Towards An Egocentric Framework for Rigid and Articulated Object Tracking in Virtual Reality

Catherine Taylor*
University of Bath
Marshmallow Laser Feast

Robin McNicholas†
Marshmallow Laser Feast

Darren Cosker‡
University of Bath

Abstract

To create immersive Virtual Reality (VR) applications and training environments, an appropriate method for allowing participants to interact with the virtual environments and objects in that scene must be considered. An approach which offers increased immersion and accurately models real-world behaviour would be advantageous – particularly within the areas of health, entertainment and engineering. Traditional consumer VR methods for facilitating interaction – e.g.controllers – are restricted by lack of tactile feed back and do not accurately represent real-world interactions with physical objects in terms of shape, limiting immersion. Ideally, physical objects would be transported into the virtual world and used as a means of interacting with the environment, via a robust tracking algorithm or motion capture system. However, achieving this in a real time markerless manner for a range of object types remains an open challenge. Moreover, costly motion capture systems or tracking algorithms which require multiple cameras are not practical for everyday VR use.

Given the recent advancements in object tracking and mesh reconstruction using neural networks, we present a novel neural network, *VRProp-Net+*, which predicts rigid and articulated model parameters of known everyday objects in unconstrained environments from RGB images at interactive frame rates. VRProp-Net+ utilises a novel synthetic training methodology and so does not require a time consuming capture or manual labelling procedure seen in many prominent neural network tracking or mesh reconstruction approaches. We present our network as part of an egocentric tracking framework, that allows prediction of object pose and shape given a moving camera. This scenario is helpful for practical VR experiences where the camera may be mounted on the VR Head Mounted Display (HMD) itself. This creates a dynamic capture volume, allowing a user to move around and interact with a virtual world without the need for multiple fixed cameras.

Index Terms: Computing methodologies—Neural networks Computing methodologies—Modelling and simulation Computing methodologies—Computer vision

1 Introduction

The method of interaction in a VR experience greatly influences the feeling immersion and the perceived realism. An intuitive approach engages the user in the virtual world and promotes realistic and natural interaction with the computer generated elements in the scene. Moreover, such an approach has much potential use in VR training environments, where the simulated procedure must closely model the real-world task. In this work we move towards an end to end system for bringing everyday objects into VR, while simultaneously allowing users to move around the environment, without requiring costly and impractical set ups – e.g. motion capture systems.

*e-mail: c.taylor3@bath.ac.uk
†e-mail: robin@marshmallowlaserfeast.com
‡e-mail: dpc22@bath.ac.uk

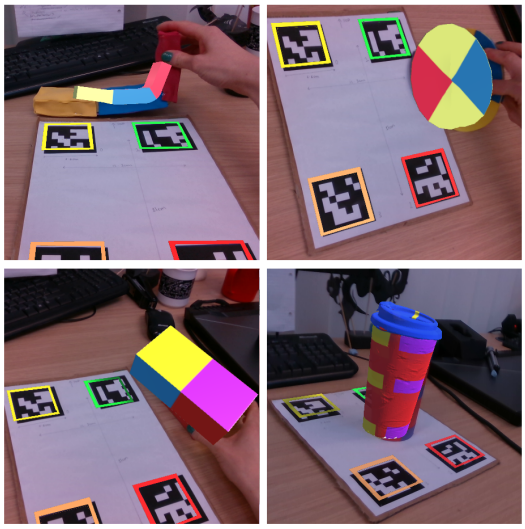


Figure 1: The pose and shape of a rigid or articulated object are predicted from RGB images captured from a moving egocentric view using *VRProp-Net+*. The predicted parameters transform a model which is rendered into a CG scene, allowing interaction with virtual objects via their physical counterpart.

Most consumer VR systems use controllers to allow communication with elements of the virtual world [4, 20]. However, controllers do not accurately model real-world interactions with physical objects and, moreover, are restricted by lack of tactile feedback. External sensors can be attached to the surface of an object in order to track its 3D position and orientation [5, 28, 36]. However, they are unable to capture any non-rigid or articulated behaviour. On the other hand, markers tracked by a motion capture system can be used to control rigid and non-rigid models [28, 35]. Despite their advantages, the non-standard hardware required makes these systems expensive and their tracking accuracy decreases when the markers are occluded, for example due to hand interactions with the object.

Alternatively, rigid or non-rigid objects can be tracked in RGB or RGBD images and used to drive rigged models [23, 27, 32, 34]. However, this is still a challenging task, in particular for articulated or non-rigid objects whose appearance can change greatly due to deformations. Neural network based approaches for object tracking or mesh reconstruction have shown great advancement in recent years and have much potential for capturing the behaviour of complex articulated and non-rigid objects [7, 10, 26, 29, 29, 37, 42]. However, many approaches are restricted by requiring multiple cameras, manually labelled data or are constrained to non real-time applications. Additionally, object tracking approaches for VR are often limited to controlled, green screen environments [29, 30].

