A Graph Cut based Adaptive Structured Light approach for real-time Range Acquisition

Thomas P. Koninckx[1] Indra Geys[1] Tobias Jaeggli[1,2] Luc Van Gool[1,2]

[1]Katholieke Universiteit Leuven,
ESAT / VISICS,
Leuven, Belgium

[2]Swiss Federal Institute of Technology (ETH)
D-ITET / BIWI,
Zürich, Switzerland

{tkoninck, igeys, tjaeggli, vangool}@esat.kuleuven.ac.be vangool@vision.ee.ethz.ch

Abstract

This paper describes a new algorithm that yields dense range maps in real-time. Reconstructions are based on a single frame structured light illumination. On-the-fly adaptation of the projection pattern renders the system more robust against scene variability.
A continuous trade off between speed and quality is made. The correspondence problem is solved by using geometric pattern coding in combination with sparse color coding. Only local spatial and temporal continuity are assumed. This allows to construct a neighbor relationship within every frame and to track correspondences over time. All cues are integrated in one consistent labeling. This is achieved by reformulating the problem as a graph cut. Every cue is weighted based on its average consistency with the result within a small time window. Integration and weighting of additional cues is straightforward.
The correctness of the range maps is not guaranteed, but an estimation of the uncertainty is provided for each part of the reconstruction. Our prototype is implemented using unmodified consumer hardware only. Frame rates vary between 10 and 25 fps dependent on scene complexity.

1. Introduction

Structured light range acquisition is becoming a mature technique for the generation of accurate 3D data. For an overview see eg. [1]. 3D digitizers based on time coded pattern projection are widespread in industry, research and cultural heritage. By using inexpensive consumer grade projection and video devices, they can offer a cost efficient and versatile alternative to many other 3D digitizers. The acquisition process can be speeded up by limiting the number of necessary input images (eg. by the use of color, as done in the work of Caspi et al. [2]).
The extreme case of range acquisition based on a single input frame has the advantage that it can capture moving objects [3, 4, 5]. In order to still solve for the correspondence between projected pattern elements and their image projections, the pattern elements should be coded using only this single input image. Most researchers use a spatial neighborhood which yields a unique identifier for the relative position in the pattern. Examples are the use of colors [6], or a set of markers with a unique shape or outline [7], or a combination of these strategies [8].
Most techniques still rely on offline computations. An exception is the work of Hall-Holt et al. [9]. In this work time coded structured light together with tracking of the pattern allows for real time range acquisition, of slowly moving objects.
We present an inexpensive system based on consumer hardware for single frame range acquisition of moving objects in real time. The reconstruction is based on optical triangulation. The online availability of the result allows the system to adapt the projection patterns for every frame to the noise levels. This is a bit similar to the offline work of Caspi et al. [2]. Related to this Horn et al. [10] proposed a technique to design an optimal sequential structured light pattern of length K. The main difference with our work is that adaptations no longer happen offline but online, and that 3D is acquired from a single projection rather than a time series.
Indeed, we will continually and as part of the online process, optimize a series of parameters for subsequent frames. This optimization will depend on the scene in order to keep the decoding of the pattern well conditioned. A weak assumption of temporal continuity underlies this adaptation. The higher the frame rate, the more valid this assumption becomes. In contrast to most other approaches we won't use very dense coding where every pattern element would have a unique identifier. For single frame acquisition this almost implies the use of color coding, which is fragile in colored scenes. On the other hand, very sparse coding together with a relative labeling approach which relies on global smoothness and strict monoticity [11] can not robustly deal with occlusions. Instead of relying on a single 'coding cue', we exploit *a weighted combination of different such cus*.

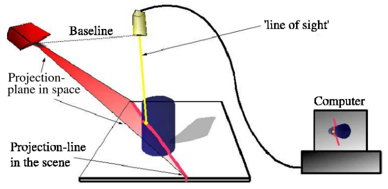

Figure 1: *Basic optical triangulation: after pattern recognition the projection planes delimited by the stripe boundaries and the 'projector center' are intersected with the corresponding lines of sight.*

These multiple sparse encodings result in a rather dense, overall encoding. This is combined with relative labeling. Via a graph cut algorithm all the above leads to optimal correspondences, together with local confidence measures. In contrast to the work by Zhang et al. [6], where the pattern is decoded line by line, our strategy directly goes after its 2D interconnections. Mutual influences between lines and the omission of image rectification also render the algorithm considerably faster and less sensitive to noise.
The rest of the paper is organized as follows. Section 2 describes the labeling problem. In section 3 the graph-cut solution for the integration of coding cues is outlined. Section 4 describes how parameters are adapted. Some results are shown in section 5, and section 6 concludes the paper.

