Prioritized Multi-View Stereo Depth Map Generation Using Confidence Prediction

Christian Mostegel[a,*], Friedrich Fraundorfer[a], Horst Bischof[a]

[a]*Institute for Computer Graphics and Vision, Graz University of Technology, Austria*

Abstract

In this work, we propose a novel approach to prioritize the depth map computation of multi-view stereo (MVS) to obtain compact 3D point clouds of high quality and completeness at low computational cost. Our prioritization approach operates before the MVS algorithm is executed and consists of two steps. In the first step, we aim to find a good set of matching partners for each view. In the second step, we rank the resulting view clusters (i.e. key views with matching partners) according to their impact on the fulfillment of desired quality parameters such as completeness, ground resolution and accuracy. Additional to geometric analysis, we use a novel machine learning technique for training a confidence predictor. The purpose of this confidence predictor is to estimate the chances of a successful depth reconstruction for each pixel in each image for one specific MVS algorithm based on the RGB images and the image constellation. The underlying machine learning technique does not require any ground truth or manually labeled data for training, but instead adapts ideas from depth map fusion for providing a supervision signal. The trained confidence predictor allows us to evaluate the quality of image constellations and their potential impact to the resulting 3D reconstruction and thus builds a solid foundation for our prioritization approach. In our experiments, we are thus able to reach more than 70% of the maximal reachable quality fulfillment using only 5% of the available images as key views. For evaluating our approach within and across different domains, we use two completely different scenarios, i.e. cultural heritage preservation and reconstruction of single family houses.

Keywords: Multi-View Stereo, Machine Learning, Confidence Measures, View Prioritization, Image Clustering, View Cluster Ranking

1. Introduction

In this work, we aim to improve the efficiency of multi-view stereo (MVS) approaches based on depth maps. This type of approach is very popular (e.g. (Goesele et al., 2007; Rothermel et al., 2012; Zheng et al., 2014; Galliani et al., 2015; Schönberger et al., 2016)) as it is inherently parallelizable and delivers state-of-the-art results. One drawback of such approaches is that they typically generate one depth map per image in the dataset. For modern cameras, this means that 3D points in the order of 10^7 are created per image. With a few hundred images this leads to billions of points, that have to be stored, visualized and/or handled by subsequent processing steps such as depth map fusion or surface reconstruction. In this work, we propose a way to significantly reduce the number of generated points, while at the same time preserving the reconstruction accuracy and completeness to a large extent (see Figure 1). This does not only speed up the depth map computation process, but also significantly reduces the load for all subsequent steps.

The key to our improved efficiency is a combination of geometric reasoning based on a preliminary scene reconstruction and machine learning to represent important properties of MVS that cannot be geometrically

[*]Corresponding author

Email addresses: mostegel@icg.tugraz.at (Christian Mostegel), fraundorfer@icg.tugraz.at (Friedrich Fraundorfer), bischof@icg.tugraz.at (Horst Bischof)

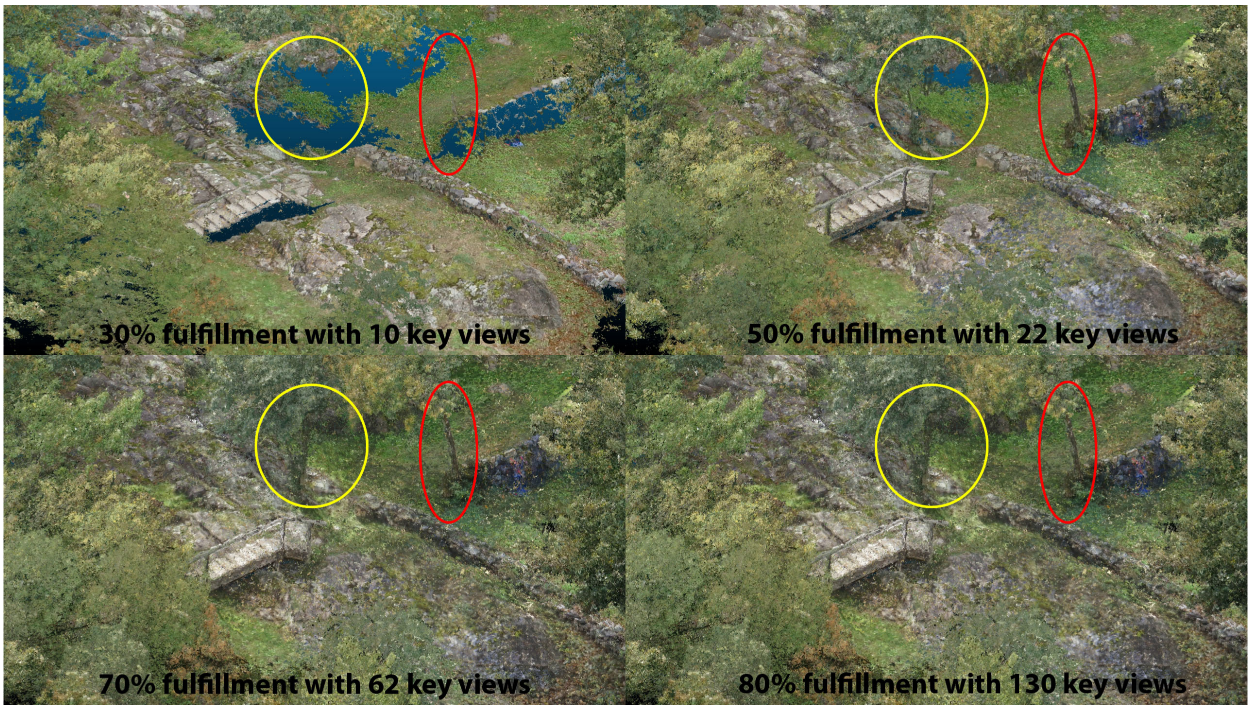

30% fulfillment with 10 key views
50% fulfillment with 22 key views
70% fulfillment with 62 key views
80% fulfillment with 130 key views

Figure 1: View Cluster Prioritization. Our approach allows us to prioritize/rank view clusters (i.e. key views with matching partners) such that a highly complete and accurate point cloud can be obtain with a very small fraction of the available images. Here, we show the point clouds from the raw depth maps of the view clusters (with 11 matching partners) ranked with our approach after reaching 30%,50%,70% and 80% of the maximal achievable quality fulfillment (i.e. completeness with respect to a desired ground sampling distance and accuracy of 1cm). Already with 50% fulfillment and only 22 key views (i.e. 1.8% of the available images), most parts of this complex scene are already contained in the reconstruction (red ellipse) and only a small part is missing (yellow ellipse). With 70% fulfillment, even strongly occluded parts such as tree trunks (see ellipses) are contained in the point cloud, although this point cloud is computed from only 62 key views (i.e. 5% of the available images). For going from 70% to 80% fulfillment, the number of necessary key views already has to be more than doubled, however, the visual difference between those two point clouds is nearly imperceptible (see also the supplemented video).

modeled. These unmodeled properties stem from the fact that MVS is an ill-posed task and to solve this task all MVS algorithms have to rely on a combination of some similarity measure (e.g. Census Transform or Normalized Cross Correlation) with a set of reasonable assumptions about the scene. The most popular assumptions include visual saliency, local planarity and a static scene. While these assumptions work well in many environments, there are also many environments where parts of the assumptions do not hold. Thus, MVS reconstructions very often contain outliers and/or fail to reconstruct certain objects completely.

In this work, we use our unsupervised machine learning framework (Mostegel et al., 2016a,b) to predict these failures and help us to reduce the number of key views (i.e. the image for computing a depth map) and necessary matching partners. Our method consists of two main steps. First, we select suitable matching partners for each view. Second, we prioritize/rank the resulting view clusters (i.e. views with matching partners) depending on their impact on a quality fulfillment function. This quality fulfillment function respects important photogrammetric parameters, such a ground resolution and 3D uncertainty, together with the scene coverage. The confidence prediction supports this whole process and allows us to obtain this ranking without having to execute the actual MVS algorithm within the ranking procedure. We formulate this quality fulfillment function as monotone submodular function and optimize this function with our ranking procedure in a greedy fashion. Although the overall problem is still NP-hard (as it includes the NP-hard maximum coverage problem), this formulation gives us strong optimality guarantees in the function space (Nemhauser et al., 1978).

This formulation has many advantages. First of all, the computed quality fulfillment function provides the opportunity to decide how many view clusters are necessary to obtain a certain quality fulfillment level. Thus, the operator can either choose to reconstruct the n best view clusters and has a estimation of the expect level of fulfillment or can simply query how large n should be to reach a certain level. The second advantage is that the inherent parallelism of MVS based on depth maps is maintained as our ranking procedure happens before executing the MVS reconstruction step. Third, the overall efficiency of the MVS reconstruction step can be significantly improved without changing the MVS algorithm itself. Thus, we were able to obtain a quality fulfillment (i.e. completeness with respect to a desired resolution and accuracy) of 70% with only 5% of the available view clusters. This leads to a speed up factor of approximately 10 and a complexity/memory reduction factor of approximately 20 for the resulting point cloud without losing much information.

2. Related Work

Our prioritization approach is related to two different research areas, which are namely: Matching partner selection and Next-Best View (NBV) planning. In the following, we discuss the relation to these two interwoven areas.

Matching Partner Selection. Most MVS approaches based on depth maps formulate some kind of heuristic to select the k best matching partners for each key view to increase the efficiency of MVS. The heuristics for matching partner selection strongly depend on how the images are acquired (structured versus unstructured) and the requirements of the MVS algorithm. If the images are acquired in a regular grid, the k closest images are a natural choice to maximize the completeness. For more unstructured settings, the connectivity in the sparse reconstruction (i.e. how many sparse 3D points are shared between two cameras) is typically a more reliable cue to determine if the dense MVS matching step will work or not. To avoid that images with insufficient parallax are chosen as matching partners, Goesele et al. (2007) combine the connectivity with geometric constraints in a greedy fashion. Their formulation down-weights connections (shared features) with a triangulation angle below $10°$ and dissimilar scale. Additionally to these two terms, Bailer et al. (2012) also add a coverage term, which favors connections that have not been covered by other selected images. Shen (2013) use a formulation without connectivity only based on the geometric constraints on the triangulation angle and the distance between images. For very small datasets where all images nearly see the same part of the scene (as in the DTU dataset (Aanæs et al., 2016)), also random selection of matching partners can lead to good results (Galliani et al., 2015). Of all formulations mentioned above, (Bailer et al., 2012) seems to be the closest related formulation to our approach. Similar to their approach, we also use the connectivity to reduce the set of potential matching partners. But instead of using hand-crafted models for the triangulation angle, our approach uses the learned weighting function (encoded as the MVS confidence). Another difference is that we use a surface mesh to evaluate geometric constraints such as triangulation angle, scale and coverage. In contrast to only using sparse points, a surface mesh provides the opportunity for geometric balancing. This means that instead of having a very unbalanced set of sparse points (with many points in richly textured parts and no points in poorly textured parts), the triangle distribution of the mesh can be easily balanced over the whole scene. This can be important for low texture structures, e.g. building facades, where very often no sparse points are extracted, whereas the micro-texture can still contain sufficient information for obtaining an accurate MVS reconstruction.

