We thank the reviewers for their valuable comments. We will add the suggested citations and fix the typos and update the manuscript and the supplement with the discussions below. To ensure reproducibility, we will release the code and trained models. We now address all the comments individually.

R1-3. First-order model. The equation $\mathcal{T}_{\mathbf{S}\leftarrow\mathbf{D}}(z) \approx \mathcal{T}_{\mathbf{S}\leftarrow\mathbf{R}}(p_k) + J_k(z - \mathcal{T}_{\mathbf{D}\leftarrow\mathbf{R}}(p_k))$ (4) defines how the coordinates in the neighbourhood of the keypoint should be transformed. Zeroth-order approximation for $\mathcal{T}_{\mathbf{S}\leftarrow\mathbf{D}}(z)$ is $\mathcal{T}_{\mathbf{S}\leftarrow\mathbf{R}}(p_k)$ which does not produce meaningful motion since all the points in a neighborhood of a keypoint are mapped to the same location. Instead, Monkey-net [21] considers zeroth-order approximation for $\mathcal{T}_{\mathbf{S}\leftarrow\mathbf{D}}(z) - z$, which corresponds to $J_k = \mathbb{1}$ in Eq.(4). It assumes that motion is constant in the neighbourhood of keypoints (pixels are simply shifted). We call this a zeroth-order model. Such model, for example, cannot handle objects moving towards the camera. In contrast, the proposed first-order model can handle such motion by introducing J_k, which models the corresponding affine transformation.

R1-3. Reference frame. To define J_k we assume there exists an abstract reference frame which is an arbitrary representation of the object. We note that the reference frame is an abstract concept that cancels out in our derivations, therefore not allowing us to visualize it. Then, Eq.(4) can be viewed as mapping the object from the driving frame to this arbitrary representation, followed by mapping it to the source frame. Note that unlike [31] there is no dependence on the reference frame in the final model, the only constraint we have on the reference is that it should be similar for both source and driving frames near keypoints locations. In fact as long as p_k (keypoint coordinates in the reference frame) is the same for source and driving keypoints, we can move the object to an arbitrary location in the reference frame without changing the target result. Therefore, the coordinate of p_k cannot be estimated and visualized.

R1. Difference of the gaussian heatmaps. Gaussian heatmaps are usually employed for pose guided generation [2]. Here, we use the difference of gaussians to indicate to the dense motion predictor where the source keypoints are. This information helps predict occlusion maps. Concatenation of source and driving heatmaps requires twice as much channels and hence is computationally less efficient. A similar representation was used in [21].

R1. Equivariance constraints. Since there is no direct supervision on the keypoints, their predictions can be unstable. Equivariance constraint forces the model to predict consistent keypoints with respect to a known geometric transformation $\mathcal{T}_{\mathbf{X}\leftarrow\mathbf{Y}}$. We used thin plate splines as they have been previously employed in unsupervised keypoint detection [15, 32] and are similar to natural image deformations. Similarly to keypoints locations we introduce equivariance constraints for jacobians in the keypoint neighbourhood.

R2. Gap between "Full" and "Pyr.+O" model. The difference between Full and Pyr.+O corresponds exactly to the addition of Jacobians and of the equivariance loss over Jacobians, e.g Eq.(11). Note that Pyr.+O and other baselines use equivariance loss over keypoint locations, e.g Eq.(10). If we disable equivariance loss over Jacobians they will become unstable, and we get the following results $\mathcal{L}_1$ - 0.073, (AKD, MKR) - (9.89, 0.052), AED - 0.22. Note this performance is significantly worse than Pyr.+O, and similar to the Baseline (see Tab. 1).

R2. Reconstruction loss. We use the following layers for each resolution conv1_2, conv2_2, conv3_2, conv4_2, conv5_2, similarly to [28]. The resolutions are 256×256, 128×128, 64×64 and 32×32. There are 5×4 terms in total. The pyramid loss implementation suggested by R2 corresponds to the loss used by [28], which in our experiments led to very blurry results (see Fig. 2). Our loss adds minimal computational overhead compared to the loss of [28].

R2. Aspect ratio. We cropped faces using square bounding boxes. If the detector outputs a rectangular bounding box of size 300×200, we first enlarge this bounding box to 300×300 and then crop. Thus, the aspect ratio does not change.

R3. Failure cases. Failure cases will be added to *Supp.Mat.* and to the associated video. Overall, failures cases are often due to not realistic inpainting (Example 1,3 and 5 in the figure below) or rare motions, underrepresented in training data (Example 2 and 4 in the figure below). Zoom-in for greater detail.

R3. Second order motion model. Second order extension of our model is possible but is a subject of future work.It will require computation of the second derivative of the inverse of $\mathcal{T}$ which is a three dimensional tensor.

R3. Fig. 1. The intuition behind Fig. 1 is that we combine keypoint displacement from motion model of [21] (see above), with J_k. We will clarify this in the camera ready.

R3. Clarifications about L161-L163. ConvNets are known to have difficulties to use information from input regions to generate spatially distant output regions. For example, it is hard for a CNN to utilize information from a region on the left to generate a region on the right. To simplify this task, similarly to [21] we warp the images such that the input regions in the source image have approximately the same coordinates as their output counterparts. For more information see [21] Sections 3.3-3.4.