2 A labeling problem

Range computations are based on optical ray-plane triangulation (see figure 1). The basic pattern we use for triangulation is a set of white parallel stripes on a black background. Every stripe boundary delimits a planar surface in space, a so-called 'triangulation plane'. In what follows we will refer to such a stripe boundary simply as a 'stripe' S. The reflection on the object of this projection pattern is identified in the camera image. Given a recognized stripe and the knowledge of the plane it originates from, the intersection with the line of sight for a pixel yields the corresponding 3D position. The purpose of the projection pattern in a single frame acquisition process is twofold. First it should be easy to detect precisely and as such support triangulation. Secondly it should make it possible to establish correspondences between the camera image and the projector pattern in order to distinguish the different triangulation planes. Here we focus on the latter issue, the former is described elsewhere [13].The black and white stripe pattern yields a maximal contrast ratio, and by adapting stripe width and pattern orientation (cfr. inf.) its detectability can be optimized. The correspondence problem boils down to determining a labeling at every time t:

$$f^t : S_j^t \rightarrow L^t \; with \; L^t = [0, n_{max}], \; j \in \{0, m\} \qquad (1)$$

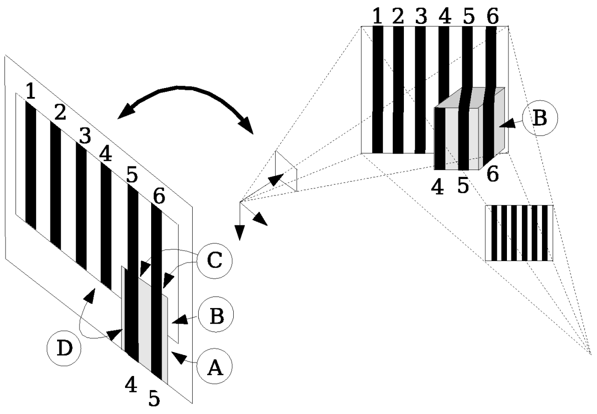

Figure 2: *Complications in labeling based pattern projection techniques: [A] 'false labels' are introduced at the cube's edge, [B] not all of the pattern is visible, [C] pieces of the pattern are wrongly interconnected, [D] monoticity is not preserved.*

in which L is the set of all possible labels n for the current projection pattern. The correct localization of a stripe S in the image together with its label n in the pattern are needed for its 3D reconstruction. The integer j is the identifier of a stripe S within the set of m detected stripes in the image. Discontinuities, textures, etc. make that a single triangulation plane can cause several detected stripes S. Thus, f^t is a *many to one* relationship and possibly $m \geq n_{max}$. As a consequence the complexity of the scene is reflected in the run time of the algorithm, and strictly spoken the timing of the algorithm is indeterministic. Note that the superscript t will be omitted at times, when the current frame is referred to.

Establishing the correct image-projection pattern correspondences is complicated by (see also fig. 2):

- **[A] ghost boundaries:** scene texture and geometry cause false boundaries to arise, which do not result from the pattern.

- **[B] horizontal occlusions:** a depth discontinuity parallel to the stripes of the pattern causes stripe boundaries to disappear in the image. A labeling based on a strictly incremental neighbor relationship which crosses such a discontinuity will be inconsistent.

- **[C] vertical occlusions:** a depth discontinuity which is not parallel with the pattern can let two different stripes S_i and S_j appear as a single stripe S in the image.

- **[D] monoticity is not preserved:** the ranking of the stripes in the projected pattern is not preserved in their image in the proximity of depth discontinuities

2.1 Cues for labeling

Before we explain how to combine all labeling cues into a consistent pattern labeling, we'll start with a concise overview of our coding strategies which yield these cues. We use 4 types: a geometric cue based on epipolar geometry, a color cue using colored rather than white stripes, and two cues based on the tracking of the geometric code and tracking of the stripes, resp.

Geometric coding

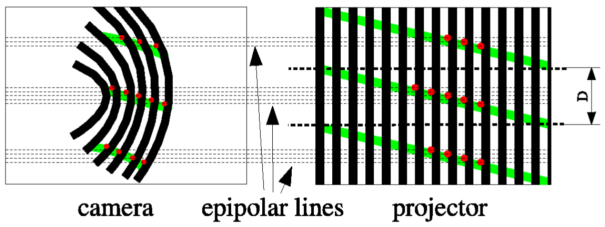

Figure 3: *Geometric coding: colored 'coding lines' underly the base stripe pattern (vertical on the right). Points of this coding line which are recognized in the camera image, are transferred to a unique intersection of a coding line and a stripe in the projection pattern. In this schematic representation the correspondences are indicated by dots.*

We submerge a colored 'coding line' in the pattern 'below' the basic stripe pattern (fig. 3), referred to in the sequel as the 'base pattern'. If such a line is not coincident with an epipolar line, the intersections of this coding line and the base pattern will all lie on different epipolar lines in the camera image. As a consequence points of such a coding line which are recognized in the camera image can be transferred to a unique point of the projection pattern by use of the epipolar geometry. This is schematically illustrated for a horizontal epipolar geometry in figure 3. The more this coding line is tilted with respect to the epipolar line the lower the noise on the transfer to the projection pattern will be. However a lower inclination allows to repeat more coding lines below eachother and thereby to increase the density of such geometric coding cues, while still respecting the constraint that a single epipolar line only should intersect a single coding line. For details see [14]. This repetition of coding lines below eachother is a first design parameter of the geometric code and will be called the vertical repetition R_v. It determines the inclination of the coding line. R_v corresponds to the number of individual coding lines encountered during a vertical transition of the pattern.
If a single epipolar line is allowed to intersect multiple coding lines of the projected pattern, code density can be increased further. This is done by moving the coding lines closer to eachother in the vertical direction. In the example of fig. 3 this situation occurs if we make the vertical distance between coding lines smaller than D. An ambiguity arises because of the multiple intersections. This can be resolved by taking a decision based on pattern tracking. The intersection closest to the one predicted by tracking of the pattern is chosen. We call the number of intersections a single epipolar can have with a coding line the horizontal repetition R_h. This is a second design parameter. We can summarize that an increase of R_v results in a *denser but noisier code*, where an increase of R_h yields a *denser code with identical noiselevel* on the decoding but *requires tracking* of the pattern. The higher R_h the less noise on the tracking can be tolerated.
We will refer to a codepoint in the image as $x_{code,i,j}$, with j the identifier of the stripe boundary on which the point resides (see expression (1)), and $i \in [0, R_v]$ an identifier for the originating coding line $l_{code,i}$. A last design parameter for the coding lines is the color C_i which is used. The decoding $x_{code,i,j} \rightarrow n_{code,i,j}$ of a codepoint to the corresponding label, yields a first set of cues. Each stripe S_j can receive zero, one or multiple such labels n_{code}.

