Appendix

A Code and Demo Video.

In the project webpage: https://metadriverse.github.io/policydissect, we provide **demonstrative videos** showing the *Policy Dissection* for human-AI shared control and the source code including all trained controllers.

B Workflow of Policy Dissection

Algorithm 1: The workflow of identifying *motor primitives*

Input : MLP policy π with L layers, I units per layer, environment $\mathbf{env}$, episodes to collect N.
Output : *motor primitives* m^j, correlation coefficient v^j related to s^j, where $j = 1, ..., J$.

for $n \leftarrow 0$ **to** N **do**
  Executing policy π in $\mathbf{env}$, recording $\{\mathbf{z}^{l,i} = [z_1^{l,i}, z_2^{l,i}, ...]\}^{I \cdot L}$ and $\{\mathbf{s}^j = [s_1^j, s_2^j, ...]\}^J$.
for $j \leftarrow 0$ **to** J **do**
  for $l \leftarrow 0$ **to** L **do**
    for $i \leftarrow 0$ **to** I **do**
      Calculate frequency discrepancy $\mathrm{Dis}(\mathbf{z}^{l,i}, \mathbf{s}^j)$ following Eq. 3:
  Determine *motor primitive* m^j for kinematic attribute s^j according to Eq. 4 Identify output v^j for m^j according to Eq. 8

After identifying the motion generation block, *motor primitives*, we can pick one or more of them to activate simultaneously and evoke a certain behavior. Therefore, a group of units are aligned with a certain behavior and *stimulation-evoked map* can be built.

C Human Subject's Comments on Shared Control

The performance of shared controls system is largely determined by the effective coordination between human subject and the AI. Some of our experiment participants report that collaboration with quadrupedal AI is more difficult than with self-driving AI. The possible reason is that human subjects are more familiar with driving vehicles than controlling legged robots. For the shared driving task, human subjects have better global planning abilities about which lane to choose and when to stop. However, for controlling the legged robots on bumpy terrain and avoiding obstacles, some of them didn't consider how the complex terrain would influence the pose of legged robots and thus had a hard time interacting with the quadrupedal AI. For example, when the robot is moving down from a small slope and is pitching forward, stopping command should be issued cautiously. Otherwise, the agent may roll or flip forward. Therefore, it would be better if human subjects have prior knowledge about the task, so that they can collaborate better with the AI trained for this task. How to implement effective human-AI teaming in general is an important but less explored direction.

D Summary of Qualitative Results

Besides the results shown in the main paper, we provide more qualitative results of human-AI shared control on policies trained in MetaDrive, Pybullet-A1, Ant, Walker and BipedalWalker environments.

As shown in Fig. 1, in the top-down view we visualize the trajectories of the PPO agent with or without human shared control on the test environment of driving task. Since the PPO agent is trained in the mild traffic density without obstacles, it is incompatible to brake and side-pass. In contrast, with human in the control loop, the lane changing and braking can be deliberately evoked by human, helping the agent solve the near-accidental cases.

We provide a summary on how to identify and use *motor primitives* to evoke behaviors for the trained agents in different environments.

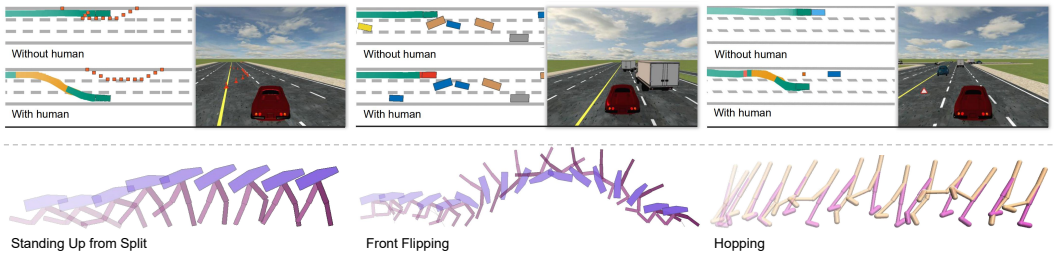


Figure 1: Qualitative demonstration of *Policy Dissection*. First row plots the driving results of the trained PPO agent with or without human-AI shared control in three driving cases. Each case plots the trajectories with human involvement, where different colors represent trajectory segments resulted from different *motor primitives*. Red and orange denote "braking" and "lane changing" respectively. The second row plots more examples of evoking behaviors in different locomotion environments.

