment.

We define S as the state space of the system (i.e. the robot and cloth). A state $s \in S$ consists of a robot state and a cloth state.

We are also given a robot model, which specifies the number and width of grippers and the inverse kinematics for each gripper. It is assumed that the grippers are not capable of distinguishing between layers of cloth and will grasp all layers of the stack at each grip location. For example, the Willow-Garage PR2 we use in our experiments has 2 grippers.

In this paper we focus on robots with a mobile base and two manipulators (like the PR2), though our framework is not limited to this type of robot. Focusing on this robot model, a robot configuration q consists of the (x, y, θ) of the base, a set of joint angles for each arm $\{\alpha_1, \ldots, \alpha_{2N}\}$ (where each arm has N joints), and booleans q^L_{grip}, q^R_{grip} indicating whether each gripper is open or closed. The configuration of the cloth is represented as a set of polygons.

Since we are building on the g-folding work in [24], we inherit the assumptions of the g-folding model:

1. The cloth has infinite flexibility. There is no energy contribution from bending.
2. The cloth is non-stretchable. No geodesic path lengths can be increased.
3. The cloth has infinite friction with itself.
4. The cloth has zero thickness.
5. The cloth is subject to gravity.
6. The cloth has no dynamics, i.e. it is moved quasi-statically.
7. If the cloth is held by a number of grippers, and one or more grippers release the cloth, no point of the cloth will move upwards as a result of gravity and internal forces within the cloth. (The *downward tendency assumption* from [24], which works well for most everyday articles of clothing. However, it does not hold for a family of objects known as pinwheels, as shown in [4].)

We extend the model used in g-folding to allow the robot to perform more types of motions (such as dragging and letting parts of articles hang off the table). We add the following assumptions:

8. The weight and density of the cloth are known.

9. There is a known uniform coefficient of static friction μ between the cloth and the surface on which it lies. The robot can exert enough force on the cloth to overcome this friction. (Note that this assumption is different from [24], which assumes that this friction is infinite.)
10. If a part of the cloth on a horizontal surface is gripped at one or more points, and all such points are moved horizontally at the same velocity, then the distance of any point of the cloth from a gripper, measured in the xy plane, will not decrease. Intuitively, this means that when a cloth is dragged across a surface, no point of the cloth will move towards a gripper due to internal forces.

4 g-states

The key to our search strategy is to restrict the movements of the robot to start and end in a *g-state*. A state $s \in S$ is a g-state if and only if a part (possibly empty) of the cloth is horizontal and a part (possibly empty) is vertical, and the cloth is not in motion and will not move independently. We define a *baseline* as a line that separates a horizontal part of the cloth from a vertical part.

Given the problem description above, there are two ways for parts of the cloth to be vertical if the cloth is in a g-state: 1) The cloth is held up by robot grippers and/or 2) the cloth is hanging off the edge of the table. In the first case, the cloth will not move as long as Theorem 1 from [24] (restated below for convenience) holds:

Theorem 1 *In our material model, a vertically hanging cloth polygon is immobilized when every convex vertex of the cloth at which the negative gravity vector does not point into the cloth polygon is fixed (i.e. be held by a gripper or be part of the baseline).*

In the second case, as long as the net horizontal force on the part of the cloth lying on the table surface can be countered by friction, points lying on the table surface (including those on the baseline) will not move. This is because such points cannot move in the plane due to friction, and they will not move upward per the downward-tendency assumption. Thus to determine if the cloth will slip over the edge, we check the following two conditions:

1. The friction between the table surface and the part of the cloth on the table must be sufficient to counter the force exerted by gravity on the vertically hanging parts of the cloth.
2. All convex vertices of the hanging part at which the negative gravity vector does not point into the cloth are part of the table edge (which is also the baseline), from which the part of the cloth hangs. This prevents the vertical part of the cloth from moving, according to Theorem 1. Examples are shown in Figure 1.

A *convex vertex* is defined as a vertex of the cloth polygon that is also a vertex of its convex hull.