Next-Best View Planning. Next-best view (NBV) planning, view planning, sensor placement, path planning for reconstruction and/or coverage, visual inspection and exploration are all closely related topics. They all have to answer the questions "what parts of the scene are already sufficiently covered?" and "how can I best improve this coverage?". In this sense, they are all related to the art gallery problem (O'rourke, 1987) or the maximum coverage problem (Feige, 1998), which both have been shown to be NP-hard.

Thus, researchers from the communities of photogrammetry, robotics and computer vision have developed and are still developing methods which are honed to very specific tasks exploiting all available domain specific knowledge to conquer this challenging task with one specific actuator/sensor setup. Some works (e.g. (Fan et al., 2016; Ahmadabadian et al., 2014a,b; Trummer et al., 2010)) thus focus on the reconstruction of small scale scenes in laboratories. For this kind of approaches we refer the reader to a recent review article of Karaszewski et al. (2016). With the increasing availability of robotic platforms, also the research in the field of structure inspection drastically increases (e.g. (Hollinger et al., 2012; Alexis et al., 2015; Bircher et al., 2016; Galceran et al., 2015)). A review of current approaches on this topic can be found in (Almadhoun et al., 2016). This field is also closely related to coverage search (e.g. (Dornhege et al., 2013; Galceran and Carreras, 2013)). For photogrammetry, there also exist several works on view planning for UAVs (e.g (Mostegel et al., 2016a; Martin et al., 2016; Schmid et al., 2012; Hoppe et al., 2012; Jing et al., 2016b,a)). While many aspects are shared with view planning, the problem tackled in this work also shows one important difference: In our case the set of images is fixed. On the one hand, this is an advantage as subtasks such as path planning do not need to be handled anymore. On the other hand, if the image constellations are suboptimal then there is no possibility to improve the situation.

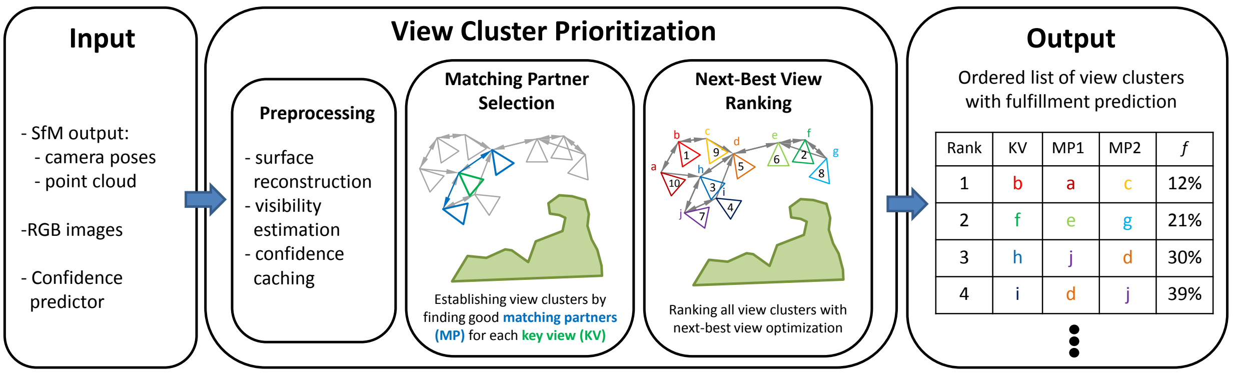
From all the works mentioned above, we see the works of Hornung et al. (2008) and Mendez et al. (2016) as most related to our task. Both approaches try to obtain a maximally complete reconstruction from a given image set by iteratively updating the geometry estimate. For updating the geometry, Hornung et al. (2008) use the complete PMVS algorithm (Furukawa et al., 2010) after each added view, while Mendez et al. (2016) execute (Weinzaepfel et al., 2013) after selecting the next-best stereo pair. We contrast from these works in several points. First of all, our aim is not an iterative update scheme with the MVS algorithm in the loop, but instead we aim to rank suitable view clusters (i.e. key views with matching partners) according to their importance for the final reconstruction. On the one hand, we are thus able to obtain a complete reconstruction with a fraction of available view clusters. On the other hand, we can pre-compute the whole key view ranking, which allows us to maintain the natural parallelization capabilities of depthmap-based MVS in the execution phase. Instead of updating the geometry estimate in each iteration, we use all measurements of the sparse reconstruction to obtain a high quality surface mesh at a low computational cost once at the start of our algorithm. The big advantage of this strategy is that we do not require the MVS algorithm in the loop, but use our MVS confidence prediction to estimate the chances of a good reconstruction. Through this formulation, we combine the advantages of NBV planning and depthmap-based MVS, i.e. data reduction and parallelism. This leads to a light-weight approach, which can be easily integrated in most photogrammetric pipelines to increase the overall efficiency.

3. View Cluster Prioritization

The basic element of our prioritization is a view cluster, which stands for a key view (i.e. an image used for generating a depth map) together with its matching partners. The aim of our view cluster prioritization is to establish a ordering of view clusters such that any subset of the ranked entries from entry 1 to n retains a maximum of information about the scene for any n. In this work, we separate the problem of view cluster prioritization into two main subproblems.

The first subproblem is matching partner selection. For this subproblem, let us assume that we have decided that we want to generate a depth map for a distinct view/image, which we will further call "key view". To generate a depth map, we require other images that observe the same part of the scene from different positions. Choosing a subset of images for this task has a lot of impact on the quality of the resulting depth map. On the one hand, choosing images with a large baseline to the key view will lead to depth values with high accuracy. On the other hand, it will also make the matching task (i.e. finding correspondences between the images) significantly harder, which will negatively influence the completeness of the depth map. Selecting more images will be beneficial for completeness and accuracy, however, the matching time also rises linear with the number of matching partners. So for increasing the efficiency, the number of matching partners is typically kept as low as possible. Thus, in this subproblem it is our aim to find the best subset of k matching partners such that k can be very small and we still obtain high quality results in terms of accuracy and completeness.

The second subproblem is next-best view ranking. For this subproblem, let us assume that we have already decided the set of matching partners for each key view. While it is possible to generate one depth map for each view, this approach leads to a massive amount of highly redundant data. Thus it is our aim to rank the view clusters (key views with matching partners) such that the most useful view clusters can


Input
- SfM output:
- camera poses
- point cloud
-RGB images
- Confidence predictor
View Cluster Prioritization
Preprocessing
- surface reconstruction
- visibility estimation
- confidence caching
Matching Partner Selection
Establishing view clusters by finding good matching partners (MP) for each key view (KV)
Next-Best View Ranking
Ranking all view clusters with next-best view optimization
Output
Ordered list of view clusters with fulfillment prediction

Rank	KV	MP1	MP2	f
1	b	a	c	12%
2	f	e	g	21%
3	h	j	d	30%
4	i	d	j	39%

Figure 2: View Cluster Prioritization. As input our approach requires the SfM output (i.e. the camera poses and the sparse point cloud), the original RGB images and the pre-trained confidence predictor. From the SfM output, we extract a surface mesh, which is then used for visibility estimation (i.e. estimation of which camera sees which mesh triangles). With this information, we then precompute the MVS confidence for each image as described in (Mostegel et al., 2016a) and cache the predictions for each visible mesh triangle. Using this data, we then find good matching partners for each view, which results in a set of view clusters. These view clusters are then ranked by their importance for the overall reconstruction in a next-best view scheme. The output is ordered list of view clusters (i.e. key views with matching partners) together with a fulfillment prediction f for each entry.

be processed first. As by product, our approach delivers a fulfillment prediction (with respect to a desired ground resolution and 3D accuracy) for each added view cluster. This information makes it possible to determine how many and which view clusters are necessary to obtain a certain level of fulfillment prior to executing the MVS algorithm itself.

In Figure 2, we show an overview of the complete ranking procedure, which is explained in full detail in the remainder of this section.

3.1. *Preprocessing*

As input our approach requires the RGB images, a pre-trained confidence predictor (Mostegel et al., 2016a) and the structure-from-motion output including the corresponding camera poses and a sparse 3D point cloud. For our metric quality constraints (ground sampling distance and 3D accuracy), we also require that the structure-from-motion output was transformed metrically. In our experiments, we used automatically detectable GCPs (Rumpler et al., 2017) for this purpose.

From this input, we then robustly extract a surface mesh with a Delaunay triangulation of the sparse point cloud (Labatut et al., 2007) with the same parameters used in (Mostegel et al., 2017). The amount of triangles of the resulting surface mesh is in the order of the number of sparse 3D points. For view cluster prioritization, the resulting geometry complexity is typically already unnecessarily high. Therefore, we first simplify the mesh reconstruction using an adaption of quadratic edge collapse decimation (Garland and Heckbert, 1997), which terminates when 95% of all triangle edges are above r times the desired ground sampling distance. In our experiments, we found $r = 20$ to be good value, as with this value all important 3D structures are still contained in the mesh, but the number of triangles is drastically reduced. To balance the triangle size independent of the 3D topology, we then refine large mesh triangles by iterative sub division until all edge lengths are below e times the desired ground sampling distance. We found that $e = 5 \times r$ lead to a good trade-off between simplicity and balance. Note that the surface mesh is computed exactly once at the beginning of our approach and is not refined with the MVS output as our prioritization approach works before the MVS algorithm is executed.

3.2. *Matching Partner Selection*

For computing a depth map with MVS, each key view requires a set of matching partners which observe the same scene from different view points. In this step of our approach, we try to find a good set of k

matching partners in the sense that completeness and accuracy are optimized simultaneously. We say that we are aiming for a *good* set (opposed to the optimal set) as even the solution space for a single key view can be extremely large. The size of the solution space for this task is the same as for unordered sampling without replacement, i.e. the binomial coefficient $\frac{n!}{k!(n-k)!}$, where n is the size of the potential set of matching partners. Even for quite small numbers, like picking $k = 6$ matching partners out of $n = 23$ possibilities, the solution space is already larger than 100k.

