DynamicFusion

Rebecca Richter

Fakultät für Informatik - Technische Universität München

Abstract

Dynamic Fusion is a method for Dynamic RGB-D Scanning published by Newcombe et al. in 2015. It uses an online stream of noisy depth maps from a single RGB-D camera and produces a 3D dense reconstruction of the moving scene in real-time. The system estimates a non-rigid warp field that provides a dense 6D transformation from a reference frame to a life frame. This enables fusion of the depth maps data into a canonical model. The reconstructed surface is estimated using a truncated signed distance function that is updated in every time step. Therefore the quality of the reconstruction is increasing over time. This report gives a short overview on the DynamicFusion method, for more details refer to [2].

1 Introduction

Traditionally 3D Scanning always involved two separate phases: one for scanning of the target object and another off-line processing phase to perform the actual reconstruction of the geometry. This usually leads to multiple iterations of scanning, reconstructing, identifying noisy parts of the geometry where more data is needed and recapturing. To make the procedure more efficient, real-time systems like [1] were introduced. They enable the user to instantly see the reconstruction and identify regions that where missed or need to be recaptured. Therefore it was a really important step in 3D scanning. Nevertheless all of these methods are limited to static scenes. [1] uses a global transformation to fuse information from different scanning angles together, so in order to make this approach work in a nice way nothing in the scene should be moving but the camera. For non-living objects this works quite well but what if the target object is living and in the worst case is an animal or a kid that will never remain in a static position. DynamicFusion solves this problem, since it is able to handle a moving scene. As in [1] DynamicFusion wants to use one reference coordinate frame where the reconstruction should be performed and all the information should be collected. But in a dynamic scene the object is not only rotating and translating, but it is also deforming. Therefore a non-rigid transformation needs to be estimated individually for each voxel of the reconstruction volume, to undo changes of the object over time. Estimating and representing these transformations in real-time and is a really important part of the DynamicFusion approach and will be explained in more detail in 2.2.
This paper gives only a short summary of the work presented in [2], so for any more details please refer to the original work. In the first part of this paper we want to describe the method, focusing in the subsections on the canonical model representation 2.1, the warp field estimation 2.2 and the extension of the warp field 2.3. The second part provides some evaluation of the results presented in [2] and gives an overview about abilities and limitations of DynamicFusion.

2 Method Description

As already mentioned in the introduction, the main goal of Dynamic Fusion is to get a dense reconstruction of a moving scene in real-time. Therefore the approach processes the following steps for each new depth map.

- Estimation of the transformation parameters in order to map the deformation nodes from the canonical frame to the new life frame
- Initialization of the dense volumetric warp field and transformation of the current reconstruction into the life frame
- Update of the current surface estimate using the new depth maps data
- Extension of the warp field in order to also capture newly added parts of the surface

The following parts of this review will provide more details about these main algorithm steps as well as on the canonical model and the warp fields representation.

2.1 Canonical Model Representation

To safe and update the reconstruction, Dynamic Fusion uses a truncated signed distance function (TSDF). This mean for every voxel center in the reconstruction volume the distance from the voxel centers position to the surface of the reconstructed object is estimated. Since its a signed distance function voxels that are assumed to be "inside" the object have an opposite signed distance then voxels that are assumed to be "outside". Also voxels that are assumed to have a large distance to the objects surface only provide a truncation value. The surface estimate of the reconstruction is the plane where the distance to the surface is estimated to be 0. The TSDF can be represented as $V : S \rightarrow [v(x_c) \in \mathbb{R}, w(x_c) \in \mathbb{R}]$, where S is the set of all voxels x_c of the reconstruction volume. $v(x_c)$ provides the current distance estimate for this voxel and $w(x_c)$ is a weighting term. To increase the quality of the reconstructed surface with each new depth map, V needs to be updated in each time step. Therefore in a first step, the already estimated warp field is used to map each voxel in the canonical frame to the new life frame. In the new life frame each warped voxel is assigned to a point on the life frame surface using perspective projection. Afterwards the distance of each voxel to the life frame surface is computed using the difference between the positions in optical z direction. Mathematically this can be described as a projective signed distance function $psdf(x_c)$ for each voxel center x_c. In a second step the TSDF function is updated, using as current distance estimate the weighted average of all distances measured for this point. Additionally the weights value is also updated. The updating scheme can be written as

$$
\begin{aligned}
V(x_c)_t &= \begin{cases} [v'(x_c), w'(x_c)]^T, & if\ psdf(dc(x_c)) > -\tau \\ V(x_c)_{t-1}, & otherwise \end{cases} \\
v'(x_c) &= \frac{v(x_c)_{t-1} w(x_c)_{t-1} + min(psdf(dc(x_c)), \tau)}{w(x_c)_{t-1} + w(x_c)} \\
w'(x_c) &= min(w(x_c)_{t-1} + w(x_c), w_{max})
\end{aligned}
\tag{1}
$$

dc(.) transforms a discrete value into the continuous domain and τ is the truncation distance. As can be seen in the formula, there is a weight value $w(x)$ corresponding to each newly added distance value. This value represents the probability that the associated depth value is really observed a this point. Therefore the $w(x)$ value is proportional to the distance of the voxel to its k-nearest deformation nodes.