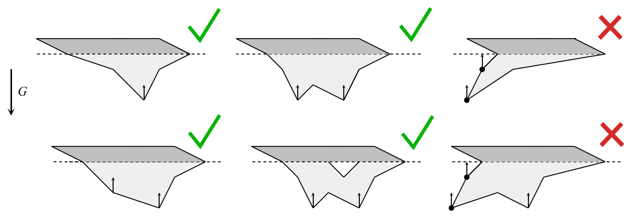


Fig. 1: Examples of parts of cloth hanging off the table in various configurations. The dotted line is the table edge (baseline). In order for the cloth to be hung, all convex vertices not at the baseline should have the negative gravity vector (small arrows) point into the cloth. These vertices are indicated by dots. In the configurations shown on the right, there are convex vertices where the negative gravity vector does not point into the cloth and which are not on the baseline, hence the cloth is not immobilized.

5 Algorithm Formulation

We use a search-based procedure to find a sequence of folding primitives that results in the successful completion of all the requested g-folds, if possible. Our planning algorithm is comprised of two components: (1) The creation of a Fold-DAG. (2) A search over motion primitives.

5.1 Fold-DAG

The user inputs a desired sequence of g-folds. However, it may not be necessary to execute the folds in the same order as given, as some folds may be performed independent of the previous folds in the sequence. For example, consider the g-fold sequence shown in Figure 2. If the robot were to perform the folds in the same order, it might have to, for example, complete fold 1, then move around to complete fold 2 before moving back to complete fold 3. However, we can see that either the g-fold in part 1 or the one in part 2 can be performed first. The g-fold in part 3 only needs part 1 to be completed, and the g-fold in part 4 only needs part 2. Capturing the dependencies between the requested g-folds can potentially lead to quicker solutions. For example, the robot can choose to complete 1 and 3 before moving around to complete 2 and 4.

We use a directed acyclic graph (DAG), which we call the Fold-DAG, to capture dependencies between the folds specifed in a folding sequence, i.e. which g-folds need to be completed before a particular g-fold can be performed. The nodes V of this graph $G = (V, E)$ are the specified folds and the edges E specify dependencies. E is produced in two steps: First, we add an edge from node $v_a \in V$ to node $v_b \in V$, $e(v_a, v_b)$ if the folded part of the cloth specified in node v_b overlaps the folded part of the cloth specified in v_a for all pairs of nodes in the graph. We then remove redundant dependencies by eliminating any $e(v_a, v_b)$ if there is a path from v_a to v_b that consists of more than one edge. For example, Fig 2 shows an input the g-fold sequence and its Fold-DAG.

When executing a sequence of folds, we are implicitly traversing the Fold-DAG. We can determine a list of *available* g-folds for each cloth configuration depending

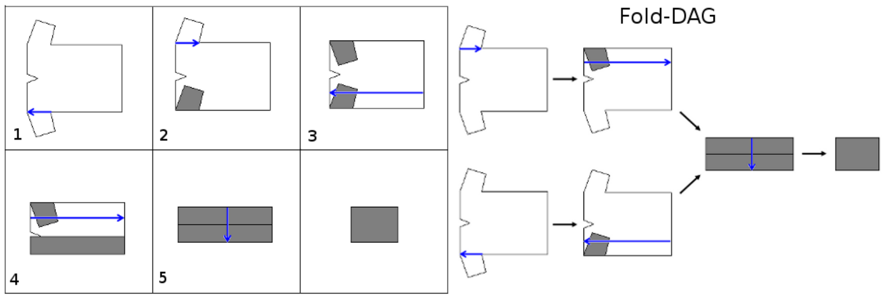


Fig. 2: The Fold-DAG generated from a set of user defined g-folds. Left: A user defined sequence of g-folds for a T-shirt. Right: The sequence when it has been transformed into a Fold-DAG. The corresponding T-shirt configurations are also shown for clarity.

on the g-folds that have been performed previously. A node v is *available* if all nodes in $V_i = \{v_i \in V | e(v_i, v) \in E\}$ have been executed, i.e. if all nodes pointing to v have already been performed.

5.2 Motion Primitives

The goal of the search process is to produce a series of motion primitives that achieves the given sequence of g-folds. If we disregard the kinematic constraints of the robot, this process is trivial since all g-folds can be performed directly. However, taking into account the constraints, a given g-fold may not be possible due to, for instance, the grip locations being too far from the robot. The search process searches over a given set of motion primitives, attempting to achieve the given g-fold sequence. The search algorithm uses the Fold-DAG to guide the search to the goal configuration (where all requested g-folds have been performed).