MetaDrive: Brake. By activating the *motor primitive* positively related to speed with a negative value, the agent can gradually decrease the speed and still stay in the same lane. The brake can be evoked to avoid collision to obstacles and cut-in vehicles which the agent has never seen in the training phase.

MetaDrive: Lane Changing. Since the agent can observe the distance to left and right side, identifying motor primitives related to sidewalk and yellow solid line can make the distance to side controllable. We use "tanh" as activation function which is symmetric. Therefore identifying only one *motor primitive* related to one side is enough to evoke lane changing behavior, since, for example, increasing the distance to left side equals decreasing the distance to right side. Lane changing is useful for sidestepping obstacles or moving to another lane with lower traffic density.

Pybullet-A1: Turning Left/Right. As shown in Fig. 2, the moving direction of legged robot can be controlled by activating the neurons related to yaw or yaw rate. Similarly, negative and positive stimulation makes the agent turn towards opposite direction.

Pybullet-A1: Stopping. We can also evoke deceleration or stopping behaviors on the legged robot by controlling the activation of speed-related units. Since a huge deceleration may make the agent roll forward, we also add a pitch control to suppress the possible front flipping.

Gym-Ant: Stopping.. The ant will stop if *motor primitive* associated with x-axis speed is activated.

Gym-Ant: Moving Up/Down. The ant can move up/down if we activate two primitives respectively related to: 1) Speed of y-axis and 2) heading direction. Similar to evoking legged robot turning behavior, opposite activation values make the agent move in opposite direction.

Gym-Ant: Spinning. The ant can spin if we activate unis related to yaw rate.

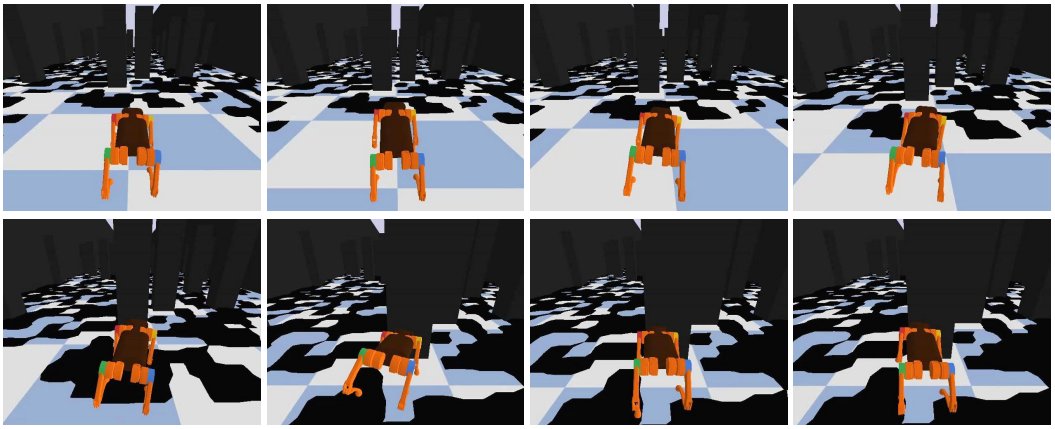
Gym-Walker: Deceleration. The agent trained in Walker stops because we stimulate the *motor primitives* positively associated with velocity with a negative stimulation. As a result, the low-level action sequence for deceleration is executed.

Gym-Walker: Hopping. We manage to find the *motor primitives* associated with torque force. We then disable the movement of the knee in red leg by stimulating the *motor primitive* with negative value. As a result, the agent hops with only the yellow leg.

