Feature Extraction based on Hierarchical Growing Neural Gas for Informationally Structured Space

Yuichiro Toda, Naoyuki Kubota

Abstract—This paper proposes a method of feature extraction from 3D point clouds for informationally structured space including sensor networks and robot partners for co-existing with people. The informationally structured space realizes the quick update and access of valuable and useful information for both people and robots on real and virtual environments. Our method is based on Hierarchical Growing Neural Gas (HGNG). This method is one of self-organizing neural network based on unsupervised learning. First, we propose 3D map building method using Kinect in order to acquire the 3D point clouds. Next, we propose the method of the feature extracting method based on HGNG. Finally, we show experimental results of the proposed method and discuss the effectiveness of the proposed method.

I. INTRODUCTION

Recently, various types of robots have been developed as the development robot technologies (RT). Today, intelligent robots receive attention by births of entertainment robots, pet robots and guide robots for a facility. In general, industrial robots can recognize around the environment, and behave in structured and static areas according to a computer program. On the other hand, these robots sometimes encounter unknown or dynamic areas in a house or social space. Therefore, it is very important for the robots to recognize and interpret the environments to perform some tasks and interaction with people. In order to realize such robots, the robots require the capabilities of human and object recognition, task planning and path planning in the dynamic environment. However, it is difficult for the robot to realize these capabilities because the sensing range of one robot can cover only a limited area even if the robot is equipped with various types of sensors. Therefore, ambient intelligence technologies of environmental systems based on the structured information have been expected recently [1-5]. By applying the ambient intelligence technologies such as wireless sensor networks to RT, the robot can use the various types of information outside of the robot. The wireless sensor networks [6-10] realize to gather the huge data on environments. However, it is very difficult to store all of huge data in real time. Furthermore, some features should be extracted from the gathered data to obtain the required information. Therefore, intelligent technology is required in wireless sensor networks. Intelligence technology and information technology have been discussed from various points of view. Information resources and the accessibility within an environment are essential for both people and robots.

Therefore, the environment surrounding people and robots should have a structured platform for gathering, storing, transforming, and providing information. Such an environment is called informationally structured space (Fig.1). The intelligent technology for the design and usage of the informationally structured space should be discussed from various points of view such as (1) data gathering of real environment and cyber space, (2) information extraction, (3) structuralization, (4) information visualization and display, (5) information search, and (6) information operations. The structuralization of informationally structured space realizes the quick update and access of valuable and useful information for both people and robots on real and virtual environments. The information is transformed into the useful form suitable to the features of robots and people. Furthermore, if the robot can share the environmental information with people, the communication with people might become very smooth and natural.

In order to realize informationally structured space, many researchers proposed environmental map building methods as sharing the environmental information between people and robots. Especially, many 3D environmental map building methods integrated color and distance information have recently been proposed since Microsoft Kinect developed [11,12]. These information visualization approaches can give people efficient information to recognize the environment because people can easily recognize the objects from the 3D environmental map. However, these maps are only the 3D point clouds for the robots. Therefore, it is difficult to use these data efficiently. In this paper, we apply unsupervised learning to 3D environmental map in order to extract several features from the 3D point clouds. By extracting the suitable features, the robot and people can share the environmental information. Our method is based on Growing Neural Gas (GNG). These methods are one of self-organizing neural network based on unsupervised learning [13-16]. By using this method, the 3D point cloud is transformed into 3D colored topological model structure. Furthermore, we propose a modified Hierarchical Growing Neural Gas (HGNG) by using several features.

This paper is organized as follow. Section 2 explains our
3D map building method. Section 3 explains unsupervised learning, proposes the methods of hierarchical neural gas and shows preliminary experimental result of unsupervised learning. Section 4 shows some experimental results of the proposed method.

