Knowledge vector representation of three-dimensional convex polyhedrons and reconstruction of medical images using knowledge vector

Shilpa Rani1,2 · Kamlesh Lakhwani3 · Sandeep Kumar4

Received: 12 May 2022 / Revised: 19 September 2022 / Accepted: 5 February 2023 / Published online: 6 March 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
Three-dimensional image construction and reconstruction plays an important role in various applications of the real world in the field of computer vision. In the last three decades, researchers are continually working in this area because construction and reconstruction is an important approach in medical imaging. Reconstruction of the 3D image allows us to find the lesion information of the patients which could offer a new and accurate approach for the diagnosis of the disease and it adds a clinical value. Considering this, a novel approach is proposed for the construction and reconstruction of the image. First, a syntactic pattern recognition-based algorithm is implemented to extract the features from the 2D image. The proposed algorithm takes an input image and extracts the features from the image and these features (Knowledge vector) consist of direction code and length. In addition, a unique and novel algorithm is developed that can rebuild an image using a knowledge vector. Reconstruction allows us to investigate the interior details of 3D images, such as the object’s size, form, and structure. The proposed algorithm performance is assessed on a medical imaging dataset, and the findings are outperformed. Performances of the proposed algorithms are evaluated on Kaggle brain MRI dataset and Medical MRI datasets which is collected from Pentagram research institute, Hyderabad. As per the experimental analysis, the proposed method gives
93.89% of accuracy on Kaggle brain MRI dataset and 97.24% on Medical MRI dataset which is better than exiting state of art methods.

**Keywords** Construction · Reconstruction · 3D medical images · Syntactic grammar · Knowledge vector

1 Introduction

Pattern recognition encloses two primary tasks: description & classification. Pattern recognition system generates the description of the object which is present in the image and then classifies the object according to the description. There are two different approaches for implementing pattern recognition: statistical and structural. These two techniques use a different method for description and classification purposes. Statistical approach [8, 13, 20] use decision-theoretic concepts for the discrimination between the objects which belongs to different class based on their quantitative features. Structural pattern recognition approach [12, 15, 26] sometimes referred as syntactic pattern recognition approach because it uses syntactic grammars concept for the discrimination between objects which belongs to different class based upon their shape, size or structural features. Some researchers combined both structural and statistical approach for pattern recognition. The statistical approach is well explored by many researchers but structural pattern recognition still doesn’t explore adequately because it requires domain knowledge for the description and classification purpose. Therefore, Structural pattern recognition techniques are domain-specific [5, 29–31].

In this paper, we proposed a novel approach basis on the Syntactic approach which represents the three-dimensional convex polyhedrons in textual form. This textual representation consists of the direction and length code of the objects which are present in the image. The textual representation is referred to as the knowledge vector. Further, this knowledge vector is considered as input and a novel algorithm is written in such a way that reconstructs a three-dimensional image using the knowledge vector. This is the First time in the field of image processing where a three-dimensional image can be reconstructed using textual information and it could be a great contribution in the field of structural pattern recognition as well as adds a new value in the field of image processing. As we discussed already that structural pattern recognition techniques are domain-specific so that we considered only medical images for our research.

1.1 Motivation and objectives

The primary motivation behind this research is to develop a novel algorithm that converts an image into a textual form using a syntactic pattern recognition technique and the secondary objective of this paper is to reconstruct a three-dimensional image using a knowledge vector. The motivation behind this research is to explore the new possibility of structural pattern recognition and this research can give a new direction to understand the three-dimensional images in textual form.

This paper is divided into 4 sections. The second section covers the motivation and objective of the research. Previous related work is discussed after the introduction. Third,
the fourth and fifth section covers the proposed methodology and evaluation parameters and results. While conclusion and future work are discussed in the last section.

2 Related work

Three-dimensional image construction and reconstruction are important parts of the field of computer vision, which is used in many real-world situations. Researchers have been working in this area for the past 30 years because construction and reconstruction are important parts of medical imaging. By putting the 3D image back together, we can find information about the patient’s lesion, which could be a new and accurate way to diagnose illness and add clinical value. Some of the authors have worked on reconstruction of the images and here we have reviewed some of the research papers which are as follows:

Yinsheng Li and Guang-Hong Chen [23] introduced DIM which is used to identify the inconsistency of the acquired projected data taken from different viewpoints. Further, the input data is divided into a subset according to the DIM value at each viewpoint angle and then the SMART-RECON algorithm is applied to the datasets for the reconstruction of the images. Practical utilization of the proposed method is tested on vivo human subject data and performance is better than existing methods. Aurelien Bustin et al. [2] proposed a method that combined the statistical and geometric 3D features. The main aim of the proposed method is to make a single isotropic volume by combining the multiple 3D volumetric data which is acquired from different orientations and recovery of sharp edges of the image and thin anatomical structure. These two aims are achieved using the Augmented Lagrangian scheme. The performance of the proposed method is evaluated on numerical simulations and the MRI datasets. Benjamin Hou et al. [18] proposed a method in which intensity information of 2D image slices is used for the registration of the image and further CNN is used for transformation of the image and for the learning of the regression function which maps the 2D slices into three-dimensional atlas spaces. Performance of the algorithm is evaluated on MRI dataset, fetal imagery with synthetic motion, and real MRI data and achieved average spatial prediction error of 7 mm. This method is a computationally better solution for 2D and 3D registration and it gives good results in real-time scenarios. Daniel P. Benalcazar et al. [1] built a 3D model by adding the depth information in 2D iris images. First, the smart phone camera captured the various 2D images from different angles and different lighting conditions, and then structure from motion algorithm is applied for reconstruction of the point cloud three-dimensional model. Finally, the best fitting three-dimensional mesh models is obtained with the help of the screened Poisson surface reconstruction method. The performance of the proposed method is evaluated on different ten subjects and the model contained approximately 11,000 three-dimensional points. The performance of the proposed 3D iris scanning method is better than the existing methods. Xuehang Zheng, Saiprasad Ravishankar [38] proposed a new PWLS based reconstruction method and the quality of the reconstructed 3D images is better than the existing PWLS-EP based method. This proposed method improved the quality of the image and image resolution and it’s more efficient in terms of computational speed. Chen Zhang [37] proposed a method for volumetric surface reconstruction and assumed that every scene consists of a cuboids structure. The proposed method having
reconstruction fidelity even if the object consists of a heavy curved and concave structure and the method can preserve the smoothness and cleanliness of the surface. This method is suitable for 3D printing of the images. Yuanhao Guo et al. [16] proposed a two-phase three-dimensional reconstruction method for light microscopy axis view images. In the first phase, an improved 3D volumetric representation is defined and in the 2nd phase, 3D reconstruction is achieved by searching the optimal surface over the confidence mapping. The performance of the proposed method is evaluated on three different datasets and the approach can represent 3D shape in a precise manner. This method can be used to determine the shape of the objects. Lu Ding et al. [6] proposed a method for optoacoustic image systems that helps to find the shape and size of the image. These methods have significant improvement in the spatial resolution, a contrast to noise ratio, and quality of the reconstructed image is reported better than existing techniques. Bin Li et al. [24] proposed a 3D-ReConStnet network that used a residual network for the extraction of the features from the 2D image. If objects are occluded in the image then the Gaussian probability distribution method is used for the learning purpose. The performance of the proposed method is evaluated on ShapeNet and Pix3D dataset and the results are comparatively better. Volumetric representation is an important approach to three-dimensional reconstruction. The main goal of volumetric representation is estimating a convex hull in three-dimensional spaces from each view [7, 10, 11, 22]. In the same way, the space carving algorithm is to recover the shape of the 3D objects by removing the voxels which are not visible in the particular view [14, 21]. But these methods require the proper segmentation and it is not always possible in some scenarios. There are many well designed three-dimensional reconstruction algorithms are available. Some algorithms are only designed for the reconstruction of peculiar and transparent objects [19]. These methods outperform on a macroscopic scale because to collect the surface-related information they setup the various lightning while capturing the images [25, 27]. In recent research [33] authors have presented a semantic reconstruction which is a combination of a data term and regularization constraints and results are outperformed on public datasets [34]. Now a day’s researchers are focusing on deep learning-based methods for improving the matching quality [17, 36] and for the reconstruction of 3D images but these methods require a large amount of training and testing data which are not available in our case.

