3D Image Reconstruction from Videos Using Patches Generated from Tracking-Learning-Detection Algorithm

Amandeep Singh
Stanford University
asingh@stanford.edu

Abstract

While there has been significant growth in 2D image and video content generation, 3D image reconstruction from videos has achieved limited user adoption owing to difficulty in generating the videos and images required for 3D reconstruction. In this paper we propose a novel pipeline that automates generation of images from a generic video stream for use by an SFM pipeline. The proposed approach uses tracking-learning-detection algorithm to generate patches that are then used as input to SFM pipeline to generate 3D images of object of interest. The proposed approach has been implemented in MATLAB and has been tested on a number of different videos to generate 3D models for objects of interest.

1. Introduction

Recent advances in semiconductor technology have democratized access to cheap, high resolution, high quality cameras. Almost all cell-phones have multiple high resolution cameras embedded within them. This democratization of high quality cameras has led to a huge increase in the volume of photo and video content being generated. However, most of the content being generated today is still 2D, and very little 3D content is being created even today.

Structure from Motion (SFM) is a well-studied problem in the field of computer vision and visual perception. SFM algorithms enable us to reconstruct 3D views of an object from multiple 2D views of the object of interest. Over the past few years SFM algorithms have advanced significantly and a number of groups have demonstrated the up-to large city scale 3D reconstruction using SFM algorithms. One such seminal example is 'Reconstructing Rome in a day' [1].

However, despite these technological advances 3D image reconstruction has not been democratized yet, and the amount of 3D content being generated is still rather limited, especially relative to the amount of 2D data that is generated every-day. One potential reason for this is the difficulty in obtaining the large number of views of the object of interest. A naïve way to obtain the different views requires the user to manually click a number of 2D images for the target of interest from different viewpoints. However, this requires care and precision on part of the user. The user has to focus on the required object and has to be careful to minimize the background noise. An alternate approach could be that the user creates a video of the target of interest while ensuring that he has captured multiple views of object and the background noise is minimized. However, even this method requires careful video generation on the user's part.

In this project we are trying to explore the possibility of generating 3D views for any object from a generic video stream. *A generic video stream is being defined as any video captured by the user without paying special attention to the target object of interest, i.e. any frame in the video may contain multiple objects in addition to the particular target object of interest.* If we can successfully generate 3D models for any object from generic video streams, it would significantly reduce the user effort required in generating 3D models, and hence, could increase the user base for 3D model generation significantly.

A naïve way of 3D reconstruction from generic video frames could be that the user manually identifies the patches with the target of interest, and then manually crops the image to generate the required multiple views for the SFM pipeline. However, this approach does not reduce user effort, but rather increases it.

An alternative approach could be that the user uses an object detector to detect the target object of interest from each of the frames, which could then be used to generate the data for SFM pipeline. However, using object detectors requires prior knowledge of the object, and also needs a lot of training data. Further, most of the object detectors cannot distinguish very well between two objects that are similar and belong to the same broad category.

Another alternative approach would be to use a tracker to track the target object of interest in each video frame. The output from the tracker could then be used to generate the data for SFM pipeline. However, most of the conventional tracking algorithms assume that the target is present in each video frame. If the object goes out of view even for a few frames, the tracking algorithm fails completely.

Therefore, in this project we propose to use tracking-

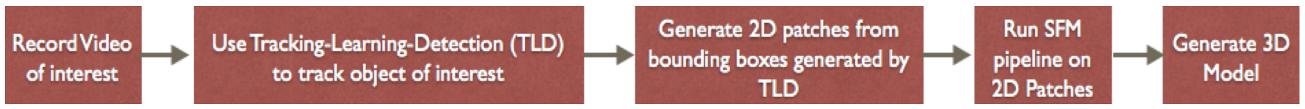
Record Video of interest
Use Tracking-Learning-Detection (TLD) to track object of interest
Generate 2D patches from bounding boxes generated by TLD
Run SFM pipeline on 2D Patches
Generate 3D Model

Figure 1 : Overall Pipeline for 3D Image Reconstruction

learning-detection (TLD) [2] based framework to obtain bounding boxes around the target object of interest from each video frame. The bounding boxes generated by TLD are then used to obtain image patches containing the target object, while eliminating the background noise such as other objects present in each video frame. These patches are then used as an input for SFM pipeline. The major advantage of using TLD based framework is that unlike object detectors it does not need prior training, and unlike trackers it has the ability to recover even if few frames in the image do not contain the target object of interest.

Therefore, the proposed pipeline essentially consists of two sequential steps, (a) Generate image patches showing multiple views for the target object of interest using TLD algorithm (b) Use SFM pipeline on the generated patches to obtain 3D model of the target object. The proposed pipeline is simple enough, and requires minimal manual intervention.

