Fusion4D: Real-time Performance Capture of Challenging Scene

Seminar: Recent Trends in 3D Computer Vision

Felix Scheidhammer and Supervisor: Benjamin Busam[1]

Chair for Computer Aided Medical Procedures
Technische Universität München

Abstract. This paper contributed a new real-time approach of multi-view performance capture, with a new pipeline making it mostly robust to large topology changes and large motions with the introduction of a Key Volume. This Key Volume gets nonrigidly aligned each frame to the observed Scene and then blended together with the observed Date using a new blending function. This volume then is incrementally updated each frame and fully reset after a certain amount of frames. ...

Keywords: nonrigid, real-time, 4D reconstruction, multi-view

1 Introduction

Even though 3D-construction is not a new topic, most system still rely on static scenes, mostly because the computation takes way more resources. And real-time approaches of 3D motion capture are even rarer, and mutli-view performance capture or interaction with Objects is momentarily solely useful in offline approaches, where one waits hours after the film is produced in strict studio settings, to get the resulting motion capture. For example, the offline Method of [Collet et al. 2015] which uses 106 Cameras in a large studio with greenscreen and controlled lighting. Collet needs 30 seconds per Frame, but the results are very accurate.
On the other side, the real-time approaches rely on one Camera, thus resulting in way less quality. For example, [Zollhöfer et al. 2014] or [Newcombe et al. 2015], which is called DynamicFusion. The first relies on a fixed Template, which is first taken of a static setup, meaning that the person in question must hold still for a certain amount of time to get the Template, which then is warped every frame to align the new Frame. With this method it is rather hard to capture children or animals, which are unlikely to hold still for the initial scan, an on the other side is it not possible to track topology changes, since the template stays fixed. DynamicFusion on the other side takes a volumetric Model to warp the observed scene, but then they also incrementally update the model per Frame with the new observed depth values., thus allowing small topology-changes and changes of the surface of the model. Both approaches cant deal with large topology changes or Motions.

Fusion4D wants to be able to deal robustly with those large topology-changes and fast motion with a multi-view setup to achieve unseen quality in real-time approaches. Secondly, they dont want to rely on priors, such like scanning or preprocessed human-skeleton systems do, but to be able to show any arbitrary scene. Fusion4D wants to achieve this with the combination of volumetric fusion and embedded deformation fields. The deformation field gives the possibility to parameterize the nonrigid scene motion whilst volumetric fusion keeps the incremental reconstruction. They accomplish that with a new Energy function for the deformation where they use dense correspondence fields to deal with large motion. And on the other side, they do not only incrementally update the reference model, but also reset it with new models periodically. They achieve high accuracy with a new robust fusing function to blend the warped reference with the observed data.

2 Overview

How do they achieve this? With a new pipeline, which will be introduced in the following lines. First, they generate from 2N IR-cameras N depthmaps using the Stereo Patchmatch algorithm, which runs in real-time with on a GPU. At the same time, they produce a segmentation mask per depthmap to be able to occlude the background. With this depthmaps they generate a dense Correspondence Field to initialize the nonrigid deformation. They also have N RGB cameras, which are only used for generating textures.
After that, they generate an Embedded Deformation model to align a certain reference Model as hierarchical voxel grid, which they denote as key volume. After the Volume is fused to the observed data, the two get fused together with the concept of volumetric fusion of [Curless an Levoy 1996], meaning that the key volume gets updated with the new values of the data and afterwards this gets blended with the data into the resulting frame. Additionally, to the update do they also fully reset the key Volume periodically with the new frame, to be able to handle large topology changes.
The reference volume (key volume) and depthmaps are stored as TSDF, truncated signed distance functions, which is very efficient for the nonrigid transformation and can also be averaged to get a denoised surface.

3 Embedded Deformation model

In the following lines will be shown, how the key volumes will get warped to align the observed scene. In their approach they use an Embedded Deformation (ED)-Model G to transform the key volume. They achieve this with laying a grid of 4mm over the volume, which then results in K sampling locations g_k as ED-nodes. Every point in the mesh of the volume is then transformed by the Transformation of the ED-nodes in its vicinity S_m, weighted by the distance towards the sampling location $w_k^m = \frac{1}{Z} exp\left(\frac{\|v_m - g_k\|^2}{2\sigma^2}\right)$, and by the global Rotation R and Translation T, which is preprocessed via Iterative Closest Points, to

get the volume overall close to the observed data. Each ED-node then contains an affine Transformation A_k and a translation t_k, which leads to the following transformation function:

$$\tau\left(v_m; G\right) := R \sum_{k \in S_m} w_k^m \left[A_k \left(v - g_k\right) + t_k\right] + T \tag{1}$$

