Contents lists available at ScienceDirect
Computer Methods and Programs in Biomedicine
journal homepage: www.elsevier.com/locate/cmpb

An efficient architecture for medical high-resolution images transmission in mobile telemedicine systems

Lijun Liu[a,b,c], Lizhen Wang[a,*], Qingsong huang[b,c], Lihua Zhou[a], Xiaodong Fu[b,c], Li Liu[b,c]

[a] *Department of Computer Science and Engineering, School of Information Science and Engineering, Yunnan University, Kunming 650091, China*
[b] *Computer Technology Application Key Laboratory of Yunnan Province (Faculty of Information Engineering and Automation, Kunming University of Science and Technology), Kunming, 650500, China*
[c] *Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China*

ARTICLE INFO

Article history:
Received 15 June 2019
Revised 17 September 2019
Accepted 21 September 2019
Available online xxx

Keywords:
Medical high-resolution image
Unbalance image pyramid
Image transmission
Mobile telemedicine
DICOM and DICOM web service

ABSTRACT

Background and Objective: The medical high-resolution image is very important in image processing and computer vision applications, which plays a critical role in image-guided diagnosis, clinical trials, consultation, and case discussion. How to efficiently access medical high-resolution images in mobile telemedicine systems is becoming a big challenge. Therefore, this work proposes an efficient pyramid architecture for optimizing medical high-resolution images transmission and rendering.
Methods: The proposed architecture consists of three core schemes: (1) unbalance pyramid scheme based on geometric relationship, (2) indexing scheme based on hash table and lattice partitioning and (3) query scheme based on similar matching. Then, we design the responsive service components: generating service, indexing service, and query service. Finally, these services are combined into a prototype system that enables efficient transmission and rendering of medical high-resolution images.
Results: The result shows that the unbalance pyramid scheme can quickly generate the pyramid structure and the corresponding image files. The indexing scheme can create the index structure and the index file in real-time. The query scheme can not only match the best layer to which the image block belongs in real-time, but also can accurately capture the query image block.
Conclusions: The prototype system based on proposed architecture is fully compliant with the DICOM standard, which can be seamlessly integrated with other existing medical systems or mobile applications, and used in various scenarios such as diagnosis, research, and education.

© 2019 Published by Elsevier B.V.

1. Introduction

Recent years have seen great development in the field of medical imaging and the amount of high-resolution images (HRIs) is fast growing [1–4]. Medical HRIs are very important in image processing and computer vision applications such as: CT scan-chest, MRI, X-rays and Optical Coherence Tomography (OCT) [5], which contain detailed anatomical structure and have been important data sources that play a critical role in image-guided diagnosis, clinical trials, consultation, and case discussion [4, 6–10]. Telemedicine is the medical practice of information exchanged from one location to another through Information and Communication Technologies (ICTs) to improve remote and flexible medical and health care services [11–13]. As smart mobile devices have excellent resolutions and have become better connected to the internet, the mobile telemedicine system allows patients to receive medical treatments from the doctors via internet technology without visiting hospitals in person [14–16]. Thus, health professionals and patients must prepare themselves to take advantage of established and enhanced health care processes to improve a patient's clinical health status [17]. Additionally, collaborative processes are very important in telemedicine domain since they allow for making right decisions in complex situations [18]. For medical high-resolution images, on the one hand, doctors need to conveniently access medical images and have discussions with other doctors for online diagnosis or teleconsultation [7, 19–21]. On the other, patients need to access their medical images and examination reports from anywhere for online consultation [7, 21]. Consequently, how to rapidly and conveniently access medical HRIs in a diversified network environment on different client devices is becoming an important requirement [7, 22].

Although considerable research efforts have been carried out on medical image transmission [23], most of them focus on three

* Corresponding author.
E-mail address: lzhwang@ynu.edu.cn (L. Wang).

https://doi.org/10.1016/j.cmpb.2019.105088
0169-2607/© 2019 Published by Elsevier B.V.

ways: compressed transmission [2, 24–27], progressive transmission [24–26, 28–30], and on-demand transmission [7, 21–23, 31, 32]. However, the most methods have some defects such as the lack of losing important information [7], repeated and invalid transmission, special coding and decoding, and time-consuming [7, 23, 33]. Therefore, transmitting and rendering medical HRIs in the mobile network is still challenging. It is worth mentioning that a typical medical HRI has at least a dozen megabytes in size, and sometimes tens of megabytes or even gigabytes [7, 34, 35]. Obviously, transmitting the entire RHI over the internet is time-consuming. The second difficulty lies in the instability and bandwidth limitations of the 4G/5G mobile network. Despite the 4G/5G mobile network has been greatly developed, the bandwidth is usually unstable and limited for transmitting medical HRIs, especially in the rural or remote areas of developing countries [7, 23]. Moreover, the HRIs cannot be fully displayed in their maximum resolution on the display devices, especially on the mobile devices, due to their limited resources (screen, resolution, memory, etc.) [23, 36, 37].