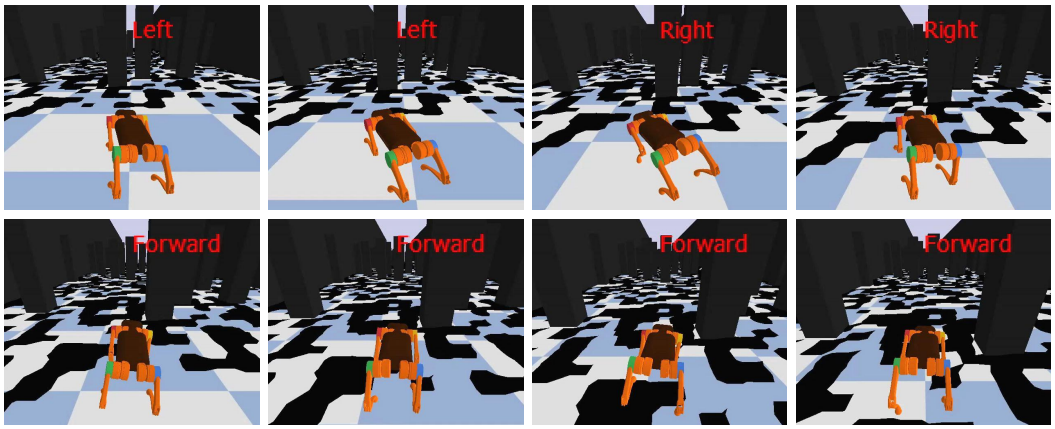
Gym-BidepalWalker: Jumping. For the agent trained in BipedalWalker, jumping is achieved by stimulating *motor primitives* related to V_z, where Z-axis is upward.

Gym-BidepalWalker: Front Flipping. We train the BidepdalWalker walking with larger torque, and thus it can jump higher if we activate units associated with V_z. BipedalWalker can conduct front flipping, which is the combination of jumping and pitching. This is resulted from combining another *motor primitive* related to pitch into control. Therefore, the walker jumps with increasing angular velocity, performing front flipping.

Gym-BidepalWalker: Standing Up from Split. In addition, a common failure mode of agent trained in BipedalWalker is that the agent performs split and can not stand up when both legs touch



(a) Insensible Agent Walking in Test Environment **Without** Human



(b) Insensible Agent Walking in Test Environment **With** Human

Figure 2: Qualitative demonstration of the insensible agents when deployed in test environment with or without human. Facilitated by human, the legged robots can turn left to avoid the collision to the front obstacle, even if it is insensible and trained to move forward as fast as possible.

the ground. We identify the *motor primitives* associated with all the motor torques. Stimulating these *motor primitives*, the agent manages to stand up and continue moving forward.

IsaacGym-Cassie: Crouching and Tiptoe. The crouching and tiptoe behaviors are related to the z-height of the main robot body, so that we can identify z-axis related units, and activate them to change the z-height.

IsaacGym-Cassie: Backflip. This behavior can be described by increasing 1. height 2. pitch and 3. knee force. Therefore, we activate three corresponding *motor primitives* selected by Eq. 4 with output calculated by Eq. 8 to make the robot back flip. Note that we also manually determine the termination time t_1 for this action, preventing it from continuously back flipping

IsaacGym-Cassie: Jumping. The jumping is represented by increasing 1. knee torque force 2. forward speed/moving distance. Therefore, activating corresponding *motor primitives* evoke jumping. Similar as the back-flip, the activation period $T = t_1 - t_0$ is predetermined, so that it only jump once after pressing the key for jumping.

IsaacGym-Cassie: Redirection. The redirection can be achieved in the same way as Gym-Ant, activating y-axis movement related units or heading related units. The forward movement can be evoked by activating x-axis related *motor primitives*.

