2D FEATURES-BASED DETECTOR AND DESCRIPTOR SELECTION SYSTEM FOR HIERARCHICAL RECOGNITION OF INDUSTRIAL PARTS

Ibon Merino[1], Jon Azpiazu[1], Anthony Remazeilles[1], and Basilio Sierra[2]

[1]Industry and Transport, Tecnalia Research and Innovation, Donostia-San Sebastian, Spain
{ibon.merino, jon.azpiazu, anthony.remazeilles}@tecnalia.com
[2]Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, Donostia-San Sebastian, Spain
b.sierra@ehu.eus

ABSTRACT

Detection and description of keypoints from an image is a well-studied problem in Computer Vision. Some methods like SIFT, SURF or ORB are computationally really efficient. This paper proposes a solution for a particular case study on object recognition of industrial parts based on hierarchical classification. Reducing the number of instances leads to better performance, indeed, that is what the use of the hierarchical classification is looking for. We demonstrate that this method performs better than using just one method like ORB, SIFT or FREAK, despite being fairly slower.

KEYWORDS

Computer vision, Descriptors, Feature-based object recognition, Expert system

1. INTRODUCTION

Object recognition is an important branch of computer vision. Its main idea is to extract important data or features from images in order to recognize which object is present on it. Many different techniques are used in order to achieve this. In recent computer vision literature, it has been a widely spread tendency to use deep learning due to their benefits throwing out many techniques of previous literature that, actually, have a good performance in many cases. Our aim is to recover those techniques in order to boost them and increase their performance or use their benefits that neural networks may not have.

The classical methods in computer vision are based in pure mathematical operations were images are used as matrices. These methods look for gradient changes, patterns... and try to find similarities in different images or build a machine learning model to try to predict the objects that are present in the image.

Our use case is the industrial area were many similar parts are to be recognized. Those parts vary a lot from one to another (textures, size, color, reflections,...) so an expert is needed for choosing which method is better for recognizing the objects. We propose a method that simulates the expert role. This is achieved learning a model that classifies the objects in groups that behave similarly to different recognition methods. This leads to a hierarchical classification that first classifies the object to be recognized in one of the previously obtained groups and inside the group the method that works better in that group is used to recognize the object.

DOI: 10.5121/ijaia.2019.10601

The paper is organized as follows. In Section 2 we present a state of art of the most used 2D feature-based methods, including detectors, descriptors and matchers. The purpose of Section 3 is to present the method that we propose and how we evaluate it. The experiments done and their results are shown in section 4. Section 5 summarizes the conclusions that can be drawn from our work.

2. BACKGROUND

There are several methods for object recognition. In our case, we have focused on feature-based methods. These methods look for points of interest of the images (detectors), try to describe them (descriptors) and match them (matchers). The combination of different detectors, descriptors and matchers vary the perfomance of the whole system. This is a fast growing area in image processing field. The following short and chronologically ordered review presents the gradual improvements in feature detection (Subsection 2.1), description (Subsection 2.2) and matching (Subsection 2.3).

2.1. 2D features detectors

One of the most used methods was proposed in 1999 by Lowe [14]. This method is called SIFT, which stands for Scale Invariant Feature Transform. The main idea is to use the Difference-of-Gaussian function (a close approximation to the Laplacian-of-Gaussian proposed by Lowe) to search for extrema in the scale space. Even if SIFT was relatively fast, a new method, SURF (Speeded Up Robust Features) [3], outperforms it in terms of repeatability, distinctiveness and robustness, although it can be computed and compared much faster.

In addition, FAST (Features from Accelerated Segment Test) [25] proposed by Rosten and Drummond introduce a fast detector. FAST outperforms previous algorithms (like SURF and SIFT) in both computational performance and repeatability. AGAST [17] is based on the FAST, but it is more efficient as well as generic. BRISK [12] is a novel method for keypoint detection, description and matching which has a low computational cost (as stated in the corresponding article, an order of magnitude faster than SURF in some cases). Following the same line of FAST based mehods, we find ORB Rublee et al. [26], an efficient alternative to SIFT or SURF. This method's detector is based on FAST but it adds orientation in order to obtain better results. In fact, this method performs at two orders of magnitude faster than SIFT, in many situations.

Figure 1 shows some detectors and the relation between them chronologically ordered.

2.2. 2D features descriptors

Lowe also proposed a descriptor called SIFT. As mentioned above, is one of the most popular feature detector and descriptor. The descriptor is a position-dependent histogram of local image gradient directions around the interest point and is also scale invariant. It has numerous extensions such as PCA-SIFT [10], that mixes PCA with SIFT; CSIFT [1], Color invariant SIFT; GLOH [18]; DAISY [27], a dense descriptor inspired in SIFT and GLOH; and so on. SURF descriptor [3] relies on integral images for image convolutions in order to obtain its speed.