In this paper, we propose a new generation, index and query method of the typical image pyramid scheme (TIPS) in the architecture for optimizing HRIs transmission and rendering, which encompasses the following key features: (1) We present a novel image layering method based on geometric relation, in which different image layer distributions can be automatically identified by adjusting an unbalance factor (β); (2) We design an index structure and indexing algorithm based on hash table and lattice partitioning to improve the query efficiency of image blocks; (3) We devise a query scheme based on similar matching, in which the query image block can be quickly matched; (4) Our prototype system based on the architecture is fully compliant with digital imaging and communications in medicine (DICOM) [38], including DICOMweb [39] and data formats, which can be seamlessly integrated with other existing medical systems or mobile applications and used in various scenarios such as diagnosis, research, and education.

The remainder of the paper is organized as follows. Section 2 reviews the preliminary knowledge of medical image and standard, key technologies and related works. Section 3 describes preliminary definitions and computational methods that are employed in this paper. After that, the system architecture, workflow, and service components are proposed in Section 4. We perform comprehensive experiments to evaluate the efficiency of our proposed approach in Section 5. At the end of this section, we discuss the related works and gives the limitations of our proposed approach before we conclude the paper in Section 6.

2. Background

2.1. Medical HRIs: medical high-resolution images

Modern medical imaging technology provides the amount of HRIs, including radiological photos, clinical images, etc., which contain detailed anatomical structure and help doctors to diagnose [1, 2, 6–8]. The distinguishing features of medical images are high dimensional, high resolution and large storage space [40]. A typical medical HRI has at least a dozen megabytes in size, and sometimes tens of megabytes or even gigabytes [23, 34]. For example, a computed radiography (CR) or digital radiography (DR) image is at least a dozen megabytes in size, a digital mammography (MG) image is tens of megabytes in size [34] and a typical slide scanned at 40 (approx. 1600 megapixels) produces a file with several gigabytes [23, 35]. Compared with other images, the backgrounds of medical images are relatively simple yet similar, and the medically useful areas (MUA) is only part of the medical image [23]. Therefore, on-demand transmission is a good solution to effectively reduce the amount of data transmission.

Table 1
DICOM services and DICOMweb services are used in this paper.

Service	Description
C-FIND	Query DICOM objects from PACS
C-GET	Transfer of DICOM objects, where receiver initiates connection
C-MOVE	Transfer of DICOM objects, where receiver does not need to initiate connection
QIDO-RS	Search for studies, series, and instances that match specified the search parameters
WADO-URI	Download DICOM files and parse the original image from the file

2.2. DICOM and DICOMweb service

The storage and communication of medical images are usually compliant with DICOM standard, which not only defines the storage format for medical images, but also defines the communication protocol for exchanging images and related meta-data among different picture archiving and communication system (PACS) [28, 38]. Currently, DICOM is a widely accepted standard for medical image storage and delivery among medical devices, which is widely used for querying studies, reviewing images and performing diagnosis. It is a bidirectional data movement across DICOM network via DICOM services such as: C-STORE, C-FIND, C-MOVE, C-GET, etc., which allow data, images and metadata to be stored, searched and retrieved in local area network [28, 41]. However, with the rapid development of telemedicine, doctors and patients are beginning to expect medical images to be available on the mobile device for making a diagnosis or consultative viewing via the Internet. In this context, DICOMweb service [39], as an extension of DICOM service (DICOM PS3.18), has received great attention and rapid development. It provides a set of RESTful service interface standards such as: WADO-URI, WADO-RS, QIDO-RS, STOW-RS, UPS-RS, etc., enabling web developers to unlock the power of healthcare images using industry-standard toolsets.

In this paper, DICOM services are used for integrating with PACS, and DICOMweb services are used to provide standard medical image query and transmission for mobile clients. The descriptions of these services used in this paper are shown in Table 1, which are implemented by using an open source toolkit (dcm4che) [38, 39, 41, 42].

