Recognition From Point Clouds and 3D Models

Eduard Sojka, Department of Computer Science, VŠB-TU Ostrava
eduard.sojka@vsb.cz, http://mrl.cs.vsb.cz

ISCAMI 2018, Malenovice, May 13, http://mrl.cs.vsb.cz/data/pacman/ISCAMI_2018.pdf

Our work presented here was partially supported by the EU H2020 686782 PACMAN project, (together with Honeywell), http://mrl.cs.vsb.cz/h2020

What are the actual problems of recognition?

- 2D recognition? (No, It is too simple. Please, excuse exaggeration.)
- 3D reconstruction/measuring itself? (As a rule, too simple again.)
- **Action recognition** (e.g. surveillance systems, robots in human-robot interaction)
- **3D recognition** (robots, augmented reality, includes the problems of occlusions, the 3D position should be determined too).

Brief taxonomy of 3D recognition approaches

- Optical images with learning (i.e. the 2D approach that is now successful)
- Optical images with 3D models without learning.
- Depth images (point clouds) with 3D models.

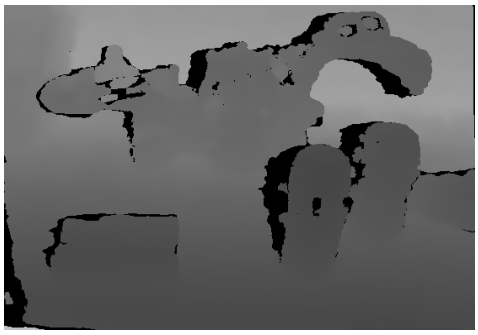
  (And where have hidden the approaches with learning?)

Why the 3D models may be useful: They may be immediately available, e.g. from CAD systems. They can also be used in the case with learning (synthetic images can be generated). Is learning useful if I have a model? Does not learning create a model from images?

Why depth images and point clouds may be useful: You see the true shape not only its flat projection as it is in the case of usual optical images.

Several remarks on the sensors

(x, y, z) "images" exist (not only intensity images). The real coordinates are usually provided. The output may be either unorganised (list of the points, **point cloud**) or organised in a matrix (**depth map**, figure below). Moreover, the map (and cloud) **may be combined with images** (RGB, IR, grayscale).



The corresponding devices are based on various principles.

Structured lighting

Time of Flight

Lidar scanner

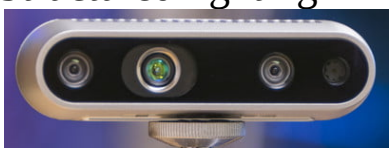


Image stereo (Furthermore, depth from defocussing)

Certain limitations may follow from the principle that is used. **What is the future of the depth sensors?**

What is expected in 3D recognition

The goal: Determine the objects that are captured in the map or point cloud together with their position.

Two motivating examples

Detection of various set of objects together with determining their location (developed for the use in an augmented reality system; http://mrl.cs.vsb.cz/data/pacman/excavator.avi).

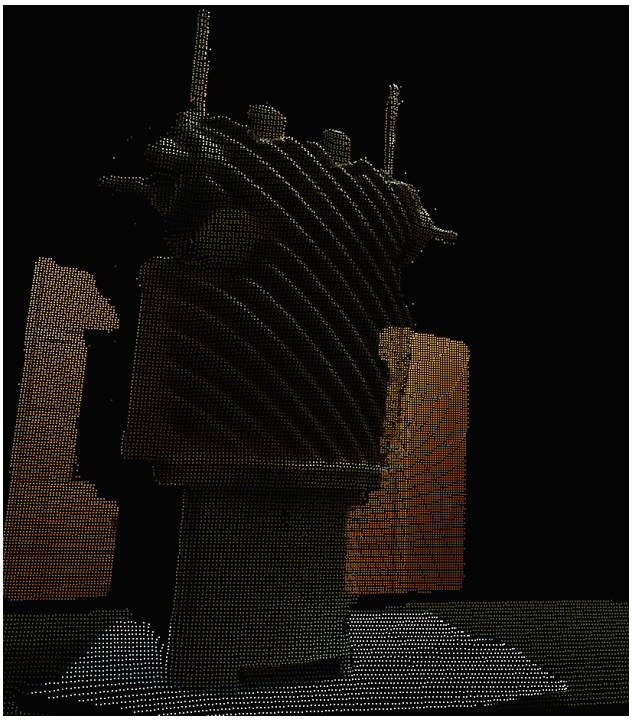
Detection and tracking the human bodies in car, especially the driver (based on a deformable model of human body; created for VW; http://mrl.cs.vsb.cz/data/driver_model/outdoor_ride2.mp4.zip).

Methods of 3D recognition

- Surprisingly, something like **template matching** method can be used (also, **geometric consistency**).
- Surprisingly, it may be regarded as a state-of- the-art method now.
- Surprisingly, easy and extremely difficult at the same time (the basic idea is simple, but the problems are everywhere).
- Will the method suffice if we want to recognise many objects in really complicated scenes … ?
- Maybe, a challenge for you ...

A pictorial introduction / explanation

Models (pre processed)



Scene

Main idea: A rigid transformation is sought for that moves a model such that it sufficiently fits to a piece of scene. (So simple.)