After study various papers on 3D reconstruction, we conclude that volumetric based approaches are giving more promising results and properly addressing the challenges of 3D reconstruction [28, 32]. This is also true that in some cases the shape of the 2D/3D images is not accurate as expected. Therefore we developed a novel approach that is working based on 3D convex polyhedrons. The 3D image consists of voxels and each voxel itself represents a cube. So a 3D image is an array of cubes. Here we are considering the structuring element of size 3X3X3 so it consists of 27 neighborhood structures and this 3X3X3 size of the structuring element is used for processing of a 3D image. The whole 3D image is scanned using the structuring element and an algorithm is developed in such a way that it converts the image into a textual form which consists of direction code and length code and it is called a knowledge vector. This knowledge vector preserves the shape of the objects which is present in the image. Further, the same knowledge vector is used as an input for the reconstruction of the original 3D image. The detail of the proposed method is given in the further section.

© Springer
3 Proposed methodology

The proposed method is designed for 2D and 3D images and an image could be a simple image that consists of several objects or it could be medical images that are more complex and contains a lot of hidden information. There is a possibility that noise could be present in the image. Therefore, noise is removed first from the image using adaptive median filter (Table 1). The sample results and algorithm are shown below:

Algorithm 4.2 Adaptive Median Filter

| Step 1: Start |
|----------------|
| Step 2: Find $P_{a1} = \text{X}_{\text{median}} - \text{X}_{\text{minimum}}$ & $P_{a2} = \text{X}_{\text{median}} - \text{X}_{\text{maximum}}$ |
| Step 3: if ($P_{a1} > 0$ and $P_{a2} < 0$), then go to step 5; else, go to step 4. |
| Step 4: Increase the size of the window. Step 4a. if window_size < $I_{\text{max}}$ repeat Step 2 and 3 else the result is $I_{xy}$. |
| Step 5: Find $P_{b1} = I_{xy} - \text{X}_{\text{minimum}}$ & $P_{b2} = I_{xy} - \text{X}_{\text{maximum}}$ |
| Step 6: If ($P_{b1} > 0$ and $P_{b2} < 0$) then result from $I_{xy}$ else result is $X_{\text{median}}$ |
| Step 7: Stop |

Once the de-noising has been done then we applied the following strategy on all the input dataset (Fig. 1)

I. Knowledge acquisition
II. Feature Extraction
III. Reconstruction of 3D images

3.1 Knowledge acquisition

Knowledge acquisition is a method on which domain knowledge is acquired for extracting the features of a specific application or domain. This is the first step of structural pattern recognition based methods. Generally, Structural approach uses syntactic grammars concept for the discrimination between objects which belongs to different class based upon their shape,

Table 1 Tabulation of medical images using adaptive median filter

| Original Image | Adaptive Median filter | Original Image | Adaptive Median filter |
|----------------|------------------------|----------------|------------------------|
| ![Image](image1) | ![Image](image2) | ![Image](image3) | ![Image](image4) |
size, or structural features. Structural pattern recognition techniques are domain-specific. Further part knowledge acquisition steps are explained in detail.

a. Syntactic Pattern recognition

The new and less explored approach of pattern recognition is syntactic pattern recognition which consists of the formal language theory concepts. Syntactic pattern recognition term is a synonym of the linguistic, grammatical, and structural pattern recognition system. The formal language theory was originated by Noam Chomsky in the 1950s with the development of a mathematical model of grammar. The concept is described as follows:

Alphabet is a finite set of symbols. A word is any finite string that consists of symbols from the alphabets. For example a valid word of alphabets \{a, b\} is \{a, b, ab, ba, bb, aa…\}. A word with no symbol is an empty word that is denoted by \(\Lambda\). A language that consists of a set of words over a finite set of alphabets. Every natural language follows some grammar rules. The grammar of the formal language consists of four tuples

\[ G = \{P_N, P_T, P_r, R\}. \]

Here

- \(P_N\) Set of variables or non-terminals.
- \(P_T\) Set of constants or terminals.
- \(P_r\) set of production rules.
**R**: Root symbol or start symbol.

**P**: Union of \( P_N \) and \( P_T \).

Here \( P_N \) and \( P_T \) are disjoint set and \( R_s \) always belongs to either \( P_N \) or \( P_T \). Set of empty words denoted by \( P^* \) which is also called free Monoid.

The language which is generated by these tuples \( G \) is called \( L(G) \). It should satisfy two conditions: Each string should only consist of terminals. Every string should derive from the root \( R_s \) by applying the suitable production rules. Production rules consist of expressions in the form of \( X \rightarrow Y \). It means string \( X \) can be replaced by string \( Y \).

Where

\[
\begin{align*}
X & \quad \text{String in } P^+ \\
Y & \quad \text{String in } P^*
\end{align*}
\]

In the proposed methodology, the set of terminals or constants are denoted by:

\[
P_T = \{ R, DR, D, DL, L, UL, U, UR, B, BR, BDR, BD, BDL, BL, BUL, BU, BUR, F, FR, FDR, FD, FDL, FL, FUL, FU, FUR \}.
\]

Root \( R \) can be changed according to the object position. Here these symbols represent the direction in which a pixel could be present. Here \( R, D, L, U, B, \) and \( F \) are elementary symbols, and \( DR, DL, UL, UR, BR, BDR, BD, BDL, BL, BUL, BU, BUR, FR, FDR, FD, FDL, FL, FUL, FU, FUR \) are composite symbols of the alphabet \( P_T \). The set of production rules are generated through the Normal algorithm which was introduced by A.A. Markov. The Normal algorithm is used to recognize the angle of change in a particular direction during tracking of the contour. This is done with the help of the look-ahead tracing (LAT) method.

### b. Picture Description Language (PDL)

Picture description language is an application of semantic/linguistic concepts of pattern recognition. The pattern can be represented in the form of string grammar and this string grammar can be obtained using a simple juxtaposition of a string. Juxtaposition is nothing but keeping the two objects together without losing their identities. This can be done with concatenation also but it involves some spatial arrangements and there is a possibility of losing the identity or losing some information related to the objects. Juxtaposition is easy and reliable because it only involves the head and tail position. Based on this permissible form of juxtaposition and because each primitive is abstracted as a directed line segment, it is evident that the structure of PDL are directed graphs and also that these structures can be handled by string grammars. Blank primitives must be used for generating disjoint structures. A null point primitive has an identical head and tail. The mechanics of PDL could be used to obtain the contour/wireframe of a pattern. To illustrate this mechanism, consider the following PDL grammar:

\[
G = \{ P_N, P_T, P_s, R_s \}
\]