The organization of the rest of paper is as follows. Section 2 below describes the problem formally. Section 3 describes the proposed pipeline in more detail, together with implementation details of TLD algorithm, and SFM pipeline. The TLD algorithm was implemented using a starter code available from homework for CS231B website. The SFM pipeline was implemented as an extension from Problem Set 2 Question 4. Section 4 then describes the experimental setup and results, followed by Section 5 which concludes the paper and gives ideas for further work on this project.

2. Problem Statement

Given a generic video stream *V* extract multiple image patches *P* which contain the target object of interest using Tracking-Learning-Detection (TLD) framework. The set of generated patches *P*, act like inputs for a SFM pipeline and are used to generate the 3D model for the object of interest.

3. Technical Approach

This section describes the technical details for the proposed pipeline. Section 3.1 below gives the details for the overall pipeline used. Section 3.2 and 3.3 give details for the implemented TLD framework and SFM pipeline.

3.1. Proposed Pipeline

Figure 1 above describes the proposed pipeline. The system receives an input video from the user. The input video is read into MATLAB to transform the input video into a set of sequential frames. The user then draws a bounding box around the object of interest. The set of sequential frames, together with the bounding box drawn by the user become the input to the TLD algorithm. The TLD algorithm then tracks the object inside the bounding box, and outputs a bounding box in identifying the location of the object in each image frame. In-case the TLD algorithm does not detect the target object in a particular image frame it returns NaN as the bounding box for that frame. However, TLD is able to find the object again when it comes back in the frame of view. The bounding boxes generated from TLD are then used to generate 2D patches of the image. It may be noted that since the patches returned from TLD might be of slightly different sizes, the patch is first resized into a dimensions 640x480. The resized image is then fed to SFM pipeline. The SFM pipeline is then used to do 3D reconstruction. It may be noted that depending on the duration of the video frame, the number of patches being input to the SFM frame can be relatively large. While the large number of patches with different views help improve the quality of image reconstruction, the run-time for SFM pipeline increases with increasing number of input patches. Thus, the number of patches to be used for the SFM pipeline is a tradeoff between image reconstruction quality, and algorithm run-time.

3.2. Tracking-Learning-Detection (TLD) Algorithm

Given the initial bounding box around the object of interest the goal of tracking algorithm is to be able to reliably track the object in subsequent frames, i.e. automatically detect the object's bounding box in subsequent frames or indicate that the object is not present in the frame. The above problem is a commonly studied problem in computer vision community, and has been called as *Long Term Tracking Problem.* While a large number of approaches have been investigated, most of them can be summarized in the form of three broad categories:

1. **Tracking based approaches**

These methods try to formulate the long term tracking problem as a tracking problem only. While these methods can be fast, they are accurate only till the time the object is in frame, and typically fail completely if the object disappears in the frame for a bit. Some of the common methods used in tracking are: Kalman filtering based approaches which try to formulate tracking problem as probabilistic graphical model, or template matching based methods such as Lucas-Kanade tracker [1].

2. Detection based approaches

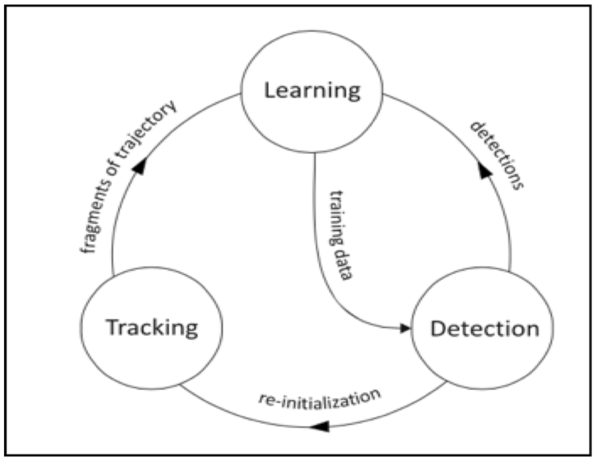

Figure 2: TLD Framework

These methods try to detect the object in every frame independently. While, it is possible to do object detection, most of the detectors require a long offline training stage, and hence are not feasible for use in real-time tracking.

3. Tracking - Learning - Detection (TLD)

TLD is a new framework that was proposed by Kalal et al [2] in 2010. This framework realizes that long term tracking cannot be solved by either tracking alone, or detection alone, but rather it decomposes the task of long term tracking into three sub-tasks: tracking, learning, and detection. The block diagram of the proposed approach is shown in Figure 2.

In this framework the tracker estimates the objects motion between consecutive frames under the assumption that the frame is visible. The detector treats every frame independently and performs full scanning of the image to localize all appearances that have been observed and learned in the past. Learning observes the performance of both, tracker and detector, estimates detectors errors, and generates training examples to avoid these errors in the future. *Given the success of TLD framework in literature we decided to use this method as our preferred method of tracking in this project.*

3.2.1 Implementation Details of TLD Algorithm

Figure 3 shows the block diagram representation of the TLD [2] algorithm. To better understand the functioning of the algorithm this algorithm was implemented using the starter code available from CS 231B ('Stanford Class on Advanced Computer Vision') homework Project. In effect, most of the detection, learning, and data augmentation part was written by us. This section summarizes the key building blocks of the algorithm, and highlights the specific sections where our implementation was different from the original paper [2].