For this purpose, we follow other works in the field (e.g. (Goesele et al., 2007; Bailer et al., 2012)) and use the connectivity between the images for a complexity reduction. Thus, we first reduce the set of potential matching partners to the top n most connected images. We evaluated the connectivity based on the number of sparse points shared with the key view. From this set we then draw y combinations. As the connectivity can be a very strong cue, we also want to ensure that the most connected images are included in the drawn y combinations. Thus, we first draw all possible combinations of the q most connected images, where q is the largest set size for which the solution space is less or equal $\frac{y}{4}$ (i.e. $\frac{q!}{k!(q-k)!} \leq \frac{y}{4}$). The rest of the $y - q$ combinations are drawn randomly from the larger set of the n most connected images. In this way at least 75% of the combinations are drawn completely random, whereas we also consider the most connected combinations. For each drawn combination of matching partners (c^k), we now evaluate the following four fulfillment functions in relation to the key camera c_{key} and a triangle t observed from c_{key}:

(1) The *coverage* is modeled as a Boolean function f_{cov}, which is true if sufficient cameras ($\geq x$) observe the triangle t and false otherwise.

(2) The *resolution* requirement (px/m^2) is defined as a truncated ratio:

$$f_{res} = \min\left\{\frac{r}{r_d}, 1\right\}, \tag{1}$$

where r_d is the desired resolution and r is the estimated resolution. We determine r by projecting the 3D triangle into c_{key} and dividing the 2D triangle area by the 3D triangle area. The desired resolution r_d can be computed from a desired ground sampling distance g_d as $r_d = 1/g_d^2$.

(3) The fulfillment of the *3D uncertainty* requirement for a desired accuracy a_d is defined as:

$$f_{unc} = \min\left\{\frac{a_d}{\sqrt{u}}, 1\right\}, \tag{2}$$

Here, u stands for the maximum Eigen value of the covariance matrix Cov_{3D} related to a triangle's centroid Hartley and Zisserman (2004).

(4) The last fulfillment function is the MVS confidence for k matching partners. In (Mostegel et al., 2016a), we presented a way to learn and predict this confidence without any ground truth for two matching partners. Here we present a way to generalize the confidence prediction step to an arbitrary number of k matching partners. For this purpose, we assume that the MVS reconstruction process follows a stochastic process and can be modeled as a combination of unary MVS confidence predictions.

Under this assumption, we can use probability theory to reason about the overall probability of obtaining a successful measurement for a pixel of a key view given a set of matching partners. For a successful measurement, most MVS approaches require successful matches to at least two matching partners. Thus, we formulate the overall probability of obtaining a successful measurement, as the probability of getting successful matches to at least two matching partners. In Appendix A, we derive this overall probability by growing a binary probability tree. In the following, we apply the general solution described in Appendix A to our specific problem.

In our problem, we are given a surface mesh with a set of triangles T and a set of images C. For a specific combination of a key view $c_{key} \in C$ with k matching partners $c^k \in C^k$, we want to predict the chances of successfully reconstructing the depth of the pixels in which a triangle $t \in T$ projects.

Thus, we now end up with the following equation for the overall probability of a successful match considering all matching partners:

$$f_{conf}(t, c_{key}, c^k) = \sum_{i=2}^{k} \left((-1)^i \cdot (i-1) \cdot \sum_{c^{(i)} \in C_k^{(i)}} \left(\prod_{j=1}^{i} f_{conf}\left(c_{key}, c_j^{(i)}\right) \right) \right), \tag{3}$$

where $C_k^{(i)}$ is the solution space for drawing subsets of i cameras from the available set of k matching partners, $c^{(i)}$ is one of these subsets and $c_j^{(i)}$ is one camera of this subset and $f_{conf}\left(c_{key}, c_j^{(i)}\right)$ is the pair-wise MVS confidence for camera $c_j^{(i)}$ with the key camera c_{key}. We compute this pair-wise MVS confidence by averaging the unary confidence values of the two cameras. We extract the unary confidence value for a triange t by averaging the pixel-wise confidence predictions of all pixels that lie inside the projection of triangle t into the individual camera. The confidence predictor is trained and executed as described in (Mostegel et al., 2016a). Note that for improved efficiency, we also precompute and store the unary confidence values for each camera-triangle pair in our preprocessing step.

Based on these four fulfillment functions, we define the fulfillment for a triangle t as:

$$f(t, c_{key}, c^k) = (\alpha f_{res}(t, c_{key}) + (1-\alpha) f_{unc}(t, c_{key} \cup c^k)) \cdot f_{cov}(t) \cdot f_{conf}(t, c_{key}, c^k), \tag{4}$$

where α defines the relative weight between the resolution and uncertainty fulfillment.

With this basis function, we now compute a combined fulfillment score for each set of drawn matching partners as

$$f(c_{key}, c^k) = \sum_{t \in T_{c_{key}}} f(t, c_{key}, c^k), \tag{5}$$

where c_{key} is the key camera, c^k is the selected set of k matching partners, the triangle set $T_{c_{key}}$ is the set of triangle visible in c_{key} with $T_{c_{key}} \in T_z$. T_z is a fixed fraction z of all the available triangle set T_{all} (i.e. $|T_z| = |T_{all}|/z$).

Of all drawn combinations of matching partners, we now select the one that maximizes this combined fulfillment. The combination of key view and selected combination, we will further call view cluster. We execute this step for all available images.

3.3. Next-Best View Ranking

In general, image acquisition for photogrammetric reconstruction is done with a lot of redundancy to ensure that the whole scene is sufficiently covered. However, a drawback of this acquisition technique is that some images do not contain any additional information compared to their neighbors. Finding the best and minimal subset for a sufficiently complete reconstruction is a very hard task. In fact, subproblems of this task are known to be NP-hard. E.g. if an oracle told us that the minimal number of key views for sufficiently covering the scene is n, we would still have to solve the NP-hard maximum coverage problem (Feige, 1998). While solving this problem optimally is computationally intractable (unless P = NP), there exist approximation approaches with theoretic quality guarantees for the found solution. In particular, if the objective function is a nonnegative monotone submodular function, then a greedy algorithm is guaranteed to select n views such that the reached objective value is within 63% of the optimal objective value for the same amount of views (Nemhauser et al., 1978). This fact (and the fact that n is not easily estimated) motivated us to solve this problem with a greedy algorithm and a nonnegative monotone submodular objective function.

Objective function. Now let us formalize the objective function which we aim to maximize, i.e. the overall fulfillment, as:

$$f_o(V, T) = \frac{1}{|T|} \sum_{t \in T} \max_{v \in V} f(t, v), \tag{6}$$

where T is the set of all mesh triangles, $|T|$ is the size of this set, t is one of these triangles, V is the set of view clusters and v is one of these view clusters. $f(t, v)$ is the triangle fulfillment as defined in Equation 4 with $v = \{c_{key}, c^k\}$. Note that maximizing this objective function for any fixed number of view clusters results in an NP-hard problem, however, the quality guarantees of Nemhauser et al. (1978) for a greedy algorithm still apply.

Algorithm. We formulate our ranking procedure as a greedy algorithm. In each iteration, we select the next-best view cluster, i.e. the view cluster with the highest fulfillment gain. We define this gain for a view cluster v_i as:

$$g(v_i, V', T) = f_o(\{V' \cup v_i\}, T) - f_o(V', T), \tag{7}$$

where V' is the set of already selected view clusters. For an efficient solution to this problem, we propose the following algorithm.

Our algorithm first starts by estimating the visibility between each view/camera and the triangle mesh. This visibility information is stored in each triangle (i.e. each triangle knows which camera has a direct line of sight to it). Based on this information, we precompute the fulfillments $f(t, v)$ for all triangles $t \in T$ with sufficient camera views ($\geq x$, related to the coverage fulfillment f_{cov}) and all view clusters. This fulfillment $f(t, v)$ is then stored within a map in each triangle t.

For each view cluster v, we now compute the initial fulfillment gain $g(v, V', T)$ with $V' = \{\}$ (i.e. an empty set of selected view clusters). Each pair of view cluster and gain values $\{v, g(v, V', T)\}$ is inserted into a priority queue, where now each view cluster is ranked by its fulfillment gain. While building up this data structure has a complexity of $\mathcal{O}(n \log n)$ with $n = |V|$, we can now use lazy updates in each iteration and avoid a significant number of unnecessary computations.

In each iteration, we now select and pop the top element of the queue, i.e. the element with the highest fulfillment gain, and add the selected view cluster v_s to the set of selected view clusters V'. Then we update the current fulfillment for each triangle t that is observed by v_s. This current fulfillment is stored inside the triangle t and computed as $f'(t, V') = \max_{v \in V'} f(t, v)$. Based on this information, we now lazily update all elements of the priority queue. This means that we select and pop the top element of the queue and update the fulfillment gain of the corresponding view cluster v_u. We store all such temporarily removed view clusters V_u in a separate data structure and keep track of the maximum fulfillment gain g_{max} over all these values. We can stop the update procedure if $g_{max} \geq g(v_{top}, V', T)$, where v_{top} is the current top element of the priority queue. Then all temporarily removed view clusters V_u are reinserted into the priority queue. Now the next iteration can start. We terminate if $g(v_{top}, V', T) = 0$ or the priority queue is empty.

Note that all computational components (the visibility estimation, the confidence prediction and the fulfillment computation) are computed only once at the beginning of algorithm and the execution of the actual MVS algorithm is avoided altogether. This makes the ranking procedure very light weight in terms of computation time. The final output of this ranking procedure is an ordered list of view clusters (key views with matching partners) together with the estimated fulfillment up to this point.

4. Experiments

In our experiments, we use two different environments. The first environment contains cultural heritage sites in the valley of Valcamonica, Italy. The cultural heritage sites consist of hundreds of open rock surfaces covered with prehistoric rock art. The sites all look very similar to each other and contain a limited amount of structures; i.e. the sites themselves are open rock surfaces are surrounded by lawn, bushes, wooden man-made structures (e.g. hand rails) and trees. The high amount of vegetation makes 3D reconstruction in this environment very hard. Thus, this environment represents the ideal use case for our machine learning approach, as it is a hard and reoccurring task. For our experiments, we use the confidence predictor of Mostegel et al. (2016a), which was specially trained for this kind of environment. For evaluation, we then use a test site which was not included in the training. This allows us to demonstrate the full potential of our approach in the same environment in which is was trained.