In order to make the search process computationally feasible, we must ensure that the search space we consider is not unmanagably large. Allowing any arbitrary motion to be considered would cover S, but it would also produce an unmanagable search space and would require simulations of the cloth dynamics (which are computationally expensive and sometimes inaccurate). We thus restrict the search to a class of motion primitives we call *g-primitives*.

Intuitively, a g-primitive is any motion primitive that both starts and ends with the system in a g-state. Formally, a *motion primitive* is a function $f : S \longrightarrow T$, where T is the set of all robot trajectories.

Definition 1. A *g-primitive* is a motion primitive whose domain and range are restricted: $f : S' \longrightarrow T'$, where $S' \subseteq S$ is the set of all g-states and $T' \subseteq T$ is the set of all trajectories that end with the system in a g-state.

Using g-primitives allows us to search over a managable subspace of S while also allowing us to perform practical tasks with the cloth. It is important to note that there are no restrictions on the intermediate states of the robot and cloth when performing a g-primitive (i.e. the cloth and robot can move arbitrarily) as long as the start and end conditions are met (see Figure 3). g-primitives are also a broader class of motion

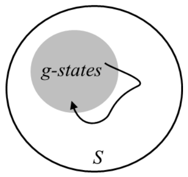


Fig. 3: The set of states S and the subset of g-states. The black path represents the sequence of states traversed by an example g-primitive.

than was allowed in [24], where all intermediate states of a fold were required to be g-states.

The g-primitives used in this paper are described in Section 6, however these are only examples of what possible primitives could be used, and by no means constitute the entirity of the class of g-primitives.

*5.3 Search Over Primitives Using A**

Given vertices of the cloth, the robot model, and the Fold-DAG we can compute a time-optimal motion sequence of primitives for the robot to execute in order to reach the desired final configuration. We define a time-optimal sequence as one which takes the least time to execute on the robot.

We used the A* search algorithm to find a sequence of g-primitives, which is a heuristic-guided search that guarantees solutions of lowest cost. To apply A*, we require functions that compute whether a state is a goal, the successors of a state, the cost of reaching a state, and a heuristic estimate of the cost to reach the goal from a given state.

We define a goal state as a state which has no g-folds remaining in the DAG. For a given state, we generate the successors by applying all the primitives in our set of primitives. If a given primitive is infeasible (due to, for instance, reachability contraints), it does not generate a successor. Since we want the search algorithm to return the plan of least execution time, the sum of costs along the path starting at the starting state is a measure of the time taken by the robot to perform all primitives that constitute the path. The primitive-specific cost functions are given in Section 7.

Our A* heuristic uses a relaxation of the problem which ignores robot constraints and approximates the time taken for the robot to perform all remaining folds in the DAG at the given state. Since robot constraints are ignored, only gripper trajectories are considered and the method of computation is identical to that used to find the time taken to move the grippers in the cost calculation. This also ensures that the heuristic is admissible.

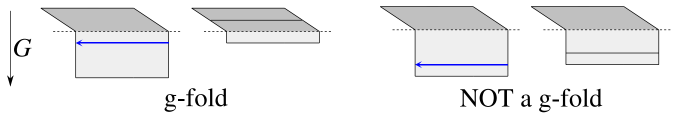


Fig. 4: Left: A valid g-fold performed on the hanging part. The dotted line shows the table edge. The hanging part is highlighted in light gray. The cloth does not move in the resulting configuration due to friction between the cloth and table. Right: A fold on the hanging part of the cloth which is not a g-fold because the cloth will move after it is released.

6 Primitives for Robotic Cloth Manipulation

Any primitive that starts and ends in a g-state is allowed. We provide three examples of primitives; g-fold, g-drag, and base motion, including the motivation for and a formal description of each.

g-fold

As defined in [24], a *g-fold* is specified by a directed line segment in the plane whose endpoints lie on the boundary of the cloth polygon. The segment partitions the polygon into two parts, one to be folded over another. A g-fold is successfully achieved when the part of the polygon to the left of the directed line segment is folded across the line segment and placed horizontally on top of the other part, while maintaining the following property:

At all times during a folding procedure, every part of the cloth is either horizontal or vertical, and each vertical part of the cloth is held either by grippers or by the edge of the table such that it is immobilized. (See Figure 2 in [24].)