Sparse color coding

We use the high intensity black-white and white-black transitions of the base pattern as the triangulation planes. We now assign a color instead of white illumination to a limited set of stripes. Saturated and high intensity colors let the stripe boundary still stand out clearly. This limits us to the use of yellow, magenta and cyan. Which color C_{code} will be used is decided on the fly based on the colors in the current scene. The orientation parallel to the pattern makes it easy to discriminate between this sparse color coding and the aforementioned geometric coding.
If S_j and S_k are consecutive edges with opposite gradient values we check the inter-stripe color. If above the detection threshold for C_{code} and isolated, we assign S_j and S_k labels $n_{color,j}$ and $n_{color,j} + 1 = n_{color,k}$. Each S_j has zero or one labels n_{color}. Whereas the n_{code}'s are dense but noisy, the n_{color}'s are rather sparse but mostly correct.
The detection of the colored stripes is based on a classification of the average color of the stripe in h,s,v-space. As only distinct stripes are used for color coding, an additional constraint is that such a colored stripe should be locally unique. The latter avoids the erroneous interpretation of the color of a textured object in the scene as a code. We use the same color for all coded stripes. We disambiguate between them by taking the one which is closest to the predicted position based on the tracking of stripes (see below).

Tracking of codepoints $\mathbf{x_{code}}$

After the initial detection of the code points $\mathbf{x^t_{code,i,j}}$ we run the following predictor-corrector style algorithm to keep track of these points over time:

- a predictor line $l_{predictor,i}$ is fitted to each set of code points $\mathbf{x_{i,j}^{t-1}}$ in the previous frame that belong to the same code line (i.e. regression over all intersections of one code line with the different stripes). Grouping of code points into such sets is a by-product of the stripe identification algorithm in [14]. As coding lines are not that inclined from the epipolar lines, fitting a straight line is a sufficient accurate prediction for our purposes.

- the intersections $l_{predictor,i} \cap S_j, \forall j$ yield predicted code points $\mathbf{\hat{x}_{i,j}^{t}}$ in the current frame. Their positions are refined to positions $x_{i,j}^{t}$ by convolving the image data around the position of S_j with the [5 $\times$ 7] environment of the projection pattern around the code point. This matched filter approach yields the new $\mathbf{x_{i,j}^{t}}$ where the maximum response is found. The search is confined to a local neighborhood around the predicted positions.

- for every $\mathbf{x_{i,j}^{t}}$ we search the closest match $\mathbf{x_{i,j'}^{t-1}}$ obeying the epipolar geometry between the current and the previous frame. The motion of a stripe S_j is limited given the frame-rate (between 10 and 20 fps dependent on scene complexity). This again allows the system to restrict the size of the search region. As the features which are tracked are projected, they will obey the same epipolar geometry between camera and projector at all times. This also holds if the underlying object deforms.

The relationship $\mathbf{x_{i,j}^{t-1}} \leftrightarrow \mathbf{x_{i,j}^{t}} \Rightarrow \mathbf{n_{i,j}^{t-1}} \rightarrow \mathbf{n_{track,i,j}^{t}}$ allows to transfer the labels on time $t-1$ to the tracked codepoints on time t.

Tracking of pattern stripes S

The same algorithm as for tracking the codepoints is used to track the stripes S_j. We now use the midpoint of the stripe as a feature point. The prediction step is omitted, and midpoints are checked for obeying inter-frame epipolar geometry. This assumes that stripes in consecutive frames have similar lengths or at least symmetrical growth. Continuity over time justifies this assumption. This yields a new set of labels $n_{trackS,j}^{t}$. Although noisy on average, the label will never be far from the real label. The higher the processing speed, the more valid this becomes.

Neighbor relationship

After the detection of all stripes a neighbor relationship is constructed. To do so we create an ordered set of stripes Q based on ranking the stripes by their horizontal position in the image. The leftmost stripe is the first, the rightmost the last. As only stripes which have an 'overlap' in the horizontal direction -i.e. share some heights- can be compared, top-down ordering is used as a second criterion. We can define this unique order as follows:

$$S_i \leq S_k \Rightarrow$$
$$\{\forall y : S_i(y) \leq S_k(y)\ OR\ max_y(S_i) \leq min_y(S_k)\}$$, with

$x = S(y)$ the x-offset (horizontal) of S on height y
y$\in$ [0, image height] the vert. pos. in the image
$max_{y,i} = \{max(y) : S_i\ exists\ on\ heigth\ y\}$,
$min_{y,i} = \{min(y) : S_i\ exists\ on\ heigth\ y\}$.