2.2 Warp Field Estimation

The main problem with depth maps describing a moving scene over time, is that each of these maps has a different coordinate frame and describes the object in a different state of deformation. In order to fuse information from all depth maps into a reconstruction, the DynamicFusion approach uses a warp field that provides for each voxel center in the reconstruction volume $S \subseteq \mathbb{R}^3$ a non-rigid 6D transformation from the canonical frame into each new life frame. These transformations need to be estimated individually for each voxel center, but performing an optimization on each single voxel would not be possible in real-time. Therefore, the DynamicFusion approach subsamples the reconstruction volume in a sparse way with so-called deformation nodes, and estimates their transformations using optimization. Then for each voxel center the corresponding transformation is computed using interpolation of the deformation nodes transformations. Each of the deformation nodes provides a position in the canonical frame pos_i, a radial basis weight rbw_i and its transformation from the canonical frame to the life frame T_{ic}. In the resulting warp field we want each 6D transformation to be represented as a transformation matrix $SE(3) \in \mathbb{R}^{4x4}$, so the warp field can be represented as a function $W : S \rightarrow SE(3)$.

Lets start with the first step, where we want to estimate the transformation for each deformation node, given a new Depth Map D_t, the current reconstruction V and the old warp field from the previous time step W_t. In order to get optimal transformation parameters for each deformation node we need to minimize the energy function

$$E(W_t, V, D_t, \varepsilon) = Data(W_t, V, D_t) + \lambda Reg(W_t, \varepsilon) \tag{2}$$

The first part of this equation is a model-to-frame ICP cost term, while the second part is a regularisation term that ensures smooth movements, based on the edge set ε representing motion dependencies between deformation nodes. The parameter $\lambda \in \mathbb{R}$ scales the influence of the regularisation term on the energy function.
To compute the $Data(W_t, V, D_t)$ term, in the first step the surface of the current reconstruction in the canonical frame is extracted and saved as a set of point normal pairs. Then these pairs are transformed into the new life frame using some transformation estimate W. Afterwards the warped canonical surface is rendered in the life frame using a rasterized rendering pipeline. Therefore only parts of the canonical surface that are currently predicted to be visible in the life frame are used and stored as point normal pairs in the life frame. In the end for each point of the rendered canonical surface the model-to-frame point plane error is computed, under the robust Turkey penalty function ψ_{data}. The data term contains the sum of all errors computed for pixels in the pixel domain of the warped canonical surface.

$$Data(W_t, V, D_t) = \sum_{u \in \Omega} \psi_{data}(n_u^T(v_u - vl_u)) \tag{3}$$

n_u and v_u are the warped point normal pairs from the canonical surface. Each of these point-normal pairs is associated via perspective projection to one point-normal from the new life frame surface. vl_u represents this associated life frame point-normal. Therefore the Data term is responsible for a good overlap of the warped canonical model and the life frame in the regions that are currently visible.
On the other hand the regularisation term mainly handles the movement and deformation estimate of regions that are occluded in the current life frame. The model is assumed to deform in a smooth. To describe this an edge set is used that provides an edge between two deformation nodes i and j if these nodes are assumed to influence the movement of each other. Each edge has a weight α_{ij} that represents the strength of the dependency between the connected nodes .It is set to the maximum of the radial basis weights of the connected deformation nodes $\alpha ij = max(rbw_i, rbw_j)$. For more details about the structure and the initialization of the edge set please refer to [2]. To compute the difference in transformation between two connected nodes i and j, the position of one node is once mapped using its own transformation estimate T_{jc} and once using the transformation of the other node T_{ic}. Afterwards the difference between the resulting positions is computed and scaled by the weight of the edge. To make the geometry deform on a smooth way, this difference in transformation should be as low as possible for all connected deformation nodes. Therefore the

regularisation term uses the sum of all these differences under the discontinuity preserving Huber penalty ψ_{reg}

$$Reg(W, \varepsilon) = \sum_{i=0}^{n} \sum_{j \in \varepsilon(i)} \alpha_{ij} \psi_{reg}(T_{ic} pos_j - T_{jc} pos_j) \tag{4}$$

In total minimizing the energy term always is a trade-off between fitting the visible geometry of the canonical model to the new life frame and smoothing the movements of the geometry. The actual minimization of the cost function is done using non-linear Gauss-Newton Optimization. For more details on the efficient optimization, please refer to [2].