In this paper, we propose a neural network solution for tracking known rigid and articulated objects in order to interact with their virtual counterparts. We colour our objects brightly to aid tracking but are not restricted to requiring multiple object view points, controlled environments or training on manually labelled data. Our custom

architecture *VRProp-Net+*, which is trained on a synthetic dataset, advances on *VRProp-Net* [29] and is able to predict the parameters of the rigid or articulated model with higher accuracy.

We also present an extension to VRProp-Net+ that tracks an object in a moving camera such that it may be used within a complete egocentric tracking framework. In a VR experience it is useful to attach a camera to the HMD so that it captures the part of the object which the user is focused on and allows a dynamic capture volume which is not restricted to a pre-defined space. Unlike previous works which have focused on tracking hand pose or have explored the interaction between hands and rigid objects, we predict both the shape and pose of rigid and articulated objects [9, 17, 22, 31].

2 Related Work

Interacting with Virtual Objects: Controllers are the most frequently used tool for interacting with virtual (i.e computer generated) objects and environments. Sequences of button presses and clicks on the controller as well as the 3D pose – from more sophisticated controllers (e.g the HTC Vive [4] or the Oculus Rift [20] controllers) – can be used to control elements of the virtual environment or change the shape, pose or appearance of objects in that scene. In contrast, Augmented Reality (AR) systems (e.g the HoloLens [15] or Magic Leap [13]) often use hand gestures to interact with the computer generated elements. While controllers and hand gestures facilitate interaction with the virtual environment, they offer limited immersion as they provide minimal tactile feedback and do not model the natural manner of interacting with a physical object.

An alternative, potentially more immersive approach, is to control the behaviour of virtual objects using physical objects. Thus, modelling real-world interactions as well as providing tactile feedback. Several commercial VR systems offer external sensors, which track the 6DoF pose of the object to which they are attached [4, 28, 36]. While these offer accurate rigid tracking, they do not capture any object articulations or non-rigid deformations. Additionally, they are obtrusive attachments which can change the weight and shape of the object which they are appended to. Markers can also be attached to the surface of an object and tracked using a motion capture system [28, 35]. The tracked pose of the markers can be used to control a rigid object or deform a non-rigid or articulated model. However, the reliability of marker-based approaches decreases when markers are occluded, which could occur when a user is interacting with an object. Moreover, motion capture systems require non-standard hardware and so are an expensive solution. Our work uses standard hardware and does not require additional attachments.

On the other hand, a physical object can be tracked in RGB or RGBD data using detected feature on the surface of an object, without the need for additional markers [23, 27, 32, 34]. Many of the state of the art approaches, though fast and accurate enough for VR, only consider tracking rigid objects and do not adapt well to objects which can deform [14,32,34]. In contrast, we consider both rigid and articulated objects. Rather than just determining the 6DoF pose of an object, the shape can be calculated by fitting a non-rigid model to the RGB or RGBD data [6,8,11,33]. Novel approaches from Tsoli *et al.* [33] and Zhang *et al.* [41] consider hand and non-rigid object tracking as a joint task. Tsoli *et al.*'s [33] work demonstrates the potential of jointly optimising hand pose and non-rigid object pose and shape but is not fast enough for VR applications. In contrast, Zhang *et al.*'s [41] *InteractionFusion* solution is suitable for real-time use. In their approach, they use two RGBD cameras which must be calibrated. In contrast, our work uses a single RGBD camera.

Neural networks: Neural networks have been used successfully for tracking objects, as well as for predicting model parameters for rigid and non-rigid objects [10, 29, 30, 42]. Works such as PoseCNN [39] and PVNet [24] demonstrate accurate 6DoF pose predictions from RGB images, even in complex, uncontrolled environments [1, 21, 38, 40]. However, these approaches only predict rigid parameters and assume the objects which they are tracking are rigid. Thus, the prediction accuracy decreases if an object undergoes a deformation. In this paper, we propose a network which can predict parameters of articulated models as well as rigid parameters.

Neural networks can be used to capture non-rigid deformations by predicting non-rigid or articulated model parameters, vertex offsets or voxels [7, 10, 26, 29, 30, 37, 42]. Kanazawa *et al.* [7] use a network consisting of an encoder and discriminator to predict the shape and pose of a human mesh, as well as the camera pose, from a single RGB image. Similarly, Zuffi *el al.* [42] use a convolutional network to recover the camera pose and the shape and pose of a zebra mesh from a single image. However, both these approaches, and many of the other state of the art works, are trained on manually labelled data. This is time consuming, and sometimes difficult to obtain for an arbitrary object. In contrast, we train our network on synthetic data and provide a method for automatically generating large datasets for arbitrary objects without the need for manual labelling. Wang *et al.* [37] also train their network – which predicts an object's deformation given an external force – on synthetic data generated using a physics engine. When combined with a GAN, they are able to predict an object's deformation – in the form of a voxel grid – from a depth map. While this network can accurately predict physical deformations, it is unable to recover the 6DoF pose of the object. Taylor *et al.* [29, 30] design a neural network based pipeline – *VRProp-Net* – for using physical objects to control the behaviour of virtual objects. While they can recover rigid or articulated parameters for a chosen object they are limited to green screen environments and require the use of green gloves. In our work, we extend this method and remove the need for such strict constraints.