II. 3D ENVIRONMENTAL MAP BUILDING

A. Feature Extraction and Matching

This subsection explains the detail of feature extraction and matching method. Our 3D map building use Kinect sensor that can get camera image and 3D distance information simultaneously [17,18]. First of all, the obtained image is transformed into gray scale image. The feature points are extracted from two images by using Speed Up Robust Features (SURF), because SURF is robust to the change of illumination and local affine distortion of images [19]. However, it takes computational cost to calculate SURF. Therefore, we use SURF implemented in General-purpose computing on graphics processing units (GPGPU) proposed by OpenCV [20]. SURF implemented in GPGPU realizes the high speed calculation using parallel processing by GPU. Each SURF is described by a vector containing 64 elements. SURF generates N possible pairs of corresponding points according to the matching threshold. The calculation time of feature extraction and matching is about 0.29s at 640x480 pixels resolution by using GPGPU. However, the pairs include many mismatched pairs when the pairs are generated. The estimation error of coordinate transform matrix is very large by using the pairs including many mismatched pairs. Therefore, the mismatched pairs are removed by estimating a homography matrix between two images. In order to estimate the homography matrix, we use RANdom SAmple Consensus (RANSAC). SIFTGPU generates N possible pairs of corresponding points according to the matching threshold. The position of a pixel on the (i-1)-th image is represented as \( x^i(x, y) \). The corresponding point calculated by homography matrix (\( a_i \)) is represented as \( (x^i', y^i') \) in the following:

\[
\begin{align*}
    x^i &= \frac{a_{11}x + a_{12}y + a_{13}}{a_{31}x + a_{32}y + 1} \\
    y^i &= \frac{a_{21}x + a_{22}y + a_{23}}{a_{31}x + a_{32}y + 1}
\end{align*}
\]

where \( a_{ij} \) is the element of the j-th homography matrix. Next, we explain the method of RANSAC. The procedure is shown in Algorithm 1.

The matching error between two points is calculated by

\[
\text{error} = \left( x^i - \frac{a_{11}x_i + a_{12}y_i + a_{13}}{a_{31}x_i + a_{32}y_i + 1} \right)^2 + \left( y^i - \frac{a_{21}x_i + a_{22}y_i + a_{23}}{a_{31}x_i + a_{32}y_i + 1} \right)^2
\]

where \((x^i', y^i')\) is the position of a pixel paired with \((x_i, y_i)\). As a result of RANSAC, we obtain the \( L \) matched pairs.

Algorithm 1 RANSAC:

1. Initialize \( j = 1 \)
2. Random sampling of 4 pairs from \( N \) pairs of corresponding points
3. \( \text{inlier}_j = 0 \)
4. Estimate the homography matrix \( (a_i) \) using 4 pairs
5. Repeat \( i = 1 \)
6. Calculate of the error of each pair using the estimated homography matrix
7. \( \text{inlier}_j = \text{inlier}_j + 1 \) if the error is lower than the threshold
8. Until \( i = N-4 \)
9. Until \( j = M \)
10. \( k = \max_{j \text{inlier}} j \)
11. Repeat \( i = 1 \)
12. Calculate of the corresponding point \( x^i \) according to \( a_k \)
13. Selection of \( x^i \) if the error is lower than the Threshold

B. 3D Map Building

After the selection of corresponding points, we must update the environmental map composed of points with color. The update of the environmental map is to obtain the position \( x^i = (x^i, y^i, z^i) \) of a pixel in the 3D space based on the position \( x_0 = (x_0, y_0, z_0) \) of the environmental map according to the relationship between \((x^i, y^i)\) and \((x_0, y_0)\). An interactive closest point (ICP) algorithm is one of the most widely used methods of matching a set \( X \) of points with point clouds \( X^i \) in 3D space [21,22]. The error function to be minimized is defined as

\[
E^{\text{CP}}(R,t) = \frac{1}{L} \sum_{j=1}^{L} \left\| Rx^j + t - x_j \right\|^2
\]

where \( R \) is the rotation matrix; \( t \) is the translation vector; We apply the unit quaternion proposed by Horn [23]. The quaternion is defined as \( \hat{q} = (q_0, q_1, q_2, q_3) \). First, the center of gravity (COG) of each point cloud is calculated in the following:

\[
\begin{align*}
    x^j &= \frac{1}{L_j} \sum_{i=1}^{L_j} x^i_j \\
    x'^j &= \frac{1}{L_j} \sum_{i=1}^{L_j} x'^i_j
\end{align*}
\]

where \( L_j \) is the number of points in each point cloud. Next, the relative position from the COG is calculated in the following:
\[ x'_i = x_i - x'_i \] \[ x'_{i\beta} = x'_{i\beta} - x'_{i\beta} \] (7)

