Imitating Humans

D.A. Forsyth

Points

- Getting poses right is important (seen motion VAE)
- Predicting batches might be better long term predictor
- LowD+Contact improves human 3D recons
- Quite simple objectives seem helpful at shaping motion
  - Quite good motions can be learned from bounds
- Motion graphs can be quite bad
  - interpolating between frames with same contacts helps
  - Linear interpolations between frames with same contacts is good physically

Motion VAE

- Synthesize x_i+1 conditioned on x_i
  - which yields an autoregressive model

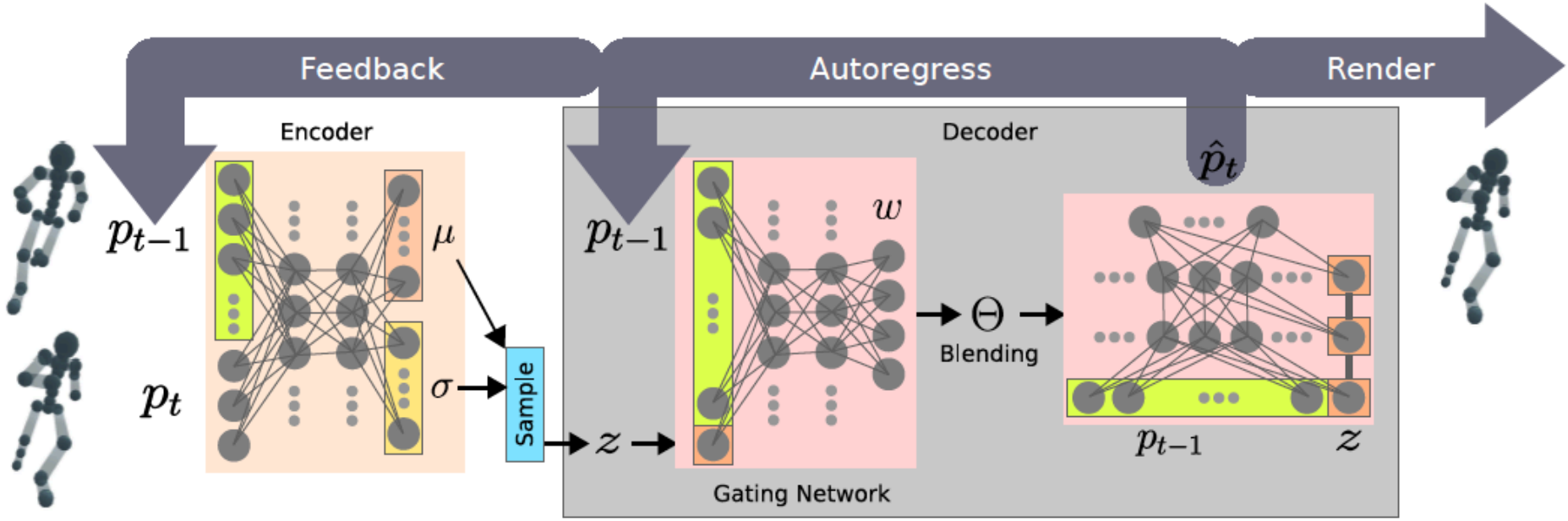


Fig. 2. The conditional VAE has two parts. The encoder takes past (p_{t-1}) and current (p_t) pose as input and outputs both μ and σ, which is then used t sample a latent variable z. The decoder uses p_{t-1} and z to reconstruct $\hat{p}_t$. For the decoder, we use a MANN-style mixture-of-expert neural network. Whe using scheduled sampling during training or at run-time, the decoder output, $\hat{p}_t$, is fed back as input for generating the next prediction.

Ling et al 20

Motion VAE - making long time motions

- Autoregressive model on its own isn't much good
  - drift
- Use RL to produce latent variables
- Thoroughly enjoyable demo at
  - https://belinghy.github.io/projects/MVAE/
- Note important point:
  - locomotion has really quite simple structure, viewed right

Motion graph evaluations

- Summary:
  - quality (equiv reachability) degrades fast with complex spatial environments
  - fixing by brute force leads to very big graphs fast
  - response to control input is poor except in easy cases

Reitsma + Pollard, 07

Motion in “chunks”

- Motion VAE, Motion graph predict next frame
- Perhaps better to think in sequences or “chunks”
- Evidence:
  - Simple segmentations of motions are easy
  - Linear blends of chunks are quite successful
  - Some MG encodings are already chunks

Simple segmentations

- Break motions into sequences by
  - acceleration peaks (old, OK)
  - change of contact (more recent, better)

In our work, as in most other approaches to interpolation, we automatically locate these key frames at changes in the contact with the environment because the physical laws governing the motion change with contact. Motions $M_1, M_2,...,M_k$ are split into phases based on these key frames and the corresponding phases are interpolated. For example, a jumping motion would consist of three phases: lift-off, flight and landing. Additional key frames can be added during long contact phases to better align the motions without violating the assumptions behind our analysis.

Safonova +Hodgins 05

Linear blends of chunks

- Take two corresponding phases (eg flight, previous slide)
  - timescale
  - interpolate linearly

We compute each phase of motion M by interpolating corresponding phases of motions $M_1, M_2,...,M_k$ with a constant set of weights, $w_1, w_2,...,w_k$:

$$M = w_1 M_1 + w_2 M_2 + ... + w_k M_k \quad (1)$$

where $\sum_{i=1}^{k} w_i = 1$. The analysis in this paper assumes that the weights sum to one so our results are limited to interpolation and do not generalize to extrapolation. The analysis is presented for interpolation of only two motions, M_1 and M_2 but generalizes to the interpolation of k motions because equation 1 can be recursively computed by interpolating two motions at a time. The weights for each interpolation sum to 1 and the final interpolation produces a motion with the weighting given in equation 1.

Consider a particular phase F. At each time t of that phase we compute motion $M(t,w)$ as follows:

$$M(t,w) = \begin{cases} P_{root}(t) = wP_{1root}(t_1) + (1-w)P_{2root}(t_2) \\ Q_i = wQ_{1i}(t_1) + (1-w)Q_{2i}(t_2), \quad \text{for } i = 1..n \end{cases} \quad (2)$$

where $w = 0..1$ is the interpolation weight, T_1, T_2 and T are the time of phase F in motions M_1, M_2 and M respectively,

Root position

Root orientation, Joint angles

Safonova +Hodgins 05

But….

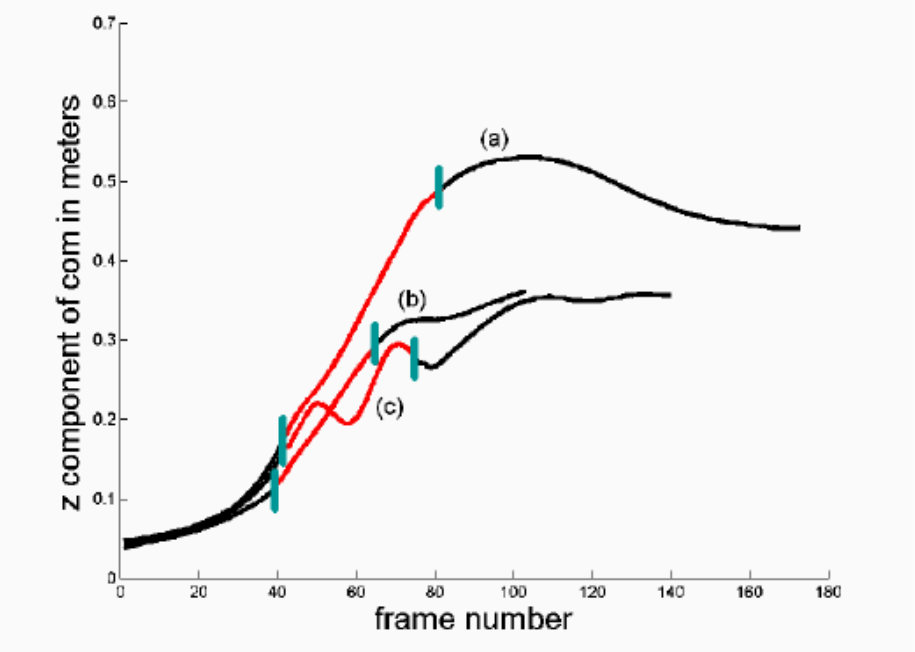


Figure 1: *The Z component of the trajectory of the center of mass for: (a) forward jump with no turn (motion M_1); (b) forward jump with 360 degree turn (motion M_2); (c) the motion that results from interpolating motions M_1 and M_2. Vertical bars are used to indicate the beginning and ending of the flight phase for each motion. The trajectory of the center of mass of the interpolated motion during flight is not a straight line as it should be.*

This is a consequence of interpolating joint angles: the locations of masses are not a linear function of joint angles, so…

If, instead, you interpolated joint positions, the joint lengths might change.