The second environment, we use for evaluation, is a suburban setting of single family houses. We will use this scenario only for evaluation and not for training. With this setting we evaluate the domain generalization

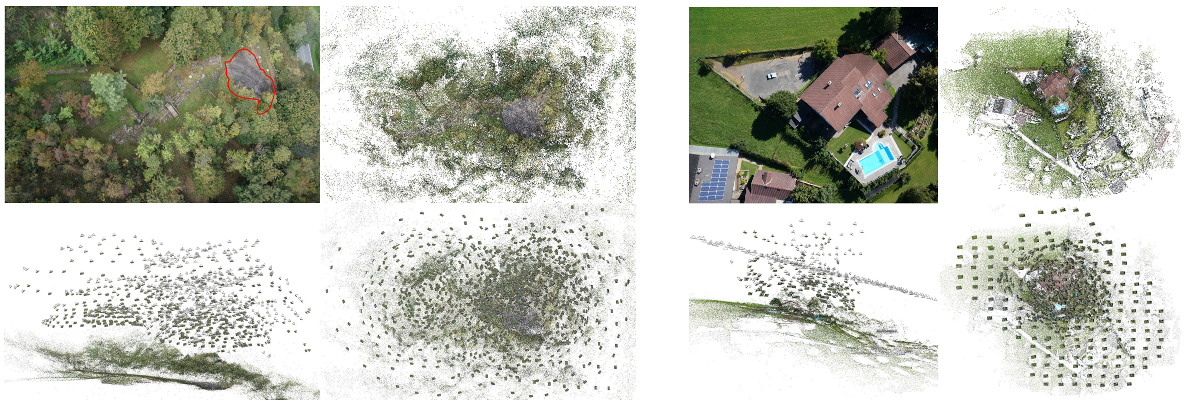

(a) Valley Dataset (b) House Dataset

Figure 3: Valley and House Dataset. Top left: One image of the dataset shows an overview of the scene. Top right: A nadir view of the sparse point cloud. Bottom right: The camera poses with the sparse point cloud from the same nadir view point. Bottom left: A side view of the camera poses with the sparse point cloud. For the Valley Dataset, ground truth of the rock formation Seradina 12C (marked in red) was acquired with a laser scanner.

properties of our confidence predictor. I.e. how does the predictor react if it is confronted with structures it has never seen before?

Using these two datasets, we structure our experiments in three main parts. First, we evaluate only the matching partner selection in Section 4.2 and then only the ranking performance in Section 4.3. From both of these experiments, we then take the best performing baseline approach and evaluate this combination against our full approach in Section 4.4.

4.1. Evaluation Details

For all our experiments, we use the same confidence predictor. This confidence predictor was trained for the environment of Valcamonica on 5000 images of 8 different sites (Mostegel et al., 2016a) with SURE (Rothermel et al., 2012) as MVS algorithm. Note that this dataset does not have any overlap with the two datasets. We selected SURE as our main MVS algorithm as it a widely accepted photogrammetric software, which yields high quality results.

For the matching partner selection, we use the $n = 22$ most connected images, draw $y = 100$ combinations and set the triangle fraction factor to $z = 10$.

For our fulfillment functions, we set the necessary number of cameras to $x = 3$ in line with the default parameter of SURE (Rothermel et al., 2012) (i.e. 2 matching partners). We set the desired ground sampling distance g_d and desired accuracy a_d to 1cm and the weighting parameter α to 0.5.

Valley Dataset. The dataset consists of 1236 registered images of a complex scene in the valley of Valcamonica (Figure 3). The images were acquired on 3 consecutive days in regular patterns (i.e. grids and domes) and in semi-structured ways using autonomous image acquisition (Mostegel et al., 2016a). The grid was acquired with 80% overlap and 8mm GSD. The domes were centered on two separate locations and the GSD was varied approximately from 8mm to 16mm. For the semi-structured autonomous image acquisition (Mostegel et al., 2016a), the quality parameters were set to 8mm ground sampling distance and accuracy. Note that this dataset has no overlap with the dataset used for training the confidence predictor. The images were processed with structure-from-motion pipeline of (Rumpler et al., 2017) and geo-referenced using the same fiducial markers as in (Mostegel et al., 2016a). This resulted in a sparse point cloud of approximately 480k points, where nearly all cameras contain 3D points with 100+ connections.

House Dataset. The House Dataset contains 485 registered images of a detached house with 3000 m^2 garden. The images were taken in a regular grid and several iterations of our autonomous image acquisition

Num Matching Partners	2	3	5	11
Matching Partner Selection [s]				
Valley Dataset	0.7	1.0	1.8	23.8
House Dataset	0.7	1.0	1.8	18.2
Next-Best View Ranking [s]				
Valley Dataset	3.5	3.6	4.0	10.8
House Dataset	2.8	3.1	3.4	8.5
Overall Average [s]				
(Valley+House)/2	3.9	4.4	5.5	30.7
Prioritization Time / Matching Time [%]				
(Valley+House)/2	4.13%	3.11%	2.36%	5.98%

Table 1: We show the average time consumption of the two main parts of our approach (i.e. Matching Partner Selection and Next-Best View Ranking) in seconds per key view. All initialization steps are included in the timings. The last two row show the overall average and the relative time consumption between our prioritization method and the matching time of SURE (Rothermel et al., 2012).

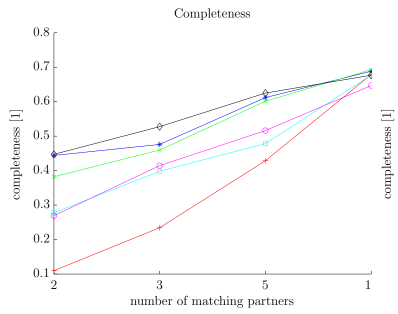
approach (Mostegel et al., 2016a). The grid was acquired with 80% overlap and 1cm GSD, and for our approach with 1cm for ground resolution and accuracy. While some scene structures are similar to the Valley environment (grass and trees), most structures have never been seen in training (roofs, house walls, swimming pool, cars, etc.).

Computation Time. We ran all our experiments on the same desktop computer with a Intel Core i7-4771 CPU, 32GB of RAM and GeForce GTX 770. We use the linux version of SURE (Rothermel et al., 2012), which only uses the CPU. With this version of SURE the average depth map computation time per matching partner on our datasets is 46.6s. In our approach, the mesh computation and the visibility casting are negligible compared to the more computationally intense parts of our approach (both parts together are finished in less than two minutes for both datasets). Our confidence prediction takes a constant time of 1s per image. In Table 1, we show the time consumption for the matching partner selection and the next-best view ranking for each dataset and number of matching partners separately. Note that our prioritization approach is very light weight and only requires 2%-6% of the time the matching procedure of SURE (Rothermel et al., 2012) itself. This means that if we reduce the number of required key views to 5%, we effectively reduce the overall run-time to 7%-11%, which speeds up the full MVS reconstruction process by a full order of magnitude.

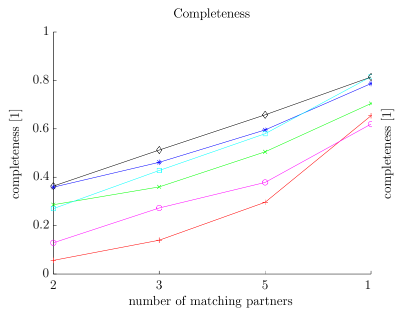
4.2. Selecting the k-best Matching Partners

In this experiment, we evaluate the first subtask of our view cluster prioritization, i.e. the matching partner selection. In this subtask, the algorithm is given a fixed set of key frames and for each key frame the k-best matching partners should be selected. Thus we evaluate three factors in this experiment, i.e. completeness, outlier percentage and RMSE (root mean square error). However, this evaluation requires ground truth. For the Valley Dataset, we have a traditional ground truth only for a small part of the scene, i.e. a rock formation called Seradina Rock 12C (Figure 3). This ground truth was captured with a Riegl VZ-400 Laser Scanner and has an accuracy of 5mm. While we also use this ground truth in our evaluation (denoted as "Laser GT"), it is limited to a mostly flat rock surface.

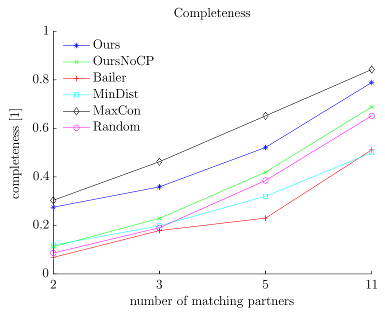
In order to evaluate the three quality factors also for the other parts of the scene (including the vegetation), we have to use a more unconventional approach. For this purpose, let us first formally define the task of the matching partner selection. In the task of matching partner selection, an algorithm shall select a subset M_k of k matching partners (images) for a specific MVS algorithm such that the resulting depth map maximally resembles the depth map produced with the same algorithm and a much larger set of matching partners M_{large} with $M_k \subset M_{large}$ and $|M_k| \ll |M_{large}|$. With this definition, we can use the depth map