Next, \( S_{ab} \) is defined as;
\[ S_{ab} = \sum_{i=1}^{l} x'_i x'^{\beta}_{i} \] (8)

According to \( S_{ab} \), a matrix \( P \) is defined as
\[
P = \begin{bmatrix}
S_x + S_x + S_x & S_x - S_x & S_x - S_x & S_x - S_x \\
S_x - S_x & S_x + S_x - S_x & S_x + S_x - S_x & S_x + S_x - S_x \\
S_x - S_x & S_x + S_x & S_x + S_x & S_x + S_x & S_x + S_x & S_x + S_x & S_x + S_x & S_x + S_x \\
S_x - S_x & S_x + S_x & S_x + S_x & S_x + S_x & S_x + S_x & S_x + S_x & S_x + S_x & S_x + S_x & S_x + S_x \\
\end{bmatrix}
\] (9)

Here the eigenvector corresponding to the maximum positive eigenvalue of \( P \) is quaternion (\( \hat{q} \)). The rotation matrix is obtained by \( \hat{q} \) in the following;
\[
R = \begin{bmatrix}
q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_0 q_2 + q_1 q_3) \\
2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_0 q_3 - q_1 q_2) \\
2(q_1 q_3 - q_0 q_2) & 2(q_0 q_3 + q_1 q_2) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \\
\end{bmatrix}
\] (10)

Furthermore, the translation vector is also obtained by \( R \) in the following;
\[ t = x'_0 - Rx'_{0} \] (11)

Figure 2 shows an experimental result of 3D modeling using the distance data and image data measured by Microsoft Kinect sensor.

C. Surface detection from 3D point clouds

In this paper, we want to extract the features of the object from the 3D point clouds efficiently. However, the 3D point clouds have many unnecessary data such as a floor or a desk surface. Therefore, we remove the unnecessary data from the 3D point clouds by detecting the plane as the surface (Fig.3 (a)). The basic algorithm of plane detection is based on RANSAC. The model uses \( a_1 x + a_2 y + a_3 z + a_4 = 0 \) in plane detection and \( a_1 - a_4 \) are the model parameters. The number of samples needed to estimate the model parameters is three points and the error function \( e_i \) is defined as follows,
\[
e_i = \frac{|a_1 x_i + a_2 y_i + a_3 z_i + a_4|}{\sqrt{a_1^2 + a_2^2 + a_3^2}}.
\] (12)

In this way, the error \( e_i \) means the distance from point \( p_i \) to estimated plane. In addition, the cost function of RANSAC can also be calculated by the number of inliers. These definitions enable to detect the plane easily. In this paper, the detection accuracy is improved by considering the normal vector of the point. Specifically, the normal vector is approximated with the normal to the local surface by using the covariance matrix of the local surface \( [14, 25] \). Figure 4 shows the flowchart of the plane detection. First of all, the 3D point clouds are classified by kd-tree \( [26] \) in order to search the neighborhood of the point \( p_i \) quickly. Next, the local surface of the point \( p_i \) is composed of the \( k \)-neighborhood surface within radius \( r \). If the local surface of the point \( p_i \) does not have \( k \) data then the point \( p_i \) is removed from the data set \( S \). The COG \( o_i \) of the local surface is calculated, and then 3x3 covariance matrix \( C \) is calculated as follows
\[
C = \sum_{i=1}^{k} (p_i - o_i) (p_i - o_i)^T.
\] (13)

The eigenvectors \( v \) and eigenvalues \( \lambda \) are computed \( Cv = \lambda v \). If \( \lambda_1 \leq \lambda_2 \leq \lambda_3 \) then the unit normal vector \( n_1 \) can be selected to be either \( v_1 \) or \( -v_1 \) associated with \( \lambda_1 \). Using the unit normal vector of the local surface enables to detect the plane elements that have the similar direction of unit normal vector. Fig.3 (b) shows an example result of the surface result.
III. UNSUPERVISED LEARNING BASED ON GROWING NEURAL GAS