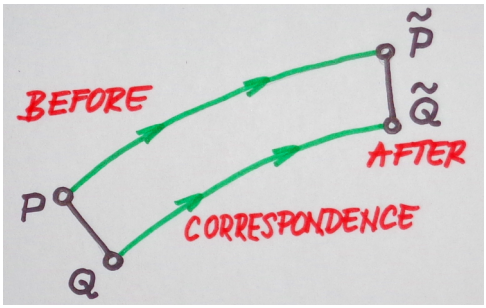
Some problems:

- How the transformation can be found?
- How the fitting can be measured?
- What fitting is sufficient?

On the rigid geometric transformation

The rigid geometric transformation (lengths and angles are preserved) is a key point. The recognition can be done like template matching, but an unknown geometric transformation (from the model to the scene) should be done before.

$$\mathbf{x}_{\text{SCENE}} = \mathbf{R}\mathbf{x}_{\text{MODEL}} + \mathbf{t}$$



The rigid transformation is determined by 6 independent parameters; 3 translations in **t**, and 3 rotation angles in **R**. They are fully determined by two pairs of **corresponding** points before and after the transformation (**we say two correspondences**), which gives 2×3 equations for 6 parameters.

The problem: Given N points m_i of the model, and N points d_i in the scene such that m_i corrseponds to d_i. Find the estimates of **R** and **t.**

$$\bar{d} = \frac{1}{N}\sum_{i=1}^{N} d_i \qquad d_{\mathrm{c}_i} = d_i - \bar{d} \qquad \bar{m} = \frac{1}{N}\sum_{i=1}^{N} m_i \qquad m_{\mathrm{c}_i} = m_i - \bar{m}$$

$$\Sigma^2 = \sum_{i=1}^{N} \| d_{\mathrm{c}_i} - \hat{\mathbf{R}}\, m_{\mathrm{c}_i} \|^2 = \sum_{i=1}^{N} (d_{\mathrm{c}_i}^{\mathrm{T}} d_{\mathrm{c}_i} + m_{\mathrm{c}_i}^{\mathrm{T}} m_{\mathrm{c}_i} - 2\, d_{\mathrm{c}_i}^{\mathrm{T}} \hat{\mathbf{R}}\, m_{\mathrm{c}_i})$$

$$\mathbf{H} = \sum_{i=1}^{N} m_{\mathrm{c}_i}\, d_{\mathrm{c}_i}^{\mathrm{T}}$$

$\mathbf{H}=\mathbf{U}\Lambda\mathbf{V}^{\mathrm{T}}$ (SVD). The estimates then are

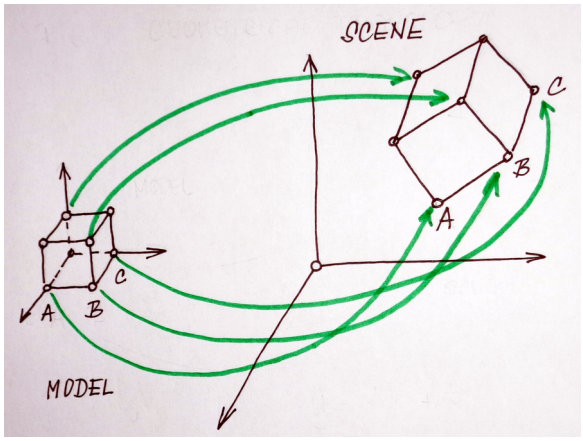
$$\hat{\mathbf{R}} = \mathbf{U} \begin{pmatrix} 1 & & \\ & 1 & \\ & & \det(\mathbf{U}\,\mathbf{V}^{\mathrm{T}}) \end{pmatrix} \mathbf{V}^{\mathrm{T}} \qquad \hat{\mathbf{T}} = \bar{d} - \hat{\mathbf{R}}\,\bar{m}$$

D. W. Eggert, A. Lorusso, and R. B. Fisher. 1997. Estimating 3-D rigid body transformations: a comparison of four major algorithms. *Mach. Vision Appl.* 9, 5-6 (March 1997), 272-290. DOI=http://dx.doi.org/10.1007/s001380050048

Why two correspondences could be preferred? More correspondences would probably lead to more precise estimation. However, it is not convenient for the RANSAC algorithm. One correspondence requires determining of Local Reference Frame (LRF). In this case only translation is needed in such a case. Determining LRF reliably is difficult.

Consistent correspondences

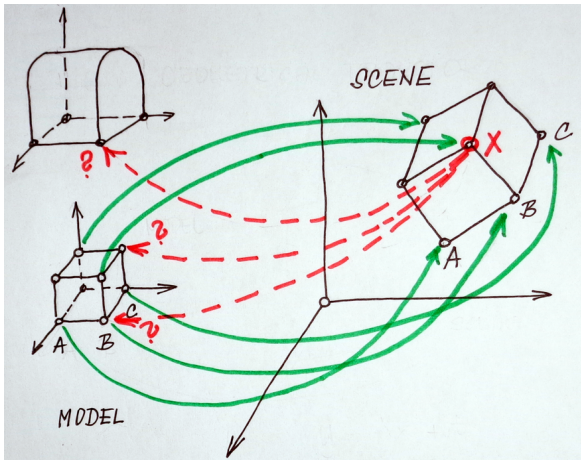
Interesting points (**keypoints**) are introduced (by intuition for now).



