Poster: Pose-assisted Active Visual Recognition in Mobile Augmented Reality

Bing Zhou[1], Sinem Guven[2], Shu Tao[2], Fan Ye[1]
[1] Stony Brook University, [2] IBM Thomas J. Watson Research Center
{bing.zhou,fan.ye}@stonybrook.edu,{sguven,shutao}@us.ibm.com

ABSTRACT

While existing visual recognition approaches, which rely on 2D images to train their underlying models, work well for object classification, recognizing the changing *state* of a 3D object requires addressing several additional challenges. This paper proposes an active visual recognition approach to this problem, leveraging camera pose data available on mobile devices. With this approach, the *state* of a 3D object, which captures its appearance changes, can be recognized in real time. Our novel approach selects informative video frames filtered by 6-DOF camera poses to train a deep learning model to recognize object state. We validate our approach through a prototype for Augmented Reality-assisted hardware maintenance.

CCS CONCEPTS

• **Human-centered computing → Mixed / augmented reality**; • **Computing methodologies → Activity recognition and understanding**;

KEYWORDS

active visual recognition; augmented reality; mobile devices

ACM Reference Format:
Bing Zhou[1], Sinem Guven[2], Shu Tao[2], Fan Ye[1]. 2018. Poster: Pose-assisted Active Visual Recognition in Mobile Augmented Reality. In *The 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18), October 29-November 2, 2018, New Delhi, India.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3241539.3267771

1 INTRODUCTION

Augmented Reality (AR) has become more prevalent in recent years, especially with the emergence of AR-enabled mobile phones. However, there are still limitations in existing technologies that prevent wider adoption of AR, one of which is lack of fine-grained visual recognition. Take technical support, a well-recognized use case for AR, as an example: AR is transforming traditional hardware repair guidance into a more intuitive and engaging experience by superimposing step-by-step 3D animated instructions on the hardware [2]. Nevertheless, existing solutions require the user to analyze the current scene, and to manually choose an applicable set of instructions to view through AR. This is not only cumbersome from user experience point of view, but also means that the user needs to have sufficient knowledge about the hardware being repaired to select the right AR content, thereby limiting the value of AR in this use case. To realize the true value of AR-assisted repair, the system needs to automatically understand the state of hardware being repaired, and present corresponding instructions to the user.

Figure 1 shows two example steps of replacing the CPU module on a server. In the proposed scenario, each state of the server will be automatically recognized, then the corresponding next-step instructions will be automatically presented to the user. Implementing such state recognition poses several new challenges, compared to traditional visual recognition methods [1]: *i) 3D object recognition requires complete visual perception, which usually cannot be captured by a single camera shot.* For instance, in Figure 1, multiple images from different viewing angles are needed to cover all components (e.g., screws, manifold, heat sink, etc.). *ii) Some states of an object may only be recognizable from few viewing angles, and conventional algorithms cannot easily recognize that, in order to differentiate these states.* For example, the server in Figure 1 cannot be recognized as in the state of 'Heat Sink Removed', unless it is viewed at an angle like that of the second image. *iii) The solution must be designed to work with the constraints of computation and power resources on mobile devices.*

A unique advantage of running AR on mobile devices is the ability to keep track of the 6-DOF pose (3D position and orientation) of a device, which can be derived from camera

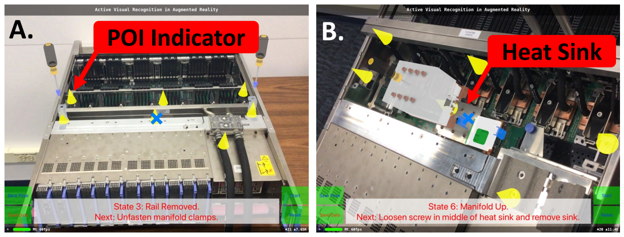
Active Visual Recognition in Augmented Reality
A.
POI Indicator
State 3: Rail Removed.
Next: Unfasten manifold clamps.
Active Visual Recognition in Augmented Reality
B.
Heat Sink
State 6: Manifold Up.
Next: Loosen screw in middle of heat sink and remove sink.