Egocentric Tracking: A tracking approach in which the camera pose is fixed has a restricted capture volume. On the other hand, using a moving egocentric viewpoint where the camera is attached to the VR HMD, creates a dynamic capture volume suitable for practical VR experiences. An egocentric view has been used for accurately tracking hand or hand and object pose [9, 17, 22, 31]. Kapadis *et al.* [9] present a network for hand and object detection and action recognition in egocentric views. However, the objects are simply detected in the 2D image and no 3D pose information is recovered. For our application, it is not enough to simply track the object, the network must recover the shape and pose. In contrast, Tekin *et al.* [31] predict the 3D pose of hand and objects from an egocentric view. Similarly, Pandey *et al.* [22] predict the 6DoF pose of a controller captured from a pair of stereo monochrome cameras, such as those found on a HMD. However, these approaches do not extend to non-rigid objects. In our work, we predict the shape and pose of rigid and articulated objects from egocentric views.

3 Approach

This section will outline our approach for using physical objects to interact with virtual objects or *VR props* [29, 30]. Section 3.1 describes the offline procedure for generating the synthetic data and Section 3.2, explains how the data is processed before being used for network training. Our network architecture – *VRProp-Net+* – is discussed in Section 3.3, before, in Section 3.4, showing how this network can be used to make predictions on real-world data for known objects. In Section 3.5, we present an approach for adding a moving camera into our system and carrying out egocentric tracking. Finally, we describe the implementation details in Section 3.6.

3.1 Synthetic Dataset Generation

For each VR prop, we generate a synthetic dataset. This allows large amounts of data to be obtained quickly for an arbitrary object, without a time consuming manual capture process [1, 2, 16, 26].

We model our chosen props using a linear blend shape model which can be manually sculpted and rigged or, using Taylor *et*

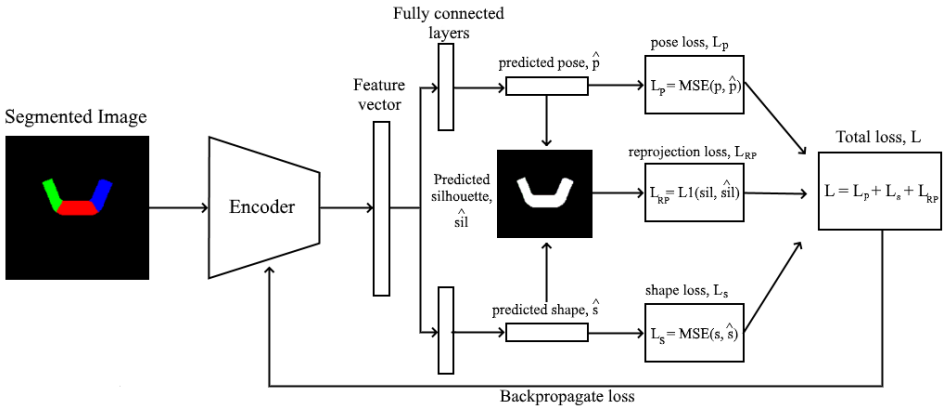


Figure 2: **VRProp-Net+**: A segmented image is input to the network and the predicted pose and shape parameters returned. The MSE of the Euclidean distance between the ground truth and predicted values is calculated for each branch. The predicted parameters can be used to update the virtual model and the predicted silhouette rendered differentiably. An L1 loss between the ground truth and predicted silhouettes is used as a reprojection error. The total error is backpropagated through the network.

al.'s [30] pipeline, generated automatically from a 3D scan of the object. The object is deformed by randomly varying the model parameters and a synthetic RGB image rendered for each frame. As discussed by Taylor *et al.* [30], depth data could be used as an alternative. However, the state of the art methods which use depth data are either unable to recover pose due to ambiguities for certain objects [25, 37] or are not suitable for real-time use [12]. On the other hand *DynamicFusion* [19] and *VolumeDeform* [6] are able to capture non-rigid behaviour in real-time from RGBD data. However, DynamicFusion cannot recover from model failure or lost tracking and VolumeDeform often fails when the deformation is large. Moreover, the addition of depth in the training data increases processing time and decreases frame rate.