Say that an object was placed to a scene by a known rigid transformation. In this case, much more than two pairs of corresponding points can be seen; we have a **consistent set of correspondences** since all of them correspond just to one object and all of them comply with a single rigid geometric transform.

Inconsistent correspondences

The essential step is to try and find the keypoints in the scene together with their certain description (**keypoints and descriptors**). For each of them, possibly corresponding candidates are then sought in the models of all possible objects. Neither the correct object, nor the correct transformation are known at this moment. The correspondence is estimated on the basis of **similarity of the descriptors**.



Highly **inconsistent set of correspondences** is usually obtained for the scene keypoints (for each of them, the correspondences with more keypoints of the correct object and incorrect objects are usually found). The problem of recognition can be reformulated. In that big inconsistent set of correspondences, find one or more confincing subsets that are consistent, i.e. the corresponding to one object and complying with a single rigid geometric transform.

The basic rough algorithm for 3D recognition

Input: Point cloud of a scene, point clouds of models (i.e. their 3D models)
Output: Identification of the object that is seen together with its position
Note: The model point clouds can be preprocessed, i.e. their keypoints together with their description (feature vectors) are precomputed. The scene with one object only is considered for now (the generalisation is straightforward).

Compute the scene keypoints and their description (feature vectors).

For each model do { // The subscript o distinguishes the objects.

- Determine the set of all correspondences, denoted by C_O, between the scene and the object point cloud (carried out by searching according to the feature vectors of keypoints)
- In C_O, find the biggest subset of correspondences, denoted by $C_{O,T}$, that are consistent; it defines one rigid transform. (Possibly more subsets could be determined, if more occurrences of the object were expected.)
- Assess $C_{O,T}$ in some way, e.g. by $|C_{O,T}|$. (Simple and fast, but not good. If you want better results, match each object back to the scene, and evaluate how it happened.)

}

Among all $\{C_{O,T},\}$ select the one with the biggest score, which gives the object (O) and its position (T). **(Looks simple, but ... :-))**

The particular problems

All the particular tasks are essential, and difficult to solve appropriately. The speed is absolutely critical. There is a lot of space for improvement of the known methods. Without having the appropriate solution, the final result will not come.

Keypoint detection: Given a scene point cloud or a model point cloud, detect the points that are important for the recognition, (salient, interesting, easily detectable, distinguished from other points, stable with respect to LOD).

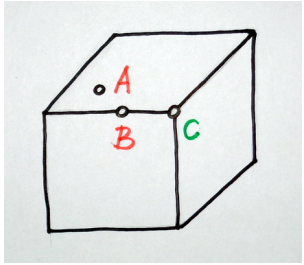
Judging the correspondence between the key points: Given a keypoint in a model and a keypoint in a scene, say how much similar the two given keypoints are.

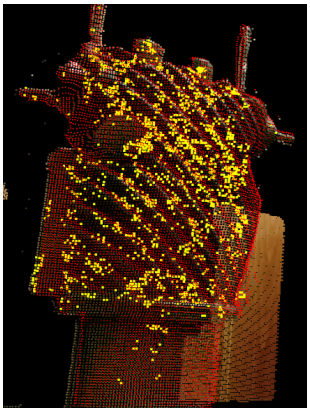
Determining the corresponding model keypoints: For a given scene keypoint, find all keypoints that seem to be the same in a given model. (Contains searching, and answering the question “how much similar two given keypoints are”.)

Finding the best consistent subset of correspondences: Given an inconsistent set of correspondences. Determine the consistent subset with the best assessment (e.g. the maximum consistent subset).

On the keypoint detection

The **keypoint** reflects the **shape of its certain neighbourhood**. What is a good keypoint? **Well-defined, salient, interesting, easily detectable, distinguished from other points, stable with respect to noise and various levels of details**.

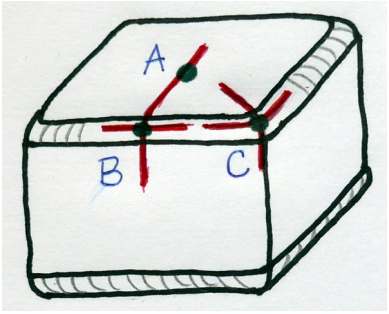


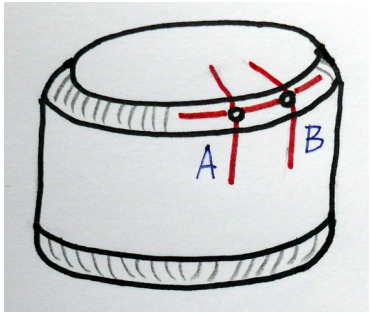


Moreover, the keypoint **should contribute to recognition effectively** (it should not contribute too much to increasing the set of false correspondences). This is difficult. If the keypoints reflect only the local geometrical properties of the surface (from a small neighbourhood), similar keypoints may appear on various places of various objects, which causes that many false correspondences are found. They must be discarded later, which is computationally expensive.

Are *A* and *B* good keypoints? How could *C* be detected?