The definitions of “blue” and “red” g-folds from [24] remain unchanged—a blue g-fold is specified by a line segment partitioning the polygon formed by the silhouette of the stacked geometry into two parts, and is successfully achieved by folding the (entire) geometry left of the line segment. A red g-fold is similarly specified, but only applies to the (potentially stacked) geometry that was folded in the previous g-fold.

The g-fold primitive is the execution of a user specified g-fold (“red” or “blue”). g-folds involving parts of the cloth that are not hanging are performed in the same way as described in section 4 of [24]. A fold for parts of the cloth that are hanging can be performed when the following conditions hold true:

1. Theorem 1 must hold true, and for this the robot must have a sufficient number of grippers, which must be able to reach and grip the necessary vertices throughout the motion required to complete the fold.
2. A g-fold parallel to the table edge along which the cloth hangs is permitted only if the resulting cloth configuration is a g-state (see Fig 4 for an example where this condition is violated).

g-drag

Kinematic restrictions on the robot arms may yield points on the cloth that need to be gripped in order to perform a g-fold unreachable from the robot's current pose. This motivates the action of dragging the article across the table, in order to make such points reachable. The g-drag primitive is defined as the translation of the part of the article on the *xy* plane of the table. A g-drag is determined by the direction of dragging and distance through which the article is dragged. See Figure 5 for examples of g-drags.

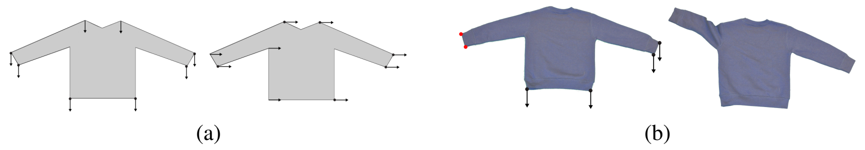


Fig. 5: (a) Two example g-drags of a long sleeve shirt in two directions. For translation without deformation of the article all convex vertices on the table surface at which the negative friction vector (small arrows) does not point into the cloth must be grasped. Grasp locations are indicated by the dots. (b) A drag of a long sleeve shirt which is not a g-drag. Red convex vertices are not gripped, which results in deformation of the left sleeve (shown on the right).

In order to ensure that the part of the cloth on the surface of the table is translated without any deformation, the following theorem must hold true:

Theorem 2 *In our material model, a cloth polygon resting on a horizontal surface can be translated without deformation if (1) every convex vertex of the cloth at which the negative vector of friction (between the table and the cloth) does not point into the cloth polygon is held by a gripper, and (2) all such vertices move at the same velocity.*

Proof. Our proof is based on the proof of Theorem 1 in [24], where van den Berg et al. prove that a vertically hanging cloth polygon can be immobilized by fixing every convex vertex of the cloth at which the negative gravity vector does not point into the cloth. Consider a cloth polygon being dragged across a horizontal surface with uniform friction. The motion does not induce deformation of the cloth if at any time during the drag, all points of the polygon have the same velocity as the gripped points. Let us pick one of these gripped points, and, without loss of generality, consider a frame of reference whose origin is at this point, such that the table surface is the *xy* plane, and the positive *x* axis is the direction along which the cloth is dragged. The origin moves with the gripped point as the cloth is dragged. Uniform friction then acts along the negative *x* axis. Using assumption 10, we know that no point of the cloth will move in the positive *x* direction due to internal forces.

Let us define a *leading string* of the polygon as a maximal sequence of edges of which the extreme vertices are convex vertices of the polygon, and no part of the polygon has a greater *x*-coordinate than these edges. A given polygon *P* can have multiple leading strings, but has at least one.