The rigid (or pose) parameters of the model are the 3D position, $\mathbf{T} = [T_x, T_y, T_z]$, and the rotation, $\mathbf{R}_{3\times3}$. These can be randomly varied with the constraint that the object must fall within the view port of the virtual camera, **VP**. Thus, to vary the 3D position of the object we simply chose a point $\mathbf{T} \in \mathbf{VP}$. The orientation of the object can be changed by sampling the angle of rotation around the x, y, z axes, rot_x, rot_y, rot_z, from a uniform distribution in the range $(0, 2\pi)$. These can be combined to a single 3×3 rotation matrix by multiplying together the individual matrices which rotate around the individual x, y, z axes, $\mathbf{R} = \mathbf{R}_x(rot_x)\mathbf{R}_y(rot_y)\mathbf{R}_z(rot_z)$.

In a blend shape model a deformation, $\mathbf{v}_{new}$, can be modelled as a linear combination of the n basis vectors (i.e blend shapes), $\mathbf{b} = [\mathbf{b}_0, ..., \mathbf{b}_n]$, and the neutral pose, $\mathbf{V}_{neutral}$, as expressed in the following equation.

$$\mathbf{v}_{new} = \mathbf{V}_{neutral} + \sum_{i=1}^{n} \mathbf{w}_i \mathbf{b}_i. \quad (1)$$

Thus, we take the articulated (or shape) parameters to be the blend shape weights $\mathbf{w} = [w_1, w_2, ..., w_n]$, which are uniformly sampled between $(0, 1)$. For each frame, the new parameters are used to update the virtual object's pose and shape and an RGB image of the scene is rendered. The virtual camera parameters are set equal to those of the physical camera so that the real and synthetic images are as similar as possible. While we use a moving camera in our experiments on real data, in the dataset generation we assume that the camera position and orientation is fixed at the origin.

3.2 Image Processing

Our network is trained on synthetic data but at run-time must make predictions on real data. Thus, to minimise the differences between these two datasets the images are first processed, by segmenting and flattening the colours and then cropping the image, before being input to the network. We propose a robust Gaussian Mixture Model (GMM) based approach for the image processing which is not restricted to controlled green screen environments. The chosen objects are textured brightly with distinct colours to aid tracking.

To begin, a GMM, which consists of K Gaussian distributions, as defined in Equation 2, is fit to each coloured section of the tracked object using the expectation maximisation (EM) algorithm.

$$\mathcal{N}(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{K}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big) \quad (2)$$

where $\mathbf{X}$ is a random variable, $\boldsymbol{\mu}_k \in \mathbb{R}^K$ is the mean of the kth Gaussian and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{K\text{x}K}$ is the covariance. Once the GMMs have been learnt, they remain constant throughout the tracking.

To segment and flatten an image, each pixel, $\mathbf{x}$, is tested using the Mahalanobis distance (Equation 3) to see if it belongs to any of the learnt distributions.

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma} (\mathbf{x}-\boldsymbol{\mu})}. \quad (3)$$

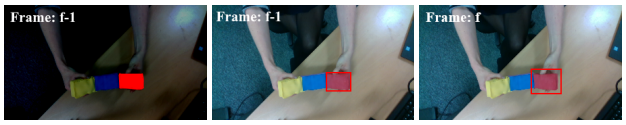
If D is below a chosen threshold, $D < \varepsilon$, then the pixel belongs to that distribution. The value of ε is selected experimentally. To segment the image, any pixels which do not belong to any of the GMMs are assumed to be background pixels and so are removed. The flattening stage removes any colour variations within object sections which may arise to lighting variations. This is done by setting all pixels which fall into the same GMM to a constant value.

To increase the efficiency of the processing algorithm and reduce the effect of background noise, we do not test every single pixel in each frame to determine if it belongs to any of the GMMs. Instead, for each object section, we test pixels within a region of interest (ROI). To determine the ROI for a section, we first find a bounding box around the segmented section in the previous frame. We take the assumption that the position of the section in the image does not vary greatly between frames and obtain the ROI by simply increasing the dimensions of the bounding box, as demonstrated in Figure 3.

Finally, the segmented and flattened image, $\mathbf{I}$, is cropped around the centre of the tracked object, obtaining a square input for the network. We take the object centre to be the 2D centroid, calculated from the non-zero pixels.

3.3 VRProp-Net+

We propose a novel CNN based architecture, *VRProp-Net+* (see Fig. 2) that is used to predict the pose, $\hat{\mathbf{p}}$, and shape, $\hat{\mathbf{s}}$, parameters



(a) Segmented region. (b) Bounding box. (c) Region of interest.

Figure 3: Finding the ROI of the red section in frame f. A bounding box is calculated around the segmented region in the previous frame $(f-1)$ and then the dimensions increased to find the ROI in f.

of VR props from RGB images, $\mathbf{I}$

$$f(\mathbf{I}) = \hat{\mathbf{p}}, \hat{\mathbf{s}} \tag{4}$$

where $\hat{\mathbf{p}} = [\hat{R}_{1,1}, \hat{R}_{1,2}, ..., \hat{R}_{3,3}]$ are the entries of the rotation matrix and $\hat{\mathbf{s}} = [\hat{w}_1, \hat{w}_2, ..., \hat{w}_n]$ are the weights of the blend shape model.