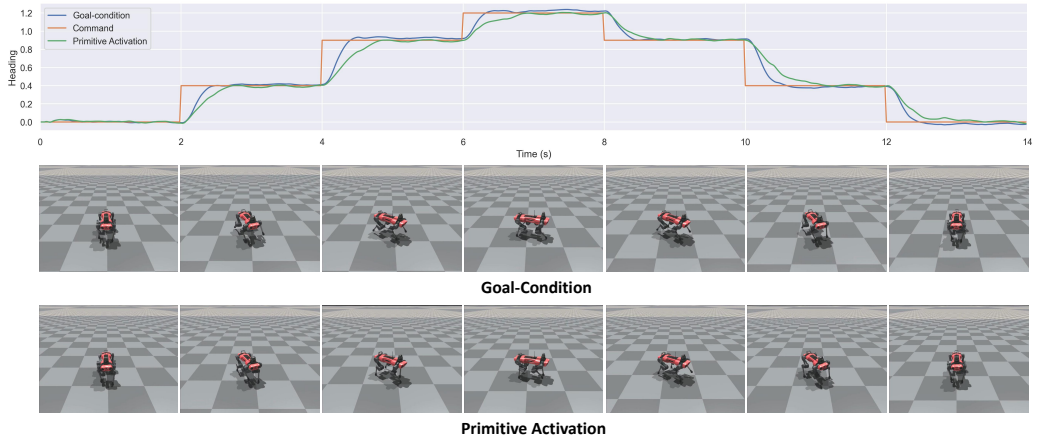


Figure 3: The visualization result of the comparison experiment. It suggests that the goal conditioned control enabled by our method is comparable to the state-of-the-art goal conditioned controller on ANYmal-C robot in IsaacGym simulator.

E Details and Visualization of Control Precision Experiment

We provide a visualization results of the control precision experiments, as shown in Fig. 3. It is obvious that both both goal-conditioned control methods can drive ANYmal-C to track the yaw command smoothly and stably. The main difference is that the goal-conditioned control enabled by *Policy Dissection* has a slightly longer response time when changing command, compared to explicit goal-conditioned control. The demo video is availabe here .

F Environment Details

We provide more environment details for MetaDrive and Pybullet-A1 such as the observations, the design of reward functions, and the termination conditions.

F.1 MetaDrive

In the driving task, the objective of RL agents is to steer the target vehicles with low-level continuous control actions, namely acceleration, brake, and steering.

Observation. The observation of RL agents is as follows:

- A 240-dimensional vector denoting the Lidar-like point clouds with $50m$ maximum detecting distance centering at the target vehicle. Each entry is in $[0, 1]$ with Gaussian noise and represents the relative distance of the nearest obstacle in the specified direction.
- A vector containing the data that summarizes the target vehicle's state such as the steering, heading, velocity, and relative distance to the left and right boundaries.
- The navigation information that guides the target vehicle toward the destination. We sparsely spread a set of checkpoints, 50m apart on average, in the route and use the relative positions toward future checkpoints as additional observation to the target vehicle.

Reward and Cost Scheme. The reward function is composed of four parts as follows:

$$R = c_1 R_{disp} + c_2 R_{speed} + R_{term}. \tag{1}$$

The *displacement reward* $R_{disp} = d_t - d_{t-1}$, wherein the d_t and d_{t-1} denotes the longitudinal movement of the target vehicle in Frenet coordinates of current lane between two consecutive time steps, provides dense reward to encourage agent to move forward. The *speed reward* $R_{speed} = v_t/v_{max}$ incentives agent to drive fast. v_t and v_{max} denote the current velocity and the maximum velocity ($80\ km/h$), respectively. We also define a sparse *terminal reward* R_{term}, which is non-zero

only at the last time step. At that step, we set $R_{disp} = R_{speed} = 0$ and assign R_{term} according to the terminal state. R_{term} is set to $+10$ if the vehicle reaches the destination, -5 for crashing others or violating the traffic rule. We set $c_1 = 1$ and $c_2 = 0.1$. For measuring the safety, collision to vehicles, obstacles, sidewalk raises a cost $+1$ at each time step. The sum of cost generated in one episode is episode cost, a metric like episode reward, but reflecting safety instead.

Termination Conditions and Evaluation Metrics. Since we attempt to benchmark the safety of shared-control system, collision to vehicles and obstacles will not terminate the episode. The episode will be terminated only when: 1) the agent drive out of the drivable area and 2) the agent arrives the destination. For each trained agent, we evaluate it in 20 held-out test environment and define the ratio of episodes where the agent arrives at the destination as the *success rate*. The definition is the same for *out of road*. Also, the average episode reward and cost on 20 test environment produce two metrics: *Episodic Reward* and *Episodic Cost*. Since each agent are trained across 5 random seeds, this evaluation process will be executed for 5 agent which has same training setting but different random seeds. We report the average and std on the 4 metrics mentioned above.