BRIEF [4] is a highly discriminative feature descriptor that is fast both to build and to match. BRISK [12] descriptor is composed as a binary string by concatenating the results of simple brightness comparison tests. ORB descriptor is BRIEF-based and adds rotation invariance and resistance to noise.

LBP (Local Binary Patterns) [22] is a two-level version of the texture spectrum method [28]. This methods has been really popular and many derivatives has been proposed. Based on this, the CS-LBP (Center-Symmetric Local Binary Pattern) [8] combines the strengths of SIFT and LBP. Later

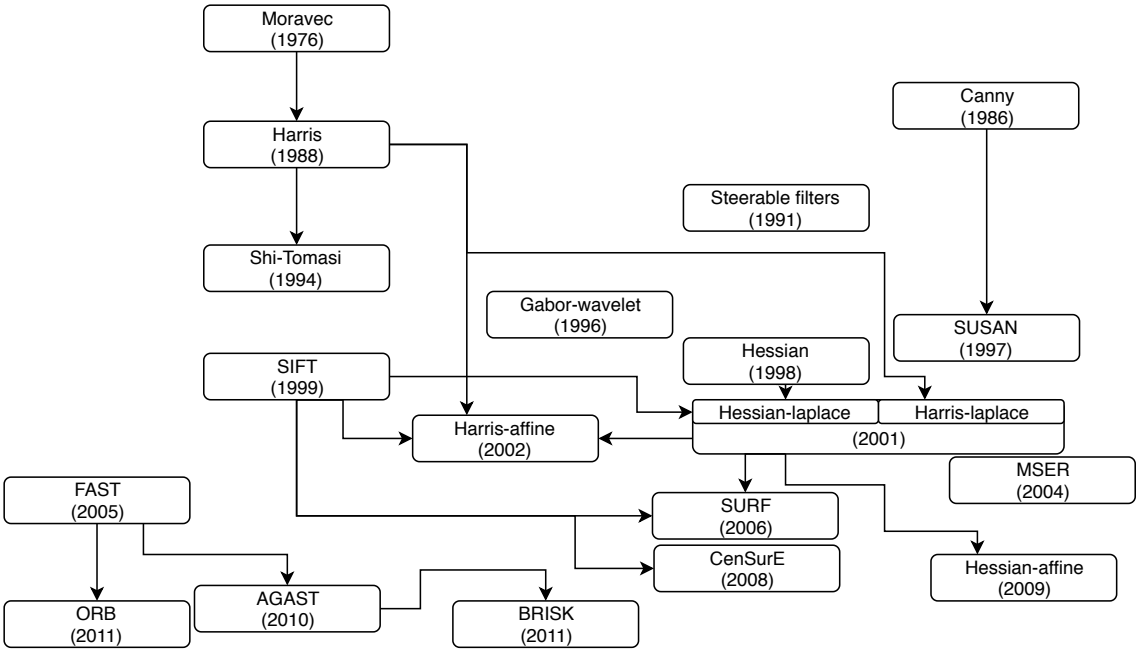

Figure 1: Recognition pipeline

in 2010, the LTP (Local Ternary Pattern) [13] appeared, a generalization of the LBP that is more discriminant and less sensitive to noise in uniform regions. Same year, ELTP (Extended local ternary pattern) [21] improved this by attempting to strike a balance by using a clustering method to group the patterns in a meaningful way. In 2012, LTrP (Local Tetra Patterns) [20] encoded the relationship between the referenced pixel and its neighbors, based on the directions that are calculated using the first-order derivatives in vertical and horizontal directions. In [23] there are gathered other methods that are based on the LBP.

MTS (Modified texture spectrum) proposed by Xu et al. [29] can be considered as a simplified version of LBP, where only a subset of the peripheral pixels (up-left, up, up-right and right) is considered.

The Binary Gradient Contours (BGC) [6] is a binary 8-tuple proposed by Fernndez et al. The simple loop form (BGC1) makes a closed path around the central pixel computing a set of eight binary gradients between pairs of pixels.

Other descriptor called FREAK [2] is a keypoint descriptor inspired by the human visual system and more precisely the retina. It is faster, usess less memory and more robust than SIFT, SURF and BRISK. They are thus competitive alternatives to existing descriptors in particular for embedded applications.

Figure 2 shows some detectors and the relation between them chronologically ordered.

2.3. Matchers

The most widely used method for matching is Nearest Neighbor (NN). Many algorithms follow this method. One of the most used is the kd-tree [24] which works well with low dimensionality. For dealing with higher dimensionalities many researchers have proposed diverse methods such as the Approximate Nearest Neighbor (ANN) by Indyk and Motwani [9] or the Fast Approximate Nearest Neighbors of Muja and Lowe [19] which is implemented in the well known open source library FLANN (Fast Library for Approximate Nearest Neighbors).