VRProp-Net+ takes in a processed image and passes it through an encoder, consisting of the basic blocks of VRProp-Net [29]. The encoder layers are then followed by a batch normalisation, producing a feature vector. As in the work by Zuffi *et al.* [42], our network has a separate branch for the pose and shape predictions. The shape prediction branch takes the feature vector as an input and passes it to a ReLu and then linear layer. This returns shape parameters. Similarly, the feature vector is passed into the the pose prediction layer, which again passes it through a ReLu and linear layer. In this instance, the pose parameters are returned. The network is trained, separately for each object, using the loss

$$L = L_s + L_p + cL_{rp} \tag{5}$$

where L_s is the shape loss, L_p is the pose loss and L_{rp} is the reprojection loss. The variable c is a constant which we determined experimentally. The shape loss, L_s, is defined as the mean square error (MSE) between the ground truth and predicted blend weights. Similarly, the pose loss, L_p, is the MSE between the ground truth and predicted rotation matrix entries. Finally, we have the reprojection loss, L_{rp}, which measures the difference between the silhouette of the ground truth and predicted mesh. Using the predicted pose and shape parameters, the virtual object can be deformed and the predicted silhouette can be differentiably rendered using Kato *et al.*'s [10] neural mesh renderer (NMR). The NMR approximates rasterisation such that it has a gradient which can be back-propagated through a network, making it suitable for use within a network loss function. Thus, L_{rp} can be found to be the $L1$-loss between the ground truth and the predicted silhouette. After each epoch, the loss is back-propagated through the network to update the parameters weights, until the error converges or falls below a chosen threshold.

3.4 Real world predictions

Once trained, VRProp-Net+ can predict the pose and shape parameters of a known physical object in RGB images, captured from a single RGBD camera. In turn, the predicted parameters can update the behaviour of a computer generated model in the VR environment.

As with the synthetic data, the RGB image is segmented and flattened. The depth map can be segmented using the processed RGB image as a mask and then both the RGB images and depth map cropped around the 2D object centroid. The 3D position of the object is found by back-projecting the centroid using the camera intrinsic matrix. The processed image is input to VRProp-Net+ which returns the predicted parameters and these used alongside the calculated 3D position to deform the virtual model.

As noted in the work by Taylor *et al.* [30], the change in shape and pose between two subsequent frames only varies a little. Therefore, to increase the speed of the algorithm, a prediction is made only for a set of key frames and the parameters interpolated between these. The orientation is smoothed using spherical linear interpolation and the blend shape weights and 3D position averaged over the current and previous 2 frames. Thus, while the network does not make predictions in real-time, the overall tracking framework is able to run at interactive rates.

3.5 Egocentric Object Tracking

We will now discuss a method to allow our network to make predictions from a moving egocentric view (i.e a sensor attached to a HMD) such that it may be used within an egocentric tracking framework. To do so, we ensure that the method correctly differentiates between camera and object movements. Within our dataset generation, we fix the camera position to $(0,0,0)$, and set the orientation to be the 3×3 identity matrix, $\mathbf{I}_3$ (i.e. the virtual camera is not rotated). Thus, we can consider that the predicted object pose from VRProp-Net+ is within a camera coordinate system - defined by the camera pose. The shape prediction is independent of coordinate system.

We begin by finding the 3D position and orientation of the moving camera in world space. To define the world coordinate system, we place fiducial markers in the tracking volume. These markers have known pose and are arranged in a plane, the centre of which we choose as our world origin. The addition of multiple markers allows the camera pose to be found even if some of the markers are occluded. In our implementation, we use Aruco fiducial makers [3] as these allow fast and accurate identification. Using the plane of marker, the camera pose in the world coordinate system can be quickly calculated. The details the algorithm used can be found in Garrido-Jurado *et al.*'s [3] original paper.

The camera pose in world space can be used to update the camera frame. As the object pose is relative to the camera pose, the object pose is transformed into world space by performing a coordinate transform into the new camera frame. The orientation of the object is transformed by simply rotating it into the updated coordinate system. For the position, we must rotate into the new frame as well as translating by the change in origin position between frames. The pose of the object in world space, alongside the predicted shape parameters, can be used to update the virtual model. This in turn can be rendered into the VR application or experience.

3.6 Implementation Details

We tested our system on 4 VR props: a rigid cube, a rigid cup, an articulated 'pizza' and an articulated sponge (see Fig. 5). The rigid objects were represented by triangular meshes and the articulated objects were blend shape models, each with 2 blend shapes. The objects were coloured brightly to aid tracking and prevent symmetry.

