Real-time gestural control of robot manipulator through Deep Learning human-pose inference

Jesus Bujalance, Fabien Moutarde

Real-time gestural control of robot manipulator through Deep Learning human-pose inference

Jesus Bujalance Martin and Fabien Moutarde

MINES ParisTech, PSL Research University, Center for Robotics, 60 Bd St Michel 75006 Paris, France

Abstract. With the raise of collaborative robots, human-robot interaction needs to be as natural as possible. In this work, we present a framework for real-time continuous motion control of a real collaborative robot (cobot) from gestures captured by an RGB camera. Through deep learning existing techniques, we obtain human skeletal pose information both in 2D and 3D. We use it to design a controller that makes the robot mirror in real-time the movements of a human arm or hand.

Keywords: collaborative robots · robot manipulator · motion control · real time · pose estimation · deep learning · ROS

1 Introduction

The first generation of manufacturing robots were always operating in human-free areas, for safety reasons. But during recent years, new types of robots have been designed for deployment in direct contact, and even cooperation, with human workers. These *collaborative* robots create the opportunity and interest for *gesture-based* control of robots by humans, for seamless and natural Human Robot Interaction.

Gestural control can mean either launching particular robot actions by just executing some predefined gestures interpreted as commands, or direct and continuous motion control of the robot by human movements. In this work, we focus on the latter. Until recently, robust servoing of robot motion on human movement was possible only using wearable sensors (e.g. ElectroMyoGram sensor, EMG, cf. [9] or [4]), or thanks to a depth camera (such as Kinect or Real-sense) allowing real-time human skeletal pose estimation and tracking (e.g. [1]).

Meanwhile, recent deep learning methods have achieved great results in both 2D and 3D human skeletal pose estimation from RGB cameras in real time. Inspired by this progress, we have designed a robot motion continuous controller based on pose estimation. To clarify, there is no gesture recognition module in this work. The objective is to have the robot mimic the movements of the human arm in real time, without any underlying understanding of these movements.

In our framework, an RGB camera captures the movements of the user. We extract the 2D and 3D poses in real-time and process them to control the robot motion. We present two implementations, one based on forward kinematics and one on inverse kinematics.

2 Pose estimation

Pose estimation is the problem of determining the position and orientation of a person from RGB images or videos. Namely, we want to obtain coordinates of keypoints such as joints, eyes, or fingers. In this section we will discuss the methods we used for both 2D and 3D pose estimation.

2.1 2D Pose estimation

Multi-person pose estimation models can be categorized as either top-down or bottom-up. For instance, AlphaPose [3] is a top-down approach since it detects every person in the scene then extracts their pose individually. Openpose [2] is a bottom-up approach since it detects all the keypoints in the scene and then puts them together to form skeletons. Other bottom-up approaches, like [8], do not separate the detection and grouping stages, obtaining a single-stage network.

We chose OpenPose over other methods because of its proven real-time performance. It also has the most active support and has been regularly updated with new features since its release. Particularly, in this work we use the hand detection module.

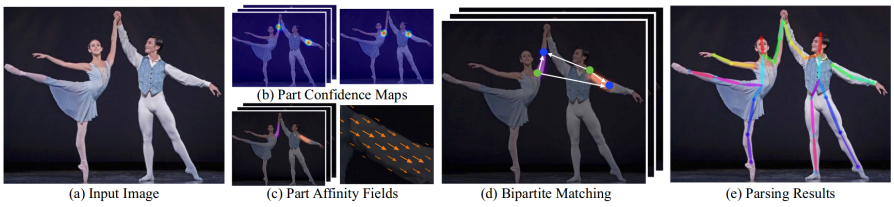


Fig. 1: OpenPose [2] pipeline.

OpenPose Figure 1 shows the general pipeline of OpenPose. First, an input image enters a multi-stage CNN which jointly predicts the set of confidence maps, one for each part, and the set of part affinities which represent the degree of association between parts. Then, bipartite matching is used to associate body part candidates and obtain the full 2D skeletons. A part refers to a keypoint such as the left elbow or the right eye.

2.2 3D Pose estimation

2D pose estimation allows us to control the robot in a 2D plane. In order to add a third dimension we need the 3D pose. There are numerous open-source methods that perform this task. We chose Human Mesh Recovery (HMR) [5], but we concede that it might be excessive to compute a full mesh when we only

care about the pose. There are other simpler yet effective approaches such as [7] that could have been used as well. Nevertheless HMR provides good results and performs well in real-time.