Safonova +Hodgins 05

Better linear blends of chunks

- Take two corresponding phases (eg flight, previous slide)
  - timescale
  - interpolate linearly

from the new center of mass position and joint angles (see Appendix A). The interpolation equation is now:

$$M(t,w) = \begin{cases} P_{com}(t) = wP_{1com}(t_1) + (1-w)P_{2com}(t_2) \\ Q_i(t) = wQ_{1i}(t_1) + (1-w)Q_{2i}(t_2), \text{ for } i = 1..n \\ P_{root}(t) = F(P_{com}(t), Q(t)) \end{cases} \quad (4)$$

where F is the function that computes the root position from the center of mass and the joint angles. With this small

Center of mass position

Joint angles

Recover root from COM, angles

Safonova +Hodgins 05

You need to handle time right, too

In the literature, the time of an interpolated motion has generally been computed as: $T = wT_1 + (1-w)T_2$. But setting time in this way results in scaling gravity by:

$$\frac{wT_1^2 + (1-w)T_2^2}{(wT_1 + (1-w)T_2)^2} \tag{8}$$

In many cases this error will be small and will not be noticeable. Reitsma and Pollard [RP03] determined that if gravity is between -9.0 and -12.7 the error is not visible to the human observer.

Instead, use

$$T = \sqrt{T_1^2 w + T_2^2 (1-w)} \tag{7}$$

Safonova +Hodgins 05

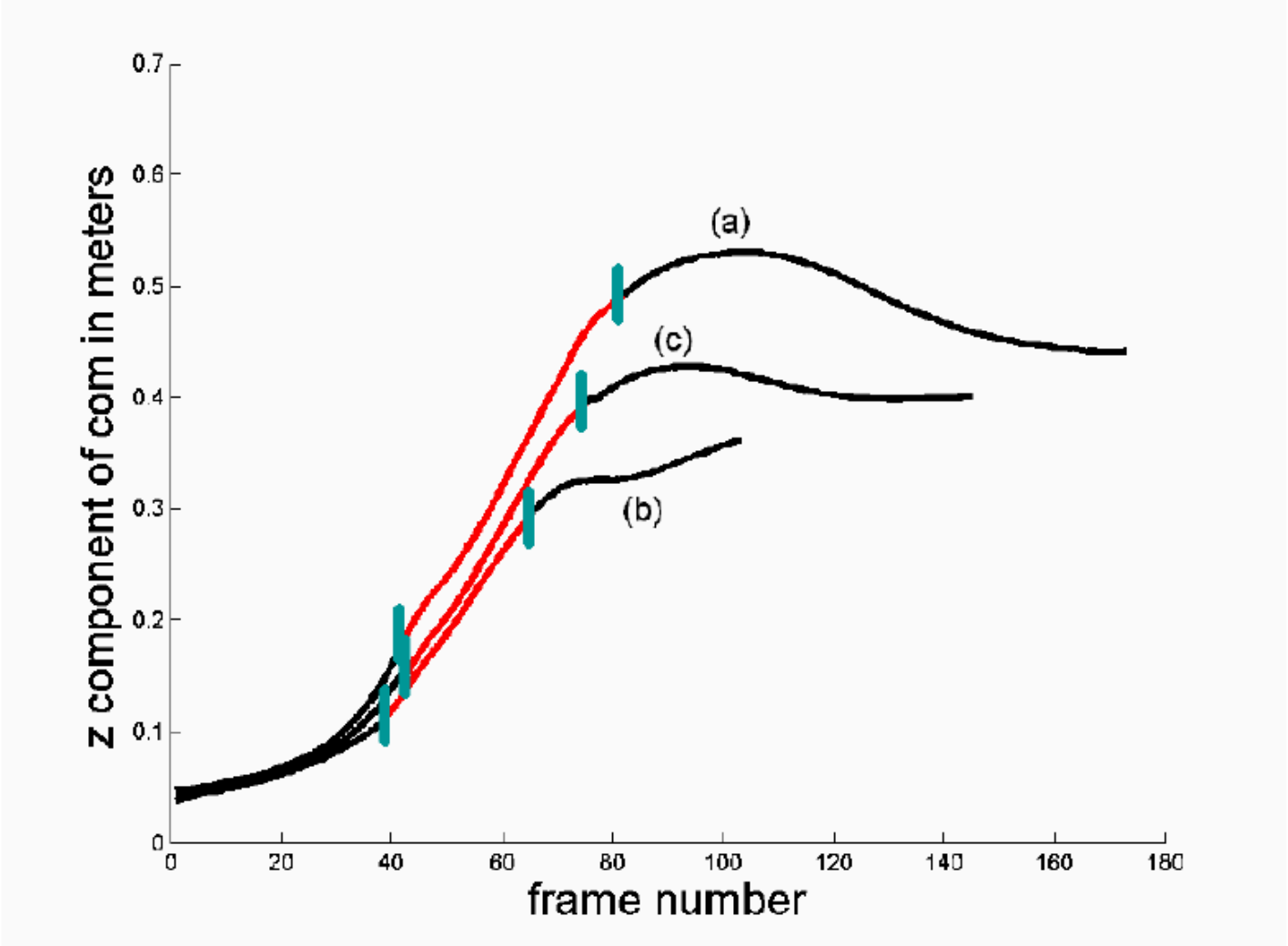


Figure 2: *Example from figure 1 but with the flight phase of the interpolated motion computed by interpolating the center of mass positions of the input motions instead of the root positions and with the time of the flight phase computed as* $T = \sqrt{T_1^2 w + T_2^2 (1 - w)}$.

Safonova +Hodgins 05

S+H N+Q

- Important point:
  - parametric blends can be manipulated safely, IF properly defined
- To know:
  - Handling a stance motion like this works, too
  - There's more detailed material on scaling
- Q:
  - recorded z isn't a perfect straight line in flight - why?
  - are there other segmentation criteria that are important?
  - note the segmentation scale is quite short
    - flight doesn't last long, contacts are continually changing
  - should one segment at longer scales, too?

Interpolated motion graph

- Rather natural consequence
- Key points:
  - the quality of search is important for the quality of generated motions
  - you can use the interpolation idea to build a richer graph
    - which allows optimal (or near optimal anytime) search

Figure 1: Optimal and greedy solutions for walking a given distance and for picking up an object.

Safonova + Hodgins 07

Interpolated motion graph

- Construct correspondences between frames
  - which allow interpolation, as above
- Now make two, interpolatable paths
  - and interpolate
- This requires considerable detail to ensure
  - graph can be represented
  - paths can be constructed
  - A* yields optimal paths
  - etc.