Degenerate cases in which $S_i(y) \leq S_k(y')$ && $S_i(y) \geq S_k(y')$ corresponding to *crossing or partially coincident* stripes are removed. The stripe with the orientation closest to the local average within a window of 5 stripes in Q around the one under consideration is retained.
$\forall S_i \in Q$ check if $\exists S_k : overlap(S_i, S_k) \neq 0$, with

```
$if(max_{y,i} \leq min_{y,k})$
    $\{overlap(S_i, S_k) = 0\}$
$else$
    $\{\forall y \in S_i\ AND\ y \in S_k :$
        $if\ (|S_k(y) - S_i(y)| \leq DIST)$
            $overlap{+}= 1$
    $\}$
```

$DIST$ = the $2-\sigma$ tolerance interval around the mean distance in x direction between consecutive stripes of Q.
The first S_k which is found when further traversing Q is always a neighbor of S_i. All consecutive stripes S_l only are a neighbor if they do overlap with S_i but do not overlap with any of the previous neighbors S_k.
The neighbor relationship is weighted by:

$$Nb(S_i, S_k) = \frac{overlap(S_i, S_k)}{yRegion(S_i, S_k)} \times \frac{(length(S_i) + length(S_k))/2}{maxLength} \quad (2)$$

$yRegion(S_i, S_k) = min(max_y(S_i), max_y(S_k)) - max(min_y(S_i), min_y(S_k))$
The first factor of expression (2) is the number of pixels which lie within the neighbor interval, relative to the maximum that would be possible. This expresses the *quality* of the overlap. The next factor renders long overlapping stripes relatively more important.
The fact that Q is ordered and that multiple neighbors of the same stripe must not overlap renders this relationship very efficient to construct. The search for neighbors can be stopped if the total overlap of all neighbors found so far, equals the length of the first stripe, or if we encounter a stripe which overlaps with one of the other neighbors.
This neighbor relationship gives us an additional cue for assigning labels:
$overlap(S_i, S_k) \neq 0 \ \Rightarrow\ n_i + 1 = n_k$

3 Optimal correspondences

Noise on the decoding and false neighbor relationships makes that the different cues for labeling will almost never be fully consistent. Increasing the quality of the coding in order to make the problem better conditioned can only be achieved at the cost of a decrease in coding density. Furthermore, relative labeling of neighbors will always remain noisy, and is a prerequisite to enable single frame range acquisition. We however do have to combine all cues in a single consistent labeling.
To achieve this we propose to search for the *weighted least squares* solution to this problem. This can be considered as an optimal approximation of all labeling cues, and as such will yield optimal camera-projector correspondences. This solution is not limited to the cues in this paper. Any additional set of cues, regardless the density or noise, can be integrated in a straightforward way.
We seek to solve the following minimization problem:

$$\begin{aligned}Min[\sum_j\{\sum_i \alpha|n_j^* - n_{code,i,j}|^2 + \beta|n_j^* - n_{color,j}|^2 + \\ \sum_i \gamma|n_j^* - n_{track,i,j}|^2 + \delta|n_j^* - n_{trackS,j}|^2 + \\ \epsilon\sum_k(Nb(S_j,S_k)\times(n_j^*+1-n_k^*)^2) \qquad \}\quad]\end{aligned} \qquad (3)$$

In which j and k run over all stripes S, i over all coding lines $l_{code,i}$. The labels $n*$ form the solution we look for. By reparametrization, the sum over i and j can be collapsed into a single summation which is evaluated over all cues available. The first four terms express that the weighted and squared differences between the resulting labeling n^* and the corresponding cue n should be minimized. The last term introduces the neighbor relationship between the stripes.

3.1 Cast into a graph cut

Given we aim at an algorithm with real-time capabilities we need a time and memory efficient way to solve (3). To do so we cast the problem into the search for a minimal cut on a graph.
A weighted graph G=<V,E> consists of a set of nodes V and a set of weighted edges E to connect them. Our graph also contains some additional special nodes called terminals, of which one is the source s and one is the sink t. The nodes V form chains that interconnect s and t. If the nodes of a chain are also sideways interconnected we get a structure as visualized in figure 4. A binary cut on this graph, divides the set of nodes V in two subsets of which one contains s and one t. We will generate such a cut, so that the cutting surface minimizes the total sum of the weights of all edges which are broken. If the two endpoints of every chain are connected to respectively s and t we know that every chain will be cut. The problem of searching this binary cut can be solved very efficiently (see the work of Kolmogorov et al. [12]).

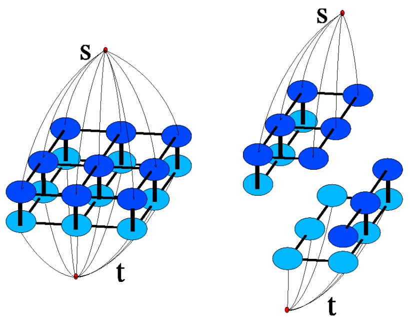

Figure 4: *A graph cut divides a set of nodes into two disjoint sets S and T.*

Now we turn back to the problem of equation 3. We will associate with every stripe S_i a chain of nodes, which interconnects s and t. The length of every chain equals n_{max}, the size of the set of all possible labels L in expression 1. Every edge of this chain represents a possible label for the corresponding stripe. The edges are initialized with unity weights. At the position of the label predicted by a cue n_x of equation 3 the edge is weakened by the corresponding α, β, γ or δ weight. Instead of only a local weakening of an edge, we place a Gaussian around each cue and let it influence a local environment on the chain of which the size corresponds to the uncertainty on the specific cue (cfr. inf.). As every chain is connected to s and t each stripe will be given a label, as the stripe certainly will be cut.
The ϵ-term in (3), corresponding to the neighbor relationship can be taken into account by a sideways interconnection of the nodes of chains which correspond to neighbors. These edges run diagonally through the node tensor, and encourage neighbors to have consecutive labels. See figure 5.