Examples of several detectors can be found in Appendix A. Quite often, the principal curvatures (or similar values) are exploited (Gaussian curvature, shape index etc). Non-maxima suppression is usually also used in some way.



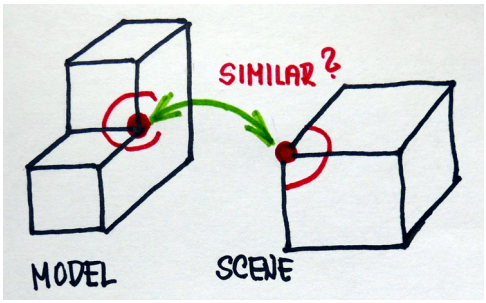


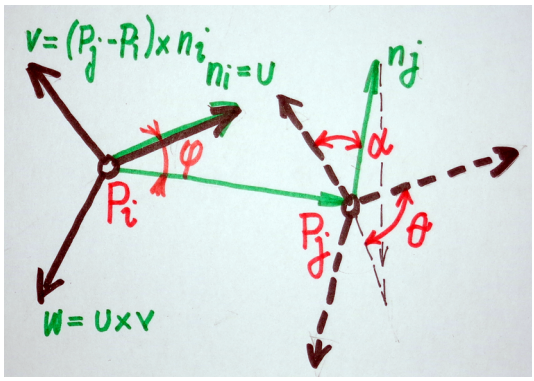
Conclusion: In spite of many efforts, something like an ideal keypoint detector does not exist. Unwanted keypoints always occur, and some expected keypoints are always missing. Certain objects always develop problems (e.g. rotational). The subsequent steps should not be to much sensitive to it.

Judging the correspondence of keypoints

Problem: Determine whether two keypoints are similar. Every key point is described by a vector containing numerical values (features, descriptors) characterising the shape of its neighbourhood (often, curvatures again). For two keypoints (P_1, P_2), we have two feature vectors $\mathbf{p}_1$, $\mathbf{p}_2$.

$\| \mathbf{p}_1 - \mathbf{p}_2 \|_2$ evaluates the similarity





An example of how the descriptors can be constructed (PFH – point feature histogram)

The algorithm (descriptors for P_i): Find all points in a certain neighbourhood of P_i. Compute the angles φ, α, θ for all point pairs in the neighbourhood. Create the histo-gram of the angles (e.g. 5×5×5=125 values).

Requirements: A local neigbourhood of a key point should be described in an accurate way, the description should be resistant to noises and stable for various levels of details. What should be the size of neigborhood? Selecting the useful features is difficult. It determines the final successfulness and the run time.

Conclusion: Many approaches appeared in the last decade with no clear winner between them, many problems appear in all cases. **The area is still open.** Examples of how the descriptors can be created can be found in the appendix.

Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. 2016. A Comprehensive Performance Evaluation of 3D Local Feature Descriptors. *Int. J. Comput. Vision* 116, 1 (January 2016), 66-89. DOI: http://dx.doi.org/10.1007/s11263-015-0824-y

Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. 2009. Fast point feature histograms (FPFH) for 3D registration. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation* (ICRA'09). IEEE Press, Piscataway, NJ, USA, 1848-1853.

Basic approaches

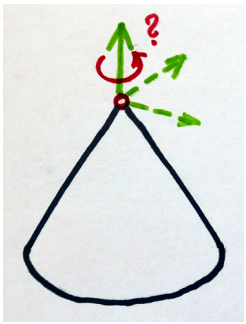
- With/Without local reference frame (it also has an overlap to recognition).
- Based on point positions directly vs. based on normals (i.e. angles and curvatures).

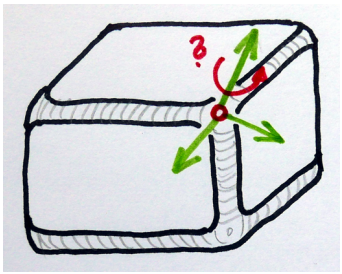
Worth discussion (1): What is the strength of the descriptor? Computational speed (e.g. PFH → FPFH). The size of descriptor (e.g. are 125 values of PFH really necessary?)

Worth discussion (2): Given a set of feature values with a decreasing information content and with noise. Does an optimal number exist of the values that should be used? Yes, it probably does exist! Many values are harmful.

Local Reference Frames: Are They Useful?

The problem: Take a keypoint and determine a new local coordinate system (*x*, *y*, *z*) in it. Can such a system be defined reliably and repeatedly (with respect to various noises, scale differences etc)?





Conclusion: The LRF would be useful if it were reliable enough, which is difficult (almost impossible) to achieve in practice.