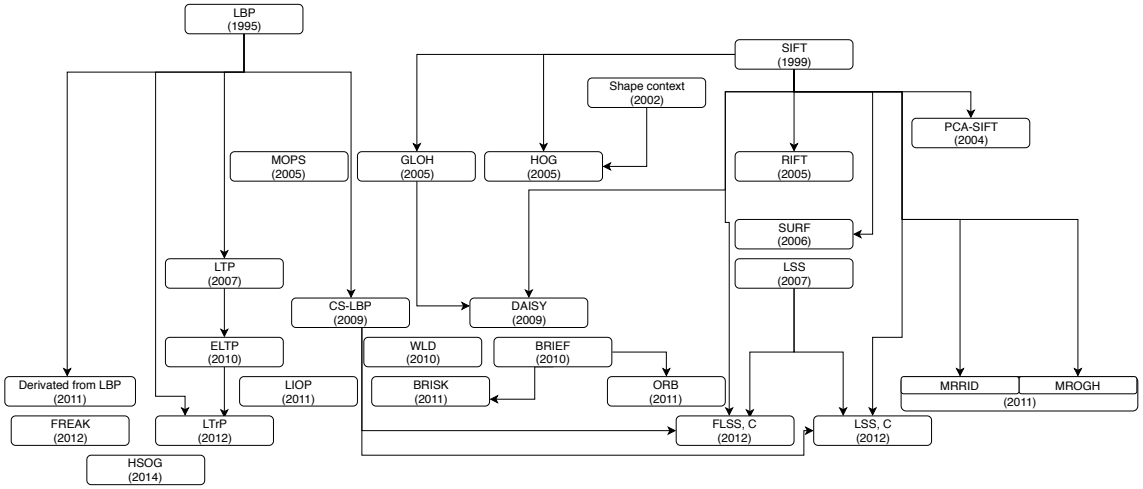

Figure 2: Recognition pipeline

3. PROPOSED APPROACH

As we have stated before, the issue we are dealing with is the recognition of industrial parts for pick-and-placing. The main problem is that the accurate recognition of some kind of parts are highly dependant on the recognition pipeline used. This is because parts' characteristics like texture (presence or absence), forms, colors, brightness; make some detectors or descriptors work differently. We are thus proposing a systematic approach for selecting the best recognition pipeline for a given object (Subsection 3.2). We also propose in Subsection 3.3 an expert system that identifies groups of parts that are recognized similarly to improve the overall accuracy. The recognition pipeline is explained in Subsection 3.1.

We start defining some notations. An industrial part, or object, is named **instance**. The images captured of each part are named **views**. Given the set of views $\boldsymbol{X}$, the set of instance labels $\boldsymbol{Y}$ and the set of recognition pipelines $\boldsymbol{\Psi}$, the function $\omega^{\Psi}_{X,Y}(y)$ returns for each $y \in \boldsymbol{Y}$ the best pipeline $\psi* \in \boldsymbol{\Psi}$ according to a metric F_1 that is later discussed. We call ψ^{**} to the pipeline that on average performs better according to the evaluation metric, this is, that maximizes the average of the scores per instance (2).

$$\omega^{\Psi}_{X,Y}(y) = \underset{\psi \in \Psi}{\operatorname{argmax}} F1^{\psi}_{y}(X,Y) = \psi^{*} \tag{1}$$

$$\psi^{**} = \underset{\psi \in \Psi}{\operatorname{argmax}} \frac{\sum_{y \in Y} F1^{\psi}_{y}(X,Y)}{|Y|} \tag{2}$$

3.1. Recognition Pipeline

A recognition pipeline Ψ is composed of 3 steps: detection, description and matching. Detectors, $\boldsymbol{\Gamma}$, localize interesting keypoints in the view (gradient changes, changes in illumination,...). Descriptors, $\boldsymbol{\Phi}$, are used to represent those keypoints in order to locate them in other views. Matchers, $\boldsymbol{\Omega}$, find the closest features between views. So, a pipeline ψ is composed by a keypoint detector γ, a feature descriptor ϕ and a matcher ω. Figure 3 shows the structure of the recognition pipeline.