2.3. Related works

To rapidly and conveniently transmit medical HRIs, considerable research efforts have been carried out, most of them focus on three ways: compressed transmission [2, 24–27], progressive transmission [24–26, 28–30], and on-demand transmission [7, 21–23, 31, 32]. The compressed transmission may get a risk of losing important information because it reduces the image resolution and quality, or catch a complex process that client needs to be customized for special decoding and reconstruction software [7]. The progressive transmission can be achieved by first transmitting a low-resolution approximation of the image, then sending further information and the original image can be recovered by sending all further information [25]. Therefore, this method improves the user experience, but cannot reduce the amount of data transmission. The on-demand transmission uses new network interactive technologies to change the whole transmission and only the required resolution images or Region of Interests (ROIs) are transmitted to the client. The on-demand transmission is usually implemented based on JPEG2000 and JPIP protocol, which has very good network transmission characteristics, including ROI, progressivity and scalability [1, 7, 22, 28, 33, 36]. However, this method also has some disadvantages, such as repeated transmission, in-

Table 2
Frequently used symbols.

Symbol	Description
P	An image pyramid
pkey	The unique identifier of an image pyramid
G	A set of layer images
S	A set of image resolutions
W, H	The width and height of an image
n	The number of layers of a pyramid
h	The vertical distance between the largest and smallest layer
h_i	The vertical distance between two adjacent layers
β	The distance ratio between two adjacent layers
R	A set of the user query request
m	The number of lattices of a pyramid
R(W)	The width of the requested image
L	A series of lattices with different resolutions
L(W)	The width of a lattice
CL	The candidate image set of a query
IB	The description of a specific image block
x, y	The x-axis and y-axis values of the upper-left of the block
w, h	The width and height of an image block

valid transmission, special coding and decoding, coding and decoding are complicated and time-consuming [7]. To further improve the performance of medical HRIs transmission and reduce the complexity, the tile-pyramid on-demand transmission without using JPEG2000 is adopted to solve redundant transmissions for each fixed scaling resolution [28, 32, 36]. As we known, the TIPS only supported a $4\times$ ratio zooming, which does not affect users reading extreme HRIs such as satellite images and microscope images, in clinical practice, it may be not convenient to read medical HRIs [10]. In this context, paper [23] designed a novel unbalanced ratio pyramid structure(URPS) to adapt to the reading habits of doctors and achieved good results. In this model, the number of image layers is generated based on rules and the layer distribution is fixed. In addition, although it is a web-based system, it does not implement the DICOMweb services, thus increasing the complexity of integration with third-party medical systems. Different from these works, our approach focuses on designing a novel unbalanced image pyramid generation, indexing, and query method to optimize the transmission and rendering for medical HRIs.

3. Computational methods

3.1. Preliminaries and problem definition

The list of symbols to be used throughout the rest of this paper is first summarized in Table 2.

Definition 1. An image pyramid is a collection of images from high to low resolution which is represented by a triplet:

$P =< pkey, G, S >$

where *pkey* refers to the unique identifier of an image pyramid, $G=\{G_{\max},G_1,\ldots, G_n,G_{min}\}$ represents the collection of images from high to low resolution and $S=\{S_{\max}(W,H), S_1(W,H), \ldots, S_n(W,H), S_{min}(W,H)\}$ is the corresponding resolutions. $G_{\max}$ is the original image that is parsed from DICOM file, $G_{\min}$ is the smallest image that is specified by the user.

Definition 2. The number of layers represents how many layers are needed in the pyramid, which is named as *n* and is determined according to Eq. (1).

$$n = \max(S_{\max}(W)/S_{\min}(W), S_{\max}(H)/S_{\min}(H)) \tag{1}$$

where *W* is denoted the width of the image and *H* is the height of the image.

Definition 3. Unbalance factor$\beta$is a constant indicating the distance ratio between two adjacent layers as shown in Fig. 1. Only the width of the image is considered because the image is scaled with the same coefficient.

From the geometric relationship, β and h can be represented by$h_i(1 \le i \le n+1)$, as shown in Eq. (2).

$$\beta = \frac{h_2}{h_1} = \frac{h_3}{h_2} = \cdots \frac{h_{i+1}}{h_i} \cdots = \frac{h_{n+1}}{h_n},$$
$$h = h_1 + h_2 + \cdots + h_{n+1} = \sum_{i=1}^{n+1} h_i \tag{2}$$

where h_i represents the vertical distance between the *i*th layer and the *i*1th layer, *h* represents the vertical distance between the largest layer ($G_{\max}$) and the smallest layer($G_{\min}$).