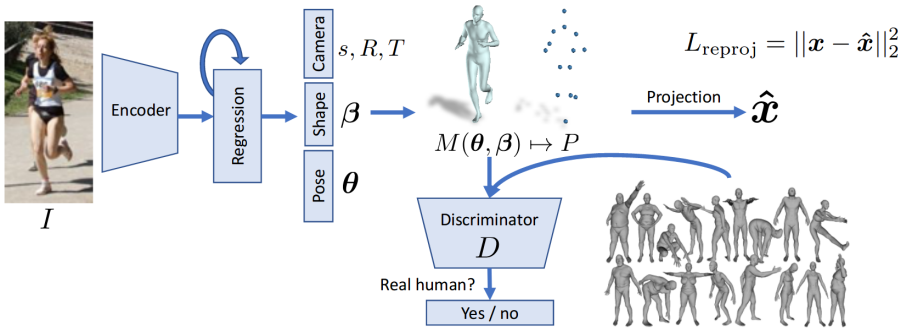


Fig. 2: HMR [5] pipeline.

Human Mesh Recovery HMR is a recent approach that directly predicts the 3D pose and shape of a human SMPL model [6], along with the camera configuration, from a single image.

The proposed framework is as follows : An image I is sent through a convolutional encoder into a 2048D latent space. The latent features are then decoded by an iterative regression to produce the pose, the shape, and the camera configuration. These 3D parameters are sent to the discriminator D, whose goal is to tell whether they come from a real human.

HMR uses a weakly-supervised adversarial framework that allows the model to be trained on images with only 2D pose annotations, without any ground truth 3D labels. Indeed, the reprojection loss used for training only requires 2D joint locations. Note also that there is no need to make a priori assumptions about the joint limits, bone ratios or other physiognomic constraints. Instead, the 3D mesh pool acts as a data-driven prior.

3 Real-time robot control

The robot we used is a Universal Robot, model UR3. Shown in Figure 4, it is a 6 DOF robot manipulator designed for collaborative tasks. The easiest way to control Universal Robots directly is through the URScript programming language. However, we chose to implement our controller in python within the Robot Operating System (ROS) framework. ROS is a middleware which provides hardware abstraction, device drivers, visualizers, message-passing, and other useful tools to manage multi-modular projects like ours.

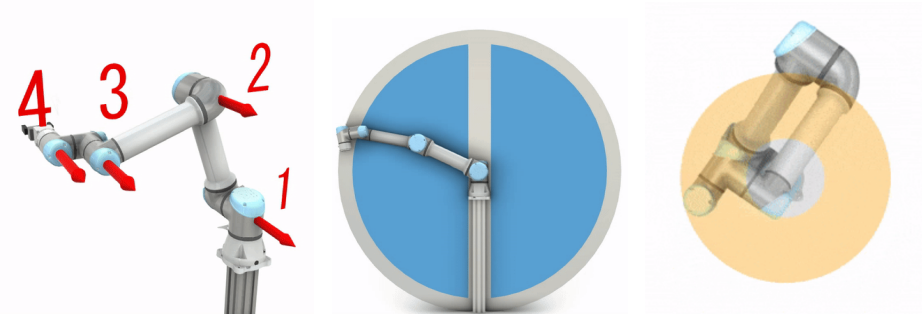
As a side note, although the UR3 robot is controlled in position (at the hardware level), we should see more velocity-controlled robots in the future as the need for real-time applications increases [10]. The same author of this paper made an open-source ROS package, *jog_arm*, which allows to control the speed of the joints or the end-effector in real time. In this section, we will discuss the advantages (and disavantages) of this package as well as the two different implementations that we developed to control the robot from a sequence of human poses.

3.1 Inverse kinematics (IK)

The most straight-forward way to control a robot is to control only the position of the end-effector. We recall the equation tying the coordinates in cartesian space x and joint space q :

$$\dot{x} = J(q)\,\dot{q} \tag{1}$$

For our robot, the Jacobian matrix J is a 6 by 6 matrix (generally 6 by #DOF). The package simply inverts the Jacobian matrix to recover the desired joint angles from the desired end-effector xyz position, corresponding to the 3D hand poisition provided by HMR. The other 3 coordinates correspond to the orientation of the end-effector, which we maintain constant (similar to fixing the 3 wrist joints). The main advantages of this Jacobian method are its speed and that the resulting trajectories can be altered in real time. It is also deterministic, avoiding unexpected behaviours. This makes it very suitable for real-time environments.