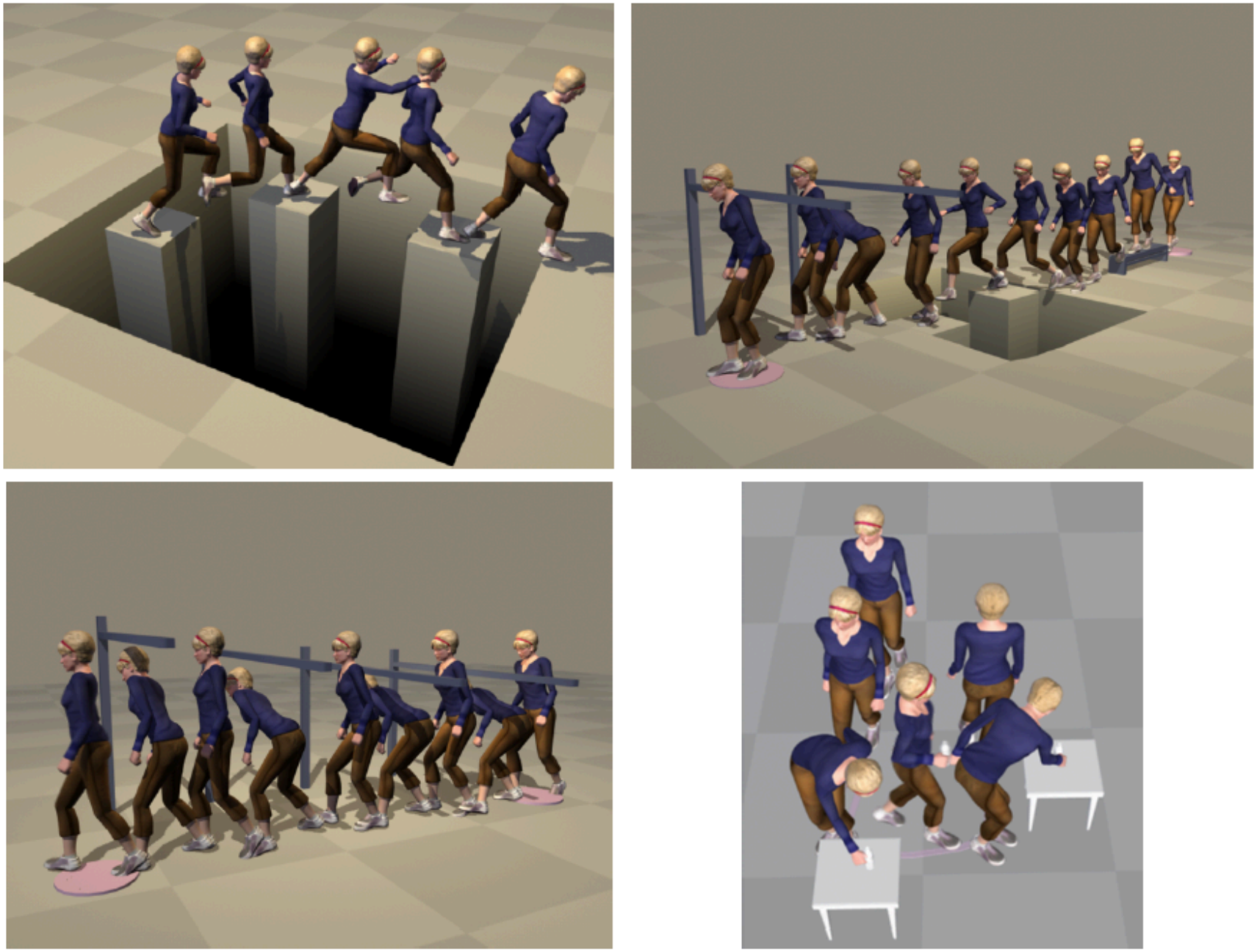


Figure 10: Synthesized motion

Safonova + Hodgins 07

Interpolation helps

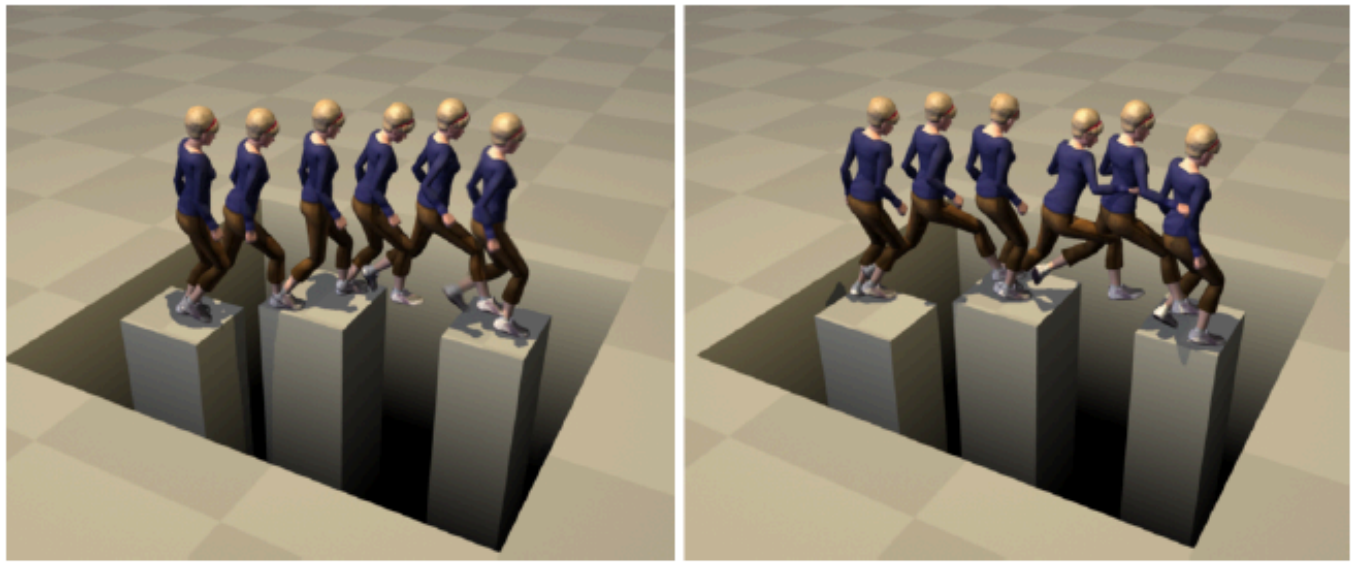


Figure 12: (Left Image) The optimal solution with interpolation. The character walks across the first column and jumps across the second column. (Right Image) The optimal solution without interpolation. The character jumps between each pair of columns. Because the behaviors change between the two solutions, we cannot warp the solution found without interpolation to match the more efficient solution found with interpolation.

Safonova + Hodgins 07

S+H, II, N+Q

- Important point:
  - you can assemble long-scale motions out of parametric blend places
- Q:
  - mostly as above
  - could one predict batches from batches?
    - cf Motion VAE frames from frames
    - YES

Predicting batches

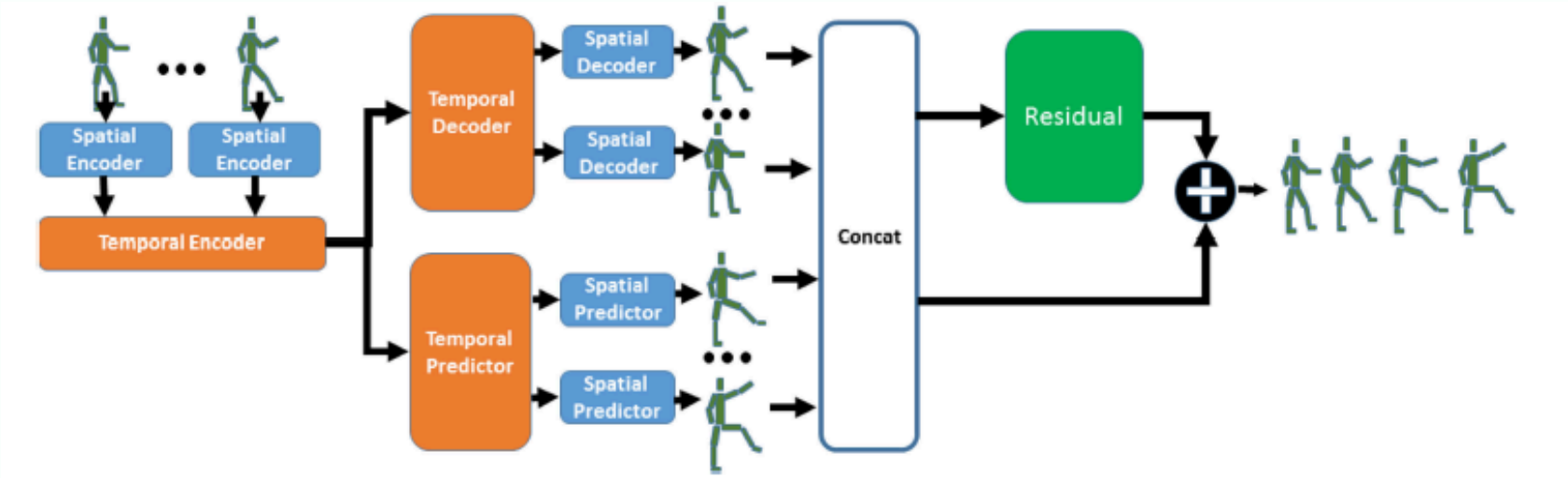


Fig. 2. The network architecture of the Spatio-temporal Recurrent Neural Networks. Detailed network structures of the Temporal Encoder/Decoder/ Predictor and the Spatial Encoder/Decoder/Predictor can be found in Figs. 3 and 4 respectively.

