THE ANNALS OF „DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE XIV MECHANICAL ENGINEERING, ISSN 1224-5615
2011

ALGORITHMS FOR OBJECTS' SHAPE RECOGNITION AND MEASUREMENT BEFORE GRIPPING

Prof.Dr.Eng.Staretu Ionel
Dr.Inf.Itu Alexandru
Drd. Inf. Moldovan Catalin
"Transilvania" University of Brasov

ABSTRACT

The paper proposes an algorithmic methodology to solve a necessary first step, namely visualization, recognition and measurement of objects, for gripping, in particular by means of anthropomorphic grippers for robots. For object shape recognition, Haar classifiers, and an Open CV library use are proposed. Based on measurement of the distance between the initial point and a second intermediate point, you can do calculations, aimed at object size, and distance remaining to the target. Thus, the algorithmic method for visualization, recognition, and measurement of objects for anthropomorphic gripping, is based on a number of existing software modules. They have been adapted for the intended purpose, and specially designed software modules were added.

KEYWORDS: image processing, Haar identifier, shape recognition, size measurement

1. INTRODUCTION

In order to simplify and streamline the gripping through anthropomorphic grippers, an algorithm method, to solve a necessary first step, namely visualization, recognition and measurement of objects to grip, is proposed.Shape recognition, in the case of the objects viewed, is possible using Haar classifiers adjustment. Based on measurements of the distance between the initial point and a second intermediate point, you can do calculations, aimed at object size, and distance remaining to the target.

Next, 3D shape of the object is generated, then the gripper approaches it, and its displacement is measured. This information is further transferred to the pre-configuration module. The shape of the object will determine the gripper pre-configuration, and the size of the object is the gripper aperture [1,3,4,6].

In normal situations, objects shape is often complex. For the research that is envisaged in this paper, we will start from four basic shapes that the visualization module will be able to recognize (parallelepipeds, spheres, cones and cylinders).

Thus, a library will be structured for the four basic shapes, which can solve the problem of gripper pre-configuration. Specifically, this paper presents the shape recognition steps for objects to grip, and these objects size measurement steps.

2. OBJECT RECOGNITION ALGORITHMS IMPLEMENTATION

2.1 Object Recognition using Haar Classifiers

In image processing, object recognizing means the presence of an object belonging to a certain predefined class, within a static frame (image), or in a video frame (streaming video).

Objects can be classified into simple and complex objects. For simple objects, recognition of objects is equivalent to determining the edges (edge detection), of specific color regions, textures, shapes, etc. In the case of complex objects, such as human face, their detection is done using different

methods based on learning specific features (classifiers).

Viola and Jones [5] proposed a new approach for object recognition, using a cascade of classifiers to describe the object to be recognized. The algorithm proposed by the two researchers aimed, firstly, at human face detecting, but classifiers can describe other classes of objects that can be recognized as well.

The classifier is thus the basic unit of object detection, and it is called Haar type classifier (Haar-like feature), being similar to a Haar-type function, fig. 1, 2 and 3.

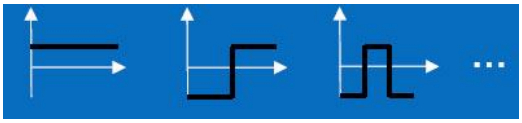

Fig. 1 Haar- type function

Fig. 2 Haar -type classifiers

Classifiers calculation is as follows:

$$feature_{i,k} = W_{i,k,1} * RectSum_{i,k,\,orange+white}\,(I) + W_{i,k,2} * RectSum_{i,white}\,(I) \quad (1)$$

where:

$$RectSum(r)=SAT(x-1, y-1) + SAT(x+w-1, y+h-1) - SAT(x-1, y+h-1) - SAT(x+w-1, y-1)$$

$$SAT(x, y) = \sum_{x'\le x, y'\le y} l(x', y')$$

w –region length (r)

l – region width(r)

Associated weight ($W_{i,\,j,\,k}$) is then compensated:

$$W_{i,k,1} * Area_{i,k,\,portocaliu+alb} + W_{i,k,} * Area_{i,k,+alb}\,(I) = 0 \quad (2)$$

Fig. 3 A classifier calculation

2.2. Open CV Library

To implement the object recognition software module, an Open CV image processing library has been used. In programming, a library is a collection of routines that have already been tested, and put into a format that allows their use in other applications too, with no need of rewriting them.