Dataset generation: A dataset was generated for each object in Unity, as discussed in Section 3.1. Our virtual camera is modelled on an Intel Realsense D435 sensor and so has parameters $\mathbf{f} = [622.084, 622.154]$ and $\mathbf{u} = [426.034, 245.07]$. The images are rendered with dimensions (848×480) and cropped to (384×384).

Image processing: To segment and flatten the tracked object, a GMM with $K = 2$ clusters was learnt for each object section and the threshold for the Mahalanobis distance was selected through experimentation as $\varepsilon = 2$. We chose 2 clusters so that the GMM can appropriately model the variation that can occur on a section of the same colour due to shading and lighting, while keeping the segmentation algorithm fast and efficient.

Network: VRProp-Net+ was implemented in Pytorch and trained on a desktop computer with an NVIDIA GeForce GTX 1070 GPU and an Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz. We use Adam optimiser with a learning rate of $1e-4$. In our training procedure, the non-rigid parameters are standardised with the condition that the mean blend weight is 0 and the standard deviation is 1. The network is trained separately for each object. We chose the loss constant, $c = 0.5$, as through experimentation found that this provided the highest accuracy predictions.

Real world predictions and egocentric Tracking: We capture our physical objects with an Intel Realsense D435 RGBD sen-

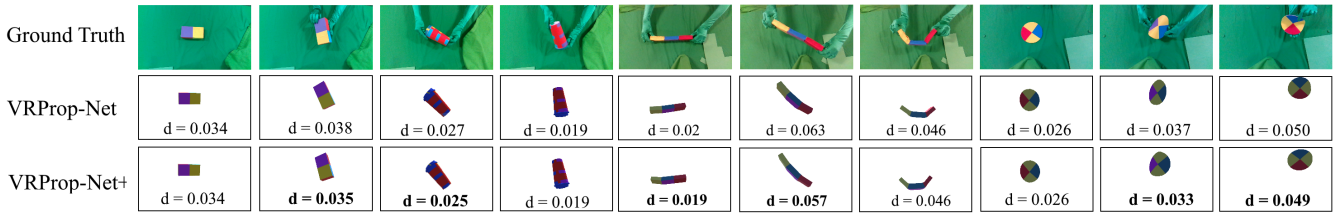


Figure 4: Comparison of the Euclidean distance, d, between real and predicted silhouettes for VRProp-Net [29] vs VRProp-Net+.

sor which was attached to the HMD of an Oculus Rift. We use Aruco [18] within OpenCV as the chosen fiducial marker library.

4 Results and Discussion

In this section, we demonstrate and discuss the results of our approach on several rigid and articulated objects. We begin by comparing the prediction accuracy of our VRProp-Net+ architecture against the closest work in literature which considers both rigid and articulated objects, VRProp-Net [29], for fixed camera sequences. We also show egocentric object tracking for our chosen objects.

Fixed camera: For each object, we trained VRProp-Net and VRProp-Net+ and used the trained network to make predictions on unseen synthetic sequences. The predicted parameters were used to deform the blend shape model and the root mean square (RMS) error between the predicted model vertices and the ground truth mesh vertices from the synthetic sequence calculated for each network.

Object	Object Type	VRProp-Net [29]	VRProp-Net+
Sponge	Articulated	0.428	**0.409**
Pizza	Articulated	0.681	**0.618**
Box	Rigid	0.160	**0.125**
Cup	Rigid	0.581	**0.423**

Table 1: Comparison of the average RMS error (cm) for a synthetic sequence of 100 frames between VRProp-Net [29] and our VRProp-Net+ for a range of rigid and articulated objects.

Table 1 shows our network was consistently able to make higher accuracy predictions than VRProp-Net for the range of rigid and articulated objects. The outputs of the 2 networks on real-data can also be compared (see Fig. 4). These tests were carried out in green screen environments as required by Taylor *et al.*'s [29, 30] work. However, our approach can make predictions in more complex scenes (see Fig. 5). Again, we see that visually both networks make appropriate predictions for the range of objects. The Euclidean Distance, d, between the real object silhouette and the silhouette of the predicted mesh can be calculated to evaluate how closely the reconstruction matches the input. We find that the predicted meshes in our network are slightly closer to the ground truth than VRProp-Net. Moreover, our system has a mean frame rate of $20 fps$, which is faster than Taylor *et al.*'s [29, 30] frame rate of $15 fps$.

We have textured our objects brightly to aid tracking, while several state-of-the art works are able to track objects using their natural texture [7, 39]. However, we are not limited to using manually labelled training data and can track both rigid and articulated objects.