Figure 1: Example states of a server under repair.

and inertial sensor data, offers opportunities for fine-grained active visual recognition. Specifically, we propose to leverage device pose data, in addition to camera-captured images, to accurately identify the state of an object, and use that to enhance AR experiences. Our contributions include:

- A video frame filtering mechanism that identifies the most informative frames using camera pose, enabling the "scan and recognize" capability.
- A deep learning pipeline that runs on mobile devices and aggregates visual information from multiple video frames for robust object state recognition.
- A prototype running on mobile devices to demonstrate our approach's practicality when applied to AR-assisted hardware maintenance scenarios.

2 SYSTEM DESIGN

Our system works in three phases: *pose-controlled video frame filtering*, *Convolutional Neural Network (CNN) based image feature extraction*, and *visual information aggregation*, as shown in Figure 2.

2.1 Pose-controlled Video Frame Filtering

In this phase, the most informative video frames at certain poses are selected, while mobile device is scanning the object. It is critical for both collecting data to train the visual recognition model, and for applying the trained model to infer object state. Our design works as per the following three steps:

Initial Pose Calibration. We first map both the object and mobile device into the same coordinate system, so that AR experiences can be launched consistently, with respect to the object and the relative camera pose. To achieve this, the user simply needs to point the mobile camera at a fixed location on the object (e.g., a barcode marker on the machine), and set that as the origin of the tracking coordinates.

Pose Clustering and Frame Labeling. We next determine camera poses from which the captured video frames are the most informative and can best distinguish different states of the object. We first identify the "point of interest" (POI) on the object (indicated as yellow marks in Figure 1). POIs are sub-areas of an object where the appearance is subject to change (e.g., a removable portion of the machine) [1].

[1]In our current design, we assume that human experts will mark these POIs. As future work, we are developing mechanisms to extract the POIs automatically from video data.

Since a POI may only be observable from a certain angle, we need to determine the camera poses that can capture usable images to recognize the appearance changes of each POI (hence the change of object state). When collecting training data, the user shall scan each POI thoroughly to gather images from different angles. Meanwhile, the camera pose data corresponding to the images are also recorded. When training the model, we first run a clustering algorithm on camera pose data to determine the 'pose groups', each of which corresponds to a POI. As a result, we can label images with pose groups to train the visual recognition model. And during inference, images fed from camera can also be automatically associated to POI based on the corresponding camera pose data.

Video Frame Filtering. Given the mechanism above, video frames can be labelled by POI. When training the model, we use all captured frames. When inferring object state, we need to select the best quality video frames as the user scans the object with camera. For that purpose, our system visually guides the user to point the camera to POIs, until the scan results in sufficient video capture of these areas. To select the best quality frames, we use inertial sensor data (accelerometer and gyroscope) to determine the most "stable" video frames for each POI when the camera is scanning. The video frames captured when the mobile device had minimum acceleration and rotation are selected as input.

2.2 CNN based Image Feature Extraction

We tune a pre-trained ResNet50 [1] as our base model for image feature extraction. We remove the last fully connected layer of the model, thus getting a 2048-D feature vector for each input image. The inference only runs whenever there is video frame update for a POI. The image features extracted for other unchanged POIs are reused, if no video frame update is available. This significantly reduces the computation overhead and battery consumption on mobile devices.

2.3 Visual Information Aggregation

We aggregate all extracted image features to feed into a fully-connected deep neural network, using an approach similar to MVCNN [4], to classify them into different object states. All image features are concatenated as input for the aggregation model. This model inference runs in real time, whenever there is any update from the input features.

3 IMPLEMENTATION

We implemented the proposed system in two parts: a *Model Training* program and a *Mobile App*.