OpenCV library is free for both commercial and research use. The library is designed to run on multiple operating systems: Mac OS X, Windows and Linux [2].

The main objective of the OpenCV library is to achieve real-time processing, both image and frame, from a video sequence. For this reason, image processing algorithms are optimized by the experts from Intel. OpenCV contains over 400 functions, covering 28 areas of research in image processing and analysis, thus being highly flexible.

The content of the library is created as follows: when a researcher has discovered a new algorithm, he sends it to Intel, where it is analyzed, optimized, and adapted to take advantage of special support processors for MMX multimedia (Matrix Math eXtension), and SSE (Streaming SIMD Extensions). That algorithm is to be included in the next version of the library, which will be launched. Processors that are currently marketed benefit from MMX and SSE technology, and therefore OpenCV is a very good choice for analysis and image processing, due to its optimization for the hardware structure of the processors.

OpenCV Library was launched in 2001, and since then it has grown to encompass all known image processing algorithms, providing a complete tool for creating applications in this area.

OpenCV library has the following features:

• C / C + + library free for image processing.
• It is optimized and developed for real-time applications.
• It does not a operating system dependent.
• Generic functionality for loading, saving and getting pictures/video.
• API (Application Programming Interface) support for programming, low and high level.
• Provides Intel Integrated Performance Primitives i nterface (API) with specific optimizations at the processor level.

The main features of the OpenCV library are:

• Handling an image data (allocation, de-allocation, copying, saving, conversion).
• Images and video stream Input / Output (both file-level and device-level - web-cam, camcorders, digital cameras).

• Support for matrix and vector operations (scalar product, vector product, vector suite, etc.).
• Various dynamic structures (lists, queues, trees, graphs).
• Algorithms and tools for image processing (filtering, contour detection, corner detection, color conversion, interpolation, morphological operations, operations on the histogram, image segmentation, support for defining areas of interest (ROI) within images, calibration tools for video input devices, tools for image recognition by classifiers, support for stereo images, neural network support).
• Structural analysis (connected components, contour processing, distance transformations, variable times, assembling mold / template, Hough-type transformation, polygonal approximating, approximating areas of interest by lines, approximating areas of interest by ellipse, Delaunay triangulation).
• Cameras calibration (setting and following patterns, calibration, fundamental matrix estimation, stereo correspondence).
• Analysis of movement (motion tracking in a video sequence, motion segmentation).
• Recognition of objects (methods based on proper value, HMM (Hidden Markov Model)). Simple Interface GUI - Graphics User Interface (display image / window level video, handlers of events from the mouse and keyboard, scroll bars).
• Ability to overlap layers and images (lines, polygons, text).
The main OpenCV modules are:

- *cv.dll* – contains the OpenCV functions.
- *cvaux.dll* – contains a series of experimental OpenCV functions.
- *cxcore.dll* – includes support for data structure and linear algebra.
- *highgui.dll* – contains specific functions for GUI - visual interface of the application.

OpenCV Library offers a number of useful tools for classifiers creation and testing, but the longest process is pictures gathering (negative, positive) and object separation from the background. To alleviate the first two steps, the author has made two helping applications that greatly simplify the process of collecting useful information that can later be used for training the classifier. The first application is called **PrintScreener** (fig. 4), its aim being to make one screen shot each 30-millisecond interval and save the capture in a default directory.