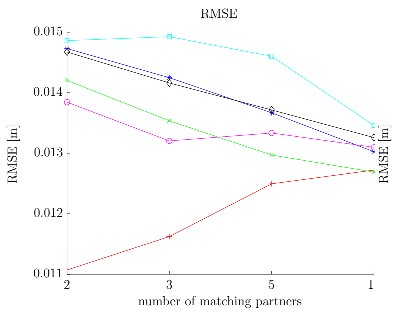
(a) Valley Laser GT



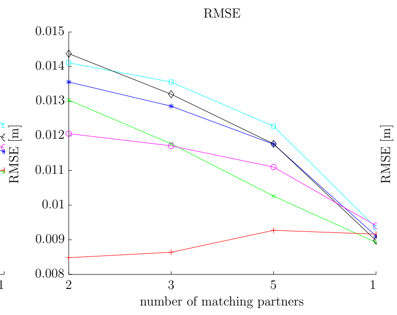
(b) Valley 22 MP



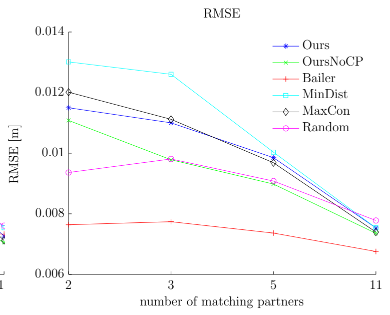
(c) House 22 MP



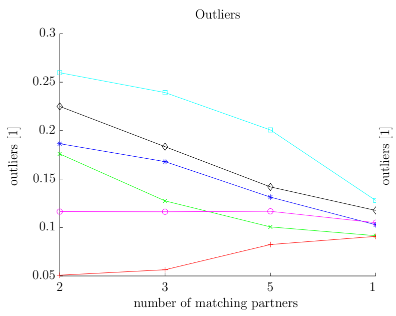
(d) Valley Laser GT



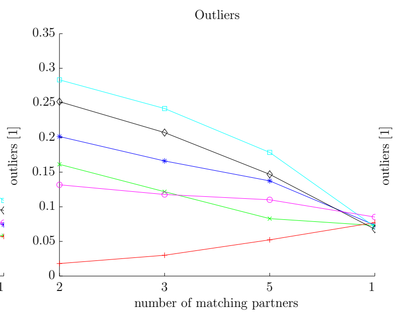
(e) Valley 22 MP



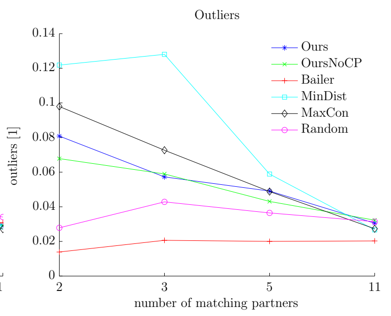
(f) House 22 MP



(g) Valley Laser GT



(h) Valley 22 MP



(i) House 22 MP

Figure 4: Matching partner selection experiment on the **Valley** and **House** Datasets. We show completeness, RMSE (Root-Mean-Square Error) and outlier ratio compared to the output with the 22 most connected matching partners (22 MP) and the laser ground truth of a rock formation (Laser GT).

produced with M_{large} as "ground truth" for assessing the quality parameters. We will further refer to this ground truth as "22 MP" as we use the most connected 22 matching partners.

Using these two kinds of ground truth, we evaluate the three quality parameters in the following manner. First, we detect outliers with respect to the ground truth and the desired accuracy. I.e. we classify all depth estimates which are more than 3 times the desired accuracy (i.e. $> 3 \cdot 1$cm) as outliers. Note that for the traditional ground truth, we additionally exclude occlusions that are more than 24cm from the rock surface from this evaluation. Then we evaluate the completeness and RMSE with respect to the ground truth using only the valid measurements (without outliers and occluded values).

From the large set of possible approaches for matching partner selection (see Section 2), we select the most relevant approaches for the given algorithm and application of photogrammetric reconstruction. Thus, we evaluate our approach against two photogrammetric standard approaches, which are both implemented by SURE (Rothermel et al., 2012), i.e. the k closest images (MinDist), the k most connected images (MaxCon). Further, we select k random images as proposed by (Galliani et al., 2015) (Rand) and k images with the carefully hand-crafted approach of (Bailer et al., 2012). Finally, we run our approach once with confidence prediction (Ours) and once without prediction (OursNoCP), i.e. $f_{conf} = 1$. For each key frame, all approaches have the task of selecting $k = 2, 3, 5, 11$ out of the 22 most-connected matching partners.

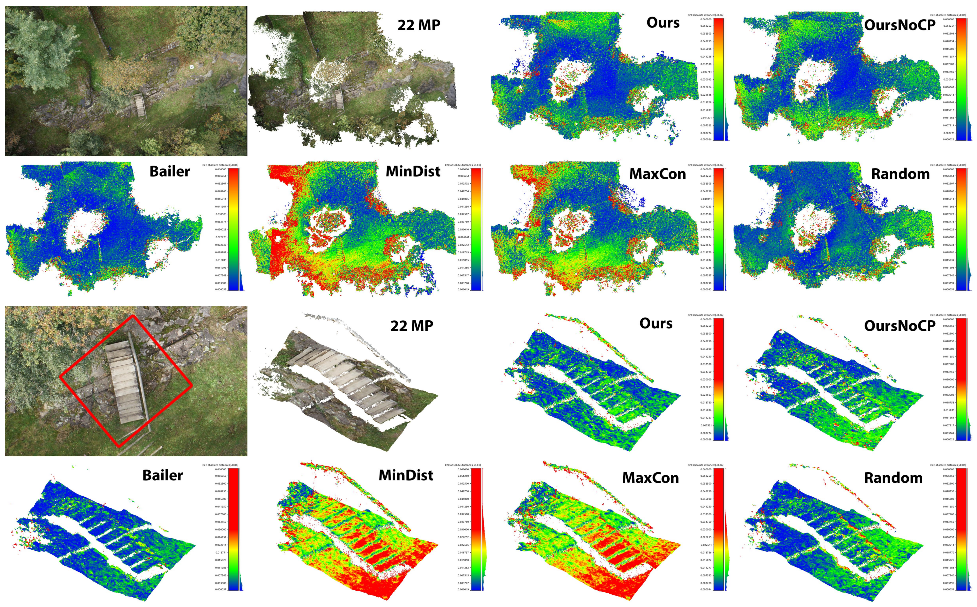


Figure 5: Impact of matching partner selection (selecting 5 out of 22) on the **Valley Dataset**. All shown reconstructions are computed with the same image as key view (top left). The top two row show the whole point clouds where the color encodes the distance to the reference (22MP); from no error (blue) to *6cm* error (red). The bottom row shows a cutout of the same reconstruction (here red indicates an error larger than *3cm*). Notice that our approach leads to a reconstruction with low error that still preserves fine details such as the hand rail. Other approaches have either a low error but also low completeness (Bailer) or have a high completeness but also a high error (MinDist and MaxCon).

For computational reasons, we do not computed all possible combination on all images of the datasets, but use a representative subset. For the *22 MP* evaluation, we will use 100 randomly selected key views for evaluation. For the *Laser GT* evaluation (of Rock 12C), we select the 50 views (of the 100) in which Seradina 12C is most prominently visible (i.e. covers the largest area in pixel).

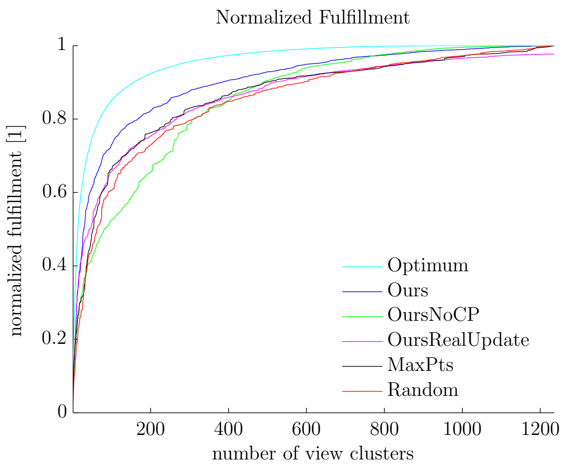
Results. In Figure 4, we show the averaged results in completeness, the root mean square error (RMSE) and the ratio of outliers.

Now let us first compare the results of *Valley Laser GT* and *Valley 22 MP* to establish the relation between traditional ground truth and the ground truth with a large number of matching partners. In this regard, the number of outliers seems to be most important. If we compare sub-figures (g) and (h), we can observe that the relative ordering between the approaches stays the same for 2,3 and 5 matching partners. From this we conclude that our approximative ground truth seems to work reasonably up to 5 matching partners.

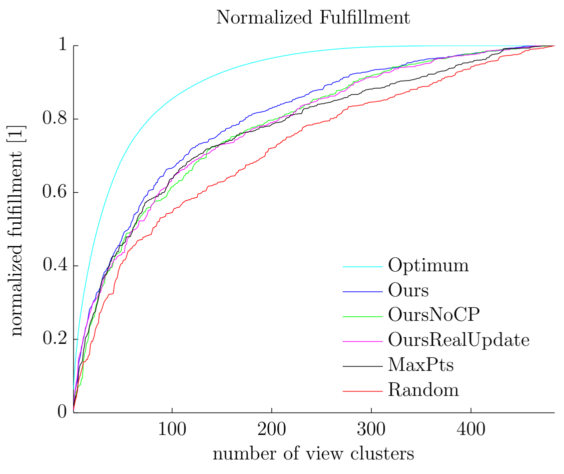
Now let us compare the individual approaches on the Valley Dataset. As expected, for all approaches more matching partners lead to higher completeness. Similarly, more matching partners lead to a lower outlier ratio for most approaches. Only for Bailer, the number of outliers seems to increase (we think this is due to the very low completeness). For all approaches, the RMSE and the outlier ratio seem to be highly correlated.

In general, we can observe a tradeoff between completeness and accuracy (outliers and RMSE). None of the evaluated approaches leads in both completeness and accuracy. Some approaches are rather tuned for completeness (MaxCon and Ours), while others more for accuracy (Bailer and OursNoCP). In contrast to only flat surfaces (Laser GT), we can see a clear gap between the two versions of our approach on the complete scene (22 MP). Further, we can see that our approach seems to be tuned very much towards completeness (very close to MaxCon), but at the same time shows a significantly lower outlier ratio than MaxCon (especially for a low number of matching partners).

If we now take a look at the results of the House Dataset, the relative ordering between MaxCon, Ours and OursNoCP is very similar. Ours stays between the other two approaches (for completeness and accuracy), but exhibits a much higher completeness than OursNoCP.



(a) Valley



(b) House

Figure 6: Ranking Performance on the **Valley** and **House** Datasets. We show the normalized fulfillment over the number of used view clusters.

Overall we conclude that our approach with learning leads to a significantly higher completeness than our approach without learning and at the same time keeps the outlier ratio significantly lower than approaches of similar completeness (especially for a low number of matching partners). This can also be observed in the example shown in Figure 5.

4.3. Ranking Only

In the previous experiment, we fixed the set of key views and evaluated the matching partner selection. In this experiment, we fix the matching partner selection and evaluate the cluster ranking performance. For each key view, we run our full approach with 5 matching partners. This leads to one view cluster per key view.

The task in this experiment is to maximize the *real* fulfillment with as few view clusters as possible. We evaluate the real fulfillment analogue to Equation 6, with the difference that all estimated and predicted values are replaced by measurements as follows. For evaluating the coverage, we use the projected depth map together with the set of successful matching partners reported by SURE. For judging whether a measurement represents a valid measurement of a triangle we use the same criterion as in the last experiment. For evaluating the theoretical uncertainty (Equation 2), we use the set of reported matching partners by SURE and check for self occlusions for each triangle. Finally, we replace f_{conf} with the actual coverage of the triangle. This means for a given triangle, we project this triangle into the key view and analyze the percentage of pixels with a valid measurement within the 2D projection of the triangle.