The TLD algorithm represents the object as a collection of size normalized positive and negative patches. By

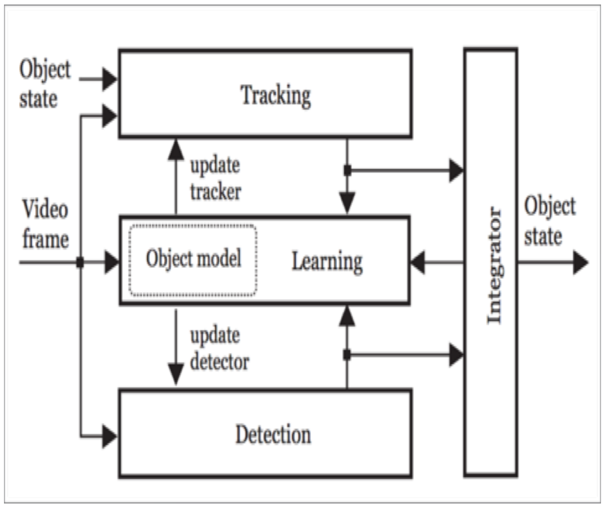

Figure 3: Block Diagram of TLD Algorithm

comparing patches in a new frame with patches in the object model, the algorithm detects the location of the object present in the new frame. Unlike the authors, where RAW pixel features were used for feature representation [2], we decided to use HoG features for feature representation. HoG features helps us to be more invariant to geometric and photometric transformations.

Tracker: The tracker estimates the objects motion between frames under the assumption that the object is still visible. The authors use a modified version of Lucas-Kanade tracker as the tracking stage. We used the same strategy. Essentially, the tracker estimates motion of a number of points within objects bounding box and uses this information to estimate the motion of the object within consecutive frames. The algorithm estimates each point's motion independently and subsequently uses all these predictions to estimate the median translation vector which is used to estimate the new transformed bounding box.

Detector: The detector scans the input image by a sliding window and for each patch decides about presence or absence of the object. The authors propose a three state cascade object detector. In the first stage, a patch variance filter is applied which rejects all patches for which gray-value variance is smaller than 50% of the variance of the patch that was selected for tracking. In the second stage, the authors used an ensemble classifier, and finally in the third stage a nearest neighbor classifier. In our implementation, we decided to use an SVM classifier as the detector, instead of using ensemble classifier. The SVM classifier is easy to implement and has been commonly used in Object Detectors. Additionally, we added few constraints to help the detector. Since, we know that the object cannot suddenly change its location and scales completely, we added maximum delta on how much the size of the bounding box can change from frame to frame. Similarly, maximum delta was added to specify the maximum change in object location positions.

Learner: The learning strategy implemented was similar to the original implementation in [2]. The P-expert discovers new appearances of the object that could be used to improve the detector. The N-expert generates the negative training examples. It provides examples of the background to help detector better discriminate the background from object. Model was updated only for reliable appearances. An appearance was defined as Reliable as an appearance that agrees with both the detector and tracker (defined by thresholds).

Integrator: Integrator combines the bounding box of the tracker and the bounding boxes of the detector into a single bounding box which is output from the TLD. The integrator implementation is similar to the one in the original paper.

3.3. Generating Patches from Bounding Boxes from TLD

TLD outputs a list of bounding boxes, giving the location of bounding box in each frame, or NaN if the object was not detected in a particular frame. Given the co-ordinates of bounding box, patch was extracted from the original image. It may be noted that the patch was resized appropriately to ensure that all generated patches are of identical sizes.

3.4. SFM Pipeline

Given the 2D generated patches, structure from motion pipeline was used to convert the set of 2D images to a 3D image. The pipeline used to implement this 2D to 3D generation process is shown in Figure 4 below.