The keypoints detection and description are described previously in the background section. In the matching, are two groups of features: the ones that form the model (train) and the ones that

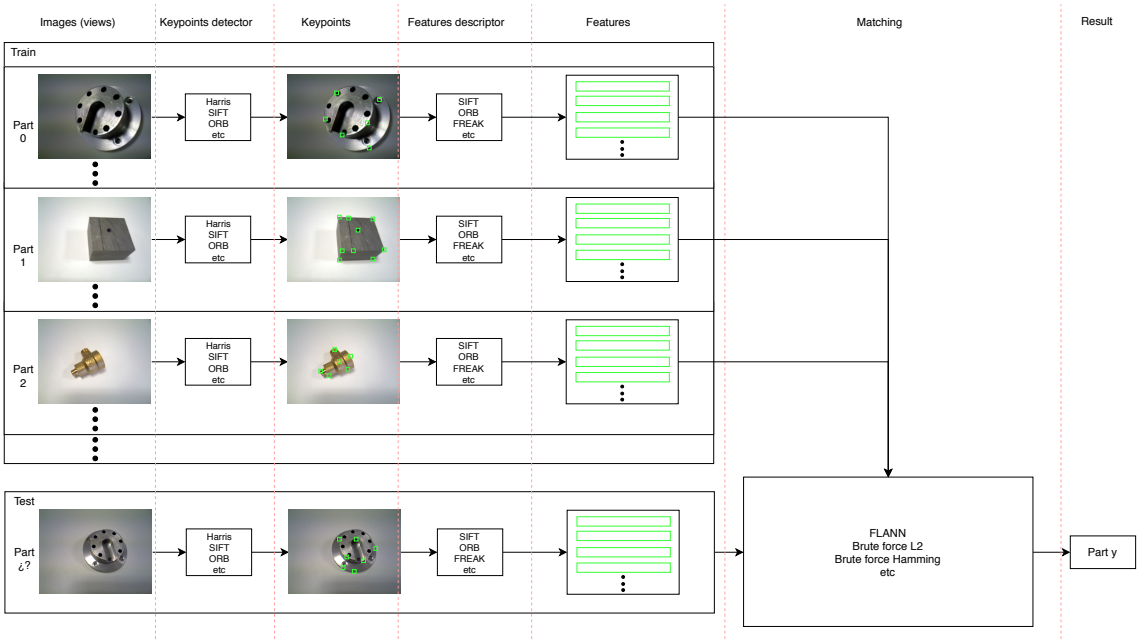

Figure 3: Recognition pipeline

need to be recognized (test). Different kind of methods could be used to match features, but, mainly, distance based techniques are used. This techniques make use of different distances (L2, hamming,...) to find the closest feature to the one that needs to be labeled. Those two features (the test feature and the closest to this one) are considered a match. In order to discard ambiguous features, we use the Lowe's ratio test [15] to define whether two features are a "good match". Assuming f_t is the feature to be recognized, and f_{l1} and f_{l2} its two closest features from the model, then (f_t, f_{l1}) is a good match if:

$$\frac{d(f_t, f_{l_1})}{d(f_t, f_{l_2})} < r \tag{3}$$

where $d(f_A, f_B)$ is the distance (Euclidean or L2 distance: Equation 4, Hamming distance: Equation 5, where δ is the kronecker delta (Equation 6),...) between features A and B, and r is a threshold that is used to validate if two features are similarly close to the test feature and discard it. This threshold is set at 0.8. Now a simple voting system is used for labeling the view. For each view from the model (train) the number of good matches are counted. The good matches of each instances are summed and the test view is labeled as the instance with more good matches.

$$d_E(P, Q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{4}$$

$$d_H(P, Q) = \sum_{i=1}^{n} \delta(p_i, q_i) \tag{5}$$

$$\delta(p_i, q_i) = \begin{cases} 0 & if \quad x_i = y_i \\ 1 & if \quad x_i \neq y_i \end{cases} \tag{6}$$

Table 1: Example of a confusion matrix for 3 instances.

		Actual instance			
		object 1	object 2	object 3	
Predicted instance	object 1	40	10	0	50
	object 2	0	30	25	55
	object 3	10	10	25	45
		50	50	50	150

3.2. Recognition Evaluation

As we have said, we have the input views X, the instance labels Y and the pipelines Ψ. To evaluate the pipelines we have to separate the views in train and test. The evaluation method used for it is Leave-One-Out Cross-Validation (LOOCV) [11]. It consists of $|X|$ iterations, that for each iteration i, the train dataset is $(X - x_i)$ and the test sample is x_i. With this separation train-test we can generate the confusion matrix. Table 1 is an example of a confusion matrix for 3 instances.

As mentioned in the introduction of Section 3, we use the metric F_1 value [7] for scoring the performance of the system. The score is calculated for the tests views from the LOOCV. F_1 score, or value, is calculated per each instance (7). This metric is an harmonic mean between the precision and the recall. The mean of all the F_1's, $\bar{F_1}$ (8) is used for calculating ψ^{**}.

$$F_1(y) = 2 \cdot \frac{precision_y * recall_y}{precision_y + recall_y} \tag{7}$$

$$\bar{F_1} = \frac{\sum_{y \in Y} F_1(y)}{|Y|} \tag{8}$$

The precision (Equation 9) is the ratio between the correctly predicted views with label y (tp_y) and all predicted views for that given instance ($|\psi(X) = y|$). The recall (Equation 10), instead, is the relation between correctly predicted views with label y (tp_y) and all views that should have that label ($|label(X) = y|$).

$$precision_y = \frac{tp_y}{|\psi(X) = y|} \tag{9}$$

$$recall_y = \frac{tp_y}{|label(X) = y|} \tag{10}$$

3.3. Expert system

The function ω gives a lot of information about objects but it needs the instance to return the best pipeline for that instance which is not available a priori. Indeed, this is what we want to identify. We use the information that would provide ω to build a hierarchical classification based in a clustering of similar objects.