3.2. UIPS: Unbalance image pyramid scheme

The core of the pyramid scheme is how to determine the number of layers, calculate the resolution of each layer and how to represent and store the pyramid. First, the number of layers can be determined by Eq. (1), and then the image resolution of each layer is calculated according to the geometric relationship of layers (Fig. 1). Finally, we propose a simple scheme to store the description information of the pyramid.

3.2.1. Calculation of layer resolution

As shown in Fig. 1 and Eqs. (1) and (2), if the unbalance factor is equal to $1(\beta=1)$, the distance between each layer is equal $(h_1=h_2=\cdots=h_{n+1})$, when the unbalance factor is not equal to 1 $(\beta \neq 1)$, the unbalance factor is sized in a form of geometric progression, and the geometric proportion isβ. Therefore, the width of the *i*th layer can be calculated by Eq. (3).

$$S_i(W) = \begin{cases} S_{\max}(W) - i \times \dfrac{(S_{\max}(W) - S_{\min}(W))}{n+1} (\beta = 1, 1 \le i \le n) \\ \dfrac{\beta^i - \beta^{n+1}}{1-\beta^{n+1}} \times (S_{\max}(W) - S_{\min}(W)) + S_{\min}(W) (\beta \neq 1, \\ \hfill 1 \le i \le n) \end{cases} \tag{3}$$

Proof. According to the geometric relation of the layers, the image width of the *i*th layer can be calculated from Eq. (3.1).

$$S_i(W) = 2 \times \frac{h - \sum_{j=1}^{i} h_j}{\tan\partial} + S_{\min}(W) \tag{3.1}$$

Then, tan ∂can be expressed as Eq. (3.2) from the geometric relation.

$$\tan\partial = \frac{2\times h}{S_{\max}(W) - S_{\min}(W)} \tag{3.2}$$

If the unbalance factor is equal to $1(\beta=1)$, the distance between each layer is equal $(h_1=h_2=\cdots=h_{n+1})$. Therefore, the sum of the distance from the first layer to the *i*th layer can be expressed by Eq. (3.3)

$$\sum_{j=1}^{i} h_j = i \times h_1 (\beta = 1) \tag{3.3}$$

Similarly,

$$h = \sum_{j=1}^{n+1} h_j = (n+1) \times h_1 \tag{3.4}$$

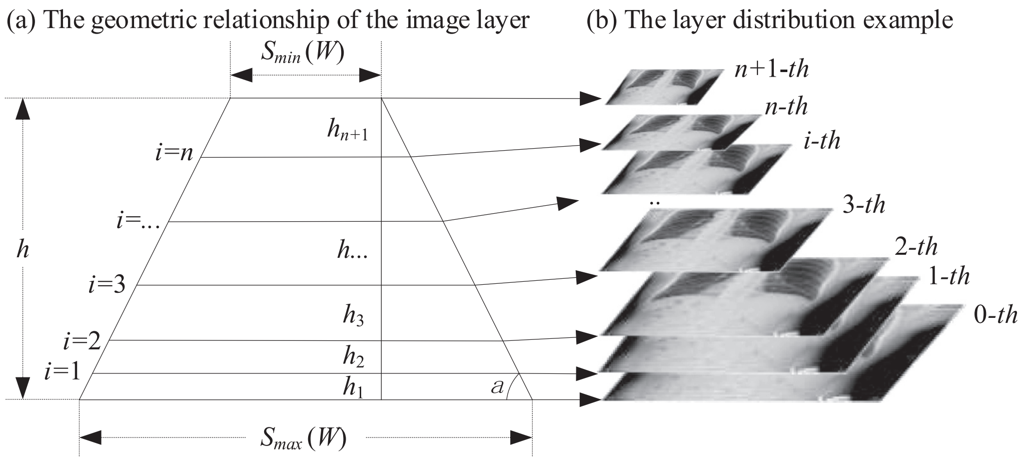


Fig. 1. Geometric relation and distribution example of the layer.