We want our solution to be unique, which means that we want every chain to be cut only once. We can achieve this by avoiding that the cutting surface can 'fold back' in the node tensor. See figure 5. This is achieved by putting a constraint on the minimal weight in the direction of a chain versus the variation in the weights which interconnect different chains. Define $min(A)$ as $((1-\alpha-\beta-\gamma-\delta)$, which corresponds to the minimal cost which can occur in the longitudinal direction of a chain. B corresponds to the cost of interconnecting two chains j and k: $(\epsilon \times Nb(S_j,S_k)(n_j^*+1-n_k^*)^2)$. As can be seen in figure 5, a cutting surface can turn back to avoid the edge with cost $max(B)$ at a minimum cost of $2min(A)+min(B)$. This leads to the following constraint

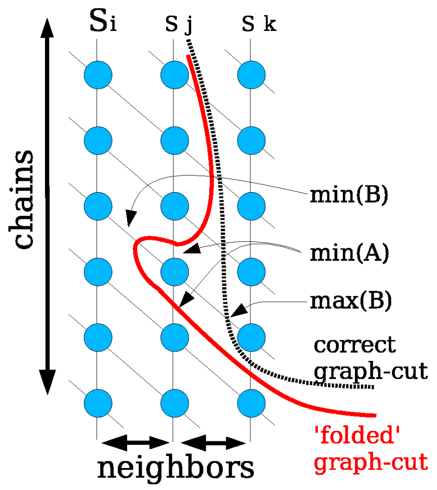

Figure 5: *A graph cut can fold back at a minimal cost of 2*min(A)+min(B) to avoid an edge which has a maximum cost of max(B). In order to have a unique solution for every stripe S this folding should be avoided.*

on the edge weights:

$$2min(A) > \Delta(B) \tag{4}$$

The graph cut is speeded up considerably by limiting the length of each chain to a window around the solution predicted by a much faster pre-labeling. For this prelabeling we assign labels recursively based on the neighbor relationship and the geometric coding. Inconsistencies overwrite eachother. The search window around the predicted labeling is adapted to be always larger than twice the average mislabeling on this prediction. A minimum window of 20 labels takes into account the possible large and unpredictable errors based on the 'overwriting' of cues during this prelabeling.

3.2 Inconsistent solutions - uncertainty on the solution

The solution for the labeling problem solves most but not all of the complications listed in the intro of section 2. False stripes [A], missing stripes [B], and non monoticity [D] are dealt with appropriately. If enough cues point to a labeling which is different from the one predicted by the neighbor relationship, the latter will be automatically ignored. In the case of false stripes, one ends up with several stripes on the same scanline with the same label. This is recognized and the stripe which deviates most from the local average orientation is removed.
However no solution for the wrongly interconnected stripes which can occur around vertical occlusions is proposed so far. A theoretically sound solution for this would be to upgrade the labeling problem, which is almost 1 dimensional to a full 2D labeling. This can be done by splitting all stripes in small pieces which are interconnected in a bottom-top way to represent the original stripe. The interconnections can be weighted by the edge strength of the stripe. This subdivision allows the graph cut to not only separate neighboring stripes, but also to cut stripes in the longitudinal direction. This increase of the problem dimensionality however does not allow us to solve the problem at high speeds anymore. Furthermore it introduces too much freedom: we don't want stripes to split anywhere, but only where needed because a problem arises.
To this end we define the local *consistency* of the solution. Define the residual for stripe S_j on cue k, in which $k \in [0, 4]$ refers to the cues described in section 2.1, as $r_{j,k} = n_{k,j} - n_j^*$. With r_j we refer to the total residual for S_j, weighted by the corresponding α, β, γ, δ and ϵ factor. Now the consistency for S_j is defined as:

$$consistency(S_j) = \frac{e^{-r_j^2}}{e^{maxCues - \#cues}} \tag{5}$$

$maxCues$ corresponds to the maximum number of cues a stripe can receive: $maxCues = [2 \times R_v + 3]$ (geometric coding + code tracking, color coding, stripe tracking, neighbor relationship), $\#cues$ reflects the actual number of cues received. This consistency is a measurement for the local uncertainty on the labeling - and thus the 3D-reconstruction - assigned by the graph cut. The more a label of a stripe respects the cues available, the more the consistency will approach 1. The denominator ensures that a stripe which is more heavily coded receives a higher weight.

Now we have a way for identifying the problematic case of situation [C]. We place a threshold on the residual (and thus the consistency) of each stripe, if too low the neighborhood of this stripe is examined. After this conflict resolution the set of stripes Q is re-ordered and fed again into the graph cut. As a practical threshold a weighted residual of 5 labels was used in our tests as a threshold to switch to this problem resolution strategy. If needed this process can be repeated several times. This search for a global optimum based on the graph-cut, and afterwards the optimization of the consistency for this solution, boils down to an expectation-maximization approach with the graph-cut as E-step and the conflict resolution as M-step.