Since some parts work better with some particular pipelines because of their shape, color or texture, we try to take advantage of this and make clusters of objects that are classified similarly well by each pipeline. For example, two parts that have textures may be better recognized by pipelines that use descriptors like SIFT or SURF rather than non textured parts. We call these clusters typologies. This clustering is made using the algorithm K-means [16], that aims to partition the

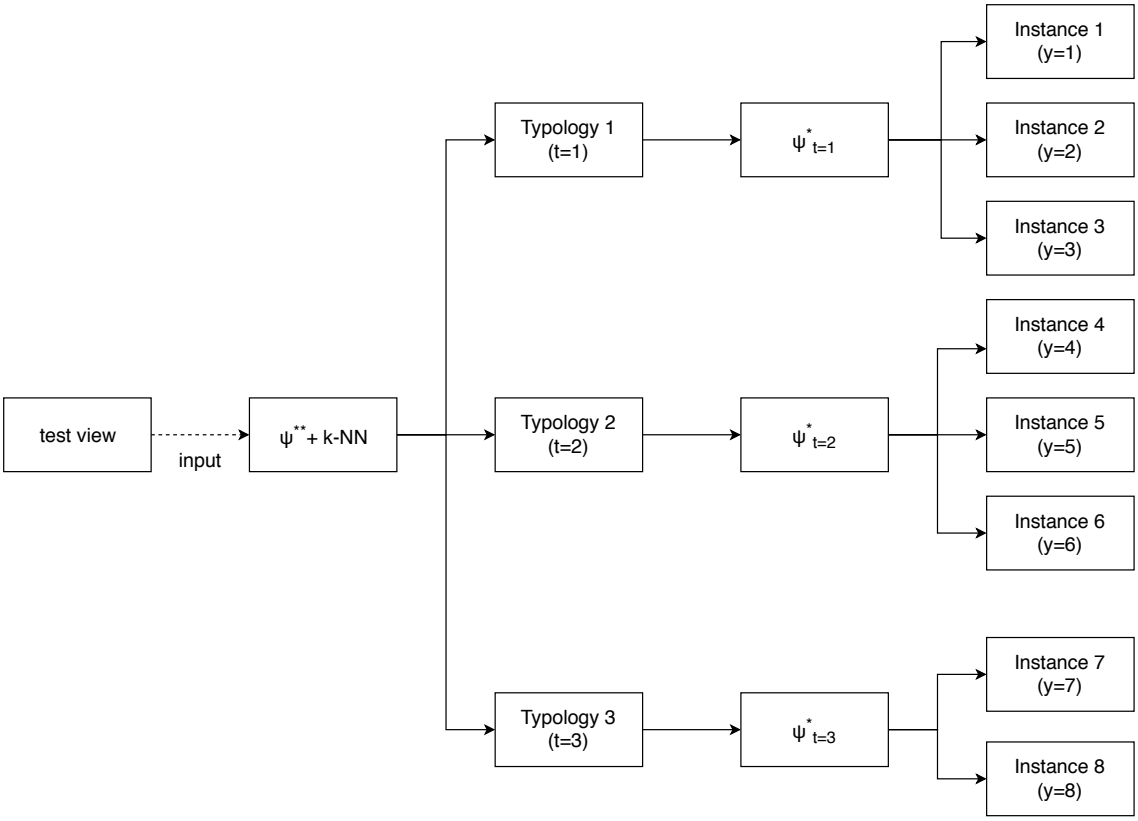

Figure 4: Hierarchical classification

objects into K clusters (where $K < |Y|$) in which each object belongs to the cluster with the nearest centroids. The input is a matrix with the instances as rows and for each row the F_1 value of each pipeline. The inputs for this algorithm are for each instance an array of the F_1 value obtained with every pipeline. The election of a good K may highly vary the result since if almost all the clusters are composed by 1 instance the result would be close to just using ψ^{**}. After obtaining the K typologies, the ψ_T^*'s (11) are calculated, i.e., the best pipeline for each typology.

$$\psi_T^* = \underset{\psi \in \Psi}{\operatorname{argmax}} \frac{\sum_{y \in T} F_1{}_y^{\psi}(X, Y)}{|T|} \tag{11}$$

The first step of the hierarchical recognition is to recognize the typology with the ψ^{**}. Given the typology t as the typology predicted, the ψ_t^* is used to recognize the instance y of the object. We call the hierarchical recognition Υ. The Figure 4 shows an scheme of the hierarchical recognition for clarification.