(a) Wrist singularity (b) Elbow singularity (c) Shoulder singularity

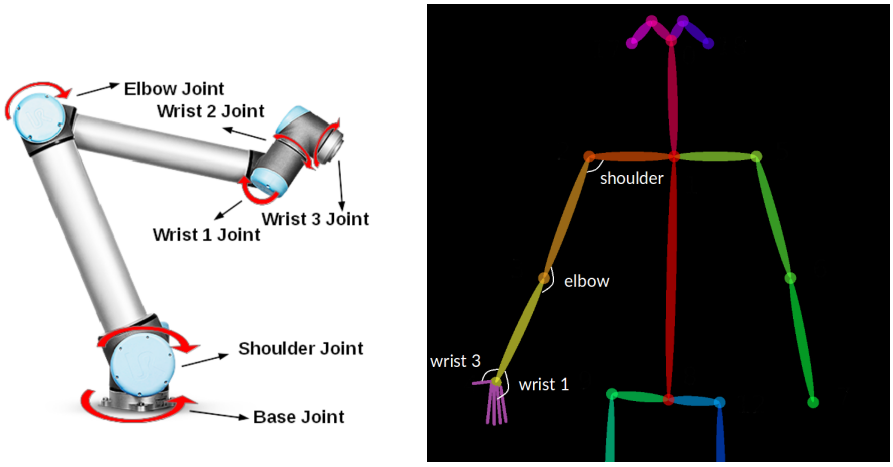
Fig. 3: Singularities of UR robots (source: www.universal-robots.com)

Singularities When dealing with an IK approach, one must be aware of singularities and how to handle them. As shown in Figure 3 the UR3 robot has

three different types of singularities. At a singularity, the mobility of the manipulator is reduced. They occur when $det(J) = 0$, and can produce undesired behaviours such as infinite solutions (wrist singularity) or a solution with infinite joint speeds (shoulder singularity).

The *jog_arm* package is able to reverse out of singularities. Indeed, the robot will decrease its speed as it is approaching a singularity and halt before reaching it. More complicated planners should be able to avoid them altogether by taking more sophisticated IK solutions, but this simple Jacobian inversion method cannot plan around them (or obstacles, joint limits, etc). Because of these limitations, this technique is generally only useful over short distances.

3.2 Forward kinematics (FK)



(a) UR3 robot (b) OpenPose skeleton and mapping

Fig. 4: Joint mapping for forward kinematics.

If we opt for a FK approach, the pose information allows us to control the entire robot, not just the end-effector. However, we come across the problem of mapping the joints between a human arm and the robot manipulator, which don't share the same amount of DOF. Our hand-crafted joint mapping is as follows. The shoulder, elbow and wrist 1 mappings come out naturally. The wrist rotation around the forearm axis (wrist 3 joint) is computed from the angle between the forearm and the thumb of the skeleton. We only require the 2D skeleton from OpenPose here, as shown in Figure 4.

The base joint is the only one operating outside of the 2D plane (the wrist 2 joint remains fixed). A calibration step measures the length in pixels of the upper arm whenever a human is detected for the first time. When we detect

a shorter forearm, we compute a base joint angle accordingly. To distinguish between the arm coming towards or moving away from the camera, we could compare the depth of the elbow and shoulder provided by HMR, but it comes with extra computation time.

The main limitation of this method is that it only works properly if the person is facing the camera and the movements are mostly planar, which limits the range of motion of the base joint to approximately $\pm 45^o$. Because of this limited range, we choose not to compensate for the differences between the projected angles given by OpenPose and the actual angles of the user. A future version should correct these distortions.

Gripper For both IK and FK we include a gripper controller which distinguishes between two states : fully open and fully closed. Based on the OpenPose hand detection, we detect a closed hand if the vectors wrist-knuckle and knuckle-fingertip have opposite directions for all fingers (excluding the thumb).

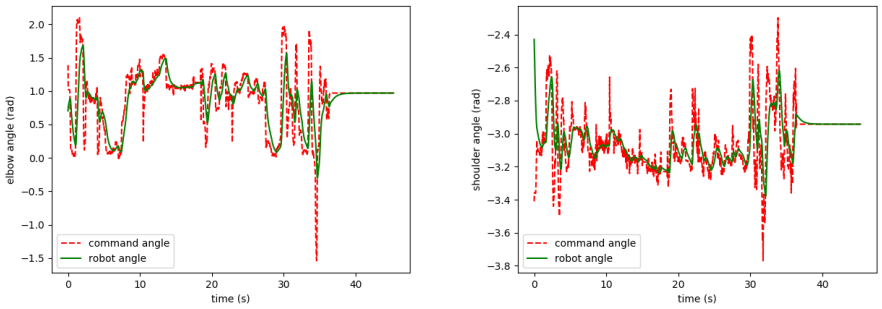
4 Experiments

When testing our controller, we ran into a few issues, most coming from HMR. As discussed in section 2.2, the network is optimized to provide a credible human mesh, not just the pose. Therefore, its precision regarding joint positions is not as good as that of OpenPose. We avoid this issue by simply prioritizing OpenPose outputs, and counting on HMR just for the third dimension even in the IK case. Also, a momentum-like exponential average of the commands proved to give smoother results. Another issue comes when both arms of the user are within the 2D plane of the body. Quite often, HMR will think that the person is facing backwards. Indeed, if the face is not well detected, neither the reprojection loss nor the adversarial loss can discriminate against a mesh facing backwards in this situation. To solve this issue we could simply mirror the backwards mesh, but often it won't be satisfying. Another option is to retrain HMR adding a loss to the discriminator which penalizes meshes facing backwards.