Combining Eqs. (3.1)–(3.5) can be derived:

$$\begin{aligned} S_i(W) &= 2\times\frac{(n+1)\times h_1 - i\times h_1}{2\times(n+1)\times h_1}\times(S_{\max}(W)-S_{\min}(W)) \\ &\quad + S_{\min}(W) \\ &= S_{\max}(W) - i\times\frac{S_{\max}(W)-S_{\min}(W)}{n+1} \end{aligned} \tag{3.5}$$

If the unbalance factor is not equal to 1($\beta \neq 1$), the distance between each layer is sized in a form of geometric progression, and the geometric proportion isβ. From the summation formula of the geometric progression, the sum of the distance from the first layer to the *i*th layer can be expressed by Eq. (3.6)

$$\sum_{j=1}^{i} h_j = \frac{h_1\times(1-\beta^i)}{1-\beta}(\beta\neq 1) \tag{3.6}$$

Similarly,

$$h = \sum_{j=1}^{n+1} h_j = \frac{h_1\times(1-\beta^{n+1})}{1-\beta}(\beta\neq 1) \tag{3.7}$$

Combining Eqs. (3.1), 3.(2), 3.(6) and 3.(7), Eq. (3.8) can be derived:

$$\begin{aligned} S_i(W) &= 2\times\frac{h-\sum_{j=1}^{i} h_j}{\tan\partial} + S_{\min}(W) \\ &= 2\times\frac{h_1\times(\beta_i-\beta^{n+1})}{\tan\partial\times(1-\beta)} + S_{\min}(W) \\ &= \frac{\beta^i-\beta^{n+1}}{1-\beta^{n+1}}\times(S_{\max}(W)-S_{\min}(W)) + S_{\min}(W) \end{aligned} \tag{3.8}$$

Combining Eq. (3.5) and Eqs. (3.8), Eq. (3) can be derived:

$$S_i(W) = \begin{cases} S_{\max}(W) - i\times\dfrac{(S_{\max}(W)-S_{\min}(W))}{n+1} & (\beta = 1, 1\leq i\leq n) \\ \dfrac{\beta^i-\beta^{n+1}}{1-\beta^{n+1}}\times(S_{\max}(W)-S_{\min}(W)) + S_{\min}(W) & \beta\neq 1, 1\leq i\leq n) \end{cases} \tag{3.9}$$

To ensure that the image is not deformed, the height of the *i*th layer can be calculated by Eq. (4).

$$S_i(H) = \frac{S_i(W)}{S_{\max}(W)}\times S_{\max}(H) \tag{4}$$

Algorithm 1 detailed the steps of layer resolution calculation. The width of the layer is first calculated by Eq. (3), as shown in lines 3-7, and then its corresponding height is determined by Eq. (4), as shown in line 8.

Algorithm 1
The LRC algorithm (Layer resolution calculation algorithm).

```
Name: LRC($S_{\max}(W,H)$, $S_{\min}(W,H)$, $n$, $i$, $\beta$)
Input: $S_{\max}(W,H)$: the resolution of the image $G_{\max}$;
       $S_{\min}(W,H)$: the resolution of the n+1-th image $G_{\min}$;
       $n$: the number of layers;
       $i$: the i-th layer;
       $\beta$: the unbalance factor;
Output:$S_i(W,H)$: the resolution of the $i$-th layer.
1.  begin
2.    $S_i(W,H) \leftarrow \Phi$
3.    if ($\beta = 1$) then
4.      $S_i(W) \leftarrow S_{\max}(W) - i\times\{(S_{\max}(W)-S_{\min}(W))/(n+1)\}$
5.    else if ($\beta \neq 1$) then
6.      $S_i(W)=\{(\beta^i-\beta^{n+1})/(1-\beta^{n+1})\}\times(S_{\max}(W)-S_{\min}(W))+S_{\min}(W)$
7.    end if
8.    $S_i(H) \leftarrow (S_i(W)/S_{\max}(W))\times S_{\max}(H)$
9.    $S_i(W,H) \leftarrow S_i(W),S_i(H)$
10.     return $S_i(W,H)$
11. end
```

3.2.2. *Storage structure of UIPS*

Based on Definition 1, each pyramid can be described as a node with two attributes: pyramid key (*pkey*) and layer set. To record the user's request behaviors, we added a request attribute, which consists of request time and request resolution, into the pyramid node. Fig. 2 depicts the logical storage structure of UIPS.

The logical storage structure is stored as XML file on server side, and the corresponding images are stored in the respective directories. Fig. 3 (a) details the directory structure and Fig. 3 (b) is the corresponding XML code.

3.2.3. *Generation algorithm*

Algorithm 2 depicts the detailed steps of the generation method for UIPS, which can be divided into three steps. First, load the storage structure from the XML file (py-scheme.xml) and calculate how many layers are needed using Eq. (1) (line 4). Then, loop through each layer (lines 6-12), including calculating the resolution of each layer (Algorithm 1), creating a new layer and setting the layer information, generating the corresponding image and storing it into local disk. Finally, generate the corresponding XML code and write it to local disk (lines 13–16).