The proof now proceeds in a similar fashion to the proof in [24], which presents a recursive proof where at every step in the recursion, all *upper strings* of a polygon P are proven immobilized, and thereby every point of P that can be connected to an upper string by a vertical line segment that is fully contained within P is proven immobilized. The notion of "upward" in their case is analogous to the $+x$ direction in our case, and the "downward" force of gravity can be replaced by friction in the $-x$ direction. Leading strings take the place of "upper strings." Then, in a similar fashion, we can prove that all leading strings are immobilized. The proof in [24] uses the downward tendency assumption to show that 1) non-convex vertices of a gripped upper string will not move upward, 2) every point of P that can be connected to an upper string by a vertical line segment that is fully contained within P cannot move upward. Assumption 10 serves the same purpose in our case, ensuring that the corresponding points do not move in the $+x$ direction. The rest of the proof follows accordingly. $\square$

The possible distances an article can be dragged form a continuous space, but we discretize them to meet the requirements of our search algorithm. More details are given in the experiment description.

Move

The robot is allowed to move around the table, which makes different portions of the article reachable. The space of robot base poses is continuous and we manually select a set of base poses from this space that provides good coverage of the table surface for the arms. The allowed poses for our robot's base are shown in Figure 7a.

7 Costs for the Primitives

The cost we associated with each g-primitive is the time it takes to execute. Our motion primitives can be decomposed into sequences of base motion and arm movement, and the cost is computed by summing the execution time for each motion.

A *g-fold* consists of moving the arm(s) to the grip configuration, performing the fold, and releasing the cloth (if blue fold), while moving the base backward and forward as necessary to reach the computed gripper locations.

A *g-drag* consists of moving from the initial configuration to the grip configuration, moving the base along the direction of the g-drag, and releasing the cloth (if the next fold is not a red fold).

For the above two primitives it is sometimes necessary for the robot to back away from the table in order to make the target grippoints reachable (this is due to the PR2's limited reachability for points close to its body). Thus if the grip points are too close to the robot, we include a backward base motion in the primitive's sequence of motions as well.

The *move* primitive moves the base between pre-specified positions by turning in place, driving to the target position, and turning in place again to reach the desired rotation.

The costs for the above primitives are computed by summing the execution time necessary to complete each component of the primitive (i.e. reaching and base

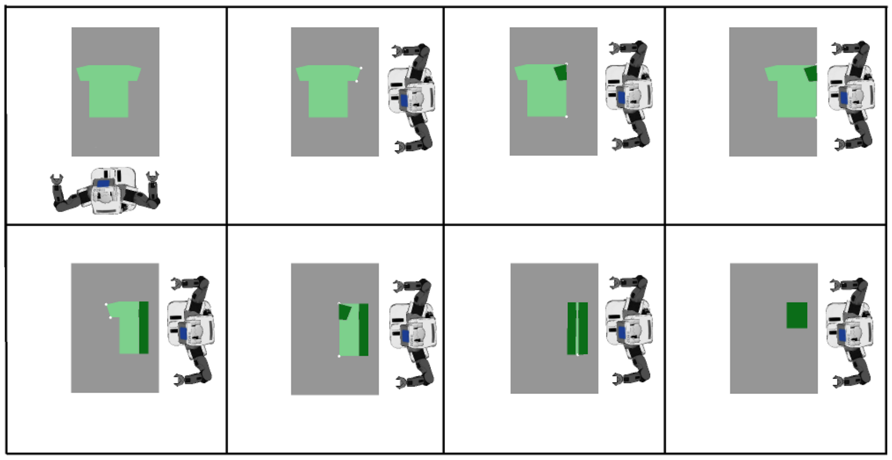


Fig. 6: Solution for the T-shirt using the g-folds of Figure 2 as input.

movement). Details about setting the execution time for the arms is discussed in Section 8. The time necessary to execute base motions is computed assuming constant linear and angular speed.

8 Planning and Simulation Experiments

We used a Willow Garage PR2 robotic platform [15] and performed experiments both in simulation and on the physical robot. For the Move primitive, our robot could move between 5 predefined positions around the table as shown in Figure 7a. For the Fold primitive, the computed grip-points were often too close to the robot's body for it to reach them without moving its base. To overcome this, the robot was allowed to move its base backward until the points could be reached, thereby enhancing the robot's reachability along the x-axis, (as defined for each position in Figure 7a). For the Drag primitive, we only allowed the robot to drag the article towards itself. The drag distance was discretized, and drags in increments of 10 cm (up to the length of the article) were considered. Grip points were computed by first computing grip locations as prescribed by the primitives while assuming point grippers. Then, following [24], we accounted for gripper width (and some cloth stiffness) to find a reduced number of required grip-points.