Training pictures collection should be grouped into positive and negative. In the case of positive image collection, the object that wants to be recognized must appear at least once. However, to train the name of the picture containing the object,

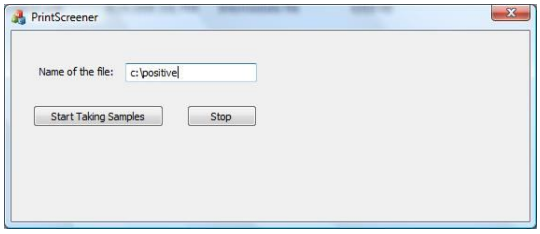

Fig. 4 PrintScreener tool

and the rectangle (area of interest), the photo that contains the object (fig. 5) is necessary. To solve this problem, and optimize the process, the author has made a second tool, **ObjectMarker,** which allows the user to make the selection with the mouse so as to fit the target object, and save the information needed to file (the file name with coordinates of four points defining the rectangle which frames the object).

Besides the two directories that contain negative and positive collection of pictures before training, you need to create a file containing their names. The reason is simple; for a well-trained classifier, at least 1000 positive, and 3000 negative pictures are necessary. All this information could not be stored in the RAM of the computer, and this is why the pictures are loaded in turn, by calling **path + their name**.

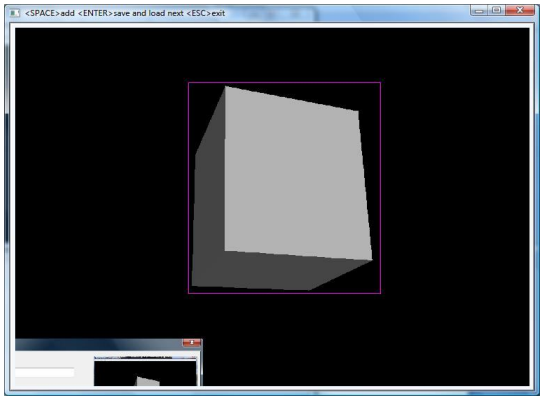

Fig. 5 ObjectMarker tool

Once the collection of pictures made, with the files that describe the photo name, and areas there is object to be recognized , step 3, training, begins.

Two tools that come with OpenCV library: *haartraining* and *createsamples* are used.

You can operate with them through the command line, in this case, using the following commands:

** createsamples.exe -info positives/train.txt -vec data/positives.vec -num 1300 -w 20 –h 20,*

where:

-info – is the path to the file containing the training set,

-vec – is the file generated that will contain information on the positive set,

-num – is the number of positive pictures,

-w – length,

-h – width.

** haartraining.exe -data data/cascade -vec data/positives.vec -bg negatives/train.txt-npos 950 -nneg 2100 -nstages 30 -mem 2500 -w 20 -h 20,*

where:

-data – is the path where the classifier will be created,

-vec – is the path to the file that contains information on the positive pictures set,

-bg – is the number of negative pictures (950),

-npos – is the number of positive pictures (2100),

-nneg – is the number of positive pictures,

-nstages – is the number of training stages,

-mem – is the memory available for training,

-w – length,

-h – width.

The training process requires a lot of hardware resources, for a classifier, for a primitive; approximately two days are necessary for processing. The result is an XML (Extensible MarkupLanguage) creation that groups information used in the recognition, in logical groups (fig. 6) .

For the application, four classifiers were created, corresponding to the four primitives that need to be recognized. In fig. 7 is illustrated the spheres detection, based on the previously built classifier.

3. IMPLEMENTING ALGORITHMS TO MEASURE OBJECTS

After the object has been identified, the next step is to measure it, in order to pre-configure the gripper, and generate its 3D model. The measurement of the object is based on a video device, and the software module, able to analyze each frame of the video, and information about the size of a pixel in the metric (mm) system captured image. The structure of a module for measuring objects is illustrated in fig. 8.

```
<?xml version="1.0" ?>
- <opencv_storage>
  - <cascade type_id="opencv-haar-classifier">
      <size>24 24</size>
    - <stages>
      - <_>
          <!-- stage 0 -->
        - <trees>
          - <_>
              <!-- tree 0 -->
            - <_>
                <!-- root node -->
              - <feature>
                - <rects>
                    <_>0 0 8 14 -1.</_>
                    <_>4 0 4 14 2.</_>
                  </rects>
                  <tilted>0</tilted>
                </feature>
                <threshold>0.0930760800838470</threshold>
                <left_val>-1.</left_val>
                <right_val>1.0000129938125610</right_val>
              </_>
            </_>
          </trees>
          <stage_threshold>1.0000129938125610</stage_threshold>
          <parent>-1</parent>
          <next>-1</next>
        </_>
      - <_>
          <!-- stage 1 -->
        + <trees>
          <stage_threshold>1.1275479793548584</stage_threshold>
          <parent>0</parent>
          <next>-1</next>
        </_>
      </stages>
    </cascade>
```