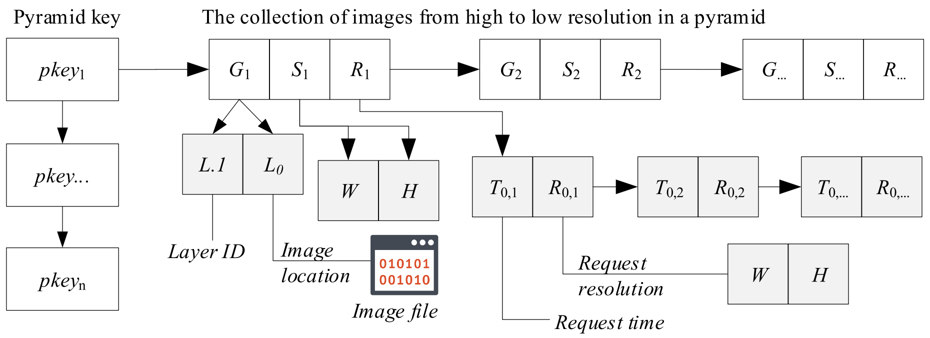
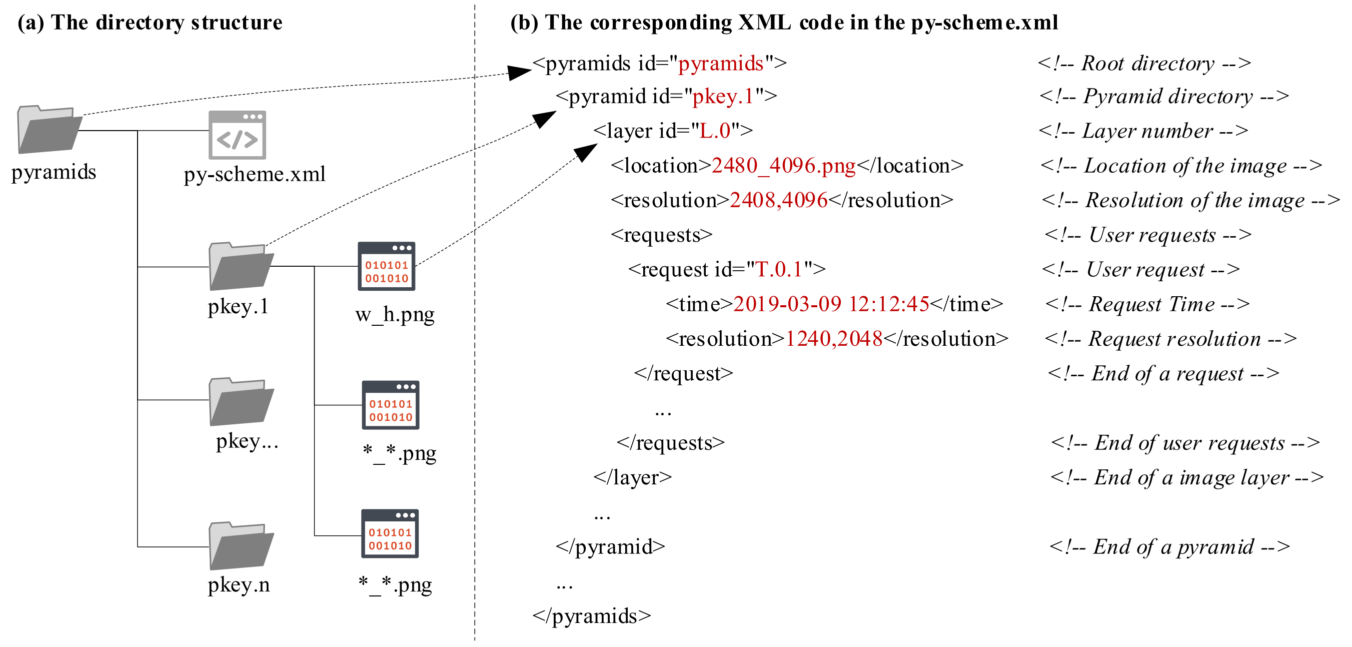


Fig. 2. Logical storage structure of UIPS.