F.2 Pybullet-A1

In the quadrupedal locomotion task, the RL training objective is to move forward on bumpy terrain as fast as possible. We use two environments: the one without obstacles as training environment, and the other one with obstacles as test environment. The environment construction, reward definition, tasks are the same as [3]. Its environment has open-source code at: https://github.com/Mehooz/vision4leg. In this work, we only slightly modify the observation as follows:

- IMU recording Yaw, Yaw rate, Pitch and Roll
- Angle of 12 joints
- Torque applied to 12 joints

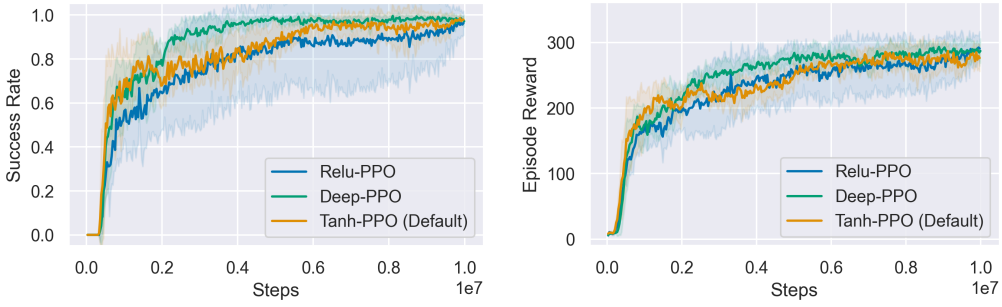
Containing historical output of these sensors over the last 3 steps, the proprioceptive observation is in 84 dimension. For LocoTransformer baseline method directly trained on test environment, the input is this state vector and first-view depth images in 64×64 over the last 4 steps.

F.3 IsaacGym

We use the same setting reported in [1], and the training code is from https://github.com/leggedrobotics/legged_gym.

G Learning Curves of Trained Agents

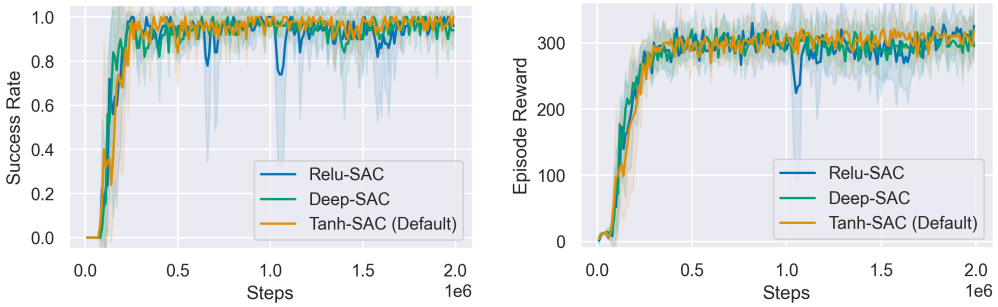
G.1 MetaDrive



(a) Success Rate of PPO Agents

(b) Episode Reward of PPO Agents

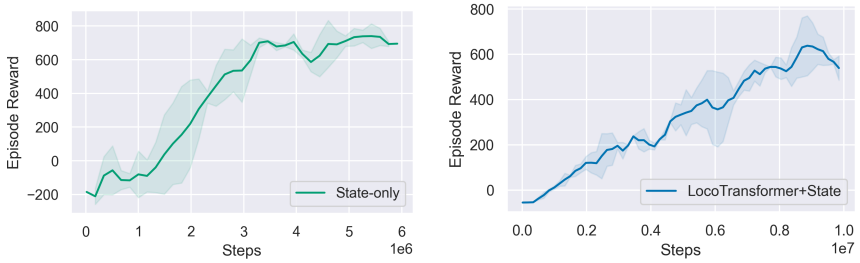
Figure 4: Success rate and episode reward of PPO agents on 50 training environments.



(a) Success Rate of SAC Agents (b) Episode Reward of SAC Agents

Figure 5: Success rate and episode reward of SAC agents on 50 training environments.

G.2 Pybullet-A1 Legged Robots



(a) Insensible Agent on **Training** Environment (b) Visual Input Agent on **Test** Environment

Figure 6: Training episode reward of two quadrupedal locomotion policies.

