MR2024-7

メタバースマップの継続的更新のための差分検出

松原 智也[*1]　杉本 麻樹[*1]　斎藤 英雄[*1]

Change Detection for Constantly Maintaining Up-to-date Metaverse Maps

Tomoya Matsubara[*1]　Maki Sugimoto[*1]　Hideo Saito[*1]

Abstract – Metaverse has been attracting more and more attention because of its potential for various use cases. In metaverse applications, the seamless integration of digital and physical worlds is vital for synchronizing information from one world to another. One way to achieve this is to reconstruct 3D environmental maps every time, which is not feasible due to computational complexity. In this paper, we propose a change detection method, combined with object classification, designed for efficiently updating the existing 3D environmental map as little as possible. Despite its simplicity, the experiment shows promising results with an object detector fine-tuned with data from the target environment.

Keywords : metaverse, digital twin, change detection, object detection

1 Introduction

Metaverse is a term that describes the concept of "beyond universe." It is a 3D virtual environment where virtual objects, including avatars that represent users, can interact with each other as well as with the surrounding physical environment [1]. The metaverse's idea of integrating virtual and physical worlds is common in Augmented Reality (AR) and Virtual Reality (VR), or more generally, Mixed Reality (MR). As multi-user online systems are involved, metaverse is often considered to be the next-generation Internet.

Because of its potential for a variety of use cases, metaverse has been attracting more and more attention in the industry. For example, VRChat[1], a social VR platform in which users can create their own worlds, invite people to them, and interact with them from all over the world, reports more than $25,000$ community created their worlds. In the domestic market, cluster[2] offers a similar platform. These platforms are based on VR, which enables interactions between users and virtual objects and thus does not support interactions with physical environments.

To achieve the integration of virtual and physical worlds, digital twins play an important role. Digital twins are virtual replications of physical entities, enabling data transmission between virtual and physical worlds [2]. These days, such 3D reconstructions can be obtained easily thanks to low-cost depth sensors (e.g., iPhone equipped with LiDAR). However, constantly keeping up-to-date 3D reconstructions is a challenging problem because reconstructing and replacing the whole environment is computationally expensive.

In this paper, we propose a change detection method, combined with object classification, to update the existing 3D environmental map as little as possible. By detecting changes between the existing map and newly captured images, the whole replacement could be avoided. Besides, detected changes provide event logs in the environment. Such information would be helpful, for example, in surveillance video analysis. Figure 1 depicts the overview of our change detection pipeline.

2 Related Works

2.1 3D Reconstruction

Sparse 3D reconstruction of an environment can be obtained from RGB-D images once camera parameters at each frame have been determined. This only involves simple matrix multiplication.

COLMAP [3, 4] is a widely used Structure-from-Motion (SfM) system for 3D reconstruction from unordered images. Its Multi-View Stereo (MVS) algorithm reconstructs dense representation after sparse reconstruction. Simultaneous Localization And Map-

[*1]慶應義塾大学
[*1]Keio University
[1]`https://hello.vrchat.com/`
[2]`https://cluster.mu/`