Model Training. The training program tunes the *ResNet50* network and trains the customized aggregation model, with filtered video frames, device pose, and corresponding object

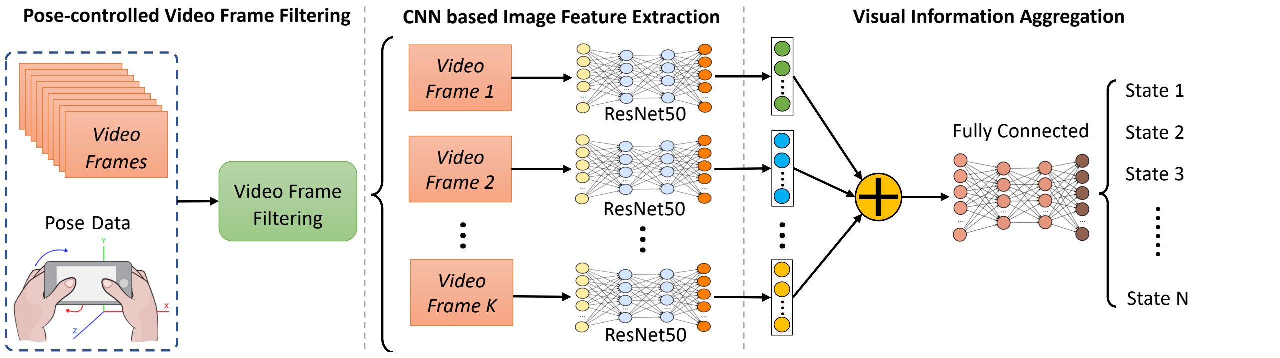

Figure 2: Video frames are filtered by pose information, and fed into a fine-tuned ResNet50 for feature extraction. A neural network is designed for aggregating visual information from filtered video frames for final classification.

states as input data. To collect training data, we scan the object at each state for approximately 2*mins*. After the model training, we convert the models to Core ML [3] format to run on iOS platform.

Mobile App. We developed an iOS-based mobile application with four modules: AR tracking, frame filtering, model inference, and rendering. We use ARKit [3] for collecting device pose data and for object tracking when rendering AR experiences. The frame filtering component selects appropriate ones as training data (in training model) or as input data (in inference mode). We run model inference using the Core ML framework, which is optimized for on-device performance by minimizing memory footprint and power consumption. 3D animations are rendered in the AR scene using SceneKit [3]. For our evaluation, we run this app on an iPad with dual core 1.8GHz A9 processor and 2GB memory.

4 EVALUATION

With the setup above, we collected about 15000 filtered video frames that represent 8 different states of a server.

State Recognition Accuracy. For each of the 8 states, we use our mobile app to scan the server and infer the current state. We repeat the recognition test 50 times for each state. As shown in Table 1, the recognition accuracy is 100% for all states except state 3 and 4, where the accuracy is 96% and 88%, respectively. These two states are easier to be confused because the visual difference between them are small: only two small screws are removed from state 3 to state 4. We can further improve the accuracy for these states, by instructing user to slow down camera movement or to adjust camera pose. Nevertheless, the overall results clearly demonstrated that our proposed solution is robust enough for practical use.

Resource Overhead. We measure the overhead of our active visual recognition mechanism on the mobile device, in CPU and memory usage, and in GPU time per frame. We compare these resource usage metrics when the mobile app is running in two modes: AR tracking only and AR with visual recognition. Table 2 shows the average values over running both modes for a duration of 3 minutes. The visual recognition increases the CPU usage by 12%, and consumes about 33MB more memory. The GPU time per frame is increased by about 10*ms* due to the inference of neural networks. With these overheads, our application can easily achieve 60 fps.

State	**1-2**	**3**	**4**	**5-8**
1-2	100%	0%	0%	0%
3	0%	96%	4%	0%
4	0%	12%	88%	0%
5-8	0%	0%	0%	100%

Table 1: Confusion matrix of results in percentage.

Mode	CPU	Memory	GPU Frame Time
AR Only	26%	95.5*MB*	1.3*ms*
AR + Visual	38%	128.4*MB*	11.4*ms*

Table 2: Resource consumption and frame time.

5 CONCLUSION

In this paper, we propose an approach that combines both visual and camera pose data to recognize the changing state of 3D objects. We develop a prototype system with an iOS-based mobile app, demonstrate the practicality of our approach in AR-assisted hardware maintenance.

REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[2] Steven J Henderson and Steven K Feiner. 2007. *Augmented reality for maintenance and repair (armar)*. Technical Report. Columbia Univ New York Dept of Computer Science.
[3] Apple Inc. 2018. Apple Developer Documentation. https://developer.apple.com/documentation.
[4] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*. 945–953.