Where

\[
\begin{align*}
P_N & \quad \{ R_s, P_1, P_2, P_3, P_4, P_5, P_6, P_7 \} \\
P_T & \quad \{ R, DR, D, DL, L, UL, U, UR, B, BR, BDR, BD, BDL, BL, BUL, BU, BUR, F, FR, FDR, FD, FDL, FL, FUL, FU, FUR \}
\end{align*}
\]
A set of production rules are represented by $Pr$ and production rules are shown below:

$$
\begin{align*}
R_s & \rightarrow R + P_1 \\
P_1 & \rightarrow DR + P_2 \\
P_2 & \rightarrow D + P_3 \\
P_3 & \rightarrow DL + P_4 \\
P_4 & \rightarrow L + P_5 \\
P_5 & \rightarrow UL + P_6 \\
P_6 & \rightarrow U + P_7 \\
P_7 & \rightarrow UR + P_8 \\
P_8 & \rightarrow B + P_9 \\
P_9 & \rightarrow BR + P_10 \\
P_{10} & \rightarrow BDR + P_{11} \\
P_{11} & \rightarrow BD + P_{12} \\
P_{12} & \rightarrow BDL + P_{13} \\
P_{13} & \rightarrow BL + P_{14} \\
P_{14} & \rightarrow BUL + P_{15} \\
P_{15} & \rightarrow BU + P_{16} \\
P_{16} & \rightarrow BUR + P_{17} \\
P_{17} & \rightarrow F + P_{18} \\
P_{18} & \rightarrow FR + P_{19} \\
P_{19} & \rightarrow FDR + P_{20} \\
P_{20} & \rightarrow FD + P_{21} \\
P_{21} & \rightarrow FDL + P_{22} \\
P_{22} & \rightarrow FL + P_{23} \\
P_{23} & \rightarrow FUL + P_{24} \\
P_{24} & \rightarrow FU + P_{25} \\
P_{26} & \rightarrow FUR
\end{align*}
$$

The result obtained by applying all the productions is

$$
R + (DR + (D + (DL + (L + (UL + (U + (UR + (B + BR + (BDR + (BD + (BDL + (BL + (BUL + (BU + (BUR + (F + (FR + (FDR + (FD + (FDL + (FL + (FUL + (FU + FUR))))))))))))))))))))
$$

The above production system creates an octagon as represented in the below Fig. 2:

The directions of the pixels are defined with the help of production rules. Suppose an alphabet $A$ is given then $A^*$ could be constructed over $A$ which consists of all the possible

---

**Fig. 2** Pictorial representation of **a** Center Plane **b** Back Plane and **c** Front Plane with corresponding pixel directions
words containing A and null word also. So any subset of A* is a formal language. According to this theory, any digital image is consists of a set of patterns and the same digital image can be represented as a language that consists of a finite array of vertex/vertices. As mentioned, the PDL is a language of representing the digital image into the regular array of vertices.

c. Structuring elements of 3D images

The size and shape of a 2D/3D processed image always depend on the structuring elements which have been chosen for processing the image. In the case of a 2D image, if the window size/neighborhood structure is 3X3 then a maximum of 16 convex polygons can be present. For a 3D image if the window size/neighborhood structure is 3X3X3 then a maximum of 256 convex polyhedrons can be present. In the same way, we can find the number of concave polygons and polyhedrons. These polygons and polyhedrons are called a structuring element which can be used for the processing of 2D and 3D images. The number of structuring elements always depends on the size of the neighborhood. This concept was introduced by Rajan (1990) and Jirawit Lerdsinmongkol (2008).

In our research, we used a neighborhood structure of size 3X3X3 so we have 256 possible structuring elements. A detailed description of 3D convex polyhedrons is given below:

### 3.1.1 Three-dimensional rectangular convex polyhedrons

The idea of constructing 3D convex polyhedrons is the same as 2D convex polygons. As we discussed earlier, a 3D image consists of voxels and each voxel itself represents a cube. So a 3D image is an array of cubes. Here we are considering the structuring element of size 3X3X3 so it consists of 27 neighborhood structures. The smallest size of possible convex polyhedrons consists of 7 neighborhoods. The labeling of the 3X3X3 array and their possible coordinates are represented further in Fig. 4 and three different planes are considered where the kth plane is the central plane and (k-1) and (k + 1) is the front and rear plane.

In a 3X3X3 size of array total of 8 corner pixels are present hence there is a possibility of any corner pixel would not be available or any two corner pixels would not be available etc. Therefore, there are 256 different possible ways of constructing the convex polyhedrons w.r.t central pixel 14. The formula is given below:

![Fig. 3 a 27-Neighbourhood b 7-Neighbourhood c 9-Neighbourhood d 15-Neighbourhood e 19-Neighbourhood Structuring elements](image-url)
\[
\sum_{k=0}^{8} C_k = 8C_0 + 8C_1 + 8C_2 + 8C_3 + 8C_4 + 8C_5 + 8C_6 + 8C_7 + 8C_8 = 256
\]

Here, these convex polyhedrons are divided into 9 categories:

- When all the corner pixels are present then the number of possible convex polyhedrons will be \(8C_0\) which is equivalent to 1.
- When one of the corner pixels is not available then the number of possible convex polyhedrons will be \(8C_1\) which is equivalent to 8.
- When two of the corner pixels are not available then the number of possible convex polyhedrons will be \(8C_2\) which is equivalent to 28.
- When three of the corner pixels are not available then the number of possible convex polyhedrons will be \(8C_3\) which is equivalent to 56.
- When all the corner pixels are not available then the number of possible convex polyhedrons will be \(8C_4\) which is equivalent to 70.
- When all the corner pixels are not available then the number of possible convex polyhedrons will be \(8C_5\) which is equivalent to 56.
Table 2 Representation of a possible combination of convex polyhedrons

**Group P** contains 1 possible combination

A = {}  
\[ P_2 = \{1, 3, 7, 9, 19, 21, 25, 27\} \]

**Group Q** contains 8 possible combinations

\[ B_0 = \{1\} \quad B_0 = \{3\} \quad B_0 = \{7\} \quad B_0 = \{9\} \]

\[ Q_0 = \{3, 7, 9, 19, 21, 25, 27\} \quad Q_0 = \{1, 3, 7, 9, 19, 21, 25, 27\} \quad Q_0 = \{1, 3, 7, 9, 19, 21, 25\} \]

**Group R** contains 28 combinations

\[ C_0 = \{1\} \quad C_0 = \{7\} \quad C_0 = \{19\} \quad C_0 = \{25\} \quad C_0 = \{27\} \]

\[ R_0 = \{1, 3, 7, 9, 19, 21, 25, 27\} \quad R_0 = \{3, 7, 9, 19, 21, 25, 27\} \quad R_0 = \{3, 7, 9, 19, 21, 25\} \]

**Group S** contains 56 combinations

\[ D_0 = \{1, 3, 7\} \quad D_0 = \{7, 9\} \quad D_0 = \{1, 21\} \quad D_0 = \{3, 7, 9\} \quad D_0 = \{7, 19\} \]

\[ S_0 = \{1, 3, 7, 9, 19, 21, 25, 27\} \quad S_0 = \{7, 19, 21, 25, 27\} \quad S_0 = \{1, 3, 7, 9, 19, 21\} \]