We used 3 seconds for the input execution time for all arm motions on the PR2. We found that going below this time decreases the path-following accuracy of the physical robot, thus resulting in imperfect folds.

We experimented with nine common clothing articles listed in Table 1 in simulation. The folding sequence is given as input to the algorithm. An example of the solution sequence for a T-shirt is shown in Figure 6.

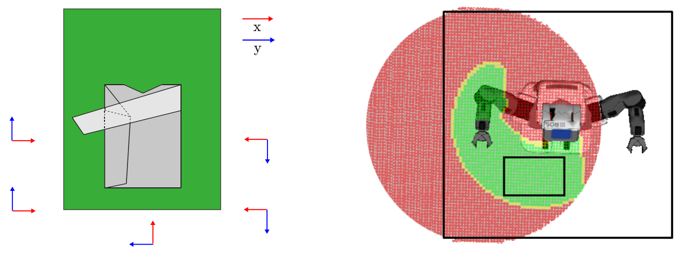
We used the `ikfast` module provided by OpenRAVE [11] in order to determine if the robot can reach a given point. This can be a time consuming operation because it involves collision-checking and searching over the free joint of the 7 DOF manipulator, and IK queries are made often by the search algorithm. In order to decrease the number of times `ikfast` is called, we created a fast approximation to IK that

Article	# Folds	L	C(s)	N Expand	N Explore	T(s)	IK time(s)	IK Queries	Overhead(s)
T-shirt	5	7	48.12	16	84	6.29	1.14	3512	0.007
Jeans	2	3	35.01	13	116	4.07	1.44	776	0.008
Shirt	7	9	76.94	84	491	28.29	3.96	5911	0.040
Tie	3	4	27.90	24	256	6.06	1.59	593	0.013
Scarf	2	3	18.90	14	177	4.94	1.37	593	0.011
Vest	2	2	24.62	8	61	1.91	0.41	383	0.004
Skirt	3	4	29.63	60	535	15.37	5.42	10087	0.026
Big Towel	3	4	27.91	22	232	8.47	3.01	6675	0.013
Hand Towel	3	3	27.35	18	161	5.04	1.90	400	0.007

Table 1: Simulation Results. L is the number of primitives in the solution path, C is the cost of the path in seconds. N expand/explore is the number of nodes expanded/explored. T is the total search time of algorithm.

we query before calling `ikfast`: offline, we define two bounding boxes for each robot arm such that any point within the inner box is reachable by that arm's wrist and any point outside the larger outer box is unreachable from the current robot base position. For each arm, the inner box is pre-computed by making IK queries on a grid of 3D points around the robot with a 1cm discretization. We then check if the wrist can reach each point in the grid. We only consider points that are above the table and less than 1m above the floor, as this region encompasses most of the points whose reachability the planner needs to check. See Figure 7b. The shape of reachable 3D points is not cuboidal, but we inscribe a cuboid in this region to create the inner box for ease of checking during the search. The outer box is constructed based on the fact that the PR2 cannot reach any point that is further away than the length of its arm at full stretch. For our experiments, any point farther than 0.75m from the center of the base in any direction lies outside the outer box. Whenever the planner makes an IK query, we first check if the translation component of the query is within the inner box for the relevant arm. If so, we immediately return that the point is reachable. If the point is outside the outer box, we immediately declare that it is unreachable. If the point is contained in the outer box but is not in the inner box, we query `ikfast`. This enabled us to reduce the time spent on IK queries by about 50%, on average.

In order to make the generated plans robust to robot execution error (for example, dragging by less than the desired amount), we introduce the concept of "IK comfort." We only declare a point reachable if both the point and four points on the



(a) (b)

Fig. 7: (a) Allowed robot base positions around the table for the Move primitive. (b) Bounding boxes used to decrease the number of IK queries on the PR2.

circumference of a circle of a set radius, centered at the given point are reachable. Also, if the gripper fails to grab the cloth at the target point during execution on the robot, it tries to grab other points that lie within the comfort radius of the target before failing. For our trials, we found that an IK comfort radius of 3 cm resulted in robust execution.