OpenPose is much more precise and robust to lightning. The only issue comes when the hand is perpendicular to the camera, but this is expected since all fingers but one are partially occluded.

Results In our setup with a GEFORCE GTX 1080 Ti, OpenPose runs at 30fps for 640x480 images. OpenPose and HMR run together at 6fps. Some videos showing the results are provided in the linked Google Drive.

Figure 5 compares, in the FK case, command angles and resulting robot angles. It shows that the delay is very short. The commands present some noise but the dynamics of the robot act as a filter and the result is a smooth trajectory. In the IK case, since only the position of the end-effector is controlled, the target/result comparison cannot be made on angles. Figure 6 therefore compares target and obtained position of the end-effector. It shows that the robot doesn't



(a) Elbow (b) Shoulder

Fig. 5: Command angle and actual robot angle (FK)

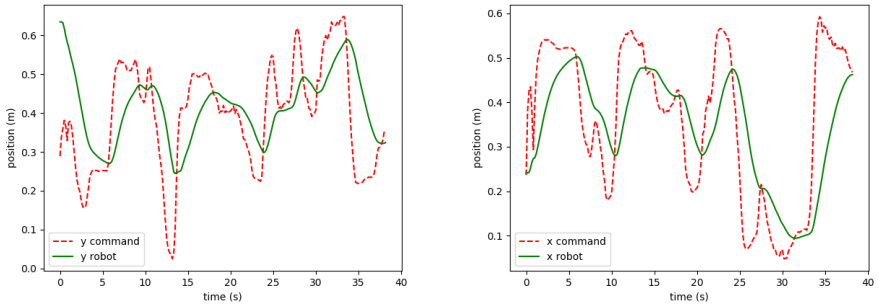


Fig. 6: Position of the end-effector in the 2D plane (IK)

follow the command as tightly and the delay is bigger, equal to 1 second. This behaviour is expected since the frequency of the commands is much lower and the post-processing is heavier. Note that in our setup with good lightning and only one person, OpenPose performs extremely well and its output is very close to the ground truth. Therefore the command curves could be interpreted as the ground truth corrupted with some additive centered noise.

Overall, the FK solution proved to be more precise and satisfying for the user. Still, the IK solution is also valuable since it allows to control the robot in the entire 3D space with no calibration.

5 Conclusions and future work

The raising trend in bringing workers and industrial robots together needs an efficient human-machine interaction. This work shows that a gestural motion

control is possible in a simple scenario and it represents a first step towards a richer human-robot collaboration.

The inverse kinematics approach wasn't responsive enough for a real-time application, but 3D pose estimation can still be useful for other tasks such as pointing to a particular object to be picked up by the robot. The forward kinematics approach was responsive and robust. A more sophisticated mapping could allow us to obtain joint angle sequences capable of reproducing a task shown in a human video demonstration. Future work will be done on imitation learning from demonstrations to have the robot perform a variety of complex tasks (e.g. pick and place).

Acknowledgements

The research leading to these results has received funding by the EU Horizon 2020 Research and Innovation Programme under grant agreement No. 820767, CoLLaboratE project.

References

1. G. Paravati F. Manuri A. Sanna, F. Lamberti. A kinect-based natural interface for quadrotor control. *Entertainment Computing*, Volume 4, Issue 3:179–186, 2013.
2. Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. 2018.
3. Haoshu Fang, Shuqin Xie, and Cewu Lu. RMPE: regional multi-person pose estimation. 2016.
4. Shin H. S. Lee K. H. Ganiev, A. Study on virtual control of a robotic arm via a myo armband for the selfmanipulation of a hand amputee. *Int. J. Appl. Eng. Res*, 11(2):775–782, 2016.
5. Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. *CoRR*, abs/1712.06584, 2017.
6. Matthew Loper, Naureen Mahmood, avier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34, 2015.
7. Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017.
8. Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017.
9. Y. Xu, C. Yang, P. Liang, L. Zhao, and Z. Li. Development of a hybrid motion capture method using myo armband with application to teleoperation. In *2016 IEEE International Conference on Mechatronics and Automation*, pages 1179–1184, 2016.
10. Andy Zelenak, Clinton Peterson, Jack Thompson, and Mitch Pryor. The advantages of velocity control for reactive robot motion. In *ASME 2015 Dynamic Systems and Control Conference*, pages V003T43A003–V003T43A003. American Society of Mechanical Engineers, 2015.