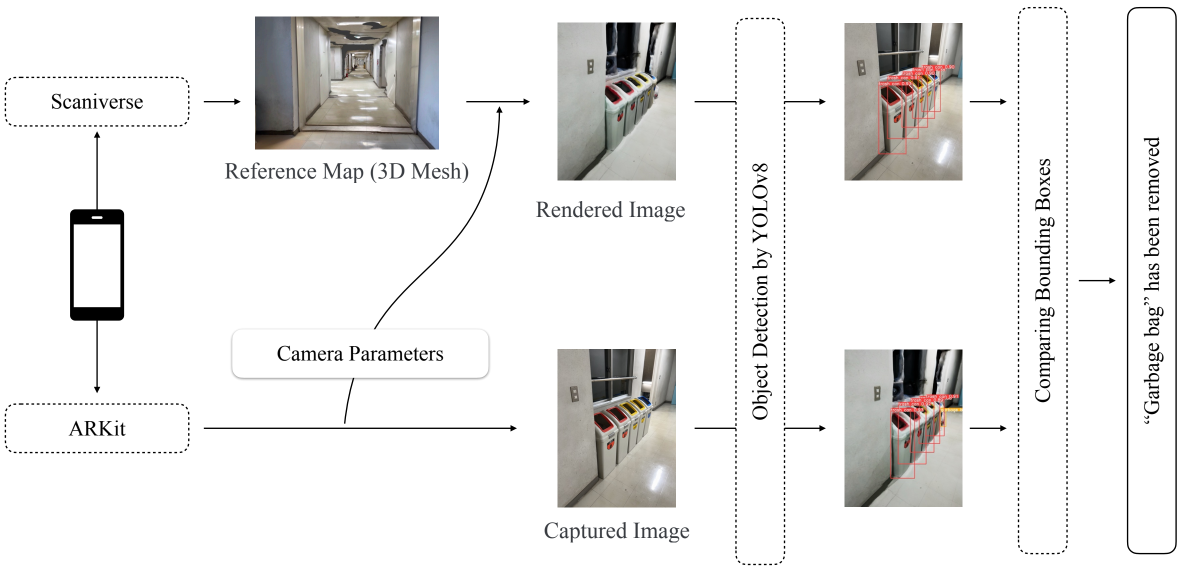


Fig. 1: Overview of our change detection pipeline.

ping (SLAM), which solves the localization of a robot, is another branch of techniques for 3D reconstruction [5, 6]. We refer readers to [7] for comprehensive reviews of recent Visual SLAM algorithms.

In addition to the reconstruction of the captured views, a lot of work on novel-view synthesis has recently been done [8, 9, 10]. Leveraging machine learning, these algorithms can provide scene renderings from a specific point of view using images captured from different points of view.

2.2 Object Detection

Object detection is a task to localize objects in images and classify them into certain categories, and it has a long history from hand-crafted features-based approaches such as the Viola-Jones algorithm [11], Histogram of Oriented Gradients (HOG) detector [12], and Deformable Part Model (DPM) [13] in the 1990s-2000s.

As deep neural networks started to surpass the performance of hand-crafted approaches, deep learning approaches have become mainstream since the 2010s. There are mainly two types of recent object detectors based on deep learning, two-stage detectors [14, 15] and one-stage detectors [16, 17]. Two-stage detectors first produce object proposals and then refine them while classifying them at the same time. One-stage detectors achieve this through a single pipeline.

2.3 Change Detection

Change detection aims to find the difference between two inputs, which are usually 2D images or 3D representations such as point clouds. Depending on the context, it is referred to as anomaly detection [18], discrepancy check [19], difference detection [20], and so on.

When the detected target is classified, especially in a pixel- or point-wise manner, it is referred to as semantic change detection. CSCDNet and SSCDNet [21] are 2D semantic change detection networks that leverage existing datasets for weakly supervised learning by dividing the task into change detection and semantic extraction. Our previous work on 3D semantic change detection [22] predicts semantic changes between two point clouds. While it does not require training, hyperparameters should be carefully tuned due to the rule-based nature of the algorithm.

3 Methods

Our change detection aims at detecting changes between an existing map, which will be called reference map hereafter, and newly captured images. The images are expected to be captured at different times from the reference map so that there are changes in the environment. We call the time the images are captured "current" in contrast to the "reference" map.

Our change detection pipeline is as follows. First, we build the reference map as a 3D mesh. Next, with some changes in the same environment (e.g., removals of objects), we capture images. Then, an object detection algorithm is applied to each frame of both the images and the renderings of the reference map from the same pose. Finally, comparing the detection results of each frame of the images and the renderings provides the change detection prediction.

3.1 Reference Map

It is worth noting that the reference map should be a dense representation with textures as our pipeline applies object detection to its rendering. As such, we use 3D mesh for the reference map.

In this paper, the reference map is reconstructed by Scaniverse[3], an iOS App for textured 3D model reconstruction for the sake of simplicity. Of course, more sophisticated reconstructions, such as novel-view synthesis [8, 9, 10], could provide more complete renderings. We leave that for future work.

3.2 Current Map

As for the current environment, it is not necessary to reconstruct 3D representations explicitly because we directly apply object detection to captured images. However, there is still a need to track the extrinsic parameters to render the reference map from the same poses as the captured images.