We start by parameterizing the motion manifold as a time series: $P(X_{t+n}, \ldots, X_{t+1} | X_t, \ldots, X_{t-m})$ where X_t is the motion frame at time t and P is the conditional probabilistic distribution of n frames from $t+1$ given $m+1$ frames before $t+1$. What the model captures is the dependencies between the past $m+1$ frames and the future n frames. Many existing data-driven models fall under this umbrella. Most of them consider the situation when $n = 1$ and $m = 0$, such as the motion graphs [2] and autoregression [12]. Some consider

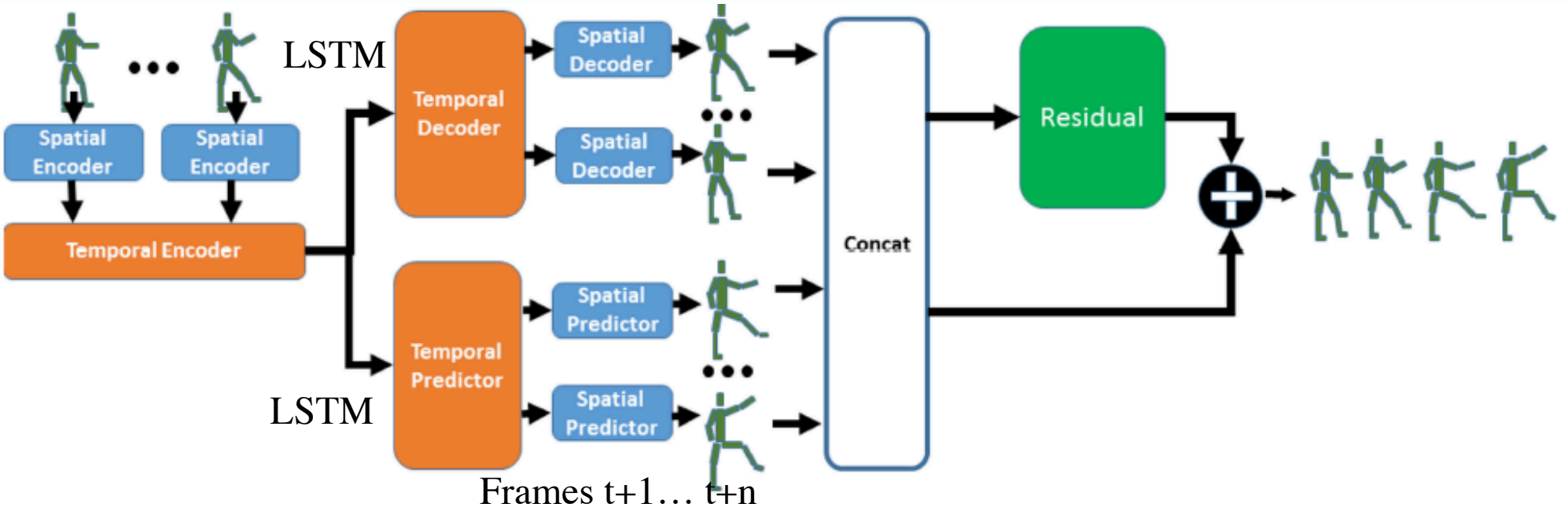
Wang ea, 21

Predicting batches

Frames t… t-m

Frames t… t-m

This appears to be some sort of temporal filter, 220c1



Frames t+1… t+n

Fig. 2. The network architecture of the Spatio-temporal Recurrent Neural Networks. Detailed network structures of the Temporal Encoder/Decoder/Predictor and the Spatial Encoder/Decoder/Predictor can be found in Figs. 3 and 4 respectively.

Wang ea, 21

Losses

4.2.1 Reconstruction Error

We use Mean Squared Error (MSE) for C_r to force STRNN to reconstruct motions. $C_r = C_d + C_p$

$$C_d = \frac{1}{m}\sum \|(\Sigma_e^{-1}(\Phi_d(\Phi_e(\Sigma_e(X_e)))) - X_e)\|^2 \quad (5)$$

$$C_p = \frac{1}{n}\sum \|(\Sigma_e^{-1}(\Phi_p(\Phi_e(\Sigma_e(X_e)))) - X_p)\|^2, \quad (6)$$

where C_d and C_p are the reconstruction loss of the decoded and predicted motions. m and n are the decoding and prediction lengths, $X = \{X_e, X_p\}$, X_e and X_p are the ground-truth decoding/predicted motions, and Σ_e^{-1} is simply the inverse function of Σ_e. Minimizing C_r results in a tight approximation of the motion manifold. We will denote this cost as *MSE* in Section 5.

4.2.2 Long-horizon (LH) Cost

C_s in Equation (4) is smoothness cost

$$C_s = \frac{1}{m+n}\sum \|\hat{X}^{t+1}_{body} - 2\hat{X}^{t}_{body} + \hat{X}^{t-1}_{body}\|^2 + \sum \|\hat{X}^{t}_{root} - \hat{X}^{t-1}_{root}\|^2, \quad (7)$$

where m and n are the decoding and prediction lengths, $\hat{X}$ is the concatenation (along the time axis) of the decoded and predicted motions, C_s governs the smoothness of the motions and has been widely used in many optimization-based character animation approaches. Note this constraint essentially penalizes big accelerations only and does not overly dampen the motion dynamics. We will denote it as *long horizon constraint (LH)* in Section 5.

- Evaluation
  - prediction error
  - quantitative

Wang ea, 21

N+Q

- Qualitatively, rather good motions
  - movie at https://www.youtube.com/watch?v=1eZxWkLj1lg
  - quite good control of footskate without postprocessing
  - BUT occasional stop and turn stuff -temporal structure is weird
- Q:
  - Prediction error is likely a very poor evaluation method
    - what is better?
  - How do they get diversity?
    - long batches have much greater diversity than frames
    - I don't think they do…
  - How could one get diversity?
    - just injecting random numbers is quite unreliable
    - what about ditching direct training loss and using an adversary?

See "DynamicFutureNet" - but not sure how this works?