4. EXPERIMENTS AND RESULTS

Our initial hypothesis is that Υ has a better performance than ψ^{**}. In order to demonstrate this hypothesis we conducted some experiments. Moreover, we want to know in which way does the number of parts and the number of views per part affect the result.

The pipelines used (detector, descriptor and matcher) are defined in Subsection 4.1. In Subsection 4.2, we explain the dataset we have created to evaluate the proposed method under the use case that is the industrial area and the results obtained. In order to compare these results with a well-known dataset in Subsection 4.3 we present the Caltech dataset [5] and the results obtained.

Table 2: Pipelines composition.

Pipeline	Detector	Descriptor	Matcher
ψ_0	SIFT	SIFT	FLANN
ψ_1	SURF	SURF	FLANN
ψ_2	ORB	ORB	Brute force Hamming
ψ_3	—-	LBP	FLANN
ψ_4	SURF	BRIEF	Brute force Hamming
ψ_5	BRISK	BRISK	Brute force Hamming
ψ_6	AGAST	DAISY	FLANN
ψ_7	AGAST	FREAK	Brute force Hamming

Table 3: F_1's of the ψ^{**}'s and Υ for each subset of our dataset. p stands for number of parts and t for number of pictures per part.

p \ t	10		20		30		40		50	
	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ
3	**0.935**	0.862	0.967	**0.983**	0.989	**1**	0.992	**1**	0.993	**1**
4	**0.899**	0.854	0.924	**0.962**	0.932	**0.966**	**0.944**	0.801	**0.91**	0.865
5	**0.859**	0.843	**0.868**	0.863	**0.883**	0.818	0.876	**0.901**	0.87	**0.912**
6	0.865	**0.967**	0.873	**0.992**	**0.891**	0.87	0.88	0.88	0.856	**0.901**
7	0.872	**0.9**	0.886	**0.986**	**0.894**	0.891	0.88	**0.876**	0.845	**0.94**

4.1. Pipelines

The pipelines we have selected are shown in Table 2. Many combination could be done but it is not consistent to match binary descriptors with a L2 distance. The combinations chosen are compatible and may not be the best combination. Global descriptors, such as LBP, does not need a detector.

4.2. Our dataset

We select 7 random industrial parts and on a white background we make 50 pictures per part from different angles randomly. That way, we have a dataset with 350 pictures. In Figure 5 are shown zoomed in examples of the pictures taken to the parts.

We use subsets of the dataset to evaluate if changing the number of views per instance and the number of instance vary the performance. This subsets have from 3 to 7 parts and from 10 to 50 views (10 views step). In Table 3 are gathered the results for all the subsets using ψ^{**} and Υ. The highest score for each subset is in bold. On average the hierarchical recognition performs better. The more parts or views per part, the better that performs the hierarchical recognition comparing with the best pipeline.

Now we focus on the whole dataset. In Figure 6 are shown the F_1's of each instance using each pipeline for this particular case. The horizontal lines mark the $\bar{F}_1$ for that pipeline. The score we obtain with our method (last column) is higher (0.94) than the best pipeline which is ψ_2 that corresponds to the pipeline that uses ORB (0.845).

Figure 5: Parts used in our dataset.

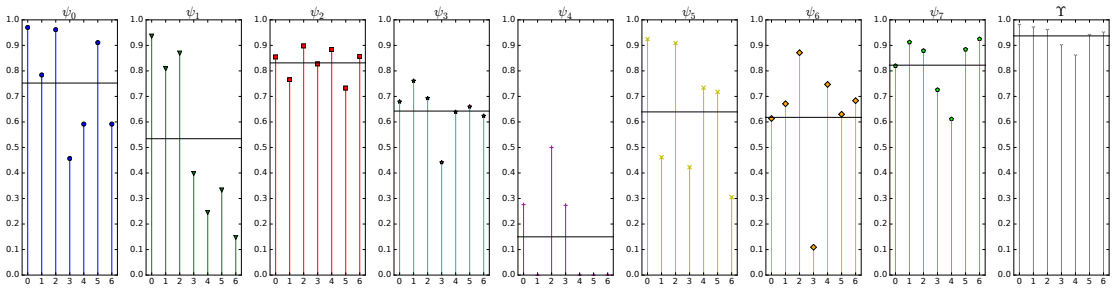

Figure 6: F_1 score for each instance and algorithm.

Table 4: Time in seconds that needs each pipeline in recognize a piece.