In this experiment, we evaluate our approach with prediction (Ours) against two variants of our approach. The first variant (OursNoCP) does not use the confidence prediction (i.e. f_{conf} is fixed to 1). The second variant (OursRealUpdate) uses the confidence prediction, but instead of also predicting the update of the objective function, it uses the real fulfillment for the update. This variant can be seen as our approach with the MVS approach in the loop.

Additionally to random ranking (Random), we also tried to come up with a good baseline algorithm that only operates on the sparse point cloud. We denote this algorithm as "MaxPts". This algorithm always greedily selects the view cluster of the camera with the highest number of connected 3D points. After selecting the camera with the maximum number of points, all these points are removed from the sparse reconstruction. Through this removal operation, the algorithm naturally tries to explore the reconstruction. Finally, we also compare our approach to the optimal greedy solution (Optimum). This algorithm requires all reconstruction to be available in every iteration and uses the real objective function for all its decisions.

Fulfillment	Opt	Ours	OursNoCP	OursRU	MaxPts	Random
Threshold	Required View Clusters [%]					
10%	0.08	**0.24**	**0.24**	**0.24**	**0.24**	0.32
20%	0.24	0.57	**0.49**	0.57	0.73	1.05
30%	0.49	0.97	1.46	**0.89**	1.62	2.18
40%	0.73	**1.46**	3.07	1.62	2.83	2.99
50%	1.13	**2.43**	6.63	3.72	4.21	5.02
60%	1.78	**4.37**	12.78	5.99	6.07	7.36
70%	3.16	**7.12**	19.17	10.92	10.60	13.03
80%	5.66	**14.00**	25.16	21.60	20.71	25.16
90%	12.70	**30.91**	39.56	42.07	39.97	47.98

Table 2: View Cluster Ranking on the Valley Dataset. We show the percentage of view clusters that are required for reaching a certain percentage of the maximum achievable fulfillment. Note that our approach only require two times the optimal number of view clusters, whereas the second best approach requires between 30-90% more than ours.

Results. In Figure 6, we show the normalized fulfillment for all approaches, such that the highest fulfillment value is set to one. In Table 2, we also show the percentage of view clusters, which are needed to reach a certain fulfillment level in steps of 10%.

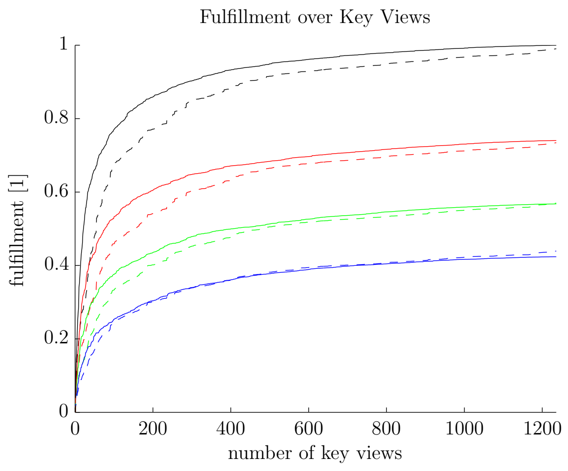
If we analyze the results of the Valley Dataset, we can see the following. First of all, our full approach performs very well. Between 30% and 70% (which is one of the most interesting regions for our task), the second best approach requires approximately 70% more view clusters than our approach. What seems really fascinating at first glance is that in this region our approach with prediction (Ours) actual performs better than with the MVS algorithm in the loop (OursRealUpdate). The reason for this astonishing result is that, with the real update, the algorithm does not have any notion of what it has tried in the past. This means, if for some reason the prediction says that there is a chance to reconstruct this object from this view point and the reconstruction actually fails, the algorithm will try the same thing with the view directly next to the last one. However, with the prediction such a behavior is naturally avoided. If we completely remove the prediction from our approach (OursNoCP), the performance degrades very soon below the baseline method. If we take a closer look at Table 2, we can see that the first 60% of fulfillment are quite cheap with our approach. For 60% we only require 4.4% of all poses (i.e. 45 view clusters). However, each further added 10% of fulfillment roughly doubles the required number of view clusters.

If we analyze the results of the House Dataset (Figure 6), we can see that our approach still performs better than the baseline approaches from 40% upwards.

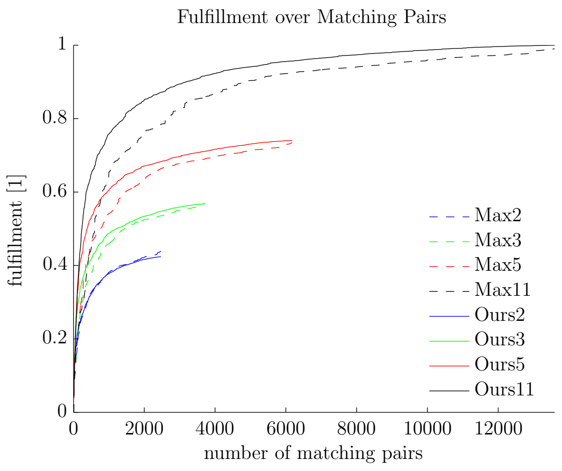
4.4. Full View Cluster Ranking

In the previous two experiments, we analyzed each of our two steps separately. In this experiment, we evaluate our full approach against the best combination of the baseline methods. We denote this approach as "Max", as it consists of the matching partner selection based on the maximum connectivity (MaxCon) and the ranking procedure based on the maximum number of visible 3D points (MaxPts). For both approaches, we evaluate the performance for different numbers of matching partners, i.e. $k = 2, 3, 5, 11$.

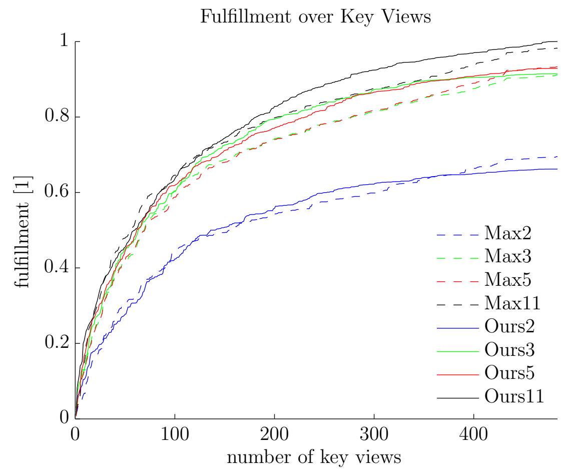
Results. In Figure 7, we show the fulfillment curves for both datasets. Note that the curves are normalized such that maximally achieved fulfillment over all approaches (i.e. Ours11 with all key views) is set to one. On a first glance, we can see that there is a significant difference in the datasets. For the Valley Dataset, there is a significant difference in the maximally reached fulfillment for different numbers of matching partners. E.g. our approach with 3 matching partners is only able to reach half of the fulfillment of the approach with the maximum number of matching partners. For the House Dataset this gap is significantly smaller and our approach with 3 matching partners is able to reach 90% of the fulfillment. We think that the main reason for this discrepancy is the difficulty of the dataset. While the Valley Dataset is strongly dominated by trees



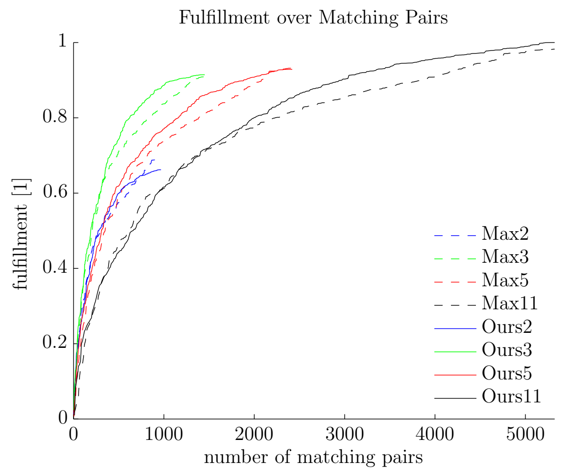
(a) Valley



(b) Valley

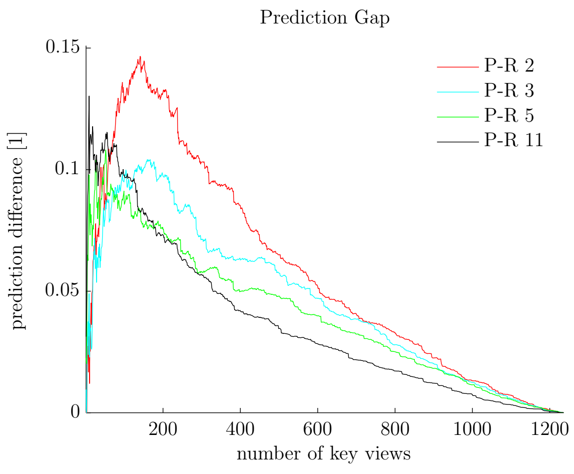


(c) House

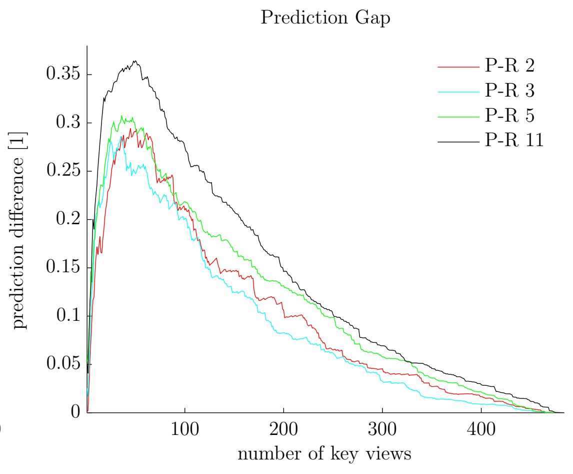


(d) House

Figure 7: Fulfillment development of our full approach on the **Valley Dataset** (top) and **House Dataset** (bottom). We show our full approach (Ours) and the baseline method (Max) for four different numbers of matching partners (i.e. $k = 2, 3, 5, 11$). The left side shows the fulfillment over the number of key views, whereas the right side shows the fulfillment over the number of matching pairs.



(a) Valley



(b) House

Figure 8: We show the prediction gap as the difference of the normalized fulfillment (i.e. Predicted minus Real) for the **Valley** and **House** Dataset. Note that the prediction gap on the Valley dataset is not very wide (especially for 5 and 11 matching partners).

which are exceedingly hard to reconstruct, the House Dataset contains mostly flat structures such as roads, short grass or roofs.