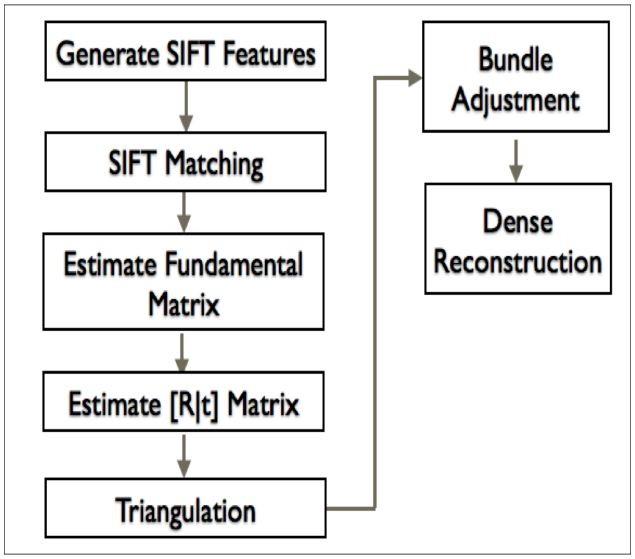

Figure 2 : SFM Pipeline

Given the images the first step for conversion from 2D to 3D involves identifying the point correspondences amongst images. To do this we used the SIFT features. SIFT features were generated for each of the images, and SIFT matching was performed to identify the top few point correspondences. It may be noted that since our images are coming from a sequence of images generated from videos, we did not consider it necessary to identify point correspondences amongst all pairs of images. Instead point correspondences were identified only for neighboring image patches, i.e. for patches generated from next image frame. Given the point correspondences, we estimated the fundamental matrix using eight-point algorithm studied in class. To minimize the effect of outliers, RANSAC algorithm was used while estimating the fundamental matrix. Given the fundamental matrix F we can easily calculate the essential matrix E.

$$E = K' * F * K$$

Following the calculation of essential matrix E, we estimated the Rotation and Translation matrices, i.e. the [R|t] matrix using the instructions given in question 4 in problem set 2. Following the estimation of [R|t] matrix, 3D points were generated using triangulation (similar to what was done in Q4, in problem set 2). The above steps were carried out to generate 3D points from each pair of images. Non-linear optimization based bundle adjustment was then used to combine the outputs from each of the pair of images, and generate the final sparse 3D reconstruction for the images. Following the sparse reconstruction, a very simple dense reconstruction module was implemented to complete the SFM pipeline. The dense reconstruction algorithm was implemented using zero mean normalized cross-correlation based matching propagation method [4]. Given the complexity involved in implementing the dense reconstruction algorithm, a part of the code from [4] was adopted and integrated with our current pipeline to complete the implementation.

4. Experimental Setup and Results

This section gives details of the various experiments we did and summarizes the results achieved. Section 4.1 outlines the summary of results for standalone TLD implementation. Section 4.2 summarizes the results for standalone SFM implementation. Section 4.3 then describes the results and limitations of our proposed approach of using the patches from TLD algorithm to generate 3D model of object of interest.

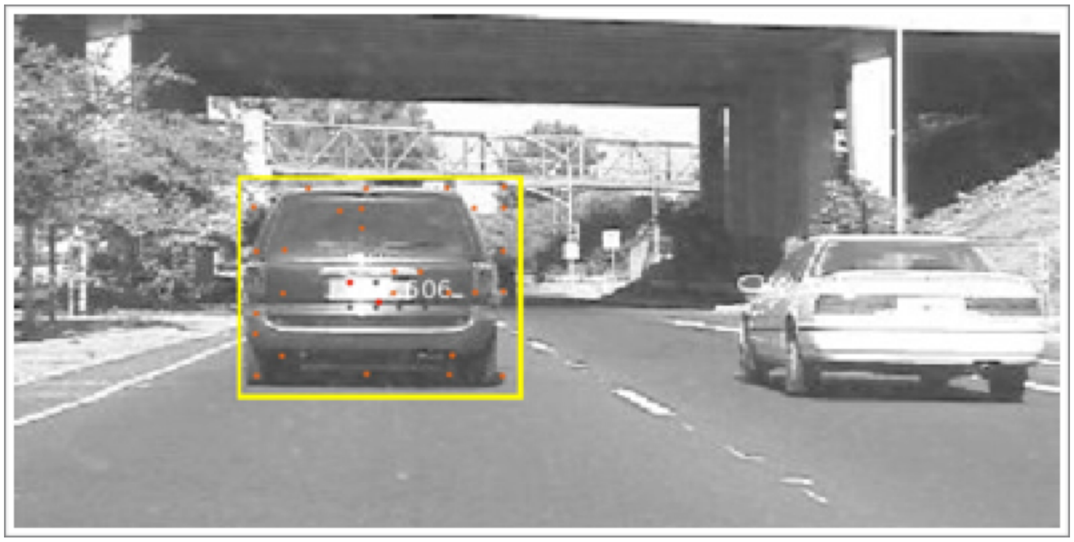

Figure 3 : Output of TLD algorithm on 'Car4' dataset

4.1. Verification of Tracking-Learning-Detection (TLD) Algorithm

The TLD algorithm was implemented in MATLAB using starter code available from CS231B course website. To ensure the correctness of the implemented TLD algorithm two metrics were used: Mean average precision, and average overlap. The TLD algorithm was verified on 'tiny_tracking_data' dataset available from CS 231B. To ensure the correctness of operation we ensured that our mean average precision score was greater than 0.78, and average overlap greater than 0.68 when our code was run on full 'Car4' dataset. Car4 dataset is part of the tiny_tracking_data dataset, and the metric scores used for measuring correctness are based on values given on CS231B website. A screenshot of the output of our algorithm on 'Car4' dataset is shown in Figure 5.