For that purpose, we use ARKit, a framework for AR applications in iOS. ARKit supports a variety of computer vision functionalities, including motion tracking and object capture. It also allows us to get estimated intrinsic and extrinsic parameters at each frame, which can be used to register the reference and current maps.

3.3 Object Detection

For object detection, we use YOLOv8[4], the latest version of YOLO [16, 23, 24, 25, 26, 27] at the time of writing. The default configuration of YOLOv8 pre-trained on COCO [28] supports 80 object categories. However, they may not be suitable depending on the dataset; some may never appear, or categories outside the pre-defined ones may appear in the target environment. As another remark, the domain gap must be considered for object detection on both the captured and rendered images.

Therefore, we fine-tune the model using a dataset of captured and rendered images in the target environment. The images are labeled with a few classes that are likely to be in the environment. It should be noted that the model fine-tuned this way may overfit the target environment and lack generalization performance. This, however, does not matter because our goal in this paper is to propose a new use case in metaverse applications and not to develop general-purpose high-quality object detectors. Once the model has been fine-tuned, it is applied to each frame of the captured and rendered images.

3.4 Change Detection

Captured and rendered images, after object detection, have bounding boxes of the detected objects. The change, either the removal or addition of objects, can be detected and classified by comparing the bounding boxes.

Let $b_i^c(n)$ denote the i-th bounding box detected in the n-th captured image. In the same way, we define $b_i^r(n)$ for the n-th rendered image. It is reminded that the captured and rendered images are aligned when they are about the same timestamp. If $b_i^c(n)$ does not have any intersected bounding box on the rendered image, it means the detected object has been added. Similarly, if $b_j^r(n)$ does not have any intersected bounding box on the captured image, the detected object has been removed. When $b_i^c(n)$ intersects $b_j^r(n)$, the object is likely to be still there. Therefore, this case is regarded as no change. In some situations, $b_i^c(n)$ and $b_j^r(n)$ may represent a different object class, but we do not consider this in this paper.

4 Results

We scan the corridor of a building with Scaniverse and use it as the reference map (see Figure 2). The reference map is compared with the current environment in which $2,202$ images are captured. Since the changes are not annotated, we report representative results.

4.1 Successful Cases

Figure 3 shows successful cases of the detection of the addition of objects. When an object is added to the environment, it is usually detected only in the captured image if the object detection model performs well enough. This can be verified in Figure 3.

[3]`https://scaniverse.com/`
[4]`https://github.com/ultralytics/ultralytics`

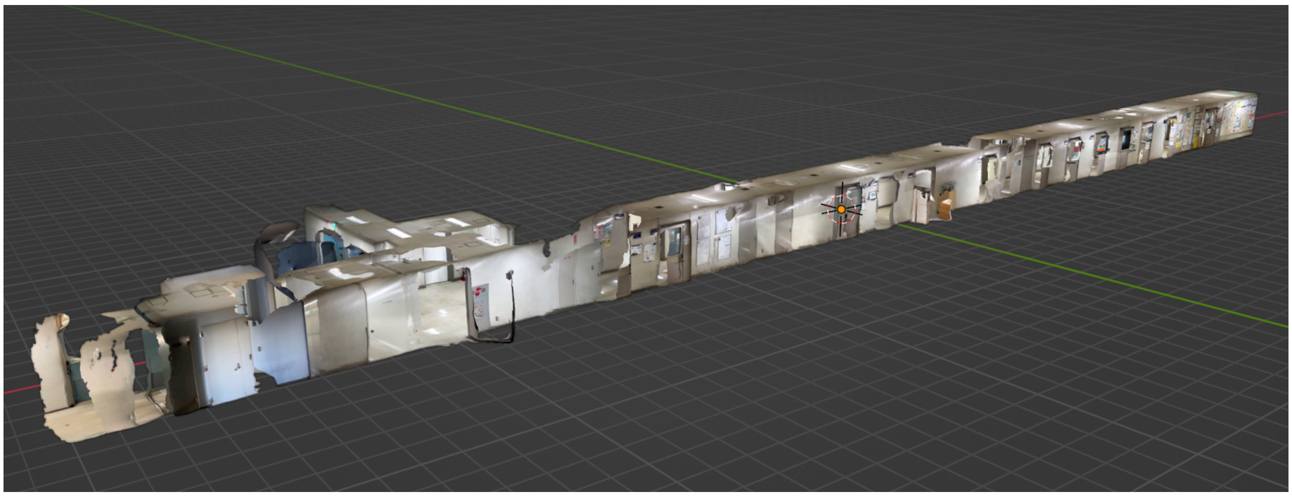


Fig. 2: Reference map (3D mesh).