Knowing some animation useful for vision

- (Finally!)
-

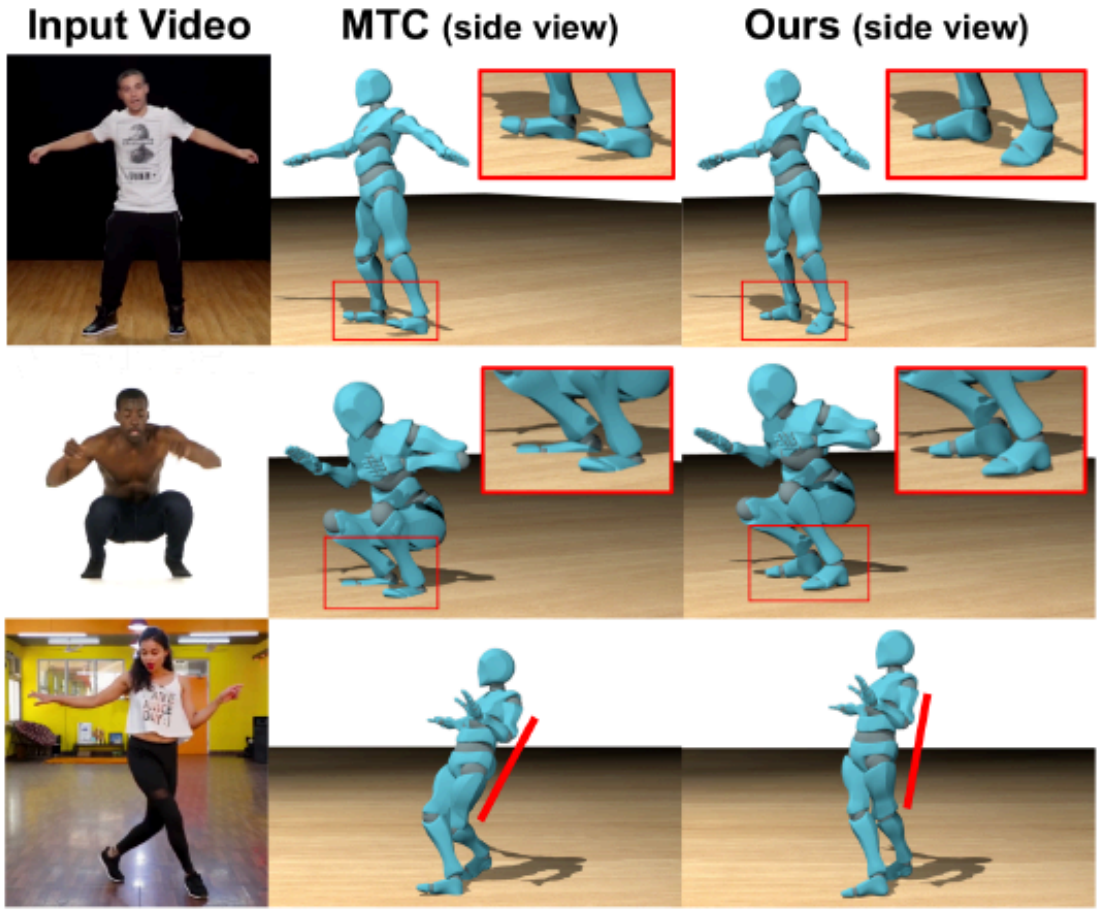


Fig. 1. Our contact prediction and physics-based optimization corrects numerous physically implausible artifacts common in 3D human motion estimations from, e.g., Monocular Total Capture (MTC) [47] such as foot floating (top row), foot penetrations (middle), and unnatural leaning (bottom).

Rempe et al 20

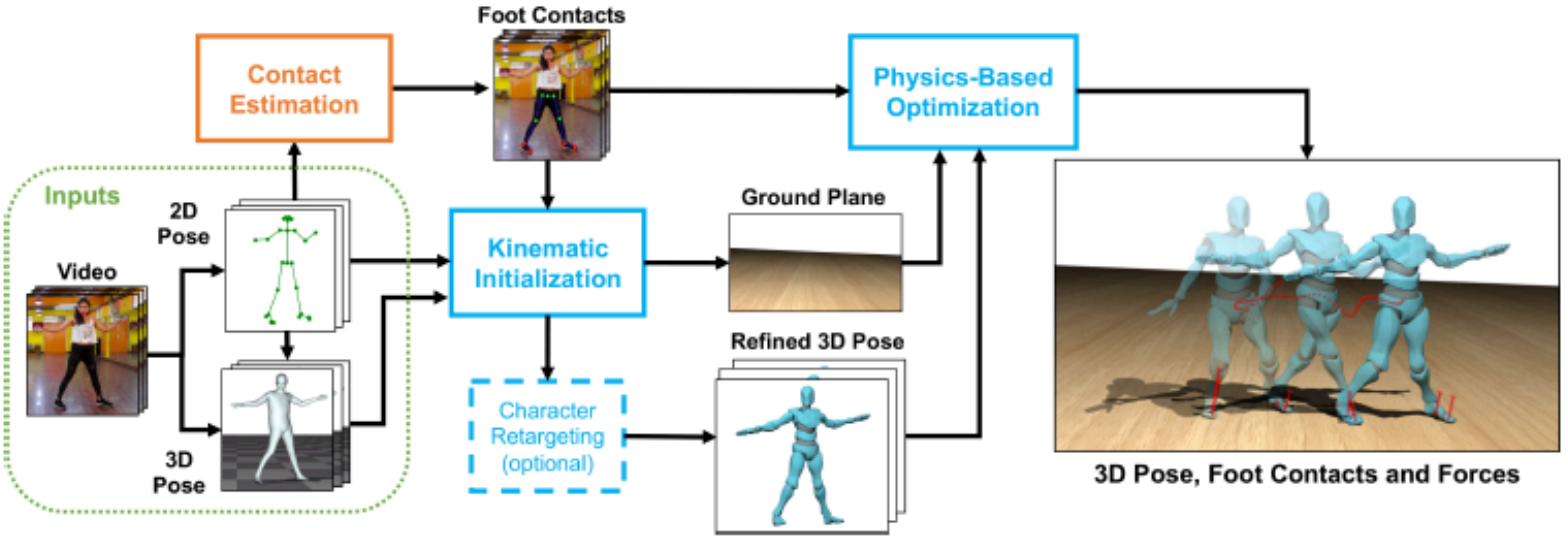


Fig. 2. Method overview. Given an input video, our method starts with initial estimates from existing 2D and 3D pose methods [4,47]. The lower-body 2D joints are used to infer foot contacts (orange box). Our optimization framework contains two parts (blue boxes). Inferred contacts and initial poses are used in a kinematic optimization that refines the 3D full-body motion and fits the ground. These are given to a reduced-dimensional physics-based trajectory optimization that applies dynamics.

Rempe et al 20

Table 2. Physical plausibility evaluation on synthetic test set. *Mean/Max GRF* are contact forces as a proportion of body weight; see text for discussion of plausible values. *Ballistic GRF* are unexplained forces during flight; smaller values are better. Foot position metrics measure the percentage of frames containing typical foot contact errors per joint; smaller values are better.

	Dynamics (Contact forces)			Kinematics (Foot positions)		
Method	**Mean GRF**	**Max GRF**	**Ballistic GRF**	**Floating**	**Penetration**	**Skate**
MTC [47]	143.0%	9055.3%	115.6%	58.7%	21.1%	16.8%
Kinematics (ours)	124.4%	1237.5%	255.2%	**2.3%**	2.8%	**1.6%**
Physics (ours)	**99.0%**	**338.6%**	**0.0%**	8.2%	**0.3%**	3.6%

Fewer frames with bad forces or foot penetration/skate/float

Table 3. Pose evaluation on synthetic and HumanEva-I walking datasets. We measure mean global per-joint 3D position error (no alignment) for feet and full-body joints. For full-body joints, we also report errors after root alignment on only the first frame of each sequence. We remain competitive while providing key physical improvements.

	Synthetic Data			HumanEva-I Walking		
Method	**Feet**	**Body**	**Body-Align 1**	**Feet**	**Body**	**Body-Align 1**
MTC [47]	581.095	**560.090**	**277.215**	511.59	532.286	**402.749**
Kinematics (ours)	573.097	562.356	281.044	**496.671**	525.332	407.869
Physics (ours)	**571.804**	573.803	323.232	508.744	**499.771**	421.931

Rempe et al 20

3D recon not much worse, sometimes better

Concept

- Notice motion VAE concept:
  - make frames, use RL controller to make paths
- We can
  - make human frames
  - make chunks of human motion
  - But joining them up is iffy
- Claim:
  - Assume we have all the frames/chunks we need, but can't join up
  - We don't need RL (don't know loss)
  - What we need is IRL or imitation learning
    - We have lots of observed real motion data
      - which is the result of a human controller joining up frames/chunks
    - Use this to impute "join up" cost/policy/etc.

Reinforcement Learning: Learning policies guided by sparse rewards, e.g., win or not the game.

- Good: simplest, cheapest form of supervision
- Bad: High sample complexity

Where is it successful so far?

- in simulation, where we can afford a lot of trials, easy to parallelize
- not in robotic systems:
  1. action execution takes long
  2. we cannot afford to fail
  3. safety concerns

Crusher robot

Fragkiadaki, ND

Ideally we want dense in time rewards to closely guide the agent closely along the way.

Who will supply those shaped rewards?

1. We will manually design them: *"cost function design by hand remains one of the 'black arts' of mobile robotics, and has been applied to untold numbers of robotic systems"*
2. We will learn them from demonstrations: *"rather than having a human expert tune a system to achieve desired behavior, the expert can demonstrate desired behavior and the robot can tune itself to match the demonstration"*

Fragkiadaki, ND

Learning from demonstrations a.k.a. Imitation Learning:

Supervision through an expert (teacher) that provides a set of demonstration trajectories: sequences of states and actions.

Imitation learning is useful when is easier for the expert to demonstrate the desired behavior rather than:

a) coming up with a reward that would generate such behavior,

b) coding up the desired policy directly.