For the Valley Dataset, our approach significantly outperforms right from the start. In Table 3, we see that between 30% and 70% fulfillment the baseline approach with 11 matching partners requires 2 to 3 times as many matching pairs than our approach for obtaining the same fulfillment level.

If we take a look at the House Dataset, we can also see a clear gap between our approach and the baseline. However, this clear gap only starts to form between 60% and 70% fulfillment. What seems interesting is that in terms of computational efficiency, three matching partners seem to be the best choice for the House Dataset.

If we take a look at the prediction performance (Figure 8), we see a significant difference between the training environment (Valley) and the unseen test environment (House). While there is a significant gap between predicted fulfillment and real fulfillment on the House Dataset, the gap is a lot smaller on the Valley Dataset. In fact, the gap closes below 10% after only 55 key views for 5 matching partners (87 for 11). This means in a known environment, the predicted fulfillment can indeed be used for estimating the actual fulfillment before executing the actual MVS algorithm.

4.5. Discussion

In all our experiments, we have shown that the MVS confidence prediction allows us to consistently outperform the exact same approach without prediction. We see this as proof that our prioritization approach was able to use the accumulated knowledge of the confidence predictor to significantly improve the performance. In this sense, we think that our approach could be a great benefit to reoccurring photogrammetric tasks in difficult environments.

On the Valley Dataset (i.e. the dataset which was captured in the same domain as the training data), we were able to achieve significantly better results for the view cluster prioritization than any other baseline. For the same level of fulfillment our approach requires 2 to 3 times less key views than the best baseline. Further, we demonstrated that the real fulfillment only lacks approximately 10% behind the predicted fulfillment. This means that the fulfillment prediction is a reasonable approach for approximating the necessary number of key views to get a good coverage of the scene. E.g. if we aim for 70% fulfillment with 11 matching partners, we can use our approach to predict a fulfillment of 80%. In this experiment, this method would lead to 69.3% fulfillment with only 60 key views (which is already highly complete – see video). If we analyze

	Max				Ours			
	2	3	5	11	2	3	5	11
Fulfillment	Required Matching Pairs [%]							
10%	0.29	0.26	0.22	0.24	**0.15**	**0.15**	0.18	0.16
20%	0.94	0.82	0.92	0.89	0.71	**0.35**	0.40	0.49
30%	2.88	1.83	1.54	2.18	2.77	1.04	**0.70**	0.81
40%	9.91	4.30	2.57	2.99	10.61	2.91	1.29	**1.13**
50%	-	10.55	5.63	4.45	-	8.94	2.83	**1.78**
60%	-	-	10.70	5.99	-	-	7.02	**2.67**
70%	-	-	30.78	10.03	-	-	22.76	**5.02**
80%	-	-	-	19.26	-	-	-	**10.52**
90%	-	-	-	36.25	-	-	-	**23.46**

Table 3: Required number of matching pairs to reach a certain level of fulfillment on the **Valley Dataset** . We compare our full approach (Ours) to the best baseline combination (Max) for a varying number of matching partners. The best values of each row are marked in bold font.

the computation time for this example in relation to using each view as key view (as done by conventional MVS), we end up with a speed up factor of 9.2 (from $n/(s + n \cdot r)$ with number selected of key views n, number selected of key views s and the ratio r of prioritization time over MVS time). If we are only interested in the saved memory, we even achieve a memory reduction factor of 20.1 (430M points vs. 8650M points). This means that our approach has a large potential to save computational time and memory with a very small overhead.

Our experiments on the House dataset demonstrated that our approach still performs reasonable even if confronted with a scene that contains mostly objects that were never seen in training. Our approach still has some benefit over the baseline methods for a high degree of fulfillment, however, the margin is significantly less than for the Valley dataset. On the House dataset, we reach 70% of the achievable fulfillment with 7.6% of the available matching pairs, while the corresponding baseline requires 9.1%. However, if we compare our the actual run-time to the best baseline, we see that our method does not lead to a run-time improvement for the same fulfillment level due to the higher computational overhead. In this sense, we can conclude that our method will not break down in an unknown environment, but the main benefit over non-learning approaches will be significantly diminished.

5. Conclusion

In this work, we presented a new approach to prioritize view clusters (key views with matching partners) for obtaining a complete and accurate 3D reconstruction at a low computational time and memory consumption. The proposed approach uses unsupervised machine learning for accumulating empirical evidence under what circumstances an MVS algorithm does not work as intended. In the prioritization, we use this accumulated knowledge to prefer constellations with a high probability of success and thus maximize the reconstruction completeness. In our experiments, we demonstrate that our approach is thus able to reach a certain level of quality fulfillment (in terms of completeness with respect to a desired accuracy and ground resolution) with up to three times less key views than the best baseline approach. In contrast to other approaches, our approach has the distinct advantage that it provides the opportunity to predict the relative fulfillment level in complex scenes. This means that the user can decide whether doubling the amount of computation time and memory consumption for reaching 80% fulfillment instead of 70% (which is hardly visible – see video) is necessary or not. On the topic of domain generalization, we have shown that our approach still performs reasonably well in an environment that has never been seen in training, however, the advantage over non-learning based approaches is significantly diminished. In this sense, we see the main application area of our approach in reoccurring photogrammetric tasks in challenging environments.

Acknowledgements

Results incorporated in this paper received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 730294 and the EC FP7 project 3D-PITOTI (ICT-2011-600545).

References

Aanæs, H., Jensen, R. R., Vogiatzis, G., Tola, E., Dahl, A. B., 2016. Large-scale data for multiple-view stereopsis. International Journal of Computer Vision, 1–16.

Ahmadabadian, A., Robson, S., Boehm, J., Shortis, M., 2014a. Stereo-imaging network design for precise and dense 3D reconstruction. The Photogrammetric Record 29 (147), 317–336.
URL `http://dx.doi.org/10.1111/phor.12076`

Ahmadabadian, A. H., Sargeant, B., Erfani, T., Robson, S., Shortis, M., Hess, M., Boehm, J., 2014b. Towards fully automatic reliable 3d acquisition: From designing imaging network to a complete and accurate point cloud. Robotics and Autonomous Systems 62 (8), 1197 – 1207.
URL `http://www.sciencedirect.com/science/article/pii/S0921889014000670`

Alexis, K., Papachristos, C., Siegwart, R., Tzes, A., Sept 2015. Uniform coverage structural inspection path-planning for micro aerial vehicles. In: Intelligent Control (ISIC), 2015 IEEE International Symposium on. pp. 59–64.

Almadhoun, R., Taha, T., Seneviratne, L., Dias, J., Cai, G., 2016. A survey on inspecting structures using robotic systems. International Journal of Advanced Robotic Systems 13 (6), 1729881416663664.
URL `http://dx.doi.org/10.1177/1729881416663664`

Bailer, C., Finckh, M., Lensch, H. P. A., 2012. Scale robust multi view stereo. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (Eds.), Computer Vision – ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part III. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 398–411.
URL `http://dx.doi.org/10.1007/978-3-642-33712-3_29`

Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., Siegwart, R., 2016. Receding horizon path planning for 3d exploration and surface inspection. Autonomous Robots, 1–16.
URL `http://dx.doi.org/10.1007/s10514-016-9610-0`

Dornhege, C., Kleiner, A., Rolling, A., Oct 2013. Coverage search in 3D. In: Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on. pp. 1–8.

Fan, X., Zhang, L., Brown, B., Rusinkiewicz, S., Nov. 2016. Automated view and path planning for scalable multi-object 3d scanning. ACM Trans. Graph. 35 (6), 239:1–239:13.
URL `http://doi.acm.org/10.1145/2980179.2980225`

Feige, U., Jul. 1998. A threshold of ln n for approximating set cover. J. ACM 45 (4), 634–652.
URL `http://doi.acm.org/10.1145/285055.285059`

Furukawa, Y., Curless, B., Seitz, S. M., Szeliski, R., June 2010. Towards internet-scale multi-view stereo. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 1434–1441.

Galceran, E., Campos, R., Palomeras, N., Ribas, D., Carreras, M., Ridao, P., 2015. Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. Journal of Field Robotics 32 (7), 952–983.
URL `http://dx.doi.org/10.1002/rob.21554`

Galceran, E., Carreras, M., 2013. A survey on coverage path planning for robotics. Robotics and Autonomous Systems 61 (12), 1258 – 1276.
URL `http://www.sciencedirect.com/science/article/pii/S092188901300167X`

Galliani, S., Lasinger, K., Schindler, K., 2015. Massively parallel multiview stereopsis by surface normal diffusion. In: International Conference on Computer Vision (ICCV). pp. 873–881.

Garland, M., Heckbert, P. S., 1997. Surface simplification using quadric error metrics. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '97. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 209–216.
URL `http://dx.doi.org/10.1145/258734.258849`

Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S. M., Oct 2007. Multi-view stereo for community photo collections. In: International Conference on Computer Vision (ICCV). pp. 1–8.

Hartley, R., Zisserman, A., Apr. 2004. Multiple View Geometry in Computer Vision, 2nd Edition. Cambridge University Press.

Hollinger, G. A., Englot, B., Hover, F., Mitra, U., Sukhatme, G. S., 2012. Uncertainty-Driven View Planning for Underwater Inspection. In: IEEE International Conference on Robotics and Automation (ICRA).

Hoppe, C., Wendel, A., Zollmann, S., Pirker, K., Irschara, A., Bischof, H., Kluckner, S., 2012. Photogrammetric camera network design for micro aerial vehicles. In: Proceedings of the Computer Vison Winterworkshop. Mala Nedelja, Slovenia.

Hornung, A., Zeng, B., Kobbelt, L., June 2008. Image selection for improved multi-view stereo. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8.

Jing, W., Polden, J., Lin, W., Shimada, K., Oct 2016a. Sampling-based view planning for 3d visual coverage task with unmanned aerial vehicle. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1808–1815.

Jing, W., Polden, J., Tao, P. Y., Lin, W., Shimada, K., Nov 2016b. View planning for 3d shape reconstruction of buildings with unmanned aerial vehicles. In: 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV). pp. 1–6.

Karaszewski, M., Adamczyk, M., Sitnik, R., 2016. Assessment of next-best-view algorithms performance with various 3d scanners and manipulator. {ISPRS} Journal of Photogrammetry and Remote Sensing 119, 320 – 333.
URL http://www.sciencedirect.com/science/article/pii/S0924271616301307

Labatut, P., Pons, J.-P., Keriven, R., Oct 2007. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In: International Conference on Computer Vision (ICCV). pp. 1–8.