In the second step of the warp field estimation, we want to compute the transformation for each voxel center in the canonical frame using interpolation of the transformations of the k-nearest deformation nodes. As already mentioned above, the transformation of a deformation node i is represented as transformation matrix, so $T_{ic} \in SE(3)$. Since it enables a more efficient and higher quality interpolation, the transformations of the deformation nodes are represented as dual-quaternions q_{ic} when the interpolation is performed. The actual transformation from a voxel center x_c is then computed using the weighted average of the transformations corresponding to the k-nearest deformation nodes. Therefore the warp function for each voxel center can be written as:

$$W(x_c) \equiv SE3 \left(\frac{\sum_{k \in N(x_c)} w_k(x_c) q_{kc}}{\left\| \sum_{k \in N(x_c)} w_k(x_c) q_{kc} \right\|} \right) \tag{5}$$

$N(x_c)$ is the set of unit dual-quaternion transformations corresponding to the k-nearest deformation nodes of x_c. The weighting term $w_k(x_c)$ depends on the radial basis weight of the node k and the position distance between node and voxel center. The function $SE3()$ maps the resulting unit dual quaternion back to a transformation matrix, so that in the end the warp function contains a transformation matrix for each voxel center. In a last step all global parts of the transformation T_{global} can be factored out, so that the resulting Warp function W_t can be written as:

$$W(x_c)_t \equiv T_{global} SE3 \left(\frac{\sum_{k \in N(x_c)} w_k(x_c) q_{kc}}{\left\| \sum_{k \in N(x_c)} w_k(x_c) q_{kc} \right\|} \right) \tag{6}$$

2.3 Extending the geometry

As already described in the previous section, with each update of the canonical model geometry the surface is able to grow. It happens if the new depth map provides information about initially occluded parts of the geometry, and therefore these parts are added to the surface reconstruction. In this section we assume that the new parts of the geometry are already represented in the TSDF function of the canonical frame. In order to also be able to update the newly estimated parts of the surface they need to be part of the warp function. Therefore we need to add new deformation nodes that subsample the new surface parts. In DynamicFusion the following steps are computed after each fusion into the canonical model, right before the computation of the next time step.

- Check for each point on the surface if its inside the radius of influence of a deformation node
- Subsample the detected unsupported area with new deformation nodes, that are at least d distance appart
- Initialize the transformation of the new nodes using interpolation of the transformations of the k-nearest old nodes. For this purpose the same interpolation scheme as for the warp field estimation is used.
- Add the new nodes and there dependencies to the edge set ε

3 Experiments and Results

Since the algorithm reconstructs a dynamic scene in real-time, it is really difficult to evaluate and present its performance on a static image. Therefore the authors of [2] published a video, that presents the results of there DynamicFunsion approach. It can easily be found on youtube searching for DynamicFusion. For the results presented there the following parameters where used: $\lambda = 200, \psi_{data} = 0.01, \psi_{reg} = 0.0001$ and $d = 25mm$ but according to [2] "the system works reliably across a range of dynamic scenes and parameters"(p. 7, [2]). The examples used in the video point out the abilities of the system, such as continuous tracking across motion in dynamic scenes which can be seen in all examples. The "Dancing Hand " example shows that the system is able to fill in initially occluded parts of the geometry while the "Dieter" example introduces the systems ability to generate consistent geometry despite loop closures. DynamicFusion is also able to handle topology changes from open to closed topology as can be seen in the "Interlocking fingers" example.
On the other hand, especially when the dynamic scenes become more challenging, the Dynamic Fusion approach also has its limitations. Since its tracking performance is limited to the assumption that the geometry deforms in a smooth way over time, large inter-frame movements can cause inaccurate surface prediciton when estimating the Data term and therefore tracking errors. The same problem is caused if there are motions in currently occluded regions. Also changes from closed to open topology can not be handled by the system. Another problem is, that with each time-step the warp field and the reconstructed scene are growing. Therefore proportionally more parts of the geometry are occluded from the camera in each time step, which makes the motion prediction for these parts become more challenging over time.

4 Conclusion

In this report the DynamicFusion approach was presented. It is able to generate a dense 3D reconstruction of a dynamic scene in real-time, using noisy depth maps from one RGB-D camera. The System uses a volumetric TSDF to represent and update the reconstruction in a reference frame and initializes a non-rigid 6D warp field to transform this model into each life frame. The system is able to track motion and generate a consistent reconstruction of objects from basic dynamic scenes including loop closures and topology changes from open to closed.

References

[1] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *In Proc. UIST*, pages 559–568, 2011.

[2] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.