How the local reference frame (LRF) can be introduced? Roughly: Take the *k*-nearest neighbours of the key point you consider and compute the covariance / scatter matrix (sometimes, certain weights may be introduced into the formula below)

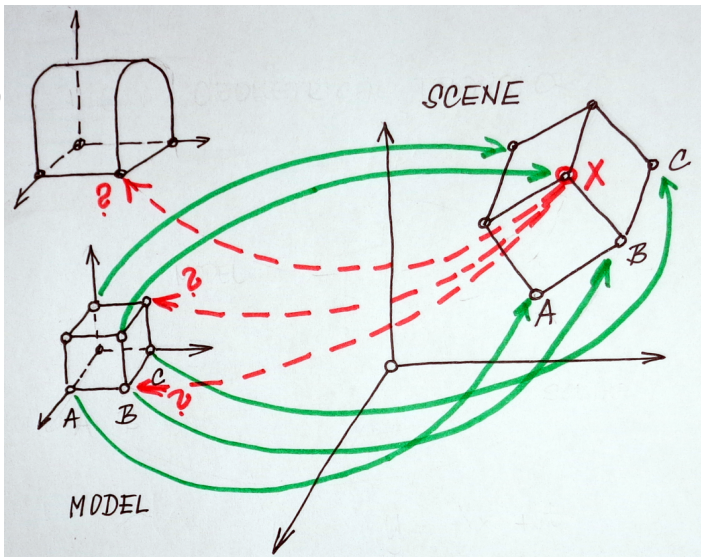
$$\mathbf{M} = \frac{1}{k}\sum_{i=0}^{k}(\mathbf{p}_i - \hat{\mathbf{p}})(\mathbf{p}_i - \hat{\mathbf{p}})^T, \;\; \hat{\mathbf{p}} = \frac{1}{k}\sum_{i=1}^{k}\mathbf{p}_i \;\; .$$

The eigenvectors of M give the axes of LRF. Create the rules for determining positive and negative directions of the axes. (A note: Usually, the eigenvector corresponding to the smallest eigenvalue is taken a the direction of normal. For certain keypoints, it can be a problem.)

Samuele Salti, Federico Tombari, Luigi Di Stefano, SHOT: Unique signatures of histograms for surface and texture description, Computer Vision and Image Understanding, Volume 125, 2014, Pages 251-264, ISSN 1077-3142, https://doi.org/10.1016/j.cviu.2014.04.011

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. 1992. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.* 26, 2 (July 1992), 71-78. DOI: https://doi.org/10.1145/142920.134011

Determining the corresponding model keypoints



Problem: Given a keypoint X in the scene, find all points Q in a model such that $\|\mathbf{p}_X - \mathbf{p}_Q\|_2$ is small enough (i.e. X,Q are sufficiently similar).

Easy from the theoretical point of view (searching for nearest neighbours in a data structure describing models).

Why it is difficult: The dimension of $\mathbf{p}$ may be hight (from tens to hundreds or even thousands). The real scenes usually contain many keypoints (say at least 10^5). You may have more objects. The objects also have many keypoints. Everything should be done in real time (say from 10 to 30 scenes per second).

Something special must be done here (parallel GPU K-d trees, oct-trees, FLANN, etc) since the speed is absolutely critical here.

Filtering the set of correspondences

The set of inconsistent correspondences may be big, say 10^5 or even more correspondences in total. Can this number be reduced in a reasonable way (i.e. the correct correspondences are preserved) before searching for the maximum consistent subset? Examples of how it could be done:

- Kalman filtering of object motion, i.e. a certain estimate of the model to object transformation is known after some time. The transformations that are too different can be eliminated.
- Creating pairs of correspondences, checking the distance on the model and on the scene side, and voting.

Finding the best consistent subset of correspondences

Problem: Given an inconsistent set of correspondences. Determine the consistent subset with the best assessment (i.e. recognise an object and its position). Say that we have N correspondences (false and correct), and that the geometric transformation is determined by two correspondences (m=2). Say that among N correspondences, I are correct correspondences (inliers); $\varepsilon = I/N$.

Algorithm (RANSAC): Generate the pairs of correspondences from N randomly, determine the transformation, check how many correspondences from N are satisfied with this transform. If the number is big enough (highest), we have a candidate for recognition.

Worth discussion (1): How many samples/trials should be done? The probability η of missing a better set of inliers of size I is

$$\eta = (1 - P_I)^k \qquad P_I = \frac{\binom{I}{m}}{\binom{N}{m}} = \prod_{j=0}^{m-1} \frac{I-j}{N-j} \approx \varepsilon^m$$

$$k = \log(\eta) / \log(1 - P_I)$$

An example and alternative formulation: N=10^5, I=10^2, m=2, η=0.03, ε=10^{-3}, k≈3.5×10^6. We are looking for a needle in a haystack. That is why good key-points, descriptors, and pre-filtering are preferred. Parallelisation is needed.

Worth discussion (2): Can a higher value of m cause that the transformation is determined more precisely? Yes, it can, but at the expense that much more trials must be done (see the example above).

Chum, Ondrej & Matas, Jiri & Kittler, Josef. (2003). Locally optimized RANSAC. 2781. 236-243. 10.1007/978-3-540-45243-0_31.

Rahul Raguram, Jan-Michael Frahm, and Marc Pollefeys. 2008. A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In *Proceedings of the 10th European Conference on Computer Vision: Part II* (ECCV '08), David Forsyth, Philip Torr, and Andrew Zisserman (Eds.). Springer-Verlag, Berlin, Heidelberg, 500-513. DOI=http://dx.doi.org/10.1007/978-3-540-88688-4_37

Final matching is necessary

The number of inliers that are found in the previous step is not a good criterion for determining the final result (e.g. since the keypoints may be distributed irregularly). The object and the transformation that were found must be checked (the object must be really matched to the scene). This would be time consuming for all RANSAC trials. Therefore, the candidates with high values of assessment from the previous step are checked (for each model, a set of matching points can determined). The winner is then decided on the basis of matching to the scene.