4.2. Verification of SFM pipeline

The SFM pipeline described in section 3.4 was implemented in MATLAB. The code written for Problem Set 2, Question 4 was used as a baseline code, and other necessary code was added to it to complete the pipeline. Most of the steps in the pipeline up-to sparse reconstruction were implemented from scratch or from Problem Sets. Given the complexity involved in implementation of a complete SFM pipeline, help was taken from [4] to implement the dense reconstruction.

To verify the correctness of implementation we used the image dataset provided in problem set 2, question 4. The input images given to the the SFM algorithm are shown in Figure 6 below. The output of the SFM pipeline showing the reconstructed 3D image is given in Figure 7 below.

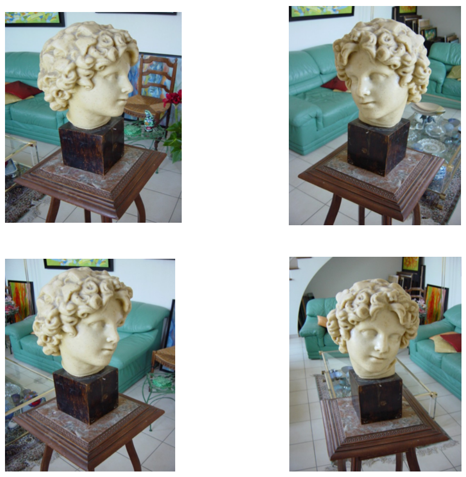

Figure 4 : Sample 2D Images used for 3D reconstruction

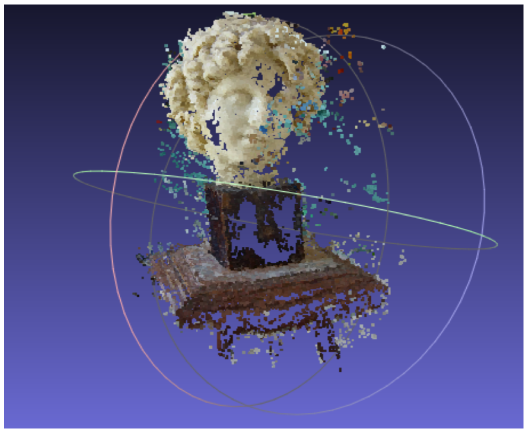

Figure 5 : Reconstructed 3D image using 2D images

4.3. Verification of Proposed Pipeline

This section describes the results achieved using the proposed pipeline. It may be noted that the dataset for this section was created on our own. We used an I-phone 6 camera to capture videos of target of interest. Care was taken to ensure that we get enough noise in each video to replicate a natural scene, and demonstrate the effectiveness of the proposed approach.

The video inputs were converted first into a sequence of frames which were then sent as input to our TLD algorithm. Patches were generated using bounding boxes generated by TLD algorithm and then input to our SFM pipeline. However, we realized that our SFM was not scalable enough to handle the large number of frames coming from the output of TLD algorithm. We then tried to select a subset of patches from the output of TLD algorithm and use as input to SFM pipeline. While this 'worked', the reconstructed image quality was rather average. We postulate that this is because every time we take a patch, we effectively reduce the resolution of the image which makes it really difficult to produce a dense good quality 3D reconstruction. We had anticipated this problem initially, and we were hoping that increasing the number of frames would help us overcome this loss in resolution. Further, we also realized that extracting the patches using TLD, changes the effective focal length, and hence decreases the accuracy of our initial focal length estimate (based on public data available on I-phone 6 camera).

Therefore, to validate our idea we decided to input the 2D patches generated from TLD algorithm to 'VisualSFM' [5] platform also in addition to our implemented SFM. VisualSFM essentially implements an SFM pipeline very similar to our proposed SFM pipeline, but has been optimized for speed using GPU based calculation of SIFT features, and GPU based SIFT matching. Further, VisualSFM has in-built algorithms which try to estimate the focal length also directly from the images, and do not rely on our initial guess estimate. This makes it a very good platform to test our overall pipeline. The figures given below demonstrate the effectiveness of our proposed pipeline in implementing a complete 3D reconstruction from videos using patches generated from Tracking-Learning-Detection algorithm.

Figure 8 below shows the outcome of running our pipeline on a video to generate 3D image of a teddy bear. As is evident from the video, the proposed pipeline is able to successfully reconstruct the 3D image using our pipeline. Also, the advantage of the proposed methodology is evident from the videos, that if we had not selected the patches, the 3D reconstructed image would have contained other objects in addition to the teddy bear. Figure 9 – 11 show the output of our proposed pipeline on other objects. Figure 9 shows the 3D reconstruction of a big printer, Figure 10 a flower, and Figure 11 shows the reconstruction of a helmet. It may be observed that since the helmet chosen had empty spaces in it, i.e. had air pockets on the surface, the reconstructed image for the helmet is not as good as the other objects.