Article	# Folds	L	C(s)	N Expand	N Explore	T(s)	IK time(s)	IK Queries	Overhead(s)
T-shirt	5	7	48.12	40	319	55.08	39.77	9950	0.010
Jeans	2	3	35.01	13	60	1.64	1.25	272	0.006
Shirt	7	9	76.94	134	920	121.70	85.99	26867	0.034
Tie	3	4	27.90	26	329	20.75	11.89	4387	0.007
Scarf	2	3	18.90	16	197	18.71	14.01	4950	0.004
Vest	2	2	24.62	14	101	9.24	6.82	2073	0.007
Skirt	3	4	29.63	22	201	19.61	14.87	4432	0.006
Big Towel	3	4	27.91	24	284	35.63	26.23	6921	0.016
Hand Towel	3	3	27.35	17	175	20.80	15.25	4214	0.004

Table 2: Simulation Results with IK comfort radius of 3 cm. See symbol definitions in Table 1.

9 Physical Experiments

We used a rectangular table with a soft working surface, so that the relatively thick grippers can easily get underneath the cloth. At the beginning, the robot can always see the entire clothing article in a known, fully spread-out configuration. The robot detects the green table surface using hue segmentation. The cloth polygon is then detected as described in [21]. We used the PR2 2D-navigation package in ROS to move the robot between the 5 base positions. The robot performed Drags by moving the base backward, and then correcting for any undershoot in base motion by using its grippers to drag the cloth further.

Clothing item	Success Rate	Av. Execution Time	Av. Vision Time	Av. Total Time
T-shirt	3/3	218s	60s	278s
Large towel	2/3	78s	20s	98s
Skirt	3/3	91s	20s	111s
Jeans	3/4	150s	55s	205s

Table 3: Success rates and timing results of physical robot executions. Vision Time includes time for initial detection as well as for visual feedback during execution. The times are averaged over all successful runs.

For several of the articles, we executed the generated plans multiple times on the robot. Table 3 shows the results of our runs. Several snapshots from folding a T-shirt, towel, and skirt are shown in Figure 8. Videos of the executions are posted at `http://rll.berkeley.edu/iser2012-folding`.

The PR2 uses visual feedback to correct for execution errors. The position of the current cloth model is updated after every g-drag or Move primitive, by fitting the model to the observed contour of the cloth on the table surface, assuming that

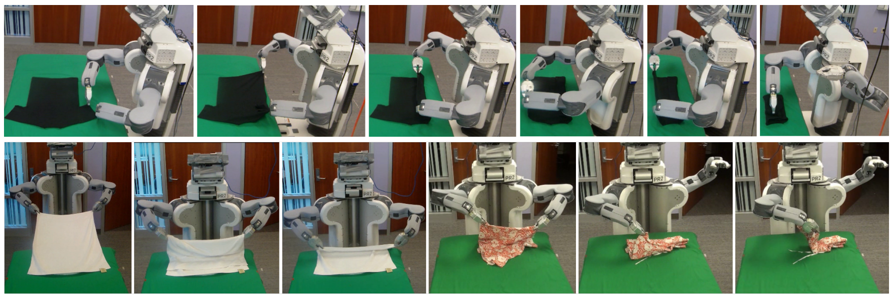


Fig. 8: T-shirt, towel and skirt folding sequences executed by PR2.

the cloth only undergoes translation and not deformation. The expected translation of the cloth is used to initialize the position of the model, following which the optimization proceeds as described in [21]. After each g-fold, the performed fold is detected and the model is updated as described in [21].

As illustrated by our success rates on the various clothing articles, our method shows high reliability on real cloth, even if it does not perfectly conform to our assumptions. For example, jeans and towels clearly violate the zero thickness assumption and the pleats of the skirt are not taken into consideration by our cloth model. Regardless we are able to achieve high success rates on these articles.

Our failures typically arise from errors in robot execution, particularly base motion. If the base does not move by the desired amount, a grip point might become unreachable. The introduction of IK comfort along with making the arms correct for base motion undershoot during a drag greatly reduces the number of such failures.

Our experiments show that the planned execution times typically underestimate the real execution times observed with the PR2. This is because the costs for each primitive used in the planning phase are highly idealized. A large part of this discrepancy can be attributed to the Move primitive. The 2D navigation package causes the robot to stop multiple times during the move. The planner also ignores certain other behaviors of robot execution. For example, the PR2 sometimes fails to grab the cloth on the first attempt, and would need to move the gripper to regrasp the cloth. Additionally, the planner assumes a constant velocity for base movement, while the robot actually spends more time accelerating and decelerating than at its full speed.