And also for the normal vectors, which then get normalized afterwards:

$$\tau^{\perp}\left(n_m; G\right) := R \sum_{k \in S_m} w_k^m {A_k}^{-T} n_m \tag{2}$$

4 Energy function

With this ED-model we can warp any point of the key-volume to the data, but how do we find the best parameters for the ED-model? For this they introduced the new Energy function:

$$E\left(G\right) = \lambda_{data} E_{data}\left(G\right) + \lambda_{hull} E_{hull}\left(G\right) + \lambda_{corr} E_{corr}\left(G\right) \lambda_{rot} E_{rot}\left(G\right) + \lambda_{smooth} E_{smooth}\left(G\right) \tag{3}$$

This penalizes misalignment, enforces regularization and other constraints, which will be presented in detail in the following chapter.

4.1 Data Term

The first Term in the Function penalizes misalignment of the warped Volume towards the observed data. This would mean the minimal distance of each vertex in the transformed Volume to the data. But that is a rather expensive function, so they chose an approximation with a point-to-plane term:

$$E_{data}\left(G\right) = \sum_{n=1}^{N} \sum_{m \in V_n(G)} \left(\tau^{\perp}\left(n_m; G\right)^T \left(\tau\left(v_m; G\right) - \Gamma_n\left(\tau\left(v_m; G\right)\right)\right)\right)^2 \tag{4}$$

This directly means the distance of the vertex and the observed data along the normal vector, as in point-to-plane. The $\Gamma_n\left(v\right)$ means a projection of the point into the nth depthmap, and if this point is visible in this map, the depthvalue is taken from the pixel and then back-projected to the 3D world.

4.2 Regularization Terms

in the next section we will see two terms. The first one is to force the affine Transformations to be as close as possible to rotations:

$$E_{rot}\left(G\right) = \sum_{k=1}^{K} \| A_k^T A_k - I_F + \sum k = 1^K \left(det\left(A_k\right) - 1\right)^2 \tag{5}$$

the first part says, that the matrix should be orthogonal, which holds for rotations only, and the second term tries to penalize all non rigid transformations plus scaling. The second term enforces smoothness over the Model:

$$E_{smooth}(G) = \sum_{k=1}^{K} \sum_{j \in N_k} w_{jk} \rho \left(\| A_j (g_k - g_j) + g_j + t_j - (g_k + t_k) \|^2 \right) \quad (6)$$

This is the sum over the transformations of all the neighbors g_j of one ED-node N_k, weighted by $exp\left(\frac{-\|g_k - g_j\|^2}{2\sigma^2}\right)$, which is a weight by the distance of the neighbor towards the Node, normalized by the average distance σ. The overall term then gets robustified by $\rho(.)$.

4.3 Visual Hull Term

The next Term in the Energy function is a new approach, which tries to say, that the model should be completely inside of a certain area, which is not in the observed free space. This could be explained via one lamp shining on the object in the scene. Then the object should be completely inside the shadow the object itself casts, since everything outside the shadow is defined as free space, and thus there shouldnt be parts of the Object. Since they use multiple cameras to observe the scene, this leads to multiple such "'shadows"', which defines the hull. Since there may be faults in the segmenation masks, those faulty areas will bring the visual hull directly up to the camera in a cone, since everything in those observed pixels might contain some part of the model, but the segmentation mask simply doesnt know if there is something, or not. Since they have N segmentation mask, this leads to N different visual hulls, which are then combined to one hull over all the maps.
So, H defines the area of the visual hull, with values of 1 inside the hull and 0 outside of it. What now should be penalized, is every vertex outside of this visual hull, meaning the distance towards the visual hull of it, where 0 says, that it is inside the hull, defined in $H(v)$.

$$E_{hull}(G) = \sum_{m=1}^{M} H(\tau(v_m; G))^2 \quad (7)$$

Since the distance of the vertex to the visual hull might get rather expensive, they use some approximation with applying Gaussian blur over the volume function H.