Figure 4 shows successful cases of the detection of the removal of objects. Similarly, the removed object is expected to be detected only in the rendered image, which is the case in Figure 4.

In both cases, the object labels are predicted correctly because the object detector is fine-tuned with the data of the target environment.

4.2 Unsuccessful Cases

When the target object, which is under no change, is far from the camera, it occupies only a small region in the captured and rendered image. In this case, the object tends to be detected in either the captured or the rendered image. This often causes false positives, that is, the prediction that the object is either removed or added despite no actual change, as seen in Figure 5.

This could be alleviated by introducing a voting strategy. The idea is to gather and take a vote from the predictions for each frame instead of immediately concluding from the prediction of a single one. Voting is also vital for the efficient update of metaverse maps because, with information on which object has been changed, the update can be as little as possible.

5 Conclusion

In this paper, we have presented a frame-wise change detection algorithm designed for efficiently updating metaverse maps. In the experiment with the fine-tuned object detector, change detection was reduced to comparing the existence of bounding boxes. However, the frame-wise prediction was shown to be prone to distant objects because of its nature.

In future work, updating metaverse maps using our method could be considered. While updating 3D representation is challenging, adding change information in the map (e.g., placing signboards or markers at the position of the change) can be done by simply unprojecting the pixels of the bounding box onto the 3D space. More extensive experiments on more environments should also be explored to study the algorithm to make it more robust to more complex situations, such as different instances of the same object being densely close to each other.

Acknowledgment

This work was supported by JST SPRING, Grant Number JPMJSP2123.

References

[1] Haihan Duan, Jiaye Li, Sizheng Fan, Zhonghao Lin, Xiao Wu, and Wei Cai, “Metaverse for social good: A university campus prototype,” in *Proceedings of the 29th ACM international conference on multimedia*, 2021, pp. 153–161.

[2] Abdulmotaleb El Saddik, “Digital twins: The convergence of multimedia technologies,” *IEEE multimedia*, vol. 25, no. 2, pp. 87–92, 2018.

[3] Johannes L Schonberger and Jan-Michael Frahm, “Structure-from-motion revisited,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[4] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys, “Pixelwise view selection for unstructured multi-view stereo,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 501–518.

[5] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós, “ORB-SLAM: a versatile and accurate

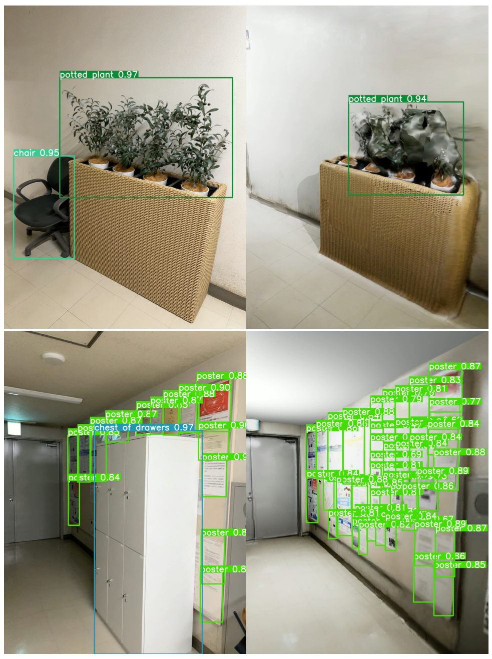


Fig. 3: Successful detection of the addition of objects: a chair in the upper case and a chest of drawers in the lower case. The left image is captured by an actual camera while the right one is rendered in Unreal Engine 5.