A. Unsupervised Learning

Unsupervised learning is performed by using only data without any teaching signals [27]. Self-organized map (SOM), neural gas (NG), growing cell structures (GCS), and growing neural gas (GNG) are well known as unsupervised learning methods. Basically, these methods use the competitive learning. The number of nodes and the topological structure of the network in SOM are designed beforehand [28]. In NG, the number of nodes is fixed beforehand, but the topological structure is updated according to the distribution of sample data. On the other hand, GCS and GNG can dynamically change the topological structure based on the adjacent relation (edge) referring to the ignition frequency of the adjacent node according to the error index. However, GCS does not delete nodes and edges, while GNG can delete nodes and edges based on the concept of ages [13,14]. Furthermore, GCS must consist of k-dimensional simplexes whereby k is a positive integer chosen in advance. The initial configuration of each network is a k-dimensional simplex, e.g., a line is used for k=1, a triangle for k=2, and a tetrahedron for k=3. GCS has applied to construct 3D surface models by triangulation based on 2-dimensional simplexes. However, because the GCS does not delete nodes and edges, the number of nodes and edges is over increasing. Furthermore, GCS cannot divide the sample data into several segments. Therefore, we use growing neural gas in order to cluster the 3D environmental space. Furthermore, we propose hierarchical growing neural gas (HGNG) in order to extract the features of the objects from the 3D point clouds.

B. Hierarchical Growing Neural Gas

This subsection explains the Hierarchical GNG (HGNG). HGNG consists of l layers of GNG. The lowest layer of GNG directly uses the input data and learns the structure of data distribution. The (l+1)-th layer of GNG uses the reference vector of the selected node in the l-th layer of GNG. Therefore, the higher layer of GNG learns the relationship among nodes in the lower layer of GNG. The procedure and notation used in HGNG are shown as follows;

\[ l \]: The number of layers
\[ w_i^j \ (i \in \mathbb{R}^n) \]: The \( i \)th dimensional vector of a node
\[ h_i^j \ (i \in w_i^j) \]: The \( m \)th dimensional specific feature vector of a node (\( m \leq n \))
\[ A^j \]: A set of nodes
\[ N_i^j \]: A set of nodes connected to the \( i \)th node
\[ c^j \]: A set of edges
\[ g_{i,j}^j \]: Age of the edge between the \( i \)th and \( j \)th nodes

Step 0. Generate two units at random position, \( w_i^1 \) and \( w_j^1 \) in \( 
\mathbb{R}^n \) where n is the dimension of input data. Initialize the connection set.

Step 1. Select at random an input data \( v_l \ (i \in \mathbb{R}^n) \) and the \( m \)th dimensional specific input vector \( v_m \ (i \in \mathbb{R}^n) \) from the dataset if \( l = 1 \). Otherwise, the \( l \)-th layer of GNG uses the reference vector of the selected node in the (\( l-1 \))-th layer of GNG

Step 2. Select the nearest unit (winner) \( s_1 \) and the second-nearest unit \( s_2 \) from the set of nodes by

\[
\begin{align*}
  s_1 &= \arg \min_{i \in A^1} \| v_l - h_i \| \\
  s_2 &= \arg \min_{i \in A^1} \| v_l - h_i \|
\end{align*}
\]

Step 3. If a connection between \( s_1 \) and \( s_2 \) does not yet exist, create the connection (\( c_{s_1,s_2} = 1 \)). Set the age of the connection between \( s_1 \) and \( s_2 \) at zero;

\[
g_{s_1,s_2} = 0
\]

Step 4. Add the squared distance between the input data and the winner to a local error variable;

\[
E_n := E_n + \| v_l - h_i \|^2
\]

Step 5. Update the reference vectors of the winner and its direct topological neighbors by the learning rate \( \eta_1 \) and \( \eta_2 \) respectively, of the total distance to the input data.

\[
\begin{align*}
  w_i^1 &= w_i^1 + \eta_1 (v_l - w_i^1) \\
  w_i^1 &= w_i^1 + \eta_2 (v_l - w_i^1) \quad \text{if} \quad c_{s_1,s_2} = 1
\end{align*}
\]

Step 6. Increment the age of all edges emanating from \( s_1 \);

\[
g_{s_1,s_2} = g_{s_1,s_2} + 1 \quad \text{if} \quad c_{s_1,s_2} = 1
\]

Step 7. Remove edges with an age larger than \( g_{\text{max}} \). If this results in units having no more connecting edges, remove those units as well.

