LaTeX Dynamic Optimization Hw 3-contact invariant optimization

Christopher Hazard, Jaskaran Grover

May 5, 2017

For this homework, we implemented Igor Mordatch's Contact-Invariant Optimization for Hand Manipulation paper (found here: https://homes.cs.washington.edu/ zoran/MordatchSCA12.pdf) We were not able to make it successfully optimize a physically valid motion for the simple example of taking a capsule object and moving it without rotation.

1 Description of the paper/optimization

Extending the idea from his previous contact-invariance paper, in which Mordatch created an additional continuous variable c_i for each proposed contact with a target object, this paper applies the same principle to hands in simulation to generate physically feasible manipulations of an object given a sequence of target poses for the object and a sequence of keyframe poses over which the motion is optimized.

The optimization variables consist of S=$[s_k]$ where s_k=$[s_H \ s_O]$ are the states of the object and the hand respectively at keyframe k with $k \in \{1, 2, , N_{keyframes}\}$. S_H=$[x_H \ \dot{x}_H \ p_i \ \dot{p}_i]$ with $x_H \in \mathcal{R}^7$being the position and orientation of the palm and p_i being the location of finger i with $i \in \{1, 2, , N_{fingers}\}$. In Mordatch's paper, he used exponential maps to represent orientations, however we opted to instead use quaternions with SLERP (spherical linear interpolation). S_O=$[x_O \ \dot{x}_O \ f_j \ r_j^O \ c_j]$ with j $\in$ 1,2,,$N_{contacts}$. f_j and r_j^O are the force and contact position (in the object frame) of contact j, x_O is the position and orientation of the object (again swapping exponential map for quaternions with SPERP), and c_j being the contact invariant weight, which is constrained to be within [0,1] (hard constraint). The total number of contacts is $N_{contacts} = N_{fingers} + 3$ (as is the case in Mordatch's paper) since we have one contact per fingertip as well as one contact for the palm and 2 contacts for the ground.

Between every two states s_k and s_{k+1} the optimization variables are splined as follows:

1. x and p are splined with catmull-rom splines (in Mordatchs paper, he used cubic spline interpolation, for which he needed derivatives for x and p as optimization variables—-since we chose catmull-rom splines, we didnt need $\dot{x}$ and $\dot{p}_O$ in our optimization)

2. f and r_o are splined linearly

3. and c is piece-wise constant (between 0 and 1) since it is the contact invariance parameter

An important difference between our implementation and the program described in Mordatchs paper is that Mordatch used inverse kinematics with an analytic kinematic model. The hand we used does not have an analytic inverse kinematics model (and we opted not to try to find one since we wanted our implementation to be generic across all types of hand designs). Therefore we would have had to either put an optimization based IK solver in our program or modify the implementation a little bit. Putting an optimization based IK-solver into our program did not seem like a good idea, because we would then be putting an optimization process inside another optimization process, so for every run of the BFGS optimizer for the outer loop of the Mordatch objective function optimization, we would have to run an iterative IK solver for every keyframe pose in the inner loop, which would result in very long running times. Instead, we decided to treat the problem as one in which we have floating point contacts instead of full rigid bodies for contact in our optimization. These floating point contacts correspond to the palm and fingertips on our original hand model, and as will be shown can be used to dictate target positions for inverse kinematics after the proposed physical motion is synthesized. Therefore the output is not guaranteed to be totally physically realistic, but good enough to a reasonable approximation.

2 Cost Function for Optimization

The total cost is the sum of these cost components at each time-step in our simulation (obviously it is not a real physical simulation) over the length of the entire motion. The entire motion is set to take 1 second, and the cost function is sampled every .05 seconds. The decision to use floating point contacts does affect some of the terms in the objective function, which we will note below. To optimize we used BFGS with finite differences. The cost function is broken up as follows:

1. Contact-Invariant cost:
   $L_{CI} = \sum_j c_j(|e_j^O|^2 + |\dot{e}_j^O|^2 + |e_j^H|^2 + |\dot{e}_j^H|^2)$
   where $e_j^O = \pi_O(r_j) - r_j$ and $e_j^H = \pi_j(r_j) - r_j$
   $\pi_O(r)$ and $\pi_j(r)$ are the projections of r onto the surface of the body and the body corresponding to contact j (in this case the palm, a fingertip, or the ground) and e_j^O and e_j^H are the projection errors onto the object and the hand respectively: note that the proposed contact point does not have to lie on either the object or the hand per se, although in a correctly optimized, physically valid scheme these points would all coincide. To simplify projection Mordatch made all of the rigid bodies in his simulation into capsules to apply a relatively easy projection formula (including the rigid bodies in the hand). In our simulation the object is a capsule and everything except the palm is a capsule as well, however we are not able to carry out projection onto the rigid bodies in our hand because of our no-inverse kinematics simplification. To accurately project onto the rigid bodies in our hand, we would need the orientation of said rigid bodies, which would require inverse kinematics to be computed at each time step we sample in our motion, which we already decided would be too computationally costly given that we don't have an analytic model of the hand. Therefore, we replaced the projection onto the hand terms with squared difference to the end effector point (which should be a very good approximation anyway).