**Group T** contains 70 combinations

\[ E_0 = \{1, 3, 7\} \quad E_0 = \{3, 7\} \quad E_0 = \{9\} \quad E_0 = \{1\} \quad E_0 = \{1, 3, 7, 9\} \]

\[ T_0 = \{1, 3, 7, 9, 19, 21, 25\} \quad T_0 = \{3, 7, 9, 19, 21, 25\} \quad T_0 = \{1, 3, 7, 9, 19, 21\} \]

\[ E_0 = \{1, 3, 7, 9, 19, 21, 25\} \quad E_0 = \{1, 3, 7, 19, 21, 25\} \quad E_0 = \{1, 3, 7, 19, 21\} \]

\[ T_0 = \{1, 3, 7, 19, 21, 25\} \quad T_0 = \{3, 7, 19, 21, 25\} \quad T_0 = \{1, 3, 7, 19, 21\} \]

\[ E_0 = \{1, 3, 7, 19, 21\} \quad E_0 = \{1, 3, 7, 19\} \quad E_0 = \{1, 3, 7\} \]

\[ T_0 = \{1, 3, 7, 19\} \quad T_0 = \{3, 7, 19\} \quad T_0 = \{1, 3\} \]

\[ E_0 = \{1, 3\} \quad E_0 = \{1\} \quad E_0 = \{} \]
| $E_m$          | $E_p$          | $E_q$          | $E_r$          |
|--------------|--------------|--------------|--------------|
| $T_{1,21,25}$ | $T_{1,21,25}$ | $T_{1,25,27}$ | $T_{1,25,27}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,9,19\}$ | $E = \{1,7,9,21\}$ | $E = \{1,9,19\}$ |
| $T_{7,9,21} = \{3,19,25,27\}$ | $T_{7,9,21} = \{3,19,25,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,9,19\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{1,7,25} = \{3,9,21,27\}$ | $T_{1,7,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{1,7,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{1,9,21} = \{3,9,21,27\}$ | $T_{1,9,21} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{9,21,25} = \{3,9,21,27\}$ | $T_{9,21,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{1,9,21,27} = \{3,9,21,27\}$ | $T_{1,9,21,27} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{3,9,21,27\}$ | $E = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{1,9,21,27} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{3,9,21,25} = \{3,9,21,27\}$ | $T_{3,9,21,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{3,9,21,25} = \{3,9,21,27\}$ | $T_{3,9,21,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{3,9,21,25} = \{3,9,21,27\}$ | $T_{3,9,21,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{3,9,21,25} = \{3,9,21,27\}$ | $T_{3,9,21,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{3,9,21,25} = \{3,9,21,27\}$ | $T_{3,9,21,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{3,9,21,25} = \{3,9,21,27\}$ | $T_{3,9,21,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{3,9,21,25} = \{3,9,21,27\}$ | $T_{3,9,21,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $T_{3,9,21,25} = \{3,9,21,27\}$ | $T_{3,9,21,25} = \{3,9,21,27\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
| $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ | $E = \{1,7,9,21\}$ |
When all the corner pixels are not available then the number of possible convex polyhedrons will be \(8C_6\) which is equivalent to 28.

When all the corner pixels are not available then the number of possible convex polyhedrons will be \(8C_7\) which is equivalent to 8.

Further, all the possible 256 convex polyhedrons are represented in Table 2 and here A, B, C, D, E, F, G, H and I represent the Possible eliminated pixels, and group P, Q, R, S, T, U, V, W, and X represents the possibility of convex polyhedron after removing the pixels.

The visualization of 256 convex polyhedrons which are listed in the above table is shown in Fig. 5.

### 3.2 Feature extraction

Extracting the features from the image is an important step in object recognition. In the proposed method we used picture description language for the representation of the features. The image is traced pixel by pixel which starts from pixel position \((1,1,1)\) to find the initial foreground pixel \((x_i, y_i, z_i)\) with the intensity value 1. This will identify the initial point of the first component present in the image. The proposed algorithm is written in such a way that it will find the next neighbor of the current pixel in all the preferred direction. Preference will be generally given to the right side of the pixel (from the current pixel). These 26 directions are already represented in Fig. 2. The naming conventions of the preferred direction are: Right(R),
Down Right (DR), Down (D), Down Left (DL), Left (L), Up left (UL), Up (U), and Up Right (UR), Back(B), Back Right(BR), Back Down Right (BDR), Back Down (BD), Back Down Left (BDL), Back Left (BL), Back-Up left (BUL), Back-Up (BU), and Back-Up Right (BUR), Front(F), Front Right(FR), Front Down Right (FDR), Front Down (FD), Front Down Left (FDL), Front Left (FL), Front Up left (FUL), Front Up (FU), and Front Up Right (FUR).

In this method, the previously recognized direction has given priority compared to other preferred directions. The tracing of each pixel will continue until the algorithm will not find any connected neighbor pixel to the current position \((x, y)\) and will remove the pixel which is already traced. If there is no neighbor pixel then the algorithm will display the knowledge vector of that component and again algorithm will continue until it will not trace all the components which are present in the image. If all the components are traced then the final knowledge vector will be displayed. This knowledge vector is nothing but a feature vector that consists of the direction and length code of each component and this knowledge vector will give the information that some objects are present in the image. The detailed algorithm is explained in the Appendix 1.

The mentioned algorithm is applied on a simple cube and knowledge vector is extracted and it is represented futher:

\[
<1,1,1>R\ D\ L\ U\ BU\ B\ R\ D\ L\ U\ *99\ 99\ 98\ 1\ 98\ 99\ 99\ 98\ <2,1,100><1,100,2>B\ *
\text{97}\ <1,100,99><100,1,2>B\ *97\ <100,1,99><100,100,2>B\ *97\ <100,100,99>
\]

The part before the * symbol is called direction code (DC) and the following part is called length code (LC).

---

Fig. 5 Three dimensional visualizations of convex polyhedrons
GUI is created to provide the better interface to the user. Sample image of GUI is shown in the Fig. 6.

### 3.3 Reconstruction of original three-dimensional image using knowledge vector

In the previous step, the knowledge vector of the image is identified which is nothing but features of the image. In this step, the feature vector is considered as an input and a novel algorithm is written which converts the knowledge vector into the image. Further, the reconstruction algorithm is explained in the Appendix 2. To make the model easy to use, a graphical user interface (GUI) was made where the text file can be chosen as an input. Once the text file is chosen, the original image is shown on the screen. The same image is shown in Fig. 7.

### 4 Result analysis

Two algorithms are implemented for the proposed work and both algorithms are implemented in python using the Anaconda tool. The experiments are done on a computer with 64GB RAM, 4GB NVIDIA Graphics card, i7 processors, and windows operating system.
4.1 Datasets

The proposed approach is evaluated on two datasets. The first dataset consists of five different folders. All five datasets consists of medical images and a detailed description is given in Table 3. This dataset is collected from Pentagram Research Centre Private Limited, Hyderabad. The second dataset is ‘Kaggle Brain MRI dataset’ which is publicly available on Kaggle [3].

4.2 Evaluation parameters

To evaluate the performance of the proposed methodology some standard image processing evaluation parameters are considered. The First parameter is Accuracy which means how accurately the algorithm can find the feature vector of the image and at the same time how accurately the algorithm can reconstruct the 3D image using knowledge vector.

The other most common parameters of evaluating the quality of image reconstruction are Peak signal to noise ratio (PSNR), Mean squared error (MSE), Structure similarity index (SSIM). These parameters work on the pixel-wise comparison between the original image and the reconstructed image. PSNR (based on the MSE metric) is a ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. The power of corrupting noise is measured by MSE and SSIM is considered as the mean and variance of the image intensities. PSNR is defined as:

$$PSNR = 10 \log_{10} \frac{MAX^2}{MSE}$$ \hspace{1cm} (1)

Where
MAXI is the maximum pixel value of the image and

\[ \text{MSE}(f, g) = \frac{1}{mn} \sum_{i}^{m-1} \sum_{j}^{n-1} (f(i,j) - g(i,j))^2 \]  

The other parameter is the execution time. The execution time is nothing but the time which is required for the reconstruction of the original image using a knowledge vector. To calculate the execution time following formula is used.

\[ \text{Execution time} = \frac{1}{\text{Performance of the Processor}} \]  