The conflict resolution itself searches for a closed loop of neighbors with an inconsistent labeling around the stripe which triggered this mechanism. We now search a path from inside this loop to a point external to the stripe region, thereby defining a cut. The path search looks for the minimal number of stripes to cut, and every cut is weighted with the gradient magnitude of the underlying stripe boundary. See figure 6.

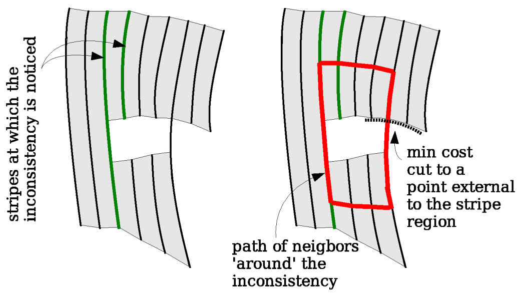
stripes at which the inconsistency is noticed
min cost cut to a point external to the stripe region
path of neigbors 'around' the inconsistency

Figure 6: *A path from inside an 'inconsistent loop' to a point outside the region covered by neighboring stripe resolves a labeling conflict.*

4 Pattern adaptations and cue weighting

The ability to decode and identify the pattern is of primary importance to our system. As we achieve rather high frame rates (10-20 fps) we exploit time continuity to change pattern settings to make the codes and pattern as detectable as possible.

The geometric code has R_v and R_h as free design parameters. R_h is chosen as high as possible as long as the average residual on the stripe tracking satisfies : $|r_2| < \frac{n_{max}}{2\times R_h}$. In this case the ambiguity of multiple coding line intersections can always be resolved. R_v, the number of times coding lines are repeated below eachother, is chosen in order to achieve high coding densities. The constraint here is that the residual on the coding remains limited: $|r_0| < T$ in which T is an arbitrary threshold. (In our tests we took for T half the inconsistency threshold.) In our experiments we use a low R_v to bootstrap the system and gradually increase R_v. The color of the coding lines is chosen to minimize interference from scene colors which are similar. The width of the base pattern is adapted in order to always have a stripe width after detection of approx 3 pix. This assures visibility and keeps the detector matched to the pattern. For more details see [14].

For the location of the color coded slots the total residual r_j for each stripe is used. As this coding normally gives a high quality coding cue, it is used to overcome problems in a region which is highly uncertain. As the detector also splits stripes based on a sudden color gradient in the longitudinal direction this also is a preemptive resolution for the next frame of a possible conflict in this area (cfr. previous section). The number of color coded slots is chosen $\leq R_h$ and is implicitly determined by the discriminant power of the tracking.

A last set of parameters which should be determined are the relative weights $\alpha,\ \beta,\ \gamma,\ \delta$ and ϵ for the different cues in equation 3.

The cues for $k \in [0-3]$ are noisy measurements of the unknown correct label n_j^* . Noise on these measurements is modeled as a normal distribution with mean 0 and standard deviation σ_k:

$n_{j,k} \ \sim \ n_{j,k}^* + N_k \ , \ \ N_k \sim \mathcal{N}(0, \sigma_k^2)$

The optimal estimation of n_j^* given the measurements $n_{j,k}$ is the one that maximizes the posterior probability:

$$\prod_{jk} P(n_j^*, \sigma_k | n_{j,k}) = \prod_{jk} \frac{P(n_{j,k}|n_j^*, \sigma_k) P(n_j^*, \sigma_k)}{P(n_{j,k})} \quad (6)$$

in which the denominator can be seen as a normalization factor. (Summation of the numerator over all (n_j^*, σ_k))

Given the noise model on $n_{j,k}$ the data likelihood becomes:

$P(n_{j,k}|n_j^*, \sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} exp(-\frac{1}{2}\frac{|n_{j,k}-n_j^*|^2}{\sigma_k^2})$

The prior $P(n_j^*, \sigma_k)$ in equation 6, can be reformulated as $P(n_j^*) \times Const$ as we have no prior knowledge about σ_k. $P(n_j^*)$ can be modeled as an exponential distribution $exp(-R(n_j^*)/\lambda)$. λ defines the width of this distribution and $R(n_j^*)$ is a regularizer which expresses that the labeling should be continuous over the stripe set Q. This is exactly what is expressed by the expression $(Nb(S_j, S_k) \times (n_j^* + 1 - n_k^*)^2)$ from equation 3.

Instead of maximizing the probability of equation 6, we minimize its negative logarithm. This leads to:

$$\begin{aligned} E = \sum_k -log(\sqrt{2\pi\sigma_k^2}) + \frac{1}{2}\sum_{jk} \frac{|n_{j,k} - n_j^*|^2}{\sigma_k^2} \\ + \frac{1}{\lambda}\sum_{jk} Nb(S_j, S_k) \times (n_j^* + 1 - n_k^*)^2 \end{aligned} \quad (7)$$

If we compare equation 7 with equation 3 we see that for the relative weights of the cues $\alpha,\ \beta,\ \gamma,\ \delta$ the variation σ_k on the corresponding residual r_k can be used as a valid steering parameter. The weight ϵ or $1/\lambda$ is left as a parameter to the user and expresses our belief in the correctness of the relative labeling. For continuous surfaces this can be taken higher than for complex irregular shapes. In practical experiments we limited the adaptations of the weights to vary within a window around an initialization to avoid drifting away to trivial solutions.