4.4 Correspondence Term

The next term says, that a point in the reference Frame should be warped close to a point in the observed data, which is the corresponding point in the motion. For this function, they amended the Global Patch Collider framework of [Wang et al. 2016], which is a learning based algorithm using decision trees to find

matches between to frames using two RBG Images as input. Using this split function with the learning parameters θ, and the pixel offset $\{p, u\}$

$$f() = L \text{ if } I_s\left(p + \frac{u}{d_s}\right) - I_t\left(p + \frac{v}{d_t}\right) < \theta \quad \text{, otherwise R} \tag{8}$$

The first amendment was, that they used the depthmaps of the last frame and the momentary frame instead of RGB to not be bound to scaling, since the depth value itself tells the algorithm, how many pixels the object will occupy if it is at a specific distance towards the camera. That is given by the scaling offsets $\{u, v\}$.
They found out, that 5 trees with 15 levels gave them the best result but they additionally didnt only take matches, if all those trees resulted in one unique match, since that was not enough for them. They introduced a voting scheme over all those trees, which says, that every tree, that found a unique match for one point will vote for this match and which match then gets the most votes for this point, will get taken as the match of this point. Since GPC uses two Images to calculate the references, they get a set of matches for every depthmap, consisting of $\{u_{nf}^{prev}, u_{nf}\}$,where the first pixel is the previous point and u_{nf} the estimated pixel in the depthmap, to which u_{nf}^{prev} should have moved. Those matches then are used for the correspondence term. But first they have to get the voxel in the old frame with this function.

$$q_{nf} = \arg\min_{v \in V} \| \Pi_n \left(\tau\left(v; G^{prev}\right) - u_{nf}^{prev}\right) \tag{9}$$

This gives us the pixel in the n'th depthmap, which was closest to u_{nf}^{prev}. Because the old ED-model warps the key-volume to the previous scene, we can project every vertex v in the model towards the depthmap D_n via $\Pi_n(v)$, and thus we get the vertex, which represented the matched pixel in the last frame. This vertex then is warped by the new ED-Model G, and the distance of this transformed point towards the estimated matched point in the current observed date u_{nf} will be taken into the correspondence term:

$$E_{corr}(G) = \sum_{n=1}^{N} \sum_{f=1}^{F_n} \rho\left(\|\tau(v; G) - P_n\left(u_{nf}\right)^2\right) \tag{10}$$

This contains the sum over all the depthmaps and their estimated correspondence matches.

4.5 Optimization

In the next chapter will be shown how this energy function $E(G)$ gets minimized, which leads to the best possible transformation for the key volume. After all the Terms of the Energy function have been presented you might see, that all of the terms are squares. This means we can find a Paremeter-set X, so that $E(G) = f(X)^T f(X)$.This means that the minimization of this function is a

standard least squares minimization problem. For the solving of this problem they use 4 iterations of the Levenberg Marquardt Algorithm:

$$\left(J^t J + \mu I\right) \tag{11}$$

The Levenberg Marquardt altorithm calculates each step this equation with this damping factor μ, accepting it, if the resulting energy gets lower and thus increasing the damping factor to be more aggressive. Or reject it, if the Energy would get higher, decrease the damping factor to make smaller steps towards the goal and then recomputing the step.
But there is one difficulty in this algorithm, and that is, that they need the derivative of this Energy function, but there are discontinuities in $E_{data}(G)$, because of the projection of the vertex to a pixel in the depthmap and then back-projecting it to the world space, since a small deviation of the vertex will lead to a completely different pixel, this function has a lot of discontinuities. So they needed a second approximation of the approximation to be able to derive the function:

$$E_{data}(G) = \sum_{n=1}^{N} \sum_{m \in V_n(G_0)} \left(\tau^{\perp}(n_m; G_0)^T \left(\tau(v_m; G) - \Gamma_n \tau(v_m; G_0)\right)\right)^2 \tag{12}$$

This equation takes the current calculated parameters of the ED-model and fixes it in G_0 and now they use these fixed parameters for the projection and back-projection and for the transformation of the normal vectors. Since the function is derived toward G this means, that the critical terms are now constants and thus the complete function is derivable.

Evaluation of $\boldsymbol{J^T J}$ and $\boldsymbol{J^T f(X)}$ To evaluate both terms for one Levenberg Marquardt iteration, they dont store the Jacobian itself, but instead already $J^T J$ and $J^T f(X)$, since J is a way larger matrix than $J^T J$ or $J^T f(X)$, which results in way less read and write commands on the hardware.
Additionally to that, $J^T J$ is a sparse matrix, only consisting of non-zero Blocks, where two ED-nodes contribute to one block at the same time. This means, that they have to compute this block only once for every combination of 2 ED-nodes. And thanks to the simple sum, the derivative is not influenced by the number of cameras, but only of the number of ED-nodes. And on the other side, those sparse blocks can be used by the CUDA, which is able to make block wise calculation.
The computation of the Jacobian each step is still not an easy task and cant be done fully in real time, but instead of further approximation, they use a linear solving approach called preconditioned conjugate gradient (PCG), with the diagonal blocks of the old $J^T J$ as precondition.