ψ_0	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	ψ_6	ψ_7	Υ
0.276	0.861	0.976	0.001	0.106	0.111	0.296	1.099	1.948

Figure 7: 6 random examples of images from the Caltech-101 dataset. The classes are: Face, Leopard, Motorbike, Airplane, Accordion and Anchor.

Table 5: F_1's of the ψ^{**}'s for each test (Caltech-101). p stands for number of parts and t for number of pictures per part.

p \ t	10		20		30		40		50	
	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ	ψ^{**}	Υ
3	0.967	0.967	0.983	0.983	0.989	0.989	0.992	0.992	0.993	0.993
4	0.975	0.975	0.987	0.987	0.975	0.975	0.969	0.969	0.963	**0.97**
5	0.98	0.98	0.99	0.99	0.967	0.967	0.96	0.96	0.956	0.956
6	0.883	0.883	0.907	0.907	0.883	0.883	0.848	0.848	0.851	**0.936**
7	0.776	0.776	0.794	**0.831**	0.78	**0.84**	0.768	**0.849**	0.783	**0.843**

A truthful evaluation of the time performance of the hierarchical classificator is a bit cumbersome since it directly depends on the clustering phase and on which are the best pipelines for each cluster. At least, it needs more time than just using a single pipeline. Given $t(\psi)$ the time need by the pipeline ψ, the time needed by Υ is approximately $t(\psi^{**}) + t(\psi^*_{T'})$ where T' is the typology guessed by the ψ^{**}. In Table 4 is shown the time in seconds that each pipeline and the Υ need to recognize a view.

4.3. Caltech-101 dataset

Caltech-101 dataset [5] is a known dataset for object recognition than could be similar to our dataset. This dataset has been tested like our dataset making subsets of the same characteristics. Some randomly picked images from the dataset are shown in Figure 7.

The results obtained for the subsets of this datasets are shown in Table 5. Same conclusions are obtained for this dataset.

4.4. Adding more descriptors

Due to the great performance that global descriptors have, we tried other experiments on the previous 2 datasets adding new descriptors that had been considered from the beginning. These experiments mix global descriptors and local descriptors. Local descriptors make use of the matching technique to recognise the objects in the images. Global descriptors, instead, are usually used

Table 6: F1s per instance and mean using added descriptors for our dataset

	0	1	2	3	4	5	6	mean
SIFT	0.962	0.785	0.951	0.455	0.598	0.906	0.598	0.751
SURF	0.857	0.5	0.8	0.196	0.667	0.25	0.5	0.539
ORB	0.6	0.53	0.723	0.715	0.8	0.9	0.8	0.845
BRIEF	0.287	0	0.5	0.287	0	0	0	0.153
BRISK	0.931	0.475	0.912	0.422	0.735	0.726	0.301	0.643
DAISY	0.623	0.675	0.871	0.121	0.75	0.637	0.687	0.623
FREAK	0.823	0.911	0.872	0.738	0.617	0.878	0.927	0.824
LBP	0.687	0.767	0.694	0.432	0.647	0.667	0.635	0.647
LBPU	0.106	0.4	0.5	0.333	0.75	0	0.25	0.334
BGC1	0.72	0.4	0.462	0.182	1	0.333	0.5	0.514
GRAD	0.8	0.667	0.5	0.546	0.8	1	0.889	0.743
GABORGB	0.909	1	0.667	0.8	0.889	0.889	0.8	0.851
LTP	0.667	0.572	0.6	0.285	0.8	0.25	0	0.453
MTS	0.261	0	0.445	0	0.75	0.286	0.364	0.301
TPLBP	0.667	0.667	0.889	0.667	0.857	0.8	0.933	0.783

among machine learning techniques to identify the objects. In this case, we have used a Support Vector Machine to learn the model that recognise the objects using the global descriptors.

The newly added global descriptors are: LBPU (Uniform LBP), BGC1, Gradded histogram, Gabor Filters, LTP, MTS and TPLBP (Three-Patch LBP). In Figure 6, are shown the F1s obtained using all the descriptors (the first 8 and the newly added global descriptors) for our dataset. The hierarchical classificator using all the descriptors achieves a F1 of 0.937. In Figure 7, instead are the results for the Caltech-101 dataset. The hierarchical classificator with this dataset achieves a F1 of 0.95.

Adding more descriptors without taking out others does not improve the hierarchical classificator. Even more, in some experiments done, the results are worse than just using the firstly proposed 8 methods. This is due to the fact that the typologies are chosen by clustering the instances. That does not provide the best combination of instances, but the ones that behave similarly to all the descriptors. So the more descriptors the more variability in clustering.