We are currently investigating ways to overcome these limitations. If the PR2 fails to grasp the article on its first attempt, it will try to grab up to 8 other points within the IK comfort radius until it grasps the cloth. The discrepancy in the computed cost of Moves and the real times may be reduced by eliminating the constant velocity assumption and averaging the actual times taken to move from one base position to another over multiple runs.

10 Conclusion

In conclusion, we described a motion planning algorithm for robotic cloth folding, enabling us to avoid computationally expensive physics simulations while taking into account kinematic constraints. We presented examples of cloth manipulation primitives that allow the robot to perform a set of user defined g-folds using our simplified cloth model. Our search algorithm allowed the robot to choose a sequence of primitives to perform all given folds in the shortest possible time (given the available primitives). At the core of our method is the consideration of real robot limitations. Our experiments show that (1) many articles of clothing conform well enough to the assumptions made in our model and (2) this approach allows our robot to perform a wide variety of folds on articles of various sizes and shapes.

References

1. B. Balaguer and S. Carpin. Combining imitation and reinforcement learning to fold deformable planar objects. IROS, 2011.
2. D. Balkcom and M. Mason. Robotic origami folding. IJRR, 27(5):613–627, 2008.
3. D. Baraff and A. Witkin. Large steps in cloth simulation. SIGGRAPH, 1998.
4. M. Bell. Flexible object manipulation. PhD Thesis, Dartmouth College, 2010.
5. M. Bell and D. Balkcom. Grasping non-stretchable cloth polygons. IJRR, 29:775–784, 2010.
6. Christian Bersch, Benjamin Pitzer, and Soren Kammel. Bimanual robotic cloth manipulation for laundry folding. In IROS, pages 1413 –1419, Sept. 2011.
7. Michael S. Branicky, Ross A. Knepper, and James J. Kuffner. Path and trajectory diversity: Theory and algorithms. In ICRA, 2008.
8. R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact, and friction for cloth animation. SIGGRAPH, 2002.
9. B. J. Cohen, S. Chitta, and M. Likhachev. Search-based planning for manipulation with motion primitives. In ICRA, pages 2902 –2908, May 2010.
10. Marco Cusumano-Towner, Arjun Singh, Stephen Miller, James O'Brien, and Pieter Abbeel. Bringing clothing into desired configurations with limited perception. In ICRA, 2011.
11. Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA, July 2008.
12. N. Fahantidis, K. Paraschidis, V. Petridis, Z. Doulgeri, L. Petrou, and G. Hasapis. Robot handling of flat textile materials. IEEE Robotics and Automation Magazine, 4(1):34–41, 1997.
13. C. Green and Alonzo Kelly. Toward optimal sampling in the space of paths. In ISRR, 2007.
14. S. Gupta, D. Bourne, K. Kim, and S. Krishnan. Automated process planning for robotic sheet metal bending operations. Journal of Manufacturing Systems, 17:338–360, 1998.
15. Kris Hauser, Tim Bretl, Kensuke Harada, and Jean-claude Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In WAFR, 2006.
16. T. Kanade. A theory of origami world. Artificial Intelligence, 13(3):279–311, 1980.
17. R. Knepper and M. Mason. Empirical sampling of path sets for local area motion planning. In ISER. Springer, 2008.
18. Manfred Lau and J.J. Kuffner. Behavior planning for character animation. In SIGGRAPH, 2005.
19. Jehee Lee, J. Chai, P.S.A. Reitsma, J.K. Hodgins, and N.S. Pollard. Interactive control of avatars animated with human motion data. In SIGGRAPH, 2002.
20. J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In ICRA, 2010.
21. S. Miller, M. Fritz, T. Darrell, and P. Abbeel. Parametrized shape models for clothing. In ICRA, 2011.
22. P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. learning and generalization of motor skills by learning from demonstration. In ICRA, 2009.
23. Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Mobile Robot Motion Planning. Journal of Field Robotics, 26(3):308–333, 2009.
24. J. van den Berg, S. Miller, K. Goldberg, and P. Abbeel. Gravity-based robotic cloth folding. WAFR, 2010.