Step 8. If the number of input data generated so far is an integer multiple of a parameter \( l \), insert a new unit as follows.

i. Select the unit \( u \) with the maximal accumulated error.

\[
u = \arg \min_{i \in A} E_i
\]

ii. Select the unit \( f \) with the maximal accumulated error among the neighbors of \( u \).

iii. Add a new unit \( r \) to the network and interpolate its reference vector form \( u \) and \( f \).

\[
w_r = 0.5 (w_u + w_f)
\]

iv. Insert edges connecting the new unit \( r \) with units \( u \) and \( f \), and remove the original edge between \( u \) and \( f \).

v. Decrease the error variables of \( u \) and \( f \) by a temporal discounting rate \( \alpha \).
\[ E_r \leftarrow E_r - \alpha E_u \]  
\[ E_f \leftarrow E_f - \alpha E_f \]  
vi. Interpolate the local error variable of \( r \) from \( u \) and \( f \).  
\[ E_r = 0.5 \left( E_u + E_f \right) \]  

Step 9. Decrease the local error variables of all units by a temporal discounting rate \( \beta \).  
\[ E_i \leftarrow E_i - \beta E_i \quad (\forall i \in A) \]  

Step 10. Continue with step 1 if an update condition of the layer (e.g., the number of nodes or some performance measure) is not yet fulfilled. Otherwise, \( l \leftarrow l + 1 \).  

Step 11. Continue with step 0 if a stopping criteria (e.g., the number of layers or some performance measure) is not yet fulfilled.

Figure 5 shows examples of learning of SOM, GCS and GNG. In the examples, the number of nodes in two-dimensional SOM is 900 (30x30). The parameters used in GCS simulations were: \( \lambda = 200; \mu_1 = 0.04; \mu_2 = 0.001; \alpha = 1.0; \beta = 0.0005 \), in GNG simulations were: \( \lambda = 200; \mu_1 = 0.05; \mu_2 = 0.005; \alpha = 1.0; \beta = 0.0005 \). We used data distribution of four rings. In Fig.5 (a), (b), SOM and GCS have many redundant nodes and edges exist outside of the sample data distribution. On the other hand, in Fig.5 (c), GNG efficiently covers the data distribution.

(a) SOM (b) GCS (c) GNG

Fig. 5. The comparison of unsupervised learning methods

In this paper, the input data is composed of the position (distance) information \((x, y, z)\) of a point, the unit normal vector \( n \) of the local surface and the color information \((R, G, B)\). However, it is difficult to cluster the objects from 3D point clouds because many relations between objects are generated in order to use these vectors simultaneously. Fig.6 (a) shows the example result of GNG by using the 3D point clouds of Fig. 3 (b). In Fig. 6 (a), the number of generated clusters is only 1. It is difficult to extract the features of an object from 1 cluster. Therefore, we use the specified vector when the winner node is selected and the accumulated error is calculated. On the other hand, all the vectors are used when the node is learned and generated. Fig. 6 (b) shows the result of the 1st layer of proposed HGNG. In Fig.6 (b), the number of clusters is 7 and the position information and color information is learned suitably as compared with the original data (Fig. 3 (b)). These nodes are used as the input data of the next layer. By doing so, we extract the features of each object efficiently.

(a) A simulation result of GNG using all the vectors \((k_1 = w_i)\).