5 Results

Figure 7 shows how the regression lines (bot. left.) predict the location of the geometric code in the next frame. Due to noise, and fast movements the distance to the real codepoints is still considerable. Trimmed codepoints are also visualized. The consistency map shows how densely coded parts of the hand are considered more trustworthy. The high believe in the reconstruction of the top of the fingers originates from a color coded stripe in that region. The low consistency of the coded mid part of the fingers is caused by

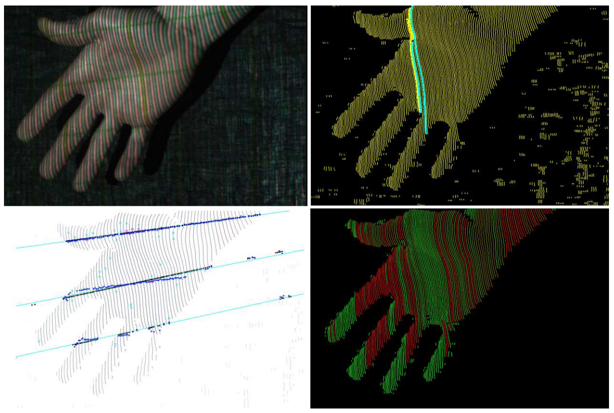

Figure 7: *Top left: input image, Top right: an inconsistency is detected, Bottom left: regression lines for codepoint tracking and trim to final location, Bottom right: resulting consistency map.*

bad geometric decoding because of the low quality prediction. The top right figure within this set shows the detection of an inconsistency at the thumb before conflict resolution.
Figure 8 shows how the inconsistency detected in figure 7 result in a major flaw in the reconstruction if no action is taken. A single stripe is wrongly detected and ’crosses’ the occlusion around the thumb. The right part of this image illustrates how a search path from within an inconsistent loop around the labeling problem resolves the error.
Figure 9 shows a sequence of shots taken of a moving and deforming hand. The camera viewpoint is moved more the bottom left to show the differences with the input images. Our pipeline was able to process this sequence at approx 15 fps.

6 Conclusion & future work

We proposed an approach for high speed 3D acquisition based on structured light. To solve the correspondence problem between camera and projector a combination of geometric coding, color coding and tracking over time is used. The correspondences are distributed over the detected pattern based on relative labeling.
In order to combine the different not fully consistent cues, a least square solution is implemented by using a graph cut. An EM-step allows for conflict resolution.
The design parameters for the pattern are adapted online to make it more robust against current noise levels on the decoding and to ease the detection. This results in a strategy for online adaptively coded structured light. Relative weights of the different cues are data determined, and adjusted over time. Based on the residual on the data approximation we can assign a local consistency measure to the reconstruction.
Future work will focus on more elaborate pattern adaptations and further optimizations for speed.

Acknowledgments

The authors gratefully acknowledges support by KULeuven Research Council (GOA).
Thomas Koninckx gratefully acknowledges support by a grant of the Flemish Institute for the Advancement of Science in Industry ‘IWT’.

References

[1] J.Batlle, E.Mouaddib and J.Salvi, Recent Pogress in Coded Struct. Light as a Technique to Solve the Correspondence Prob: Survey, Pat. Recog. vol 31, nr.7, pp.963-982, 1998.

[2] D.Caspi, N.Kyriati, J.Shamir, Range Imaging With Adaptive Color Structured Light, IEEE PAMI vol 20, nr.5, pp.470-480, 1998.

[3] P. Vuylsteke, A. Oosterlinck, Range Image Acquisition with a Single Binary-Encoded Light Pattern, IEEE PAMI, 12(2), pp.148-164, 1990.

[4] M.Proesmans, L.Van Gool, 1-Shot Act. 3D Im. Capt., Proc. SPIE, pp.50-61, 1997.

[5] K.Boyer , A.Kak, Color-encoded structured light for rapid active ranging, IEEE PAMI, v.9 n.1, pp.14-28, 1987.

[6] J.Zhang, B.Curless, S.Seitz, Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming, 3DDPVT, pp.24-36, 2002.

[7] M. Maruyama , S. Abe, Range Sensing by Projecting Multiple Slits with Random Cuts, PAMI, v.15 n.6, pp.647-651, 1993.

[8] P.Griffin, L. Narasimhan, S.Yee, Generation of uniquely encoded light patterns for range data acquisition, Pattern Recognition, 25(6), pp.609-616, 1992.

[9] O.Hall-Holt, S.Rusinkiewicz, Stripe Boundary Codes for Real-Time Struct.-Light, ICCV, pp.359-366, 2001.

[10] E. Horn, N. Kiryati, Towards Optimal Structured Light Patterns, Image and Vision Computing, Vol. 17, pp. 87-97, 1999.

[11] A. Strat, M. Oliveira, A Point-and-Shoot Color 3D Camera, 4’th Int. Conf.on 3-D Digital Imaging and Modeling, pp483-490, 2003.

[12] Y.Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision, Int. Workshop on Energy Min. Methods in Comp. Vis. and Pat. Rec., vol. 2134 of LNCS, pp 359-374, 2001.

[13] T.Koninckx, L. Van Gool, High-speed active 3D acquisition based in a pattern-specific mesh, proc. SPIE: EI Photometrics, pp.26-37, 2003.