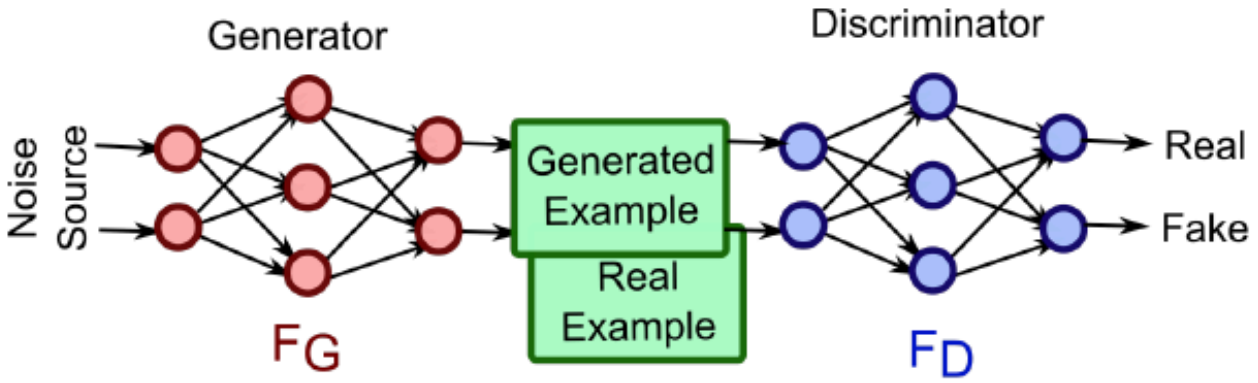
Fragkiadaki, ND

The Imitation Learning problem

The agent (learner) needs to come up with a policy whose resulting state, action trajectory distribution matches the expert trajectory distribution.

Does this remind us of something…?

GANs! Generative Adversarial Networks (on state-action trajectories)



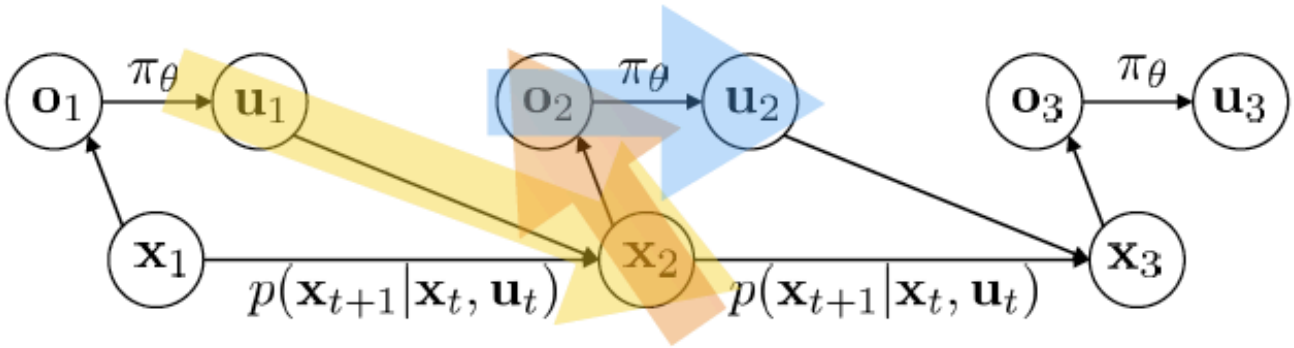
Fragkiadaki, ND Generative Adversarial Networks, Goodfellow et al. 2014

The Imitation Learning problem: Challenge

Actions along the trajectories are interdependent, as actions determine state transitions and thus states and actions down the road.

interdependent labels -> structure prediction

Action interdependence in time:



Algorithms developed in Robotics for imitation learning found applications in structured predictions problems, such as, sequence generation/labelling e.g. parsing.

Imitation Learning

For taking this structure into account, numerous formulations have been proposed:

- Direct: Supervised learning for policy (mapping states to actions) using the demonstration trajectories as ground-truth(a.k.a. behavior cloning) + ways to handle the neglect of action interdependence.

- Indirect: Learning the latent rewards/goals of the teacher and planning under those rewards to get the policy, a.k.a. Inverse Reinforcement Learning (next lecture)

Experts can be:

- Humans

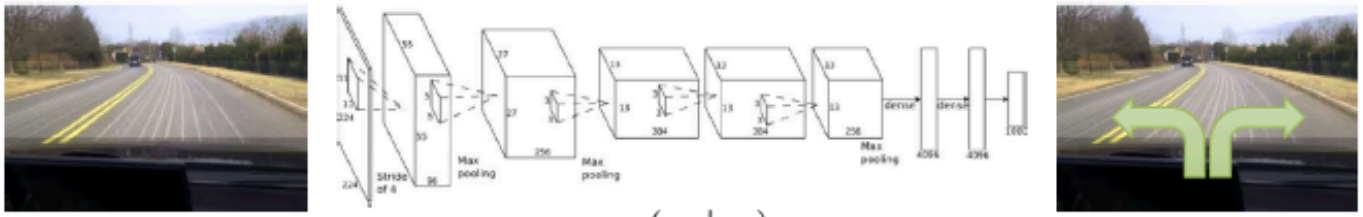
- Optimal or near Optimal Planners/Controllers

Fragkiadaki, ND

Direct possibilities

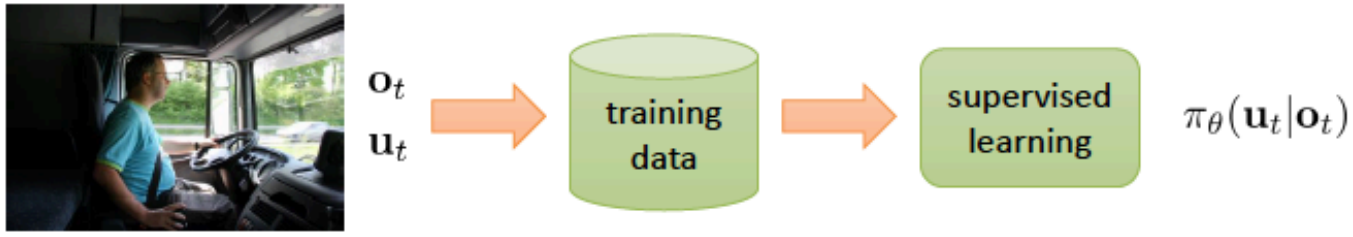
Imitation Learning as Supervised Learning

Driving policy: a mapping from (history of) observations to steering wheel angles



Behavior Cloning=Imitation Learning as Supervised learning

- Assume actions in the expert trajectories are i.i.d.
- Train a classifier or regressor to map observations to actions at each time step of the trajectory.



Fragkiadaki, ND

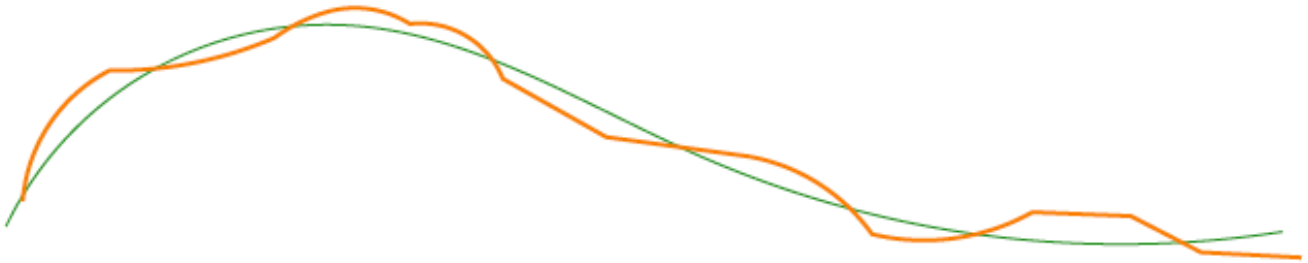
End to End Learning for Self-Driving Cars, Bojarski et al. 2016

Classifier or regressor?