Egocentric tracking: Finally, we demonstrate our egocentric tracking approach by testing our pipeline on sequences which contain a moving camera and a moving rigid or articulated object (as seen in Fig. 5). We see that for each object, our approach can accurately predict its shape and pose, even without green screens, and can correctly distinguish between camera movements and object movements. Moreover, our results demonstrate that with a moving egocentric view the capture volume need not be restricted to a pre-selected area defined by multiple fixed cameras. Instead, the capture volume is dynamic, with the only constraint being that at least one fiducal marker is reliably detected in the camera image. In future work, a valuable exploration could be carried out into how the robustness of the pose and shape prediction is affected by the size of the capture volume. In our work we have used markers to find the object pose, allowing us to have a complete framework for egocentric rigid and articulate object tracking. However, we acknowledge that a key future task would be removing the markers and determining the camera pose using features of the scene.

5 Conclusion

In this paper we have proposed a neural network based tracking system with the goal of allowing computer generated objects in virtual environments to be controlled using physical objects. We also present an egocentric tracking solution in which the camera can be attached to a HMD. The moving egocentric view creates a dynamic capture volume, allowing our system to be used in practical, everyday VR environments without the need for multiple fixed cameras tracking the object. We demonstrate our method on several rigid and articulated objects and show both fast and accurate tracking results. In addition, our network – VRProp-Net+ – achieved higher accuracy predictions than the state of the art methods for transporting real objects into virtual environments and, with our proposed segmentation algorithm, is not restricted to controlled (e.g. green screen) environments. As a future task, the system can be extended to include multiple objects and the human-object interactions as well as the object-object interaction explored. In our work, we have represented our articulated objects as blend shape models. However, in order to include a variation of objects with a variety of non-rigid behaviours into a VR scene, different models (e.g finite element meshes or rigged skeletons) could be used instead and the parameters of these models learnt by a network.

References

[1] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018.

[2] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *IEEE European Conf. on Comp. Vision (ECCV)*, 2016.

[3] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 2014.

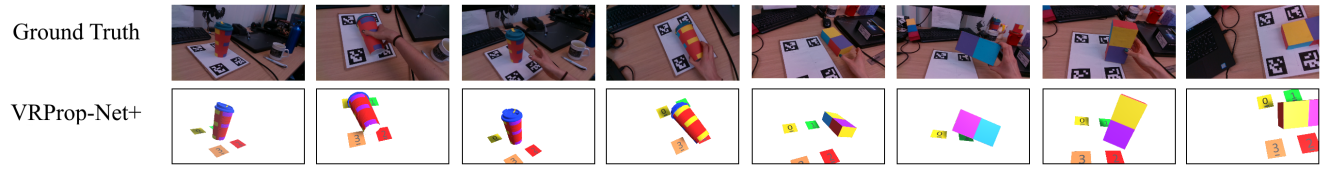
[4] HTC. Vive. `https://www.vive.com/uk/`.

[5] HTC. Vive tracker. `https://www.vive.com/uk/vive-tracker/`.

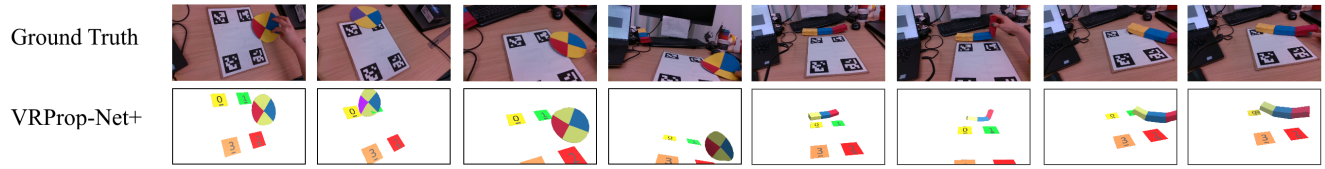
[6] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *IEEE European Conf. on Comp. Vision (ECCV)*, 2016.

[7] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *IEEE Conf. on Comp. Vision and Pattern Recognition*, 2018.

[8] A. Kanazawa, S. Kovalsky, R. Basri, and D. Jacobs. Learning 3d deformation of animals from 2d images. In *Comp. Graphics Forum*. Wiley Online Library, 2016.



(a) Rigid sequences



(b) Articulated sequences

Figure 5: Predicted pose and shape for different objects with moving camera with an egocentric view. VRProp-Net+ predicts the shape and pose in the camera frame and these transformed into the world coordinate system.

[9] G. Kapidis, R. Poppe, E. van Dam, L. P. Noldus, and R. C. Veltkamp. Egocentric hand track and object-based human action recognition. *arXiv preprint*, 2019.

[10] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2018.

[11] L. Kausch, A. Hilsmann, and P. Eisert. Template-based 3d non-rigid shape estimation from monocular image sequences. In *conf. on Vision, Modeling and Visualization*. Eurographics Association, 2017.

[12] I. Leizea, A. Mendizabal, H. Alvarez, I. Aguinaga, D. Borro, and E. Sanchez. Real-time visual tracking of deformable objects in robot-assisted surgery. *IEEE comp. graphics and applications*, 2017.

[13] MagicLeap. Magic leap. `https://www.magicleap.com/`.