A final conclusion

- Finding the geometric consistency may be regarded as a state-of-the-art method today. It is the basic method for recognition that is based on the 3D object models. (Can the tools like deep learning be useful here? How do the humans recognise the 3D objects? What about the position?)
- It works somehow (not only extremely simple scenes, more objects, real time), but it is a brute force and computationally demanding method. The limits can be seen. Can it be called an “artificial intelligence”?
- Even for solving the problems that are not extremely complicated, it is so computationally expensive that almost nothing can be done by making use of only several CPU cores. (GPU and CUDA are everywhere).
- **Problem:** If a really difficult scenes should be solved, a break through in computer architectures or in algorithms (as weel as in the sensors) must be done.
- **The main question:** What will become the state of the art method after 10 or 20 years? Perhaps, a challenge for you. Will you contribute to solution?

Appendix A: Examples of keypoint detectors

Harris 3D: The Harris corner detector is known for usual images (I is the intensity function, I_x, I_y are its derivatives, G stands for the Gaussian function, * denotes the convolution). At the corner point, the following value is big for all Δx, Δy, which leads to the solution that follows.

$$\sum_{x_i, y_i} W(x_i, y_i)\left[I(x_i + \Delta x, y_i + \Delta y) - I(x_i, y_i)\right]^2$$

$$\mathbf{M}(x,y) = \begin{bmatrix} \left\langle (I_x)^2 \right\rangle & \left\langle I_x I_y \right\rangle \\ \left\langle I_x I_y \right\rangle & \left\langle (I_y)^2 \right\rangle \end{bmatrix} \qquad \left\langle \psi(x,y) \right\rangle = \psi(x,y) * \left[G(x)G(y)\right]$$

$$\mathrm{cor}(x,y) = \det\left(\mathbf{M}(x,y)\right) - 0.04 \cdot \mathrm{trace}^2\left(\mathbf{M}(x,y)\right)$$

How it can be adapted for the depth maps and point clouds: PCA analysis of the points in a neighbourhood of the point that is being tested (i.e. find the best fitting plane). The axis corresponding to the smallest eigenvalue becomes the new z axis. In the new coordinate system, the following surface is fitted (it makes computing the derivatives possible)

$$z = f(x, y) = \frac{p_1}{2}x^2 + p_2 xy + \frac{p_3}{2}y^2 + p_4 x + p_5 y + p_6$$

It is now the same as the two-dimensional problem from above.

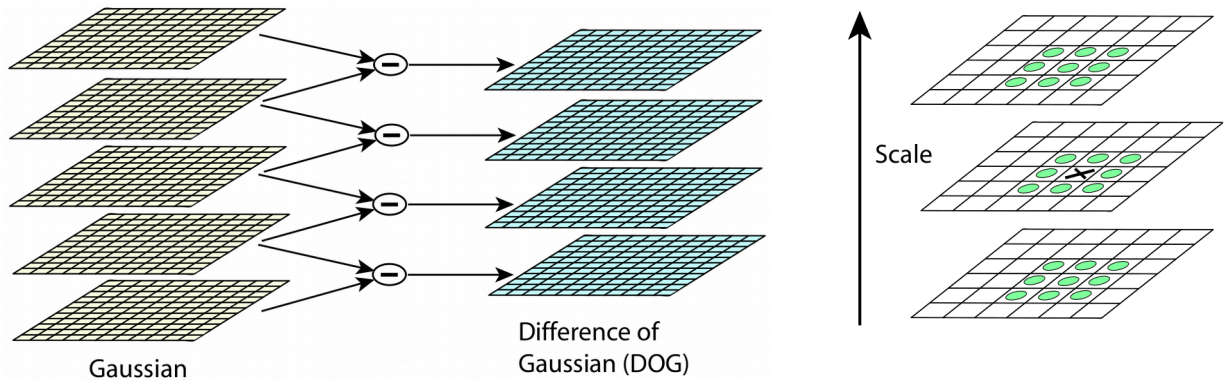
Ivan Sipiran and Benjamin Bustos. 2011. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *Vis. Comput.* 27, 11 (Nov 2011), 963-976. DOI=http://dx.doi.org/10.1007/s00371-011-0610-y

Scale invariant keypoints and feature transform (SIFT): We start here from the 2D version. Firstly, the stable candidates are determined. Images are convolved with Gaussians (increasing sigma) and subtracted.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$



Eliminating edge responses: Hessian matrix at the location and scale of the keypoint. Let α, β be the eigenvalues ($\alpha > \beta$).

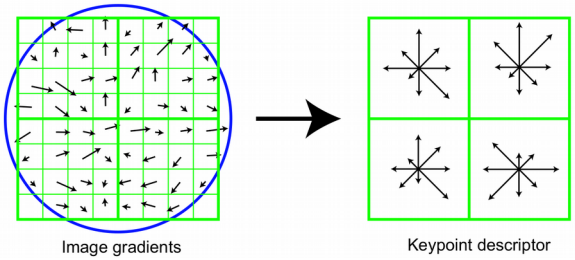
$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$$\begin{aligned} \mathrm{Tr}(\mathbf{H}) &= D_{xx} + D_{yy} = \alpha + \beta, \\ \mathrm{Det}(\mathbf{H}) &= D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \end{aligned}$$

Let r be the ratio such that $\alpha = r\beta$. The inequality should be checked

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}, \qquad \frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}.$$

Direction at a keypoint: Detected as a peak in the histogram of gradient directions in L of the corresponding scale. The descriptor below: 2×2 blocks, with a 8 bin histogram of directions (relative to the keypoint direction) in each of them (usually 4×4 blocks, however).