Because multiple actions u may be plausible at any given observation o, policy network $p_{\pi_\theta}(u_t|o_t)$ usually is not a regressor but rather:

- A classifier (e.g., softmax output and cross-entropy loss, after discretizing the action space)
- $J(\theta) = -\sum_{i=1}^{m}\sum_{k=1}^{K} 1_{y(i)=k} \log[P(y_{(i)} = k|x_{(i)};\theta)]$
- A GMM (mixture components weights, means and variances are parametrized at the output of a neural net, minimize GMM loss, (e.g., Hand writing generation Graves 2013)
- A stochastic network (previous lecture)

Independent in time errors



error at time t with probability ε

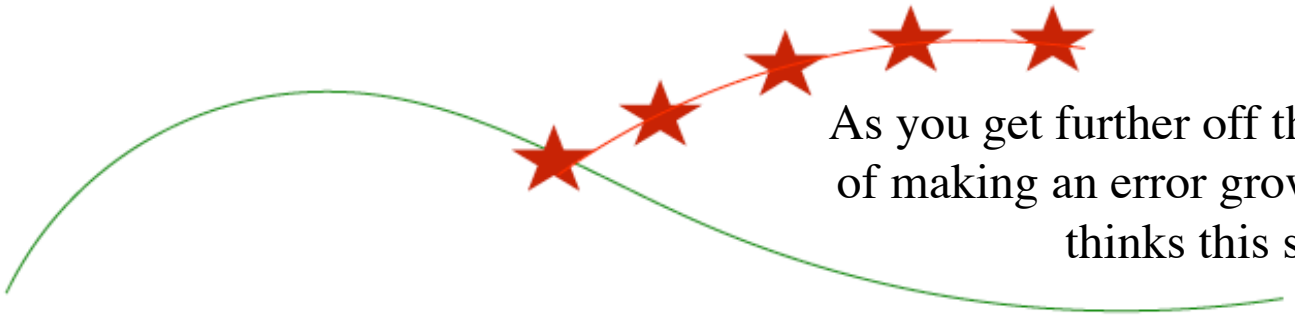
$E[\text{Total errors}] \lesssim \varepsilon T$

Fragkiadaki, ND

Important difference

- Driving:
  - mostly, goal is obvious
- Motion:
  - you can choose many frames/chunks given one
  - some form of latent variable is required to explain choice
    - how?
  - this is likely some summary of long scale goal
    - eg desired path; desired endpoint; etc

Compounding Errors



As you get further off the path, the probability of making an error grows, cause the classifier thinks this state is rare

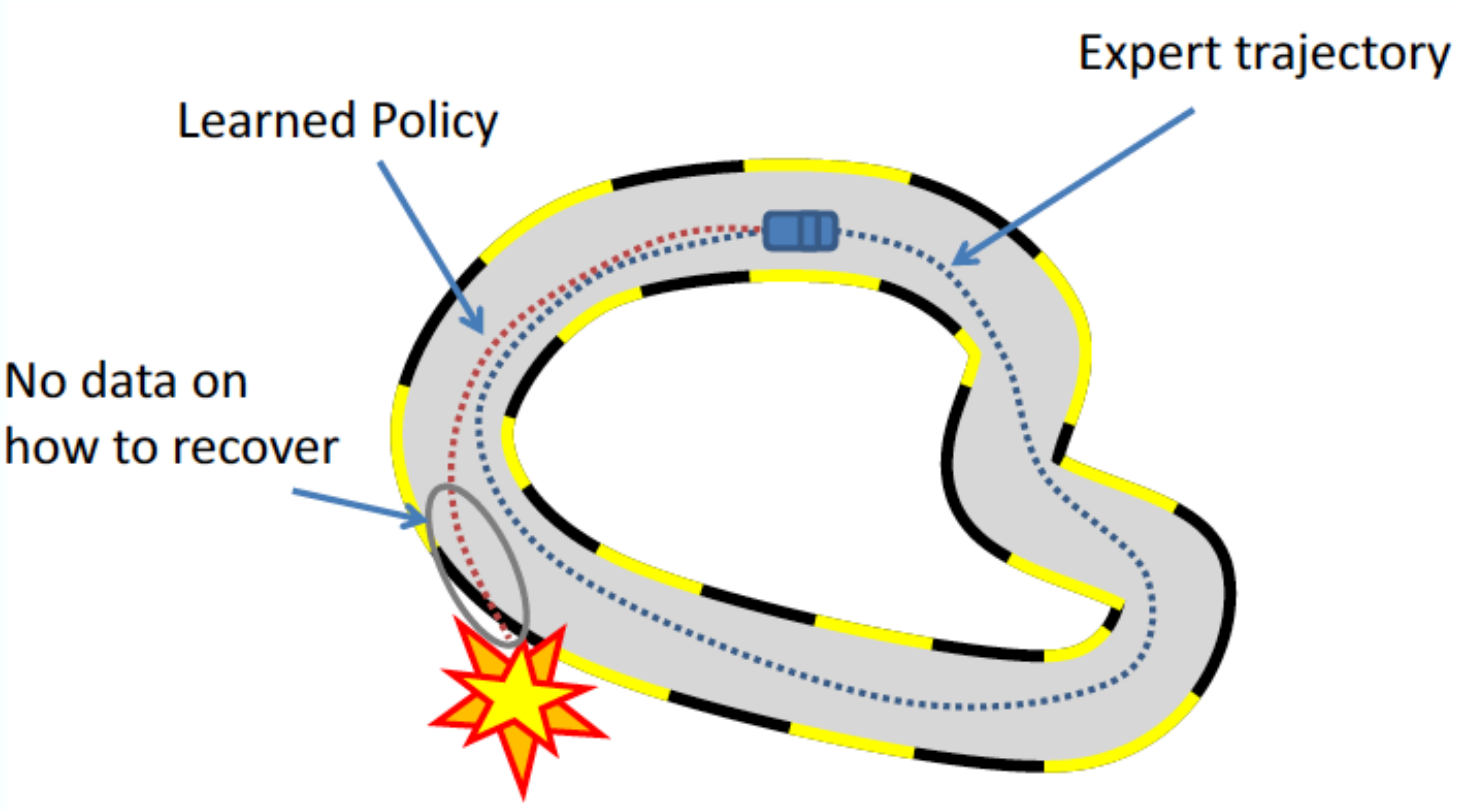
error at time t with probability ε

$E[\text{Total errors}] \leq \varepsilon(T + (T-1) + (T-2) + \ldots + 1) \propto \varepsilon T^2$

Fragkiadaki, ND A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011

Data Distribution Mismatch!

$$p_{\pi^*}(o_t) \neq p_{\pi_\theta}(o_t)$$



Fragkiadaki, ND

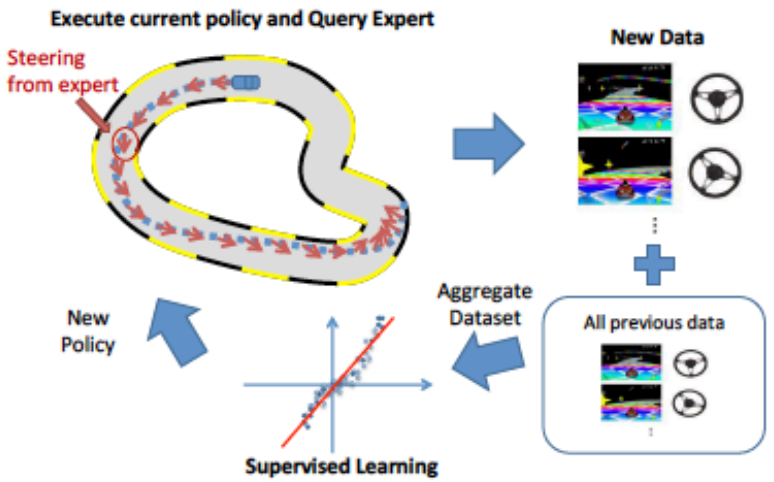
Interdependence might not matter (much)

- Likely much easier to recover from an off-policy move
  - motion “glitch”
  - no “perception”
    - latent variable changes from frame/chunk to frame/chunk

DAGGER (in simulation)

Dataset AGGregation: bring learner's and expert's trajectory distributions closer by labelling additional data points resulting from applying the current policy

1. train $\pi_\theta(u_t|o_t)$ from human data $\mathcal{D}_{\pi^*} = \{o_1, u_1, ..., o_N, u_N\}$
2. run $\pi_\theta(u_t|o_t)$ to get dataset $\mathcal{D}_\pi = \{o_1, ..., o_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions u_t
4. Aggregate: $\mathcal{D}_{\pi^*} \leftarrow \mathcal{D}_{\pi^*} \cup \mathcal{D}_\pi$
5. GOTO step 1.