G.3 Gym Environment

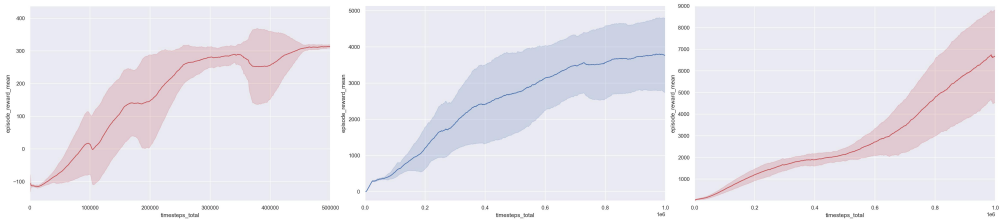


Figure 7: The episodic reward during the learning of SAC algorithms in Ant, Walker and Bipedal-Walker tasks. Though we repeat 5 times for each experiment, *Policy Dissection* is applied to the agent with highest performance to do qualitative study.

H Hyper-parameters

H.1 MetaDrive

Since SAC is not sensitive to the choice of hyterparameters, the learning rate $1\mathrm{e}{-4}$ is also suitable for Deep-SAC (6 Layer) and Relu SAC. For Deep-PPO and Relu-PPO, learning rate should be slightly decreased due to the change of neural network structure. Also note that the number of hidden units is changed to 128 per layer for Deep PPO with 6 layer. This is for keeping the total number of network variables approximate to the network used by default PPO (around 250,000 variables).

Table 1: SAC/ Deep SAC/ Relu SAC

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for target network update	0.005
Learning Rate	$1e{-4}$
Environmental horizon T	1500
Steps before Learning start	10000
Activation Function	"tanh" or "relu"
Prioritized Replay	False
Target Network Update Frequency	1
Soft Update τ	$5e{-3}$
MLP Hidden Units	256
MLP Layers	2 or 6

Table 2: PPO

Hyper-parameter	Value
KL Coefficient	0.2
λ for GAE [2]	0.95
Discounted Factor γ	0.99
Number of SGD epochs	20
Train Batch Size	30,000
SGD mini batch size	256
Learning Rate	$3e{-4}$
Clip Parameter ϵ	0.2
Activation Function	"tanh"
MLP Hidden Units	256
MLP Layers	2

Table 3: Deep PPO

Hyper-parameter	Value
KL Coefficient	0.2
λ for GAE [2]	0.95
Discounted Factor γ	0.99
Number of SGD epochs	20
Train Batch Size	30,000
SGD mini batch size	100
Learning Rate	$1e{-4}$
Clip Parameter ϵ	0.2
Activation Function	"tanh"
MLP Hidden Units	128
MLP Layers	6

Table 4: Relu-PPO

Hyper-parameter	Value
KL Coefficient	0.2
λ for GAE [2]	0.95
Discounted Factor γ	0.99
Number of SGD epochs	20
Train Batch Size	30,000
SGD mini batch size	100
Learning Rate	$5e{-5}$
Clip Parameter ϵ	0.2
Activation Function	"Relu"
MLP Hidden Units	256
MLP Layers	2

H.2 Gym Environments and Pybullet-A1 Legged Robots

Agents of Gym environments (Walker/Ant/Bidepadel Walker) are trained by SAC and share the same hyperpatrameter setting. Quadrupedal robots are trained by PPO, and we follow the same codebase and configuration used in [3].

Table 5: SAC for Gym agents

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for target network update	0.005
Learning Rate	$3e{-4}$
Environmental horizon T	1500
Steps before Learning start	1000
Activation Function	"tanh"
Prioritized Replay	False
Target Network Update Frequency	1
Soft Update τ	$5e{-3}$
MLP Hidden Units	256
MLP Layers	2

Table 6: PPO for Legged robots

Hyper-parameter	Value
KL Coefficient	0.2
λ for GAE [2]	0.95
Discounted Factor γ	0.99
Number of SGD epochs	3
Train Batch Size	16384
SGD mini batch size	1024
Learning Rate	$1e{-4}$
Clip Parameter ϵ	0.2
Activation Function	"tanh"
MLP Hidden Units	256
MLP Layers	4

References

[1] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[2] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

[3] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. *arXiv preprint arXiv:2107.03996*, 2021.