[14] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother. Global hypothesis generation for 6d object pose estimation. In *IEEE Conf. on Comp. Vision and Pattern Recognition*, 2017.

[15] Microsoft. Microsoft hololens — mixed reality technology for business. `https://www.microsoft.com/en-us/hololens`.

[16] V. Mondjar-Guerra, S. Garrido-Jurado, R. Muoz-Salinas, M. J. Marn-Jimnez, and R. Medina-Carnicer. Robust identification of fiducial markers in challenging conditions. *Expert Syst. Appl.*, 2018.

[17] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *IEEE Int. Conf. on Comp. Vision*, 2017.

[18] R. Munoz-Salinas. Aruco: a minimal library for augmented reality applications based on opencv. *Universidad de Córdoba*, 2012.

[19] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE conf. on comp. vision and pattern recognition*, 2015.

[20] Oculus. Oculus rift. `https://www.oculus.com/rift/`.

[21] A. Palazzi, L. Bergamini, S. Calderara, and R. Cucchiara. End-to-end 6-dof object pose estimation through differentiable rasterization. In *European Conf. on Comp. Vision (ECCV)*, 2018.

[22] R. Pandey, P. Pidlypenskyi, S. Yang, and C. Kaeser-Chen. Efficient 6-dof tracking of handheld objects from an egocentric viewpoint. In *IEEE European Conf. on Comp. Vision (ECCV)*, 2018.

[23] Y. Park, V. Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. In *IEEE Int. symp. on Mixed and Augmented Reality (ISMAR)*, 2008.

[24] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *IEEE Conf. on Comp. Vision and Pattern Recognition*, 2019.

[25] A. Petit, V. Lippiello, G. A. Fontanelli, and B. Siciliano. Tracking elastic deformable objects with an rgb-d sensor for a pizza chef robot. *Robotics and Autonomous Systems*, 2017.

[26] A. Pumarola, A. Agudo, L. Porzi, A. Sanfeliu, V. Lepetit, and F. Moreno-Noguer. Geometry-aware network for non-rigid shape prediction from a single view. In *IEEE Conf. on Comp. Vision and Pattern Recognition*, 2018.

[27] J. Rambach, A. Pagani, and D. Stricker. [poster] augmented things: Enhancing ar applications leveraging the internet of things and universal 3d object tracking. In *IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*. IEEE, 2017.

[28] RoadToVR. Optitrack shows hundreds of simultaneously tracked objects in single vr experience. `https://www.roadtovr.com/optitrack-hundreds-of-tracked-objects-jenga-gdc-2019/amp/`.

[29] C. Taylor, R. McNicholas, and D. Cosker. Vrprop-net: Real-time interaction with virtual props. In *ACM SIGGRAPH Posters*, 2019.

[30] C. Taylor, C. Mullanay, R. McNicholas, and D. Cosker. Vr props: An end-to-end pipeline for transporting real objects into virtual and augmented environment. In *IEEE Int. Symp. on Mixed and Augmented Reality*, 2019.

[31] B. Tekin, F. Bogo, and M. Pollefeys. H+ o: Unified egocentric recognition of 3d hand-object poses and interactions. In *IEEE Conf. on Comp. Vision and Pattern Recognition*, 2019.

[32] H. Tjaden, U. Schwanecke, and E. Schomer. Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In *IEEE Int. Conf. on Comp. Vision (ICCV)*, 2017.

[33] A. Tsoli and A. A. Argyros. Joint 3d tracking of a deformable object in interaction with a hand. In *IEEE European Conf. on Comp. Vision (ECCV)*, 2018.

[34] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall. Capturing hands in action using discriminative salient points and physics simulation. *CoRR*, 2015.

[35] Vicon. Motion capture systems. `https://www.vicon.com/`.

[36] Vicon. Origin. `https://www.vicon.com/press/2018-08-13/origin-by-vicon`.

[37] Z. Wang, S. Rosa, B. Yang, S. Wang, N. Trigoni, and A. Markham. 3d-physnet: Learning the intuitive physics of non-rigid object deformations. *arXiv preprint*, 2018.

[38] D. Wu, Z. Zhuang, C. Xiang, W. Zou, and X. Li. 6d-vnet: End-to-end 6-dof vehicle pose estimation from monocular rgb images. In *IEEE Conf. on Comp. Vision and Pattern Recognition Workshops*, 2019.

[39] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn:a convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint*, 2017.

[40] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. In *IEEE Int. Conf. on Comp. Vision (ICCV)*, 2019.

[41] H. Zhang, Z.-H. Bo, J.-H. Yong, and F. Xu. Interactionfusion: Real-time reconstruction of hand poses and deformable objects in hand-object interactions. *ACM Trans. Graph.*, 2019.

[42] S. Zuffi, A. Kanazawa, T. Berger-Wolf, and M. J. Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild". In *IEEE Int. Conf. on Comp. Vision (ICCV)*, 2019.