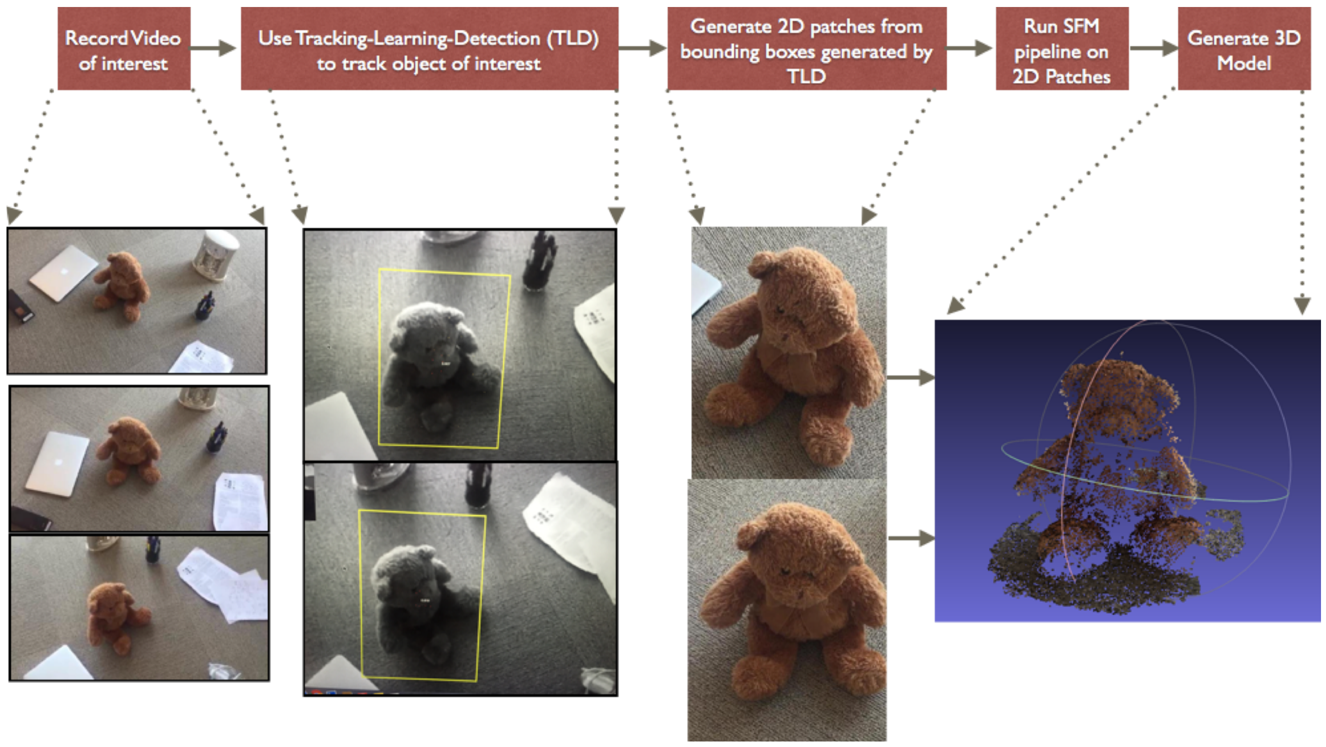
Record Video of interest
Use Tracking-Learning-Detection (TLD) to track object of interest
Generate 2D patches from bounding boxes generated by TLD
Run SFM pipeline on 2D Patches
Generate 3D Model