(a) The directory structure
(b) The corresponding XML code in the py-scheme.xml
pyramids
py-scheme.xml
pkey.1
w_h.png
pkey...
_.png
pkey.n
_.png
010101 001010

```
<pyramids id="pyramids">                          <!-- Root directory -->
  <pyramid id="pkey.1">                           <!-- Pyramid directory -->
    <layer id="L.0">                              <!-- Layer number -->
      <location>2480_4096.png</location>          <!-- Location of the image -->
      <resolution>2408,4096</resolution>          <!-- Resolution of the image -->
      <requests>                                  <!-- User requests -->
        <request id="T.0.1">                      <!-- User request -->
          <time>2019-03-09 12:12:45</time>        <!-- Request Time -->
          <resolution>1240,2048</resolution>      <!-- Request resolution -->
        </request>                                <!-- End of a request -->
          ...
      </requests>                                 <!-- End of user requests -->
    </layer>                                      <!-- End of a image layer -->
    ...
  </pyramid>                                      <!-- End of a pyramid -->
  ...
</pyramids>
```

Fig. 3. Directory structure and the corresponding XML code.

Algorithm 2
Generation algorithm for UIPS.

```
Name: GPM(pkey, $G_{\max}$,$S_{\min}(W,H)$, $\beta$, $f_{xml}$)
Input: pkey: a key used to uniquely identify a pyramid;
       $G_{\max}$: the original image;
       $S_{\min}(W,H)$: the resolution of the image$G_{min}$;
       $\beta$: the distance scale factor;
       $f_{xml}$: the XML file of the storage scheme
Output: G,i.e., the collection of each layer image;
$f_{xml}$,i.e., the XML file of the storage scheme
1.  begin
2.    $P < pkey, Layers < G, S > > \leftarrow loadSch(f_{xml})$
3.    $S_{\max}(W,H) \leftarrow (G_{\max}.rows, G_{\max}.cols)$
4.    $n \leftarrow \max(S_{\max}(W)/S_{\min}(W), S_{\min}(H)/S_{\min}(H))$
5.    $Layers(layer, S(W, H)) \leftarrow \Phi$
6.    for each $i$ in $n$ do
7.      $S_i(W,H) \leftarrow LRC(S_{\max}(W,H), S_{\min}(W,H), n, i, \beta)$
8.      $G_i \leftarrow shrink(G_{\max}, S_i(W,H))$;
9.      $layer \leftarrow createLayer(layerId_i, S_i(W,H), location_i)$
10.       $Layers.put(layer, S_i(W,H))$
11.       $writeToDisk(G_i, location_i)$
12.   end for
13.   $P.put(pkey, S_{\max}(W,H), S_{\min}(W,H), Layers)$
14.   $xml \leftarrow buildXml(P)$
15.   $writeToFile(xml, f_{xml})$
16. end
```

3.3. Indexing scheme

In this subsection, we first describe the logical index structure based on hash table and lattice. Then, we depict the indexing process for the index structure.

3.3.1. Index structure

We present a simple index structure by using hash table and lattice partitioning. For the convenience of description, we give the following definitions:

Definition 5. The index structure can be represented as a triplet:

$$indexS = < pkey, L, G, LD >$$

where *pkey* identifies the unique pyramid, *L* is a series of lattices with different resolutions, *G* represents the image collection, *LD* defines the mapping between *L* and *G*.

Definition 6. Lattice division maps the layers with similar resolution to a specific lattice. Before lattice division, the number of the lattices should be determined by the maximum, minimum resolution, and the width of a lattice. Assuming $L(W)$ represents the width of a lattice, obviously, the number of the lattices can be calculated by Eq. (5).

$$m = \frac{S_{\max}(W) - S_{\min}(W)}{L(W)} \tag{5}$$

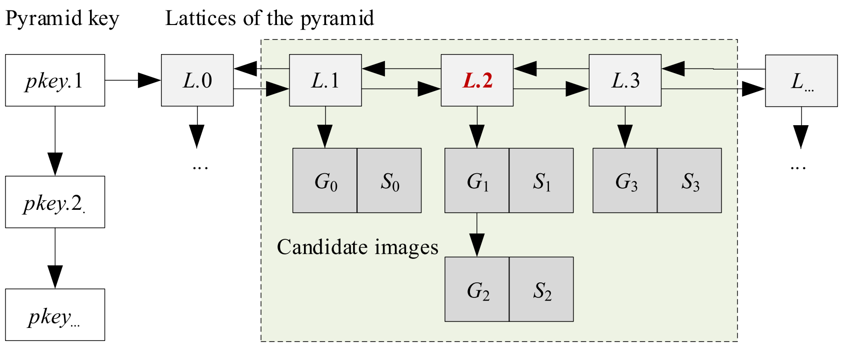


Fig. 4. Logical structure of the indexing scheme.

where m presents the number of the lattices . The lattice division function divides the different resolution into a specific lattice, which is a mapping function from resolution to lattice number. Similarly, the mapping function can be described as Eq. (6).

$$LDF(R_i(W)) = \left\lfloor \frac{R_i(W) - S_{\min}(W)}{L(W)} \right\rfloor \tag{6}$$

where $R_i(W)$ denotes the width of the image requested by user.