[14] T. Koninckx, A. Griesser, L. Van Gool, Real-time Range Scanning of Deformable Surfaces by Adaptively Coded Structured Light, 3DIM03, pp. 293-302, 2003.

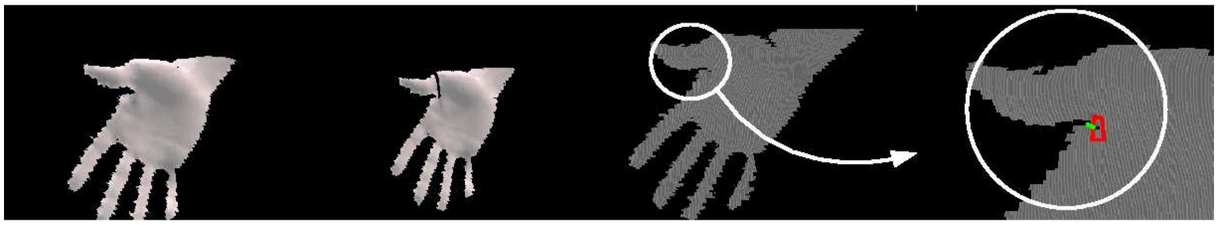

Figure 8: *Left: wrongly detected base pattern. A stripe crosses an occlusion. A major error in the reconstruction results. Right: conflict resolution.*

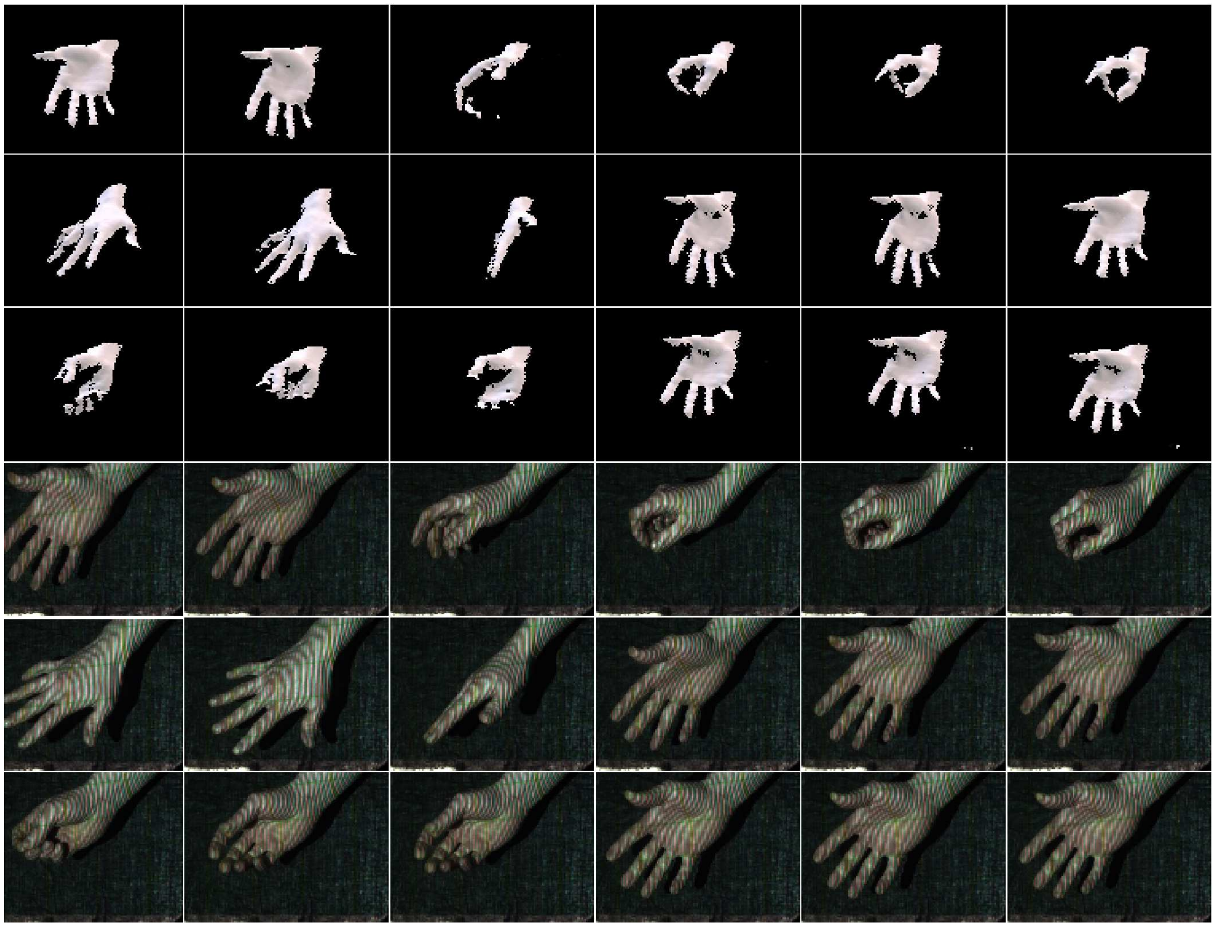

Figure 9: *Top: a series of reconstructions for a moving and deforming hand. Every 5'th frame is shown. Bottom: the corresponding input images. Note that the camera viewpoint for the reconstruction is slightly moved to the bottom left, in order to visualize jitter and/or flaws.*