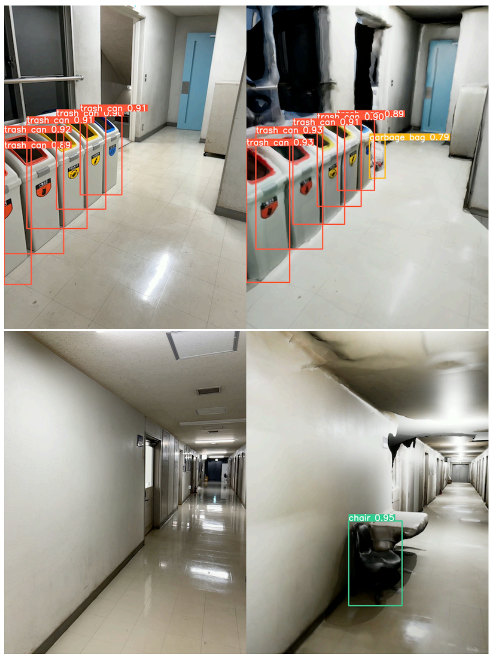


Fig. 4: Successful detection of the removal of objects: a garbage bag in the upper case and a chair in the lower case. The left image is captured by an actual camera while the right one is rendered in Unreal Engine 5.

monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] Raúl Mur-Artal and Juan D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[7] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel, "A comprehensive survey of visual slam algorithms," *Robotics*, vol. 11, no. 1, pp. 24, 2022.

[8] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[9] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman, "Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–12, 2023.

[10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, 2023.

[11] Paul Viola and Michael Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Ieee, 2001, vol. 1, pp. I–I.

[12] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Ieee, 2005, vol. 1, pp. 886–893.

[13] Pedro Felzenszwalb, David McAllester, and Deva Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE conference on computer vision and pattern recognition*. Ieee, 2008, pp. 1–8.

[14] Ross Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

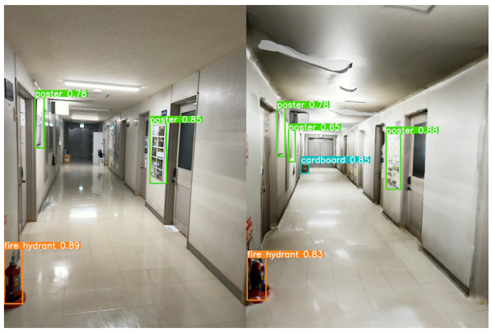
[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

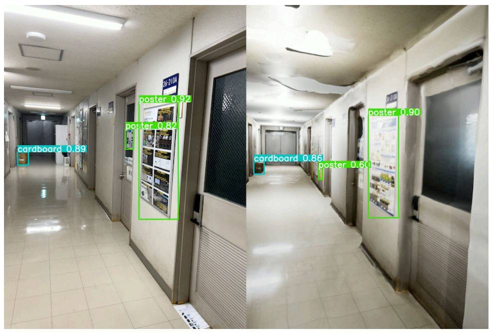
[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[18] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[19] Oliver Wasenmüller, Marcel Meyer, and Didier Stricker, "Augmented reality 3d discrepancy check in industrial applications," in *2016 IEEE Interna-*



(a) Failure



(b) Success

Fig. 5: False positive cases. (a) A distant cardboard box is detected only in the captured image (left), which leads to predicting the removal. (b) After some frames, when the camera gets closer to the cardboard box, it is detected in both images.

tional Symposium on Mixed and Augmented Reality (ISMAR), 2016, pp. 125–134.

[20] Wataru Shimoda and Keiji Yanai, "Self-supervised difference detection for weakly-supervised semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5208–5217.

[21] Ken Sakurada, Mikiya Shibuya, and Weimin Wang, "Weakly supervised silhouette-based semantic scene change detection," in *2020 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 6861–6867.

[22] Tomoya Matsubara and Hideo Saito, "Point cloud change detection in indoor environments," in *Proceedings of the 15th Asia-Pacific Workshop on Mixed and Augmented Reality co-located with TAICHI 2023, Taipei, Taiwan, August 18-19, 2023*. 2023, vol. 3467 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[23] Joseph Redmon and Ali Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[24] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[25] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[26] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al., "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[27] CY Wang, A Bochkovskiy, and HYM Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arxiv 2022," *arXiv preprint arXiv:2207.02696*, 2022.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.