Martin, R. A., Rojas, I., Franke, K., Hedengren, J. D., 2016. Evolutionary view planning for optimized uav terrain modeling in a simulated environment. Remote Sensing 8 (1), 26.
URL http://www.mdpi.com/2072-4292/8/1/26

Mendez, O., Hadfield, S., Pugeault, N., Bowden, R., 2016. Next-best stereo: extending next best view optimisation for collaborative sensors. In: British Machine Vision Conference (BMVC).

Mostegel, C., Prettenthaler, R., Fraundorfer, F., Bischof, H., 2017. Scalable surface reconstruction from point clouds with extreme scale and density diversity. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Mostegel, C., Rumpler, M., Fraundorfer, F., Bischof, H., June 2016a. Uav-based autonomous image acquisition with multi-view stereo quality assurance by confidence prediction. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

Mostegel, C., Rumpler, M., Fraundorfer, F., Bischof, H., 2016b. Using self-contradiction to learn confidence measures in stereo vision. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Nemhauser, G. L., Wolsey, L. A., Fisher, M. L., 1978. An analysis of approximations for maximizing submodular set functionsi. Mathematical Programming 14 (1), 265–294.

O'rourke, J., 1987. Art gallery theorems and algorithms. Vol. 57. Oxford University Press Oxford.

Rothermel, M., Wenzel, K., Fritsch, D., Haala, N., 2012. SURE: Photogrammetric Surface Reconstruction from Imagery. In: Proceedings LC3D Workshop.

Rumpler, M., Tscharf, A., Mostegel, C., Daftry, S., Hoppe, C., Prettenthaler, R., Fraundorfer, F., Mayer, G., Bischof, H., 2017. Evaluations on multi-scale camera networks for precise and geo-accurate reconstructions from aerial and terrestrial images with user guidance. Computer Vision and Image Understanding 157, 255 – 273, large-Scale 3D Modeling of Urban Indoor or Outdoor Scenes from Images and Range Scans.
URL http://www.sciencedirect.com/science/article/pii/S1077314216300327

Schmid, K., Hirschmüller, H., Dömel, A., Grixa, I., Suppa, M., Hirzinger, G., 2012. View Planning for Multi-View Stereo 3D Reconstruction Using an Autonomous Multicopter. Journal of Intelligent and Robotic Systems 65, 309–323.

Schönberger, J. L., Zheng, E., Frahm, J.-M., Pollefeys, M., 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. Springer International Publishing, Cham, pp. 501–518.
URL http://dx.doi.org/10.1007/978-3-319-46487-9_31

Shen, S., May 2013. Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. IEEE Transactions on Image Processing 22 (5), 1901–1914.

Trummer, M., Munkelt, C., Denzler, J., 2010. Online Next-Best-View Planning for Accuracy Optimization Using an Extended E-Criterion. In: International Conference on Pattern Recognition (ICPR). pp. 1642–1645.

Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C., Dec 2013. Deepflow: Large displacement optical flow with deep matching. In: 2013 IEEE International Conference on Computer Vision. pp. 1385–1392.

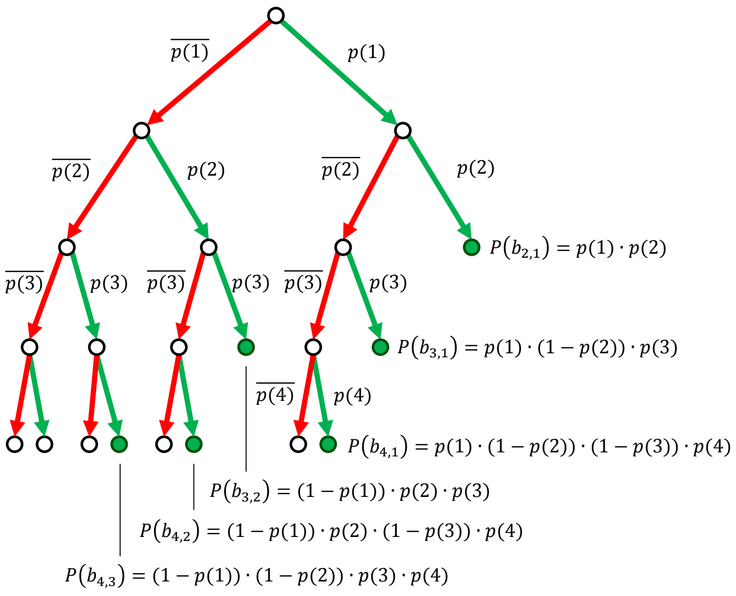
Zheng, E., Dunn, E., Jojic, V., Frahm, J. M., June 2014. Patchmatch based joint view selection and depthmap estimation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1510–1517.

Appendix A. Probabilistic Confidence Extension

For extending the confidence prediction framework to an arbitrary number of matching partners, we use a traditional probability tree. In Figure A.9 we show such a probability tree for 4 matching partners. In the following, we use such a general probability tree to derive a probabilistic formulation for obtaining a successful 3D measurement from k cameras.

Let us first formalize the probability tree for k cameras as a binary tree with a depth k. Every depth level d represent more or less what happens if we add a d^{th} camera to the previous set of $d-1$ cameras. Every vertex in the tree represents an event, which is successful with a probability $p(d)$, where d is the depth of the tree (also corresponding to the d^{th} matching partner). Let us call a path from the root vertex to a leaf vertex (i.e. a vertex without child vertices) simply "branch".

Now let us grow the binary tree. As we are only interested in having at least 2 successful matches, we can stop growing a branch when it has reached two positive events (we will further call this kind of branch as "successful branch"). As such a successful branch has always exactly two successful events, we will further denote a successful branch as $b_{y,x}$, where y is the depth/id of the last successful event and x is the depth/id of the preceding successful event. This means that only branches with less than 2 successful events are



$\overline{p(1)}$
$p(1)$
$\overline{p(2)}$
$p(2)$
$\overline{p(2)}$
$p(2)$
$P(b_{2,1}) = p(1) \cdot p(2)$
$\overline{p(3)}$
$p(3)$
$\overline{p(3)}$
$p(3)$
$\overline{p(3)}$
$p(3)$
$P(b_{3,1}) = p(1) \cdot (1 - p(2)) \cdot p(3)$
$\overline{p(4)}$
$p(4)$
$P(b_{4,1}) = p(1) \cdot (1 - p(2)) \cdot (1 - p(3)) \cdot p(4)$
$P(b_{3,2}) = (1 - p(1)) \cdot p(2) \cdot p(3)$
$P(b_{4,2}) = (1 - p(1)) \cdot p(2) \cdot (1 - p(3)) \cdot p(4)$
$P(b_{4,3}) = (1 - p(1)) \cdot (1 - p(2)) \cdot p(3) \cdot p(4)$

Figure A.9: Probability tree. We formulate the MVS confidence for a key view with k cameras as matching partners as a stochastic process (here $k = 4$). Each edge in the tree relates to an event. For a matching partner d, the event can either be positive (green color) with a probability $p(d)$ or negative (red color) with a probability $\overline{p(d)} = 1 - p(d)$. A positive event corresponds to a successful match between the matching partner d and the key view at a specific pixel location, which results in a 3D measurement that conforms with our uncertainty model. For obtaining a reliable 3D reconstruction, we require at least two matching partners with a successful match. Thus, each path in the tree with at least two positive events can be seen a successful branch. Each successful branch $b_{y,x}$ is defined by the first two positive events at level x and y respectively. The overall probability of a successful measurement can be obtained by summing all successful branches. Note that the order events in the tree is irrelevant for the overall probability, which can be better seen in Equation A.4.

grown in the next depth level. It follows that in the process of growing from level $d-1$ to d (i.e. adding a d^{th} camera), the tree gains exactly $d-1$ new successful branches (one for each camera already in the tree), while all old successful branches remain unchanged. It also follows that each new successful branch $(b_{d,i})$ has exactly one previous camera i (with $1 \geq i < d$) with a successful event associated. Thus the probability along such a new successful branch $b_{d,i}$ is given by:

$$P(b_{d,i}) = p(d) \cdot p(i) \prod_{j \in \{1,2,\ldots,d-1\} \setminus i} (1 - p(j)), \tag{A.1}$$

where $p(x)$ is the probability of a successful match with camera x. If we now add up all branches with 2 successful events at level d, we end up with the following probability for obtaining at least 2 successful events:

$$P_{min2} = \sum_{i=2}^{d} \sum_{a=1}^{i-1} P(b_{i,a}) = \sum_{i=2}^{d} \sum_{a=1}^{i-1} p(i) \cdot p(a) \prod_{j \in \{1,2,\ldots,i-1\} \setminus a} (1 - p(j)) \tag{A.2}$$

Now we can expand the equation to

$$P_{min2} = \sum_{i=2}^{d} p(i) \cdot \left(\left(\sum_{c=1}^{i-1} p(c) \right) - 2 \left(\sum_{c^{(2)} \in C_{i-1}^{(2)}} p\left(c_1^{(2)}\right) \cdot p\left(c_2^{(2)}\right) \right) \right.$$
$$\left. +3 \left(\sum_{c^{(3)} \in C_{i-1}^{(3)}} \prod_{a=1}^{3} p\left(c_a^{(3)}\right) \right) - ... + (-1)^i (i-1) \left(\sum_{c^{(i-1)} \in C_{i-1}^{(i-1)}} \prod_{a=1}^{i-1} p\left(c_a^{(i-1)}\right) \right) \right), \tag{A.3}$$

where $C_{i-1}^{(x)}$ is the solution space for drawing subsets of x cameras from the available set of $i-1$ cameras, $c^{(x)}$ is one of these subsets and $c_a^{(x)}$ is one camera of this subset. Now we can contract the equation again to

$$P_{min2} = \sum_{i=2}^{d} \left((-1)^i \cdot (i-1) \cdot \sum_{c^{(i)} \in C_d^{(i)}} \left(\prod_{a=1}^{i} p\left(c_a^{(i)}\right) \right) \right) \tag{A.4}$$

Note that this equation and Equation 3 are equivalent and that only the parameterization was changed; i.e. $d \mapsto k$, $P_{min2} \mapsto f_{conf}(t, c_{key}, c^k)$ and $p\left(c_a^{(i)}\right) \mapsto f_{conf}\left(c_{key}, c_a^{(i)}\right)$.