4.3 Results

The performance evaluation of any algorithm always depends on the input dataset. Five different datasets are used for the evaluation of the algorithm and different parameters are chosen for the evaluation purpose. The quality of the reconstructed image is represented in the Table 4 and it is visible in some images that reconstructed images are more clear and having sharp edges because the first algorithm works based on convex polyhedrons and our algorithm

| Dataset name            | Details of MRI Image |
|-------------------------|----------------------|
| Medical MRI Dataset     | Toutatix             |
|                         | No. of Images=256    |
|                         | Width=256            |
|                         | Height=256           |
|                         | Depth=256            |
|                         | Max Gray level: 256  |
| Cerebrix                | No. of Images=43     |
|                         | Width=256            |
|                         | Height=256           |
|                         | Depth=43 Max Gray level: 256 |
| Cetautomatix            | No. of Images=256    |
|                         | Width=256            |
|                         | Height=256           |
|                         | Depth=256 Max Gray level: 114 |
| MRI                     | No. of Images=256    |
|                         | Width=256            |
|                         | Height=256           |
|                         | Depth=256 Max Gray level: 181 |
| Cell Farct              | No. of Images=256    |
|                         | Width=256            |
|                         | Height=256           |
|                         | Depth=256 Max Gray level: 256 |
| Kaggle Brain MRI Dataset| No. of Images=253    |
|                         | Width=845            |
|                         | Height=845           |
|                         | Depth=253            |
|                         | Max Gray level: 256  |
Table 4  Reconstruction of Image on different datasets

|                  | Toutatix | Cerebrix | Cetautomatix | MRI         | Cell Farcet |
|------------------|----------|----------|--------------|-------------|-------------|
| Front View       | ![Image](image1) | ![Image](image2) | ![Image](image3) | ![Image](image4) | ![Image](image5) |
| Left View        | ![Image](image6) | ![Image](image7) | ![Image](image8) | ![Image](image9) | ![Image](image10) |
| Right View       | ![Image](image11) | ![Image](image12) | ![Image](image13) | ![Image](image14) | ![Image](image15) |
| Top View         | ![Image](image16) | ![Image](image17) | ![Image](image18) | ![Image](image19) | ![Image](image20) |
| Bottom View      | ![Image](image21) | ![Image](image22) | ![Image](image23) | ![Image](image24) | ![Image](image25) |
| Side View        | ![Image](image26) | ![Image](image27) | ![Image](image28) | ![Image](image29) | ![Image](image30) |
| Zoom In          | ![Image](image31) | ![Image](image32) | ![Image](image33) | ![Image](image34) | ![Image](image35) |
| Zoom Out         | ![Image](image36) | ![Image](image37) | ![Image](image38) | ![Image](image39) | ![Image](image40) |
Many researchers are working on the reconstruction of a 3D image because the minimally invasive procedure is getting popular nowadays (Fig. 8). After all, it provides better accuracy and minimum recovery time. In the above-mentioned method, the surgical navigation system will always play an important and critical role. It helps the doctors to focus on a particular position, edge, or outline of the image. But Most of the surgical navigation systems are based on two-dimensional medical images but very little development in the field of three-dimensional medical images. Therefore, a platform is developed for the reconstruction of the three-dimensional images. This System takes the series of 2D/3D medical images as an input and converts these series of images into textual form first and then in the second stage this textual information/knowledge vector would be taken as an input and reconstruction of the 3D

| Dataset          | PSNR     | SSIM   |
|------------------|----------|--------|
| Toutatix         | 32.59 ± 1.20 | 0.93 ± 0.01 |
| Cerebrix         | 31.67 ± 1.47 | 0.91 ± 0.03 |
| Cetautomatix     | 29.86 ± 1.20 | 0.87 ± 0.03 |
| MRI              | 32.57 ± 1.20 | 0.90 ± 0.01 |
| Cell Farct       | 31.26 ± 1.80 | 0.89 ± 0.01 |
| Kaggle Brain MRI Dataset | 32.50 ± 1.17 | 0.90 ± 0.02 |

Table 5 PSNR and SSIM value on different medical datasets

![Accuracy and execution time comparison](image)

Fig. 8 Comparative analysis of accuracy and execution time

Table 6 Comparative analysis between proposed work and other state of art methods in terms of accuracy and execution time on kaggle Brain MRI dataset

| Authors          | Techniques used                   | Accuracy | Execution time(Sec.) |
|------------------|-----------------------------------|----------|----------------------|
| Kalaiselvi T et al. [35] | FCM and SRG based segmentation method | 62.84%   | 7.12                 |
| Baijiang Fan [9]    | Novel region based growing algorithm          | 90.52%  | 12.51               |
| Jun Chen et al. [4]  | 3D-AlexNet                          | 92.11%   | –                    |
| Jun Chen et al. [4]  | Inception-V4                       | 87.60%   | –                    |
| Jun Chen et al. [4]  | ResNet 50                          | 85.70%   | –                    |
| Proposed work       | Syntactic Pattern based approach     | 93.89%   | 4.02                 |
image would be done in this stage. It displays the human organs on the computer screen which gives convenience to the doctors for the diagnosis of the disease. This three-dimensional reconstruction gives the facility to doctors to focus on a particular point and find the nature of the lesion and its surrounding tissues, rotation of the image, traverse the image from inside, zoom in and zoom out, thus it helps the doctors to diagnose the diseases and its severity and improve the accuracy and decrease the diagnosis time. The result of the proposed method is evaluated on the basis of qualitative analysis and quantitative analysis. PSNR and SSIM are calculated on all five datasets and it is represented in Table 5. Most of the reconstruction algorithm takes more time for reconstruction of 3D images. As compared to the existing methods, proposed method takes very less time in the reconstruction of the 3D images and results are shown in the Tables 6 and 7. The other parameter of evaluation for the performance of the proposed algorithm is Accuracy. As per the experimental analysis, the proposed method gives 93.89% of accuracy on Kaggle brain MRI dataset and 97.24% on Medical MRI dataset which is better than exiting state of art methods. The average execution time on Medical dataset and Kaggle brain MRI dataset is 1.12 sec. and 4.02 sec which is again lesser than the existing methods.

### 5 Conclusion and future work

In this paper, we have presented a syntactic pattern recognition based approach that can represent the image into textual form and a novel algorithm is proposed for the reconstruction of three-dimensional images. Reconstruction allows us to explore the internal details of the 3D images such as the size, shape, and structure of the object which could take us one step ahead in the field of medical image processing. GUI is set up so that the image can be seen correctly, and it has features that let the image be rotated in all directions and zoomed in and out so that the image’s inner details can be seen clearly. Performances of the proposed algorithms are evaluated on a medical image dataset and results are outperformed in real-time. As per the experimental analysis, the proposed method gives 93.89% of accuracy on Kaggle brain MRI dataset and 97.24% on Medical MRI dataset which is better than exiting state of art methods. This approach is better than other existing approaches because there are very few steps of reconstruction of a 3D image which makes this approach unique.

The classification task of a structural pattern recognition system is difficult to implement because the syntactic pattern recognition embodies the precise criteria which discriminate among groups and, therefore, they are by their very nature domain- and application-specific. In the future, we will try to extend the paper by implementing classification techniques for object recognition.
Appendix 1

| Input: Given Image | Output: Knowledge vector |
|-------------------|--------------------------|

Begin
Initialize
   Dimension of the image
   i,j,k=dim
   length = []
   direction = []
arr = []
loader = []
image = []
arr2 = []
arr_i = []
arr_j = []
arr_k = []

Assign the dimension values to the variable dim.
Create an empty container of the same size.
Create a 3D array (img) of the same size as the dimension variable and initialize it as zero.