Fig. 6 Haar classifier for spheres

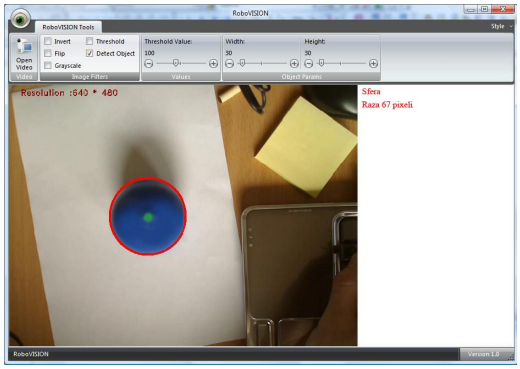

Fig.7 Primitives detection

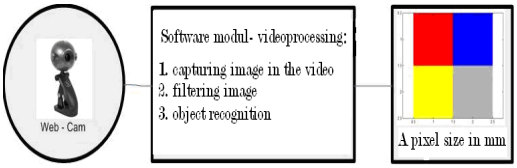

Fig. 8 The structure of a module for measuring objects

To determine the size of the detected objects, a method based on finding the report between the actual object size in millimeters, and its size in pixels, is used. In the case of a static sequence, each pixel can be associated with size in millimeters, as seen in fig. 9.

However, gripping strategy requires object analysis while the gripper is moving to the object. Thus, the software module will always capture video frames to a resolution that is known (640 * 480 pixels) where the size of the object detected, in pixels, varies depending on the distance of the frame capture.

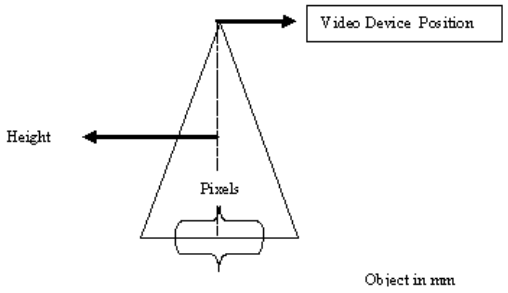

Fig.9 Measuring objects

The closer the video device will be to the object, the more it will tend towards the maximum size of the video frame. To assign a standard, before the actual measurement of an object, calibration is needed that involves a known-size object (a standard) in the zero position of the video device. By this procedure, the ratio of the object size between millimeters and pixels, is assigned. Based on the detected standard, the variation of the pixel size can be found, according to the video capture distance. The process is illustrated in fig. 8. The value of the mm / pixel ratio can help in obtaining crucial information in the gripping process: *the distance to the object (see & 2)*.

Thus, the distance covered between two points helps in the calibration process of the measurement system, and in setting the mm / pixel ratio. In real environment, the distance covered along Y axis can be found from the robotic arm controller, which will have the gripper attached, and in virtual environment, the 3D model of the object, recognized and measured, will be generated. The object is classified (sphere, parallelepiped, cylinder or cone) and its size in pixels is calculated. To solve this phase Open Inventor software module developed by Silicon Graphics Inc. (SGI) was selected. One of the main reasons why this format was chosen is that it allows a complete description of the 3D scene graphs with polygons rendered in the form of objects. Open Inventor also includes lighting facilities, descriptions of materials and textures and special effects. An Open Inventor file has the extension iv and describes a text file in ASCI format (format that uses letters and characters in the English alphabet) that illustrates through nodes and elements the 3D scene. Thus, Open Inventor format can be viewed as a descriptive language of a virtual scene. In addition, having a standard description of virtual scene, interoperability between applications that have an Open Inventor files interpreter implemented can be achieved.