5 Data Fusion

After the deformation model has been found, there are two tasks which can be done now:

the reference frame will be fused into the observed data to get the final model, which then will be displayed, and on the other side will the reference volume be updated with the new observed data.
The reference model itself was containing voxels x^r with their depth value d^r and a weight w^r. After it is warped via the ED-model, the voxels have a new position, which is aligned to the data, stored in an SDF. Each voxel in the data frame then takes the warped voxels $\tilde{x}^r$ near to it and averages their depth values and weights, weighted by the distance towards the warped reference voxel $exp\left(-\frac{Vert\tilde{x}^r - x^{r2}}{2\sigma^2}\right)$. To get even better results are the depth and weight values itself not taken, but a corrected version, which uses the gradient field of the SDF.

5.1 Fusion at the Data Frame

The primitive blending of this accumulated SDF and the reference frame to get the resulting scene will lead to some problems. The first problem is, that there may be a collision of reference voxels, it different Parts of the Model are warped into the same position, for example if the observed person touches his nose with his finger. To deal with this, they store at the prior step that reference voxel, that is closest to the date voxel and then they reject any voxel that is farther away, which wants to vote for this data voxel.
On the other side, there may still be misalignment between the reference voxels and the data, and those false values should not get blended into the result. This misalignment is the minimum of the distance value in the fused SDF to the surface and the distance towards the hull, if the voxel is outside of the hull. This error will be referenced by e_{x^r}, to work as a blending weight later on. If this error is bigger than some threshold, it will be completely discarded.

Volume Blending After the generating of the fused Date Volume and the upper corrected reference Volume, we want to blend those two together to get the final Volume. This cant be done naively, since there are misalignments as shown above. to deal with this, they blend it together by some weight, which is $(1 - e_{voxel})$. This voxel error is the average of the normalized misalignment of a reference voxel projected to each depthmap.

5.2 Fusion at the reference Frame

Since the warped reference frame still contained some errors, the voxel, which were rejected will be completely discarded, and instead the closest data voxel will be taken. And afterwards every voxel will update its depth- and weight-values by projecting it to the depthmaps and taking the average of it.
This will keep the quality of the model over the frames, but it still cant handle large topology changes. To accommodate this, the reference model will be completely discarded and reset by only taking the blended result and taken as future key volume.

6 Results

With their new approach it is possible to capture challenging scenes, for example large motions of kicks and punches, large topology changes like throwing a ball from one person to another. They dont rely on priors, so it is possible to even capture high quality reconstruction of dogs or arbitrary objects in real-time, even if there are a lot of objects in the scene.
They achieve the 30 frames per second criteria by implementing everything in CUDA on GPU. They used in their 24-Camera setup 12 Computer, each with two NVIDIA Titan X GPUs. They precompute every task for one trinocular camera, meaning Stereo PatchMatch algorithm $21ms$, generating the segmentation masks $4ms$ and estimating the dense correspondence field 5. And then everything will be sent to one Master Computer with a single NVIDIA Titan X GPU, which first preprocesses everything $3ms$ and then computes the rigid pose estimation via ICP $2ms$, the ED-Model with the optimization $20ms$ and then fusion $6ms$. This together would be approximately 60ms, and not the expected 33ms, but once the 12 Computers are ready, they can already start computing for the next frame.
In regards to results, they show that their approach leaves the other real-time approaches behind by far and are almost as accurate as the offline approach of[Collet et al. 2015], which uses more cameras for the computation. They are maybe as accurate, if they used as many cameras as Collet did.

7 Limitations

Even though they have shown so much robustness in the motion capturing, there are still some limitations towards their approach.
For one, if the motion is too large, the tracking of the motion will get lost, meaning that the key volume wont be warped well to the data and only the fused observed data will get used for the blending, which results in some noisy output, or even oversmoothed for small details, like a face. On the other side, if there are segmentation faults in a frame, those faults would create some holes in the model, because the visual hull would be wrong at those areas. And secondly, at the moment, where the large topology changes occur, will most likely produce artifacts, since this approach handles large topology-changes by resetting after a certain number of frames, and cant handle it directly in each frame.

8 Conclusion

This paper presented compelling results in their approach of real-time multi-view motion capture. they introduced a new Pipeline, using a novel Energy Function with the new defined visual hull and an amended learning based Correspondence Field estimation. They combined the idea of updating the reference frame intrinsically and always using the observed data with the concept of key volumes.

References

[July 2016] Dou, M. et al.: Fusion4D: Real-time Performance Capture of Challenging Scenes (SIGGRAPH2016)