David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision* 60, 2 (November 2004), 91-110. DOI: https://doi.org/10.1023/B:VISI.0000029664.99615.94

3D version of SIFT - ThrIFT: Instead of images, a density density map D is introduced; $n(B)$ stands for the number of points in a volume B. The indices i, j, k are the indices of equally-sized boxes in spatial dimension.

$$D(i,j,k) = \frac{n(B_{ijk})}{\underset{(i,j,k)\in I}{\operatorname{argmax}}\left\{n(B_{ijk})\right\}}$$

$$L(x,y,z;\sigma) = (D \otimes g(\sigma))(x,y,z) \qquad g(x,y,z;\sigma) = \exp\left(\frac{-x^2-y^2-z^2}{2\sigma^2}\right)$$

$$\mathcal{H}(\mathbf{x},\sigma) = \begin{pmatrix} L_{xx}(\mathbf{x},\sigma) & L_{xy}(\mathbf{x},\sigma) & L_{xz}(\mathbf{x},\sigma) \\ L_{yx}(\mathbf{x},\sigma) & L_{yy}(\mathbf{x},\sigma) & L_{yz}(\mathbf{x},\sigma) \\ L_{zx}(\mathbf{x},\sigma) & L_{zy}(\mathbf{x},\sigma) & L_{zz}(\mathbf{x},\sigma) \end{pmatrix} \qquad L_{xx} = D \otimes \frac{\partial^2}{\partial x^2} g(\sigma)$$

$$Interest(\mathcal{X}) = \underset{(\mathbf{x},\sigma)}{\operatorname{arglocalmax}} |\operatorname{Det}(\mathcal{H}(\mathbf{x},\sigma))|$$

A constant threshold and the non-maximal suppression within a 3×3×3×3 window are applied. Surface normal is a principal direction of the density map at that point, i.e. the surface normal is a direct generalisation of the gradient orientation used in SIFT.

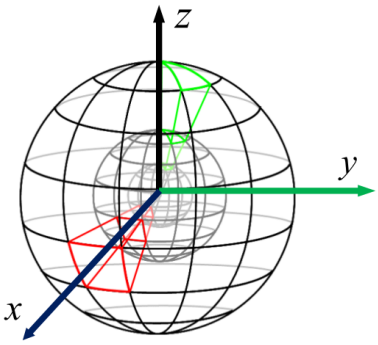
- Flint, A.; Dick, A.; Hengel; and van den, A. In *Digital Image Computing Techniques and Applications, 9th Biennial Conference of the Australian Pattern Recognition Society on*, pages 182--188, 2007

Some other noteworthy detectors Trajkovic, FAST, AGAST, NARF:

- M. Trajkovic and M. Hedley, “Fast corner detection,” Image and Vision Computing, Vol. 16, No. 2, pp. 75–87, 1998.
- Edward Rosten and Tom Drummond. 2005. Fusing Points and Lines for High Performance Tracking. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2* (ICCV '05), Vol. 2. IEEE Computer Society, Washington, DC, USA, 1508-1515. DOI: https://doi.org/10.1109/ICCV.2005.104
- Edward Rosten and Tom Drummond. 2006. Machine learning for high-speed corner detection. In *Proceedings of the 9th European conference on Computer Vision - Volume Part I* (ECCV'06), Aleš Leonardis, Horst Bischof, and Axel Pinz (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 430-443. DOI=http://dx.doi.org/10.1007/11744023_34
- Elmar Mair, Gregory D. Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. 2010. Adaptive and generic corner detection based on the accelerated segment test. In *Proceedings of the 11th European conference on Computer vision: Part II* (ECCV'10), Kostas Daniilidis, Petros Maragos, and Nikos Paragios (Eds.). Springer-Verlag, Berlin, Heidelberg, 183-196.
- Steder, Bastian., Rusu, Radu Bogdan., Konolige, Kurt., and Burgard, Wolfram, NARF: 3D Range Image Features for Object Recognition, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2010

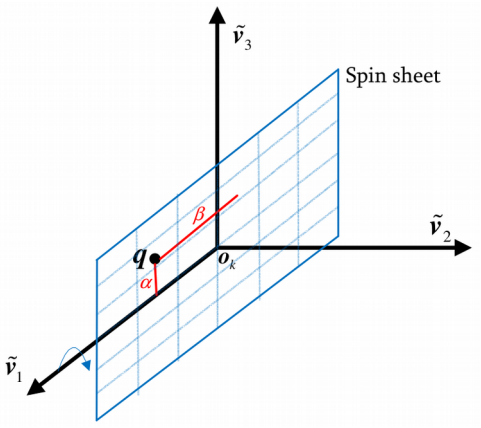
Appendix B: **Descriptors Based on Spatial Histograms**