Open Inventor standard enables primitives to be described through a class called Shapes as follows:

1. Sphere

Sphere { radius }

2. Parallelepiped

Cube { width height depth }

3. Cylinder

Cylinder { parts ALL radius height }

4. Cone

Cone { parts ALL bottom Radius heigh }.

Thus, when the object was recognized and measured, by the results, the CPrimitiveWriter class developed in the application will create an Open Inventor file where it will add the Shape section and it will save this file to disk. Then the virtual simulation module will load the object to the virtual scene and will use it for various tests and simulations. Also the ratio between the sizes of the object in pixels, calculated at different distances, is used to determine the distance to the object. If you know the relationship between millimeters and pixels, you can calculate the actual size of the object to be gripped. In fig. 10 there is the file generated by CprimitiveWriter class where the recognized primitive and its size are added to the penultimate line. On the right there is the sphere loaded in virtual scene. Also, on the second and

third lines of the file Open Inventor information related to the material and weight can be added. In this case, there is a glass sphere of 200 grams. This information is generated only to have some implicite values within the virtual simulation models, as they can be changed afterwards. Thus, the gripper aperture is found, and together with the information related to the shape of the object, the corresponding pre-configuration is assigned.

```
#Inventor V2.1
ascii
#glass
#200
Separator { # Body

Transform {
translation 0 0 0 }
        Material
{ # A bronze color:
ambientColor .33
.22 .27
diffuseColor .78
.57 .11
 specularColor .99
.94 .81
shininess .28
  transparency .6
        }
  Sphere { radius
67 }
}
```

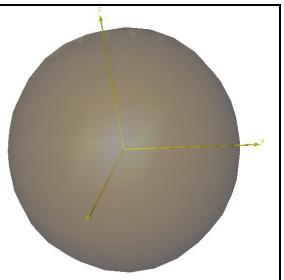

Fig.10 3D sphere geometric virtual model

4. CONCLUSIONS

The algorithmic method for viewing, identifying, and measuring objects for anthropomorphic grip, is based on a series of existing software modules, which have been adapted for the intended purpose, and to which, specially designed software modules were added.

Thus, it was created a specialized software module, to manage interaction with video devices attached to a computer that provides access to every frame of the video sequence, for analysis.

To achieve this module, the Microsoft DirectShow technology, available in the DirectX package, created especially for multimedia and 3D, was used. Video analysis is performed by a module specialized in the recognition of four primitives - spheres, parallelepipeds, cones, and cylinders. For object recognition, the algorithm proposed by Viola and Jones, based on classifiers, was used. Another software module is a module specialized in measuring objects. To achieve this, we used an algorithm based on the comparison of two video frames, containing the object to be measured, taken from different distances. Research continues, by generating the 3D model of the object to be gripped, by anthropomorphic gripper pre-configuration and, ultimately, by achieving gripping, each of these stages having specific features.

REFERENCES

[1]. **Bard, C., Troccaz, J., Vercelli, G.** - *Shape Analysis and Hand Pre-shaping for grasping.* IEEE/RSJ International Workshop on Intelligent Robots and Systems, Nov 3-5 1991, Japan, pp. 64-69.
[2]. **http**://opencv.willowgarage.com/wiki/
[3]. **Itu, A.** - *Contribution to Gripping Strategies in Real and Virtual Environment using a Three-Fingered Anthropomorphic Gripper*, *PhD. Thesis, Transilvania University of Brasov*, Brasov,Romania,2010.
[4]. **Staretu, I.** - *Gripping Systems*(in Romanian), Lux Libris Publishing House, Brasov, Romania, 2010.
[5]. **Viola, P. and Jones, M. J.** - *Robust Real-Time Face Detection* in International Journal of Computer Vision, 57(2), pp. 137--154, May 2004.
[6]. **Winges, S.A., Weber,D.J., Santello, M.** - *The Role of Vision on Hand Preshaping during Reach to grasp.* Exp Brain Res 152, pp. 489–498, 2003.