Give the path of the dataset and read the images from the dataset and store the value in array im.
   for k in range(dim):
      for i in range(dim):
         for j in range(dim):
            if im[i][j][k]!=0
               then assign img[m][i+1][j+1]==255
               store_value function(i, j, k)
               append
                i value in arr_i
                j value in arr_j
                k value in arr_k

Traversing function(x, y, z)
   Initialize the x, y, z value to i, j, k
   Initialize the img array as 0
   Call Store_value (i, j, k)
   Initialize the loader as [0 0]

While (image[k, i, j]!=[0 0])
   #Right direction
      if img(k, i, j+1)==255:
         if R is available in loader then
            image[k, i, j+1] = 0
            j=j+1
            call store(i, j, k)
            Loader[1]=loader[1]+1
         else:
            append the direction at loader[0]
            append the length with {} at loader [1]
            clear the loader
            append R
            append 0 to the loader
   #Down right direction
   else if img(k, i+1, j+1)==255:
      if DR is available in loader then
         image[k, i, j+1] = 0
         j=j+1
         call store(i, j, k)
         Loader[1]=loader[1]+1
      else:
         append the direction at loader[0]
         append the length with {} at loader [1]
         clear the loader
         append DR
         append 0 to the loader
   #Down direction
   else if img(k, i+1, j)==255:
      if D is available in loader then
         image[k, i, j+1] = 0
         j=j+1
         call store(i, j, k)
         Loader[1]=loader[1]+1
      else:
         append the direction at loader[0]
         append the length with {} at loader [1]
         clear the loader
         append D
         append 0 to the loader
   #Down left direction
   else if img(k, i+1, j-1)==255:
      if DL is available in loader then
         image[k, i, j+1] = 0
         j=j+1
         call store(i, j, k)
         Loader[1]=loader[1]+1
      else:
         append the direction at loader[0]
         append the length with {} at loader [1]
         clear the loader
         append DL
         append 0 to the loader
   #Left direction
   else if img(k, i, j-1)==255:
      if L is available in loader then
         image[k, i, j+1] = 0
         j=j+1
         call store(i, j, k)
         Loader[1]=loader[1]+1
      else:
         append the direction at loader[0]
         append the length with {} at loader [1]
         clear the loader
         append L
         append 0 to the loader
   #Down left direction

36469
Multimedia Tools and Applications (2023) 82:36449–36477
append the length with {} at loader[1]
clear the loader
append L
append 0 to the loader
#up left direction
else if img(k, i-1, j-1)==255:
if UL is available in loader then
image[k, i, j+1] = 0
j+=1
call store(i, j, k)
Loader[1]=Loader[1]+1
else:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append UL
append 0 to the loader
#Up direction
else if img(k, i-1, j)==255:
if U is available in loader then
image[k, i, j+1] = 0
j+=1
call store(i, j, k)
 Loader[1]=Loader[1]+1
else:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append U
append 0 to the loader
#Up Right direction
else if img(k, i-1, j+1)==255:
if UR is available in loader then
image[k, i, j+1] = 0
j+=1
call store(i, j, k)
Loader[1]=Loader[1]+1
else:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append UR
append 0 to the loader
#Back direction
else if img(k+1, i, j)==255:
if B is available in loader then
image[k, i, j+1] = 0
j+=1
call store(i, j, k)
Loader[1]=Loader[1]+1
else:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append B
append 0 to the loader
#Back right direction
else if img(k+1, i, j+1)==255:
if BR is available in loader then
image[k, i, j+1] = 0
j+=1
call store(i, j, k)
Loader[1]=Loader[1]+1
else:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append BR
append 0 to the loader
#Back down right direction
else if img(k+1, i+1, j+1)==255:
if BDR is available in loader then
image[k, i, j+1] = 0
j+=1
call store(i, j, k)
Loader[1]=Loader[1]+1
else:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append BDR
append 0 to the loader
# Back down direction
else if img(k+1, i+1, j)==255:
if BD is available in loader then
image[k, i, j+1] = 0
j+=1
call store(i, j, k)
Loader[1]=Loader[1]+1
else:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append BD
append 0 to the loader
# Back down left direction
else if img(k+1, i+1, j-1)==255:
if BDL is available in loader then
image[k, i, j+1] = 0
j+=1
call store(i, j, k)
Loader[1]=Loader[1]+1
else:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append BDL
append 0 to the loader
# Back left direction
else if img(k+1, i, j-1)==255:
if BL is available in loader then
```python
image[k, i, j+1] = 0
j=j+1
call store(i, j, k)
Loader[1]=loader[1]+1
def:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append BL
append 0 to the loader
# Back up left direction
def if img(k+1, i-1, j-1)==255:
if BUL is available in loader then
image[k, i, j+1] = 0
j=j+1
call store(i, j, k)
Loader[1]=loader[1]+1
def:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append BUL
append 0 to the loader
# Back up direction
def if img(k+1, i-1, j)==255:
if BU is available in loader then
image[k, i, j+1] = 0
j=j+1
call store(i, j, k)
Loader[1]=loader[1]+1
def:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append BU
append 0 to the loader
# Forward right direction
def if img(k-1, i+1, j+1)==255:
if FDR is available in loader then
image[k, i, j+1] = 0
j=j+1
call store(i, j, k)
Loader[1]=loader[1]+1
def:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append FDR
append 0 to the loader
# Forward down right direction
def if img(k-1, i+1, j)==255:
if FD is available in loader then
image[k, i, j+1] = 0
j=j+1
call store(i, j, k)
Loader[1]=loader[1]+1
def:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append FD
append 0 to the loader
# Forward down direction
def if img(k-1, i+1, j-1)==255:
if FDL is available in loader then
image[k, i, j+1] = 0
j=j+1
call store(i, j, k)
Loader[1]=loader[1]+1
def:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append FDL
append 0 to the loader
# Forward down left direction
def if img(k-1, i+1, j)==255:
if FD is available in loader then
image[k, i, j+1] = 0
j=j+1
call store(i, j, k)
Loader[1]=loader[1]+1
def:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append FD
append 0 to the loader
# Forward left direction
def if img(k+1, i-1, j-1)==255:
if BL is available in loader then
image[k, i, j+1] = 0
j=j+1
call store(i, j, k)
Loader[1]=loader[1]+1
def:
append the direction at loader[0]
append the length with {} at loader[1]
clear the loader
append BL
append 0 to the loader
# Forward direction
```

---

**Multimedia Tools and Applications (2023) 82:36449–36477**
append FDL
append 0 to the loader

# Forward left direction
else if img(k-1, i, j-1)==255:
    if FL is available in loader then
        image [k, i, j+1] = 0
        j=j+1
        call store(i, j, k)
        Loader[1]=loader[1]+1
    else:
        append the direction at loader[0]
        append the length with {} at loader [1]
        clear the loader
        append FL
        append 0 to the loader

# Forward up left direction
else if img(k-1, i-1, j-1)==255:
    if FUL is available in loader then
        image [k, i, j+1] = 0
        j=j+1
        call store(i, j, k)
        Loader[1]=loader[1]+1
    else:
        append the direction at loader[0]
        append the length with {} at loader [1]
        clear the loader
        append FUL
        append 0 to the loader

# Forward up direction
else if img(k-1, i-1, j)==255:
    if FU is available in loader then
        image [k, i, j+1] = 0
        j=j+1
        call store(i, j, k)
        Loader[1]=loader[1]+1
    else:
        append the direction at loader[0]
        append the length with {} at loader [1]
        clear the loader
        append FU
        append 0 to the loader

# Forward up Right direction
else if img(k-1, i-1, j+1)==255:
    if FUR is available in loader then
        image [k, i, j+1] = 0
        j=j+1
        call store(i, j, k)
        Loader[1]=loader[1]+1
    else:
        append the direction at loader[0]
        append the length with {} at loader [1]
        clear the loader
        append FUR
        append 0 to the loader