Unique Shape Context (USC): Improves its predecessor *3D Shape Context* by introducing a full LRF. The neighbourhood is simply divided into bins, and the histograms **Tri-Spin-Image (TriSI):**of occurrence of points in particular bins is created



Federico Tombari, Samuele Salti, and Luigi Di Stefano. 2010. Unique shape context for 3d data description. In *Proceedings of the ACM workshop on 3D object retrieval* (3DOR '10). ACM, New York, NY, USA, 57-62. DOI=http://dx.doi.org/10.1145/1877808.1877821

Tri-Spin-Image (TriSI): Improves its predecessor *Rotation Projection Statistics* by introducing a full LRF. For each axis of LRF, the following is repeated: For each $\boldsymbol{q}$ from the neigbourhood of the key point, determine α and β. Collect the values of α and β in a two-dimensional ($B \times B$) histogram. Three histograms are obtained in this way that are then compressed by PCA.



$$\alpha = \sqrt{\|\boldsymbol{q} - \boldsymbol{o}_k\|^2 - (\tilde{\boldsymbol{v}}_1 \cdot (\boldsymbol{q} - \boldsymbol{o}_k))^2},$$

$$\beta = \tilde{\boldsymbol{v}}_1 \cdot (\boldsymbol{q} - \boldsymbol{o}_k).$$

Guo, Yulan & Sohel, Ferdous & Bennamoun, Mohammed & Lu, Min & Wan, Jianwei. (2013). TriSI: A Distinctive Local Surface Descriptor for 3D Modeling and Object Samuele Salti, Federico Tombari, Luigi Di Stefano, SHOT: Unique signatures of histograms for surface and texture description, Computer Vision and Image Understanding, Volume 125, 2014, Pages 251-264, ISSN 1077-3142, https://doi.org/10.1016/j.cviu.2014.04.011.Recognition.

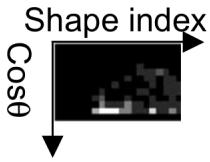
Guo, Y., Sohel, F., Bennamoun, M. et al. Rotational Projection Statistics for 3D Local Surface Description and Object Recognition, Int J Comput Vis (2013) 105: 63. https://doi.org/10.1007/s11263-013-0627-y

Appendix C: Descriptors Based on Normals/Curvatures

Local Surface Patch (LSP): It is based on fitting the quadratic surface to a window of points in a depth map (a certain restriction can be seen here), which is followed by computing the main curvatures. A value, called shape index (S_i), is introduced and used for detecting the keypoints and creating the descriptors:

$$f(x,y) = ax^2 + by^2 + cxy + dx + ey + f$$

$$S_i(p) = \frac{1}{2} - \frac{1}{\pi} \tan^{-1} \frac{k_1(p) + k_2(p)}{k_1(p) - k_2(p)}.$$

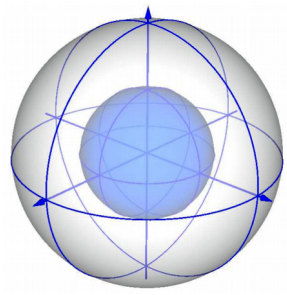


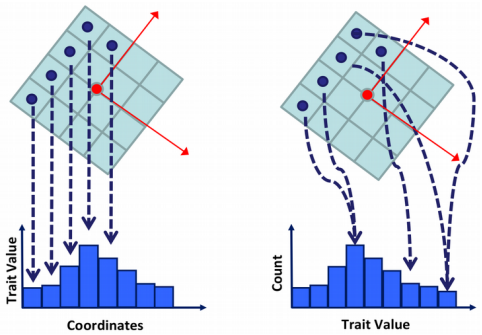
For creating the descriptors of a key point, the neigbours are used that are not too distant and not too different in the normal direction (θ). A two dimensional histogram is created based on S_i and θ.

Hui Chen, Bir Bhanu, 3D free-form object recognition in range images using local surface patches, Pattern Recognition Letters, Volume 28, Issue 10, 2007, Pages 1252-1262, ISSN 0167-8655, https://doi.org/10.1016/j.patrec.2007.02.009.

Signature of Histograms of Orientations (SHOT): LRF is determined as usual (a weight is introduced, R stands for the radius of neighbourhood, d_i is the distance from the keypoint). An algorithm based on the use of several points closest to the median distance is proposed to disambiguate the x and z axes (see the paper).

$$\mathbf{M} = \frac{1}{\sum_{i:d_i \le R} (R - d_i)} \sum_{i:d_i \le R} (R - d_i)(\mathbf{p}_i - \mathbf{p})(\mathbf{p}_i - \mathbf{p})^T$$





The neigbourhood is divided into sub-volumes (8 azimuth, 2 elevation, and 2 radial divisions). For each sub-volume, the histogram of differences between the normal orientation at the keypoint and point is created (11 bins). The histograms from the particular volumes are then concatenated into the signature. A colour version also exists.

Samuele Salti, Federico Tombari, Luigi Di Stefano, SHOT: Unique signatures of histograms for surface and texture description, Computer Vision and Image Understanding, Volume 125, 2014, Pages 251-264, ISSN 1077-3142, https://doi.org/10.1016/j.cviu.2014.04.011.