Figure 6 : 3D Reconstruction from videos of a Teddy Bear

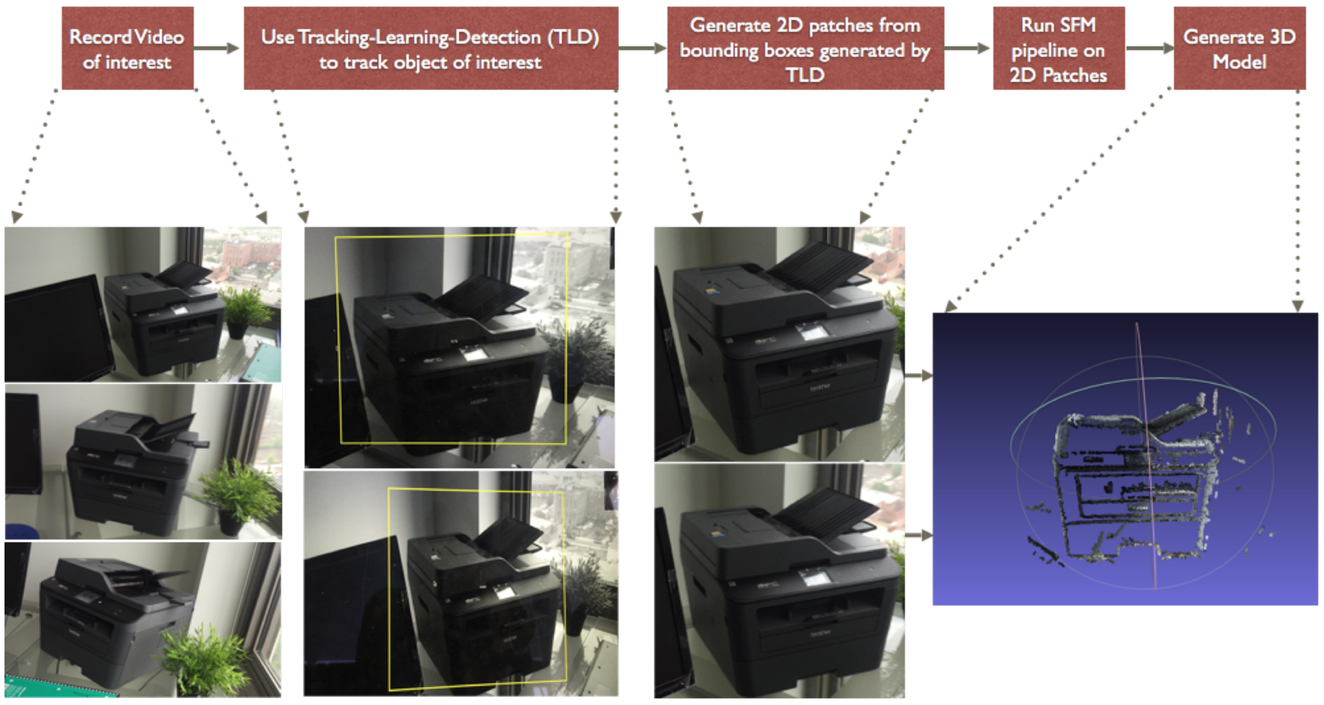
Record Video of interest
Use Tracking-Learning-Detection (TLD) to track object of interest
Generate 2D patches from bounding boxes generated by TLD
Run SFM pipeline on 2D Patches
Generate 3D Model

Figure 9: 3D Reconstruction from videos of a big Printer

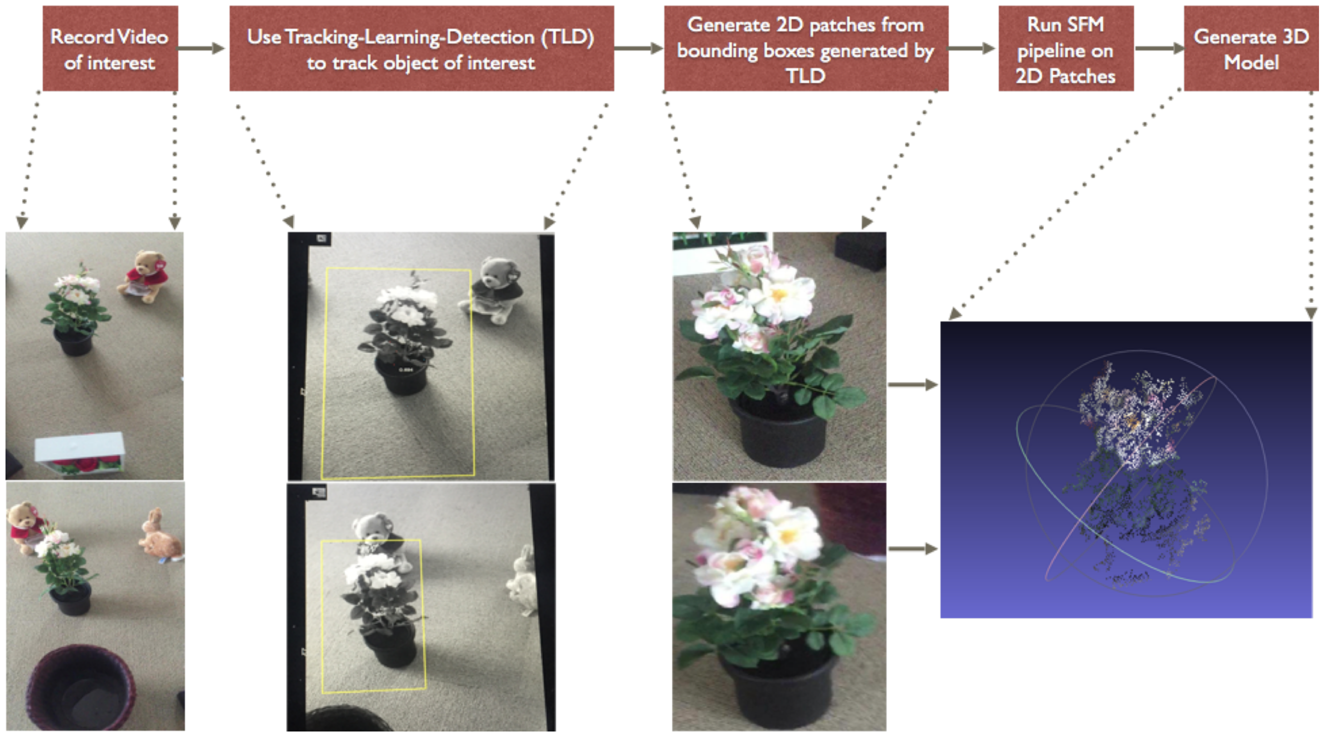
Record Video of interest
Use Tracking-Learning-Detection (TLD) to track object of interest
Generate 2D patches from bounding boxes generated by TLD
Run SFM pipeline on 2D Patches
Generate 3D Model