# Back down left direction
else if img(k+1, i+1, j-1)==255:
    if BDL is available in loader then
        image [k, i, j+1] = 0
        j=j+1
        call store(i, j, k)
        Loader[1]=loader[1]+1
    else:
        append the direction at loader[0]
        append the length with {} at loader [1]
        clear the loader
        append BDL
        append 0 to the loader
else:
    Break
    append the direction at loader[0]
    append the length with {} at loader [1]
    clear the loader
    pop up the direction
    pop up the length
    add the direction to the arr
    append "*" to arr
    add the length to arr
    Append the i, j, k value in this format <{},{},{}>
    Clear the direction array
    Clear the length array

for k in range (dim):
    for i in range (dim):
        for j in range (dim):
            if img[k][i][j]==255:
                Then append < { }, { }, { }> in arr call
                traversing function by passing the value of i, j and
                k. (traversing(i,j,k))
                Initialize str=""
                for element in arr:
                    str=str+ element
                open a text file and write the content of str into the
                text file
                Close the text file.
End
## Appendix 2

**Input:** Text file  
**Output:** Image

**Begin**
Read a text file and store it in an `str` array  
Declare array `p`, `q`, and `r`, and `s`, `x`, and `str1`  
Read the value of `str` and replace `"\<"` with `"---"`  
and store in `str1` array  
Split the array values on basis of this `"---"` and store  
the result in `x`  
for `i` in the range of(len(`x`)):  
do the following replacement in `x` and again  
keep the value in `x`  
replace `"\>"` with `"\*"`  
replace `"\<"` with `"\*"`  
replace `"\<"` with `""`  
replace `"\>"`, with `""`  
Declare 3 variables with the name `Direction`, `Length`,  
and `Start`.  
For `i` in range(length(`x`)):  
`a=x[i]`  
Split the value of array `a` on basis of `"\*"` and store in  
result in an again.  
If the length of the array is equal to 1 or 3 then:  
Assign the value of `a[0]` to `Start`.  
Split the value of array `Start` on basis of `","` and  
store in result in `Start` again.  
Assign the value of `Start [0]`, `Start [1]`, and `Start`  
`2` to `i`, `j`, and `k`.  
Append the value of `j`, `i`, `k` in `p`, `q`, `r`.  

| If the length of a is equal to 1  
| Then print the value of a  
| else  
| Assign `a[0]` value to `Start` Array.  
| Assign `a[1]` value to the `Direction` array.  
| Assign `a[2]` value to the `Length` array.  
| Assign `a[3]` value to the `End` array.  
Call `Printer Function` bypassing `Direction`,  
`Length`, and `Start` Parameter.  
Display the 3D Image on Screen Using graph plot. |

**Printer function**(`Direction`, `Length`, `Start`):  
Split the `Start` array on basis of `"\,"` and store the  
resultant array in `the Start` array.  
Split the `Length` array on basis of `"\,"` and store the  
resultant array in `the Length` array.  
Split the `Direction` array on basis of `"\,"` and store the  
resultant array in `the Direction` array.  
Store the `Start [0]` value in variable `i`.  
Store the `Start [1]` value in variable `j`.  
Store the `Start [2]` value in variable `k`.  
Now append the value of `i`, `j`, `k` in `p`, `q`, `r`.  
Display the value of `Length` array and `Direction`  
array.  
for `m` in range(`Length(Direction)`-1):  
Calculate the `Length[m]` and store the result in  
variable `L`.  
#Right Direction  
**If** `Direction[m]`="R"  
`z=j+L`
while(j<z):
    Increment j value by 1.
    Append the value of j, i, k in p, q, r.
#Down Direction
Else If Direction[m]=="D"
    z=i+L
    while(i<z):
        Increment i value by 1.
        Append the value of j, i, k in p, q, r.
#Left Direction
Else If Direction[m]=="L"
    z=j-L
    while(j<z):
        Decrement j value by 1.
        Append the value of j, i, k in p, q, r.
#up Direction
Else If Direction[m]=="U"
    z=i-L
    while(i<z):
        Increment i value by 1.
        Append the value of j, i, k in p, q, r.
#Down right Direction
Else If Direction[m]=="DR"
    z=j+i
    while(j<z):
        Increment i & j value by 1.
        Append the value of j, i, k in p, q, r.
#Down Left Direction
Else If Direction[m]=="DL"
    z=i+j
    while(i<z):
        Increment i & j value by 1.
        Append the value of j, i, k in p, q, r.
#up Right Direction
Else If Direction[m]=="UR"
    z=j+i
    while(i<z):
        Increment i & j value by 1.
        Append the value of j, i, k in p, q, r.
#Back Direction
Else If Direction[m]=="B"
    z=k+L
    while(k<z):
        Increment k value by 1.
        Append the value of j, i, k in p, q, r.
#Back right Direction
Else If Direction[m]=="BR"
    z=j+L
    while(j<z):
        Increment j & k value by 1.
        Append the value of j, i, k in p, q, r.
#Back down Direction
Else If Direction[m]=="BD"
    z=i+L
    while(i<z):
        Increment i and k value by 1.
        Append the value of j, i, k in p, q, r.
#Back Left Direction
Else If Direction[m]=="BL"
    z=j-L
    while(j>z):
        Increment k value by 1.
        Decrement j value by 1
        Append the value of j, i, k in p, q, r.
#Back up Direction
Else If Direction[m]=="BU"
    z=i-L
    while(i>z):
        Increment k value by 1.
        Decrement i value by 1
        Append the value of j, i, k in p, q, r.
#Back down right Direction
Else If Direction[m]=="BDR"
    z=j+i
    while(j>z):
        Increment i, j and k value by 1.
        Decrement i value by 1
        Append the value of j, i, k in p, q, r.
#Back down left Direction
Else If Direction[m]=="BDL"
    z=i+j
    while(i>z):
        Increment i and k value by 1.
        Decrement j value by 1
        Append the value of j, i, k in p, q, r.
#Back up Right Direction
Else If Direction[m]=="BUR"
    z=i+L
    while(i>z):
        Increment j and k value by 1.
        Decrement i value by 1
        Append the value of j, i, k in p, q, r.
#Forward Direction
Else If Direction[m]=="F"
    z=k-L
    while(k>z):
        Increment k value by 1
Append the value of j, i, k in p, q, r.

# Forward Right Direction

Else If Direction[m]=="FR"
    z=j+L
    while(j<z):
        Increment j value by 1.
        Decrement k value by 1
        Append the value of j, i, k in p, q, r.

# Forward Down Direction

Else If Direction[m]=="FD"
    z=i+L
    while(i<z):
        Increment i value by 1.
        Decrement k value by 1
        Append the value of j, i, k in p, q, r.

# Forward left Direction

Else If Direction[m]=="FL"
    z=j-L
    while(j>z):
        Decrement j and k value by 1
        Append the value of j, i, k in p, q, r.

# Forward up Direction

Else If Direction[m]=="FU"
    z=i-L
    while(i>z):
        Decrement i and k value by 1
        Append the value of j, i, k in p, q, r.

# Forward down right Direction

Else If Direction[m]=="FDR"

z=j+L
    while(j<z):
        Increment i and j value by 1.
        Decrement k value by 1
        Append the value of j, i, k in p, q, r.

# Forward down left Direction

Else If Direction[m]=="FDL"
    z=i+L
    while(i<z):
        Increment i value by 1.
        Decrement j and k value by 1
        Append the value of j, i, k in p, q, r.

# Forward up left Direction

Else If Direction[m]=="FUL"
    z=j-L
    while(j>z):
        Decrement j, i and k value by 1
        Append the value of j, i, k in p, q, r.

# Forward up right Direction

Else If Direction[m]=="FUR"
    z=i-L
    while(i>z):
        Increment j value by 1.
        Decrement i and k value by 1
        Append the value of j, i, k in p, q, r.