(b) A simulation result of GNG using only the position vector at the winner node selection and the accumulated error calculation

Table 1 The parameters of HGNG

<table>
<thead>
<tr>
<th></th>
<th>The 1st layer</th>
<th>The 2nd layer</th>
<th>The 3rd layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>300</td>
<td>150</td>
<td>75</td>
</tr>
<tr>
<td>( \mu_1 )</td>
<td>0.08</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>( \mu_2 )</td>
<td>0.008</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>( \beta )</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
</tr>
<tr>
<td>( N )</td>
<td>800</td>
<td>400</td>
<td>200</td>
</tr>
<tr>
<td>( g_{max} )</td>
<td>88</td>
<td>40</td>
<td>30</td>
</tr>
</tbody>
</table>

IV. EXPERIMENTAL RESULTS

A. Feature extraction from 3D point clouds

This section shows experimental results of proposed method. Table 1 shows the parameters of HGNG in this experiment (Superscript indicates the number of layer and \( N \) is maximum number of the node.). Figure 7 and 8 show the experimental results of HGNG. Fig. 7 uses the color information as the 2nd layer and the unit normal vector of the local surface as the 3rd layer. Fig. 8 uses the unit normal vector as the 2nd layer and the color information as the 3rd layer. In Fig. 7, the numbers of cluster of the 2nd layer and the 3rd layer is 32 and 2, respectively. In Fig. 8, the numbers of cluster of the 2nd layer and the 3rd layer is 5 and 7, respectively. In these results, the unit normal vector of the

(a) A simulation result of HGNG using all the vectors \((k_1 = w_i)\).

(b) A simulation result of HGNG using only the position vector at the winner node selection and the accumulated error calculation

Fig. 6 An example of the 1st layer GNG (These figures plots the position information of the nodes and edges and The color of edge indicates the color information of the node.)
local surface can extract the suitable normal vectors of the objects since the surfaces can segment top and side surfaces. In addition, the clustering result of Fig. 7 (a) can extract the color features of the objects as compared with the original 3D point clouds. However, the color information of the clustering result near the gray scale image in Fig. 8 (b) since the number of nodes in each clusters are so large that the color information is blended. In this way, proposed HGNG can extract more abstract features of the objects as the number of layers increases. On the other hand, if we use the color information as the 1st layer, HGNG cannot cluster the color features (Fig. 9). In Fig. 9, the number of clusters is only 1. The reason is that the original color information is distributed over the RGB color coordinate space and it is difficult to remove the unnecessary edges in GNG. From these results, the suitable features of the objects can be extracted from the 3D point clouds by using only the specified feature vector at the winner node selection and the accumulated error calculation.

**B. Object recognition**

Next, we conduct on the experiment of specific objects recognition by using each feature. Figure 10 shows the results of the 3D modeling and the surface detection. In this experiment, there are four different cups on the desk. Figure 11 shows experimental result of HGNG. In this experiment, the 1st layer uses the 3D position information as the specific feature and the 2nd layer uses the color information. In Fig. 11, the topological structure and the color features can be extracted from the 3D point clouds. The numbers of clusters of the 1st layer and the 2nd layer are 8 and 29, respectively. After extracting the features of each object, the robot can share the environmental information with a human by using a sound recognition system:

1. Human: Please look at the biggest cup on the desk.
2. Robot: OK! Is the color of the cup blue? (Fig. 12)
3. Human: Yes, it is. Is there a black cup on the desk?
4. Robot: Yes, there is. (Fig. 13)
In Fig. 12, the robot can detect the biggest cup by using the number of nodes including each cluster and the 3D positions of the nodes. After detecting the object, the robot can recognize the color of the cup by associating the 3D positions with the color features of the nodes. On the other hand, the robot can detect the black cup in Fig. 13 by using the color features of the cluster. In this way, the robot can share the environmental information by extracting the features of the objects.

V. CONCLUSION

In this paper, we proposed feature extraction based on Hierarchical Growing Neural Gas (HGNG) from 3D point clouds in order to share the environmental information with human and a robot. First, we explained the 3D environmental map building method by using Kinect in order to acquire the 3D point clouds. Next, we proposed the modified HGNG. In our HGNG, only the specified feature vector uses the winner node selection and the accumulated error calculation. The experiments showed that the proposed method could extract each abstract feature from the 3D point clouds and the robot could share the environmental information by using the features of the objects.

However, the feature of unit normal vector didn’t use the experiment of the specific object recognition. By using unit normal vector, we consider the shape of the object can be detected. Therefore, we will propose the shape detection method by using the extracting feature. Furthermore, we will apply the proposed method in dynamic environment in order to realize the human-friendly interaction system of the communication robot.

REFERENCES