5. CONCLUSION

We proposed a hierarchical recognition method based in clustering similar behaviour by the recognition pipelines. It has been demonstrated that on average works better than just recognizing with classical feature-based methods achieving high F_1 Scores (in the biggest case, 0.94 for our dataset and 0.843 for the Caltech-101). The performance of the hierarchical method is highly dependant of the feature-based method used to build it. In order to improve its performance new combinations of descriptor may be proposed. Not just adding and increasing the algorithm bag, but selecting the best combination of algorithms to obtain the best performance.

As we stated, once we recognize a piece we need its pose to tell the robot where to pick it. This has been let for future work. The use of local features enables the possibility to estimate objects pose using methods such as Hough voting schema, RANSAC or PnP.

6. ACKNOWLEDGMENT

This paper has been supported by the project SHERLOCK under the European Unions Horizon 2020 Research Innovation programme, grant agreement No. 820689.

Table 7: F1s per instance and mean using added descriptors for Caltech-101

	0	1	2	3	4	5	6	mean
SIFT	0	0	0	0.5	0.667	0.461	0.2	0.261
SURF	0	0.25	0.428	0.706	0	0	0	0.197
ORB	0.693	0.753	0.723	0.715	0.859	0.944	0.8	0.783
BRIEF	0	0.125	0.286	0.8	0	0.333	0.182	0.246
BRISK	0.167	0	0.25	0.889	0.445	0.222	0.182	0.307
DAISY	0	0	0	0.25	0	0	0.222	0.067
FREAK	0	0.182	0.545	0.909	0	0.461	0.5	0.371
LBP	0.909	1	1	0.833	1	0.667	0.889	0.899
LBPU	1	1	0.889	0.8	0.923	0.933	0.857	0.914
BGC1	0.667	0.923	1	0.75	0.889	0.923	0.857	0.858
GRAD	0.75	0.857	1	0.909	1	0.857	0.667	0.862
GABORGB	1	0.72	1	0.744	1	1	0.85	0.902
LTP	0.727	0.445	1	1	0.889	1	0.286	0.763
MTS	0.857	1	0.8	1	1	1	0.5	0.879
TPLBP	1	1	1	0.667	0.857	0.857	0.667	0.864

7. REFERENCES

[1] A. E. Abdel-Hakim and A. A. Farag. Csift: A sift descriptor with color invariant characteristics. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1978–1983. Ieee, 2006.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517. IEEE, June 2012.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *Computer Vision ECCV 2006*, pages 404–417. 2006.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792, 2010.

[5] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59 – 70, 2007.

[6] A. Fernndez, M. X. lvarez, and F. Bianconi. Image classification with binary gradient contours. *Optics and Lasers in Engineering*, 49(9-10):1177–1184, 2011.

[7] C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European Conference on Information Retrieval*, pages 345–359, 2005.

[8] M. Heikkil, M. Pietikinen, and C. Schmid. Description of interest regions with local binary patterns. *Pattern Recognition*, 42(3):425–436, March 2009.

[9] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*.

[10] Y. Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages 506–513. IEEE, 2004.

[11] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[12] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary Robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, November

2011.

[13] W. Liao. Region Description Using Extended Local Ternary Patterns. In *2010 20th International Conference on Pattern Recognition*, pages 1003–1006, August 2010.

[14] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, September 1999.

[15] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[16] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297. University of California Press, 1967.

[17] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European conference on Computer vision*, pages 183–196. Springer, 2010.

[18] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[19] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP*, 2(331-340):2, 2009.

[20] S. Murala, R. P. Maheshwari, and R. Balasubramanian. Local Tetra Patterns: A New Feature Descriptor for Content-Based Image Retrieval. *IEEE Transactions on Image Processing*, 21 (5):2874–2886, May 2012.

[21] L. Nanni, S. Brahnam, and A. Lumini. A local approach based on a local binary patterns variant texture descriptor for classifying pain states. *Expert Systems with Applications*, 37 (12):7888–7894, 2010.

[22] T. Ojala, M. Pietikinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, January 1996.

[23] M. Pietikinen, A. Hadid, G. Zhao, and T. Ahonen. Local Binary Patterns for Still Images. In *Computer Vision Using Local Binary Patterns*, Computational Imaging and Vision, pages 13–47. Springer London, 2011.

[24] John T. Robinson. The k-d-b-tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, SIGMOD '81, pages 10–18, 1981.

[25] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 1508–1515 Vol. 2. IEEE, 2005.

[26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, November 2011.

[27] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5): 815–830, May 2010.

[28] L. Wang and D.C. He. Texture classification using texture spectrum. *Pattern Recognition*, 23(8):905–910, 1990.

[29] B. Xu, P. Gong, E. Seto, and R. Spear. Comparison of Gray-Level Reduction and Different Texture Spectrum Encoding Methods for Land-Use Classification Using a Panchromatic Ikonos Image. *Photogrammetric Engineering & Remote Sensing*, 69(5):529–536, May 2003.