END
Acknowledgments  The constant support of Prof. Dr. E. G. Rajan, President, Pentagram Research, and Director, Avatar MedVision US LLC NC USA has resulted to bring the results of the research paper. The authors express their sincere thanks to Mr. Pawan Rao, a Data Scientist in the Pentagram Research Centre, and Hyderabad for his valuable suggestion during the preparation of this research paper.

Code availability  Implementation of algorithm is available.

Data availability  Data and other related material are available.

Declarations

Conflict of interest  On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethical approval  This article does not contain any studies with human participants performed by any of the authors.

Consent to participate  We are voluntarily agreed to participate in this research study.

Consent for publication  All the authors have given their consent for the manuscript to be published in your Journal.

References

1. Benalcazar DP, Bastias D, Perez CA, Bowyer KW (2019) A 3D Iris Scanner from multiple 2D visible light images. IEEE Access 7:61461–61472
2. Bustin A, Voilliot D, Menini A, Felblinger J, de Chillou C, Burschka D, Bonnemains L, Odille F (2018) Isotropic reconstruction of MR images using 3D patch-based self-similarity learning. IEEE Trans Med Imaging 37(8):1932–1942
3. Chakrabarty N (2019) Brain MRI images dataset for brain tumor detection, Kaggle, 2019, Available online: https://www.kaggle.com/navoneel/brain-mri-images-forbrain-tumor-detection. Accessed 1 April 2021
4. Chen J, Wan Z, Zhang J, Li W, Chen Y, Li Y, Duan Y (2020) Medical image segmentation and reconstruction of prostate tumor based on 3D AlexNet. Computer Methods and Programs in Biomedicine, pp 105878.
5. Choudhary S, Lakhwani K, Kumar S (2022) Three-dimensional objects recognition & pattern recognition technique: related challenges: a review. Multimed Tool Appl 23(1):1–44
6. Ding L, Déan-Ben XL, Razansky D (2017) Efficient 3-D model-based reconstruction scheme for arbitrary optoacoustic acquisition geometries. IEEE Trans Med Imaging 36(9):1858–1867
7. Djelouah A, Franco J, Boyer E, Le Clerc F, Perez P (2012) N-tuple color segmentation for multi-view silhouette extraction. European conference on computer vision. Springer, pp 818–831
8. Duda RO, Hart PE, Stork DE (2001) Pattern classification, 2nd edn. Wiley, New York
9. Fan B, Rao Y, Liu W, Wang Q, Wen H (2017) Region-based growing algorithm for 3D reconstruction from MRI images. In: 2nd international conference on image, vision, and computing (ICIVC), IEEE, pp 521–525
10. Fitzgibbon A, Cross G, Zisserman A (1998) Automatic 3d model construction for turn-table sequences. European workshop on 3D structure from multiple images of large-scale environments. Springer, pp 155–170
11. Franco J, Boyer E (2003) Exact polyhedral visual hulls. British Machine Vision Conference vol 1, pp 329–338
12. Fu KS (1982) Syntactic pattern recognition and applications. Prentice-Hall, Englewood Cliffs, New Jersey
13. Fukunaga K (2009) Introduction to statistical pattern recognition, 2nd edn. Academic Press, Boston
14. Furukawa Y, Ponce J (2009) Carved visual hulls for image-based modeling. Int J Comput Vis 81(1):53–67
15. Gonzalez RC, Thomason MG (1978) Syntactic pattern recognition: an introduction. Addison-Wesley, Reading, Massachusetts
16. Guo Y, Zhang Y, Verbeek FJ (2017) A two-phase 3-d reconstruction approach for light microscopy axial-view imaging. IEEE J Sel Top Signal Process 11(7):1034–1046
17. Han X, Leung T, Jia Y, Sukthankar R, Berg A (2015) Matchnet: unifying feature and metric learning for patch-based matching. In: IEEE conference on computer vision and pattern recognition, pp 3279–3286
18. Hou B, Khanal B, Alansary A, McDonagh S, Davidson A, Rutherford M, Hajnal JV, Rueckert D, Glocke B, Kainz B (2018) 3-D reconstruction in canonical coordinate space from arbitrarily oriented 2-D images. IEEE Trans Med Imaging 37(8):1737–1750
19. Ihrke I, Kutulakos K, Lensch H, Magnor M, Heidrich W (2010) Transparent and specular object reconstruction. In: Computer Graphics Forum, Wiley Online Library, vol 29, no 8, pp 2400–2426
20. Jain AK, Duin RPW, Mao J Statistical pattern recognition: a review. IEEE Trans Pattern Anal Mach Intell 22(1):4–37
21. Kutulakos K, Seitz S (2000) A theory of shape by space carving. Int J Comput Vis 38(3):199–218
22. Lazebnik S, Funukawa Y, Ponce J (2007) Projective visual hulls. Int J Comput Vis 74(2):137–165
23. Li Y, Chen G-H (2018) An empirical data inconsistency metric (DIM) driven CT image reconstruction method. IEEE Trans Med Imaging 38(2):337–348
24. Li B, Zhang Y, Zhao B, Shao H (2020) 3D-ReConstnet: a single-view 3D-object point cloud reconstruction network. IEEE Access 8:83782–83790
25. Liu D, Chen X, Yang Y (2014) Frequency-based 3d reconstruction of transparent and specular objects. In: IEEE conference on computer vision and pattern recognition, pp 660–667
26. Pavlidis T (1977) Structural pattern recognition. Springer-Verlag, Berlin
27. Qian Y, Gong M, Yang Y (2016) 3D reconstruction of transparent objects with position-normal consisten-
cy. In: IEEE conference on computer vision and pattern recognition, pp 4369–4377
28. Rani S, Lakhwani K, Kumar S (2020) Three-dimensional wireframe model of medical and complex images using cellular logic array processing techniques. In: International conference on soft computing and pattern recognition. Springer, Cham, pp 196–207
29. Rani S, Ghai D, Kumar S, Kantipudi MVV, Alharbi AH, Ullah MA (2022) Efficient 3D AlexNet architecture for object recognition using syntactic patterns from medical images. Comput Intell Neurosci: 1–19
30. Rani S, Ghai D, Kumar S (2022) Reconstruction of simple and complex three-dimensional images using pattern recognition algorithm. J Inf Technol Manag 235–247
31. Rani S, Ghai D, Kumar S (2022) Object detection and recognition using contour based edge detection and fast R-CNN. Multimed Tool Appl 22(2):1–25
32. Rani S, Ghai D, Kumar S Reconstruction of wire frame model of complex images using syntactic pattern recognition. J Inf Technol Manag 8:1–13
33. Savinov N, Hne C, Ladicky L, Pollefeys M (2016) Semantic 3d reconstruction with continuous regularization and ray potentials using a visibility consistency constraint. IEEE
34. Seitz S, Curless B, Diebel J, Scharstein D, Szeliski R (2006) A comparison and evaluation of multi-view stereo reconstruction algorithms. In: IEEE conference on computer vision and pattern recognition, vol 1, pp 519–528
35. Thiruvenkadak M, Nagaraja P (2015) Brain tumor segmentation of MRI brain images through FCM clustering and seeded region growing technique. Int J Appl Eng Res 10(76):427–443
36. Zbontar J, LeCun Y (2016) Stereo matching by training a convolutional neural network to compare image patches. J Mach Learn Res 17:1–32
37. Zhang C (2019) CuFusion2: accurate and denoised volumetric 3D object reconstruction using depth cameras. IEEE Access 7:49882–49893
38. Zheng X, Ravishankar S, Long Y, Fessler JA (2018) PWLS-ULTRA: an efficient clustering and learning-
based approach for low-dose 3D CT image reconstruction. IEEE Trans Med Imaging 37(6):1498–1510

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.