Problems:

- execute an unsafe/partially trained policy
- repeatedly query the expert

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011

DAGGER (in simulation)

Dataset AGGregation: bring learner's and expert's trajectory distributions closer by labelling additional data points resulting from applying the current policy

1. train $\pi_\theta(u_t|o_t)$ from human data $\mathcal{D}_{\pi^*} = \{o_1, u_1, ..., o_N, u_N\}$
2. run $\pi_\theta(u_t|o_t)$ to get dataset $\mathcal{D}_\pi = \{o_1, ..., o_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions u_t
4. Aggregate: $\mathcal{D}_{\pi^*} \leftarrow \mathcal{D}_{\pi^*} \cup \mathcal{D}_\pi$
5. GOTO step 1.

Notice you might not actually need a human here - if your states are discretized, and you have enough data, you might get this by matching

Problems:

- execute an unsafe/partially trained policy
- repeatedly query the expert

Fragkiadaki, ND A Reduction of Imitation Learning and Structured Prediction

Further algorithmic possibilities

- Aggrevate
- Aggrevated

- rough sketch in linked movies

Indirect possibilities

Inverse Reinforcement Learning

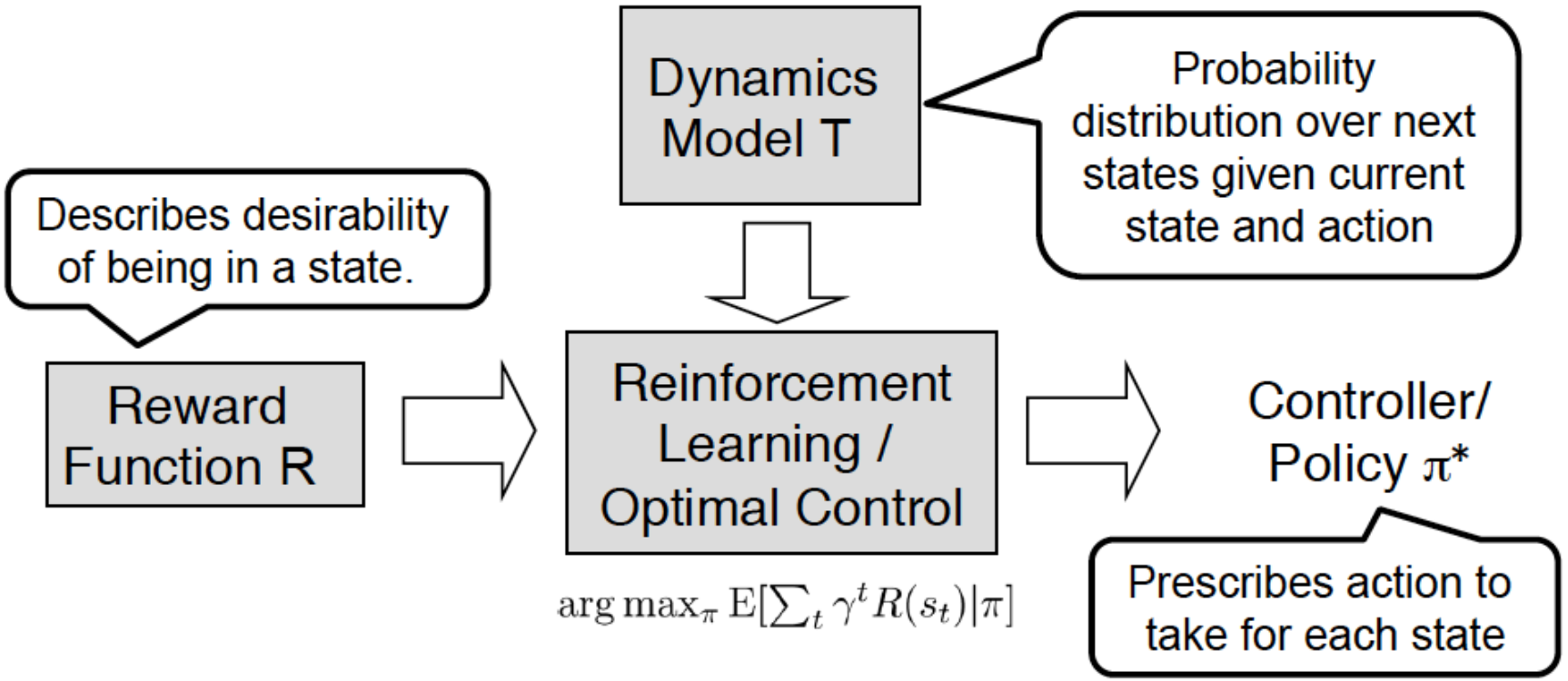


Diagram: Pieter Abbeel

Actually, we don't really have pi; we have observations of what happens under pi, which is not quite the same thing

Given π, let's recover R!

Problem Setup

- **Given**:
  - State space, action space
  - *No* reward function
  - Dynamics (sometimes) $T_{s,a}[s_{t+1}|s_t, a_t]$
  - Teacher's demonstration:

    $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$
    $(=$ trace of the teacher's policy $\pi^*)$

- **Inverse RL**
  - Can we recover R?
- **Apprenticeship learning via inverse RL**
  - Can we then use this R to find a good policy?
- **Behavioral cloning (*previous*)**
  - Can we directly learn the teacher's policy using supervised learning?

This is really like structured prediction

Strategy for structured prediction

- Construct a parametric cost function $H(\mathcal{X}, \mathcal{Y}; \theta)$
- So that, for training X*

$$\underset{\mathcal{Y}}{\operatorname{argmin}} \; H(\mathcal{X}^*, \mathcal{Y}; \theta)$$

- is close to correct Y*
  - (see movies for some details on construction)

Fragkiadaki, ND

For sequences

- Some natural choices
  - cost function:

$$V(x_1, y_1; \theta) + E(y_1, y_2; \theta) + V(x_2, y_2; \theta) + E(y_2, y_3; \theta) + \ldots$$

    - we want to find best y for given x
      - easily done with dynamic programming
  - Make V, E linear in theta
    - might involve complicated feature constructions
    - BUT simplifies learning

This yields

- The cost function has the form

$$\mathcal{H}(x, y; \theta) = \theta^T G(x, y)$$

- Choose theta so that for all training pairs x*, y*

$$\theta^T G(x^*, y^*) \leq \theta^T G(x^*, y)$$

- Note
  - this isn't one inequality - it's one inequality per possible y!
  - also, likely not feasible
  - also, doesn't prefer y's that are "close" to y*

So rearrange inequalities

- Force G(x*, y) to grow:

$$\theta^T G(x^*, y^*) + \epsilon D(y, y*) \leq \theta^T G(x^*, y)$$

- Rearrange, slack variable, and deal with many y:

$$\xi = (\max(0, \max_y \theta^T (G(x^*, y^*) - G(x^*, y)) + \epsilon D(y, y^*)$$

And now solve optimization problem

Don’t choose large theta - this helps generalization

$$\frac{1}{2}\theta^T\theta + \sum_i \xi_i$$

$$\xi_i = (\max(0, \max_y \theta^T(G(x^*{}_i, y^*{}_i) - G(x^*{}_i, y)) + \epsilon D(y, y^*{}_i)$$

Which is much nastier than it looks

Don't choose large theta - this helps generalization

$$\frac{1}{2}\theta^T\theta + \sum_i \xi_i$$

$$\xi_i = (\max(0, \max_y \theta^T(G(x^*{}_i, y^*{}_i) - G(x^*{}_i, y)) + \epsilon D(y, y^*{}_i)$$

To take a step, we'll need to know the sequence that maximizes this

Strategy

- Subgradient descent
  - slacks aren't differentiable, but it doesn't really matter (piecewise linear)
  - when you know the maximising y, the slacks are linear in theta
- Repeat
  - pass through data, computing maximizing y
    - can be brutally expensive
  - this gives slacks as linear function of theta
  - differentiate, take a gradient step

LEARCH=IRL via structured prediction

- Adopt dual representation of policies in MDP
- Then it all boils down to what we've seen