The only downside of this is that the object can't contact the hand anywhere else except the designated end effector points, but that is not an onerous restriction.

2. Physics Violation cost:
   $L_{physics}(s) = ||f_{tot} - \dot{P}||^2 + ||m_{tot} - \dot{L}||^2 + \lambda \sum_j ||f_j||^2$
   $f_{tot} = \sum_j c_j f_j + f_{ext}$
   $m_{tot} = \sum_j c_j f_j \times (r_j - x_O) + m_{ext}$
   $\dot{P}$ and $\dot{L}$ are change in linear and angular momentum, calculated via the difference between position and orientation of the object between sampled times. There are no external moments in our simulation, and the only external force is gravity. We set the regularization for forces to be relatively low to encourage search.

3. Friction cone constraint—did not include: Moratch included friction cone constraints, however this depends on the orientation of the end effector rigid bodies, which we would have to obtain from inverse kinematics

4. Finger pad constraints—did not include: this was a term Mordatch included to encourage objects to be gripped at the base of the fingertip, however this requires inverse kinematics to solve. It is also not a valuable term in the optimization (more of an add-on to make better motions)

5. Hard constraint for contact weights—-contact wieghts are constrained to be between 0 and 1, so we turned this into a quadratic soft constraint with a high penalty coefficient

6. Task Cost: the cost for moving the object to a specified orientation/position at a given keyframe in the motion—this is just a quadratic cost comparing desired position with calculated position (and orientation). For our simulation, we just wanted to move a capsule object from one point to another without regard to its orientation (i.e. we only have one task cost term)

7. Additional costs included:
   - **penetration depth cost:** for each active contact point ($c_i > 0$), if the point lies on the inside of the capsule being manipulated, penalize proportional to the penetration depth of the contact point with respect to the object
   - **kinematic constraint cost for floating point contacts:** since we made the no-inverse kinematics modification described above, we still need to constrain the placement of the contact points so that they fall within the workspace of the given hand: for this, since the contact points just represent the palm and fingertips, we enforce a quadratic constraint that penalizes (heavily) whenever the distance between a fingertip and the palm exceeds a known maximum limit—-this should be good enough to guarantee an IK solution for a fully actuated hand design

3 Demonstration of output

See attached video

Notes on video/explanation: The task objective is just to move the object to a specified point by the end of the manipulation (ignoring orientation). That part of the optimization is satisfied (the task cost is 0). The physics cost and the contact invariant cost are not satisfied though. The floating end effector points (in yellow) are the points corresponding to the center of the palm and the centers of the fingertips. The blue points are the proposed locations of the contact points on the object. As is evident in the demonstration, these points are not converging to either the yellow points or the surface of the object, which they should do if the contact point is active (positive c_i). The first part of the animation shows what the optimization comes up with (calculated positions, orientations, forces, etc.) and the second part tests the physical validity of the motion by just applying forces to the object in a physical simulator. If the generated motion is physically valid, then the second simulation should be roughly the same as the first simulation—that is not the case, indicating that the optimizer didn't converge to a solution.

(Stefanos) The code is attached, however you won't be able to run it since it is dependent on some extensive libraries that you probably don't want to bother downloading. We've included the relevent files for you to look through. By the way, ignore the hand model in the demo, it is just the hand model we used to generate end effector points, but past that it isn't relevent to the simulation (only the floating contact points are). In the first simulation the object position is set by the results of the optimization so the hand object has no effect on the object (just the floating contact points), in the second where we test physical validity, we just move the hand out of the way so it doesn't interfere in the physical simulation.

4 Comments on difficulties/why it didn't work

The most important part of the optimization was the physics term for solving force balancing and moment balancing equations on the object (force and moment balancing arent needed for the hand obviously since it is assumed to be fully actuated). Since this wasnt being satisfied in our earlier experiments, we increased the weight on the physics constraint to essentially make it a hard constraint. This backfired: it caused the contact invariant terms c_i to all go to 0 except for the contact point with the groundessentially it minimized the cost by pretending that it could just raise the ground with the object and accept the penalty from doing so.

That led us to try removing contact points with the ground altogether (so gravity would still remain in the force balancing equations for the object, but the ground would no longer be able to provide a contact force). This did not solve the problem: instead it caused the c_i's to not all be driven to 0, and had some of them stay at about .5, but for whatever reason it still didn't satisfy the force balancing equation.

We tried varying the initialization values of the c_i's as well but that also didnt turn up anything as to why the optimizer wasnt working: basically all of the c_i's are initialized to .5 (rather than 0 or 1)

The most important problem that is occurring is that the contact points aren't converging to the object (the capsule): although there is a penalty for distance between the proposed contact point and its projection onto the object when the contact is active, the optimizer seems unable to move the contact point any closer to the object. We thought that this might have something to do with penalization for penetration depth (contact points being inside the object), but varying the penetration penalty didn't change the results noticeably.

We've gone through the code several times and can't find any obvious errors, so we believe that either there is a non-obvious error lurking in the code, the changes we made to the methodology from the original paper had unintended consequences, or we are stuck at a local minimum and our weights in the objective function are not well-tuned. At this point, we would need to start from a simple example again and iteratively build up the optimization almost from scratch.