Figure 10: 3D Reconstruction from videos of a Flower

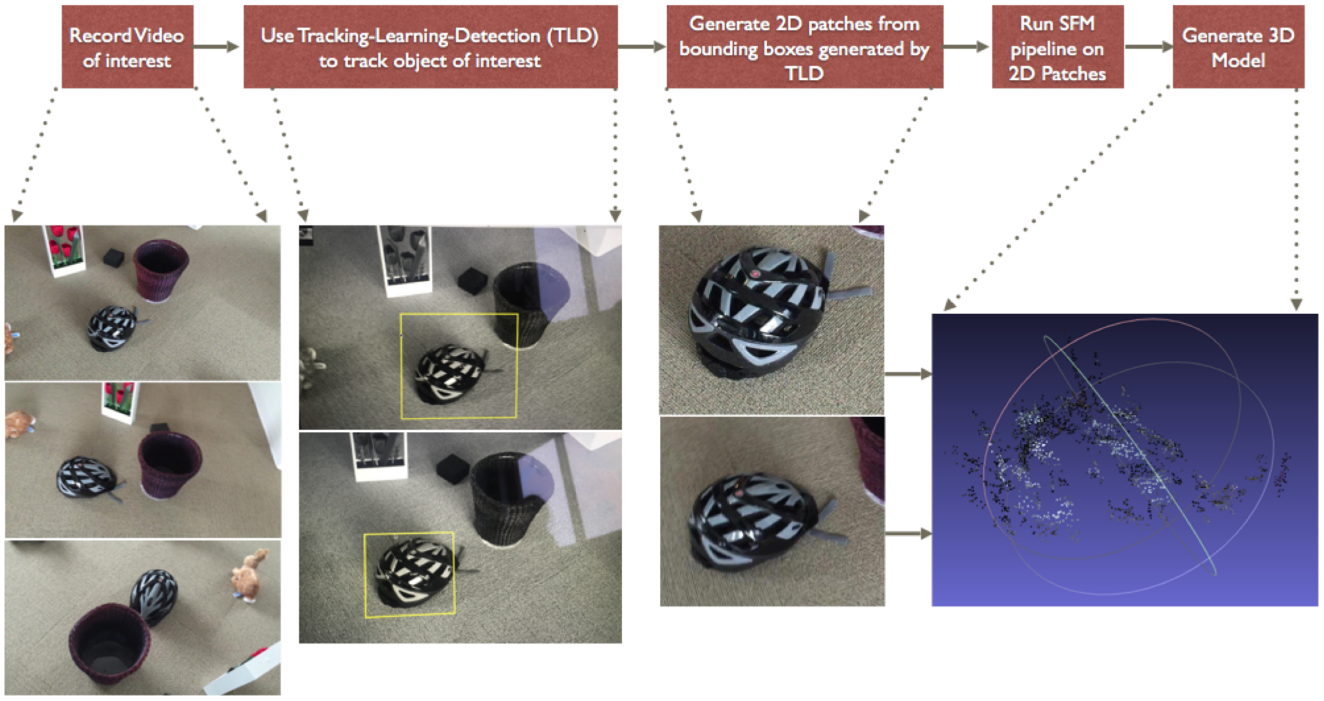
Record Video of interest
Use Tracking-Learning-Detection (TLD) to track object of interest
Generate 2D patches from bounding boxes generated by TLD
Run SFM pipeline on 2D Patches
Generate 3D Model

Figure 11: 3D Reconstruction from videos of a Helmet

5. Conclusion

In this paper we demonstrated an innovative approach that could be used to automatically generate 3D models from videos. The proposed pipeline of using a TLD based tracker to generate 2D patches for an SFM pipeline simplifies the 3D model generation considerably. It significantly reduces the manual labor required to produce videos or set of images for input to an SFM pipeline.

While being an innovative and interesting approach, implementing the complete pipeline helped us further appreciate the computational complexity involved in full 3D image reconstruction. To fully realize the full potential of our pipeline and reach the goal of democratizing 3D content generation a lot of work still needs to be done to improve SFM algorithms further to allow them to run in real-time on mobile devices in a power efficient manner.

Acknowledgements

We would like to thank Prof. Silvio, and the entire CS231A course staff for their help with this project.

Code : The dropbox link to the code has been submitted via the link given by TA's on PIAZZA.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S Seitz, R Szeliski. Building Rome in a Day. In CACM Vol 54 (2011)

[2] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," Pattern Analysis and Machine Intelligence 2011.

[3] CS 231B: Stanford course on advanced computer vision.

[4] COS 429: Course on Computer Vision

[5] Changchang Wu, "Towards Linear-time Incremental Structure from Motion", 3DV 2013