Based on the above analysis, we present an index structure that consists of a hash table, a lattice array and a mapping function as depicted in Fig. 4. The hash table is used to store pyramid keys, and the corresponding value is a lattice array that is used to record the information of the layers. The number of the lattice can be calculated by Eq. (5) and each lattice is mapped to an image set and their relationships are determined by the mapping function (Eq. (6)).

3.3.2. Indexing algorithm

Indexing algorithm focuses on establishing the index structure from the pyramid storage structure. Algorithm 3 illustrates the detailed steps of the indexing process, which consists of two steps: loading storage file (XML), creating index structure (lines 4–15), where index create includes calculating lattice division, and writing the *pkey* and lattices into index structure and file.

Algorithm 3
Indexing algorithm.

```
Name: Indexing($f_{xml}$,$L(W)$)
Input: $f_{xml}$: the XML file that stores the pyramid structure
       $L(W)$: the width of the lattice
Output: indexS: the index structure of pyramids.
1.  begin
2.      $indexS < pkey, L < array > > \leftarrow \Phi$
3.      $xmlTree \leftarrow loadXmlSch(f_{xml})$
4.      for each pyramid node pNode in xmlTree do
5.        $S_{\max}(W) \leftarrow pNode.getS_{\max}(W)$
6.        $S_{\min}(W) \leftarrow pNode.getS_{\min}(W)$
7.        $m \leftarrow (S_{\max}(W) - S_{\min}(W))L(W))$
8.        $L < Array > \leftarrow L < array < m > >$
9.        for each layer node layerNode in pNode
10.         $S(W, H) \leftarrow layerNode.getResolution(W, H)$
11.         $k \leftarrow \lfloor (S(W) - S_{\min}(W))/L(W) \rfloor$
12.         $array < k > .add(layerNode)$
13.       end for
14.       $indexS.put(pNode.getPkey(), L.put(array))$
15.     end for
16.     return indexS
17. end
```

3.4. Query scheme

Query scheme aims to query a specific image block from the index structure according to the requested parameters. For the convenience of description, we give the following definitions:

Definition 7. Briefly, a typical query can be represented as:

$QIB = (pkey, R(W, H), IB)$

where *pkey* identifies a unique pyramid, $R(W,H)$ represents the resolution of the requested image and *IB* is the description of an specific image block.

Definition 8. The image block description (*IB*) can be modeled as a four-tuple:

$IB =< x, y, w, h >$

where- x and y refer to the x-axis values and y-axis values of the upper-left of the block, respectively;- w and h refer to the width and height of the block, respectively.

Definition 9. The fuzzy matching function (*FMF*), also called as similarity function, calculates the similarity between a given resolution and a candidate image.

$$FMF(R(W), S(W)) = \frac{1}{1 + |R(W) - S(W)|} \tag{7}$$

where $R(W)$ represents the width of a requested image and $S(W)$ denotes the width of a candidate image.

Definition 10. The candidate images are determined by lattice division (Definition 6). The mapping function first maps the requested resolution into a specific lattice. The current lattice, the direct precursor and successor of the lattice are selected as candidate lattices. Then, the images in these lattices are used as candidate images. For example, when the requested image resolution is mapped to the lattice $L.2$, the images contained in the lattice $L.1$, $L.2$, and $L.3$ are selected as candidate images.

Algorithm 4 illustrates the detailed steps of querying an image block, which can be divided into four steps, i.e., querying a pyramid (lines 2–3), selecting candidate images (lines 4–12), matching the optimal image layer (lines 13-19), and capturing the query image block (lines 20-21). First, obtain a lattice array from a specific pyramid according to the pyramid key (*pkey*). Then, calculate the lattice to which the requested image belongs according to Eq. (6) and get candidate images from this lattice and its direct adjacent lattices. Third, using Eq. (7) to calculate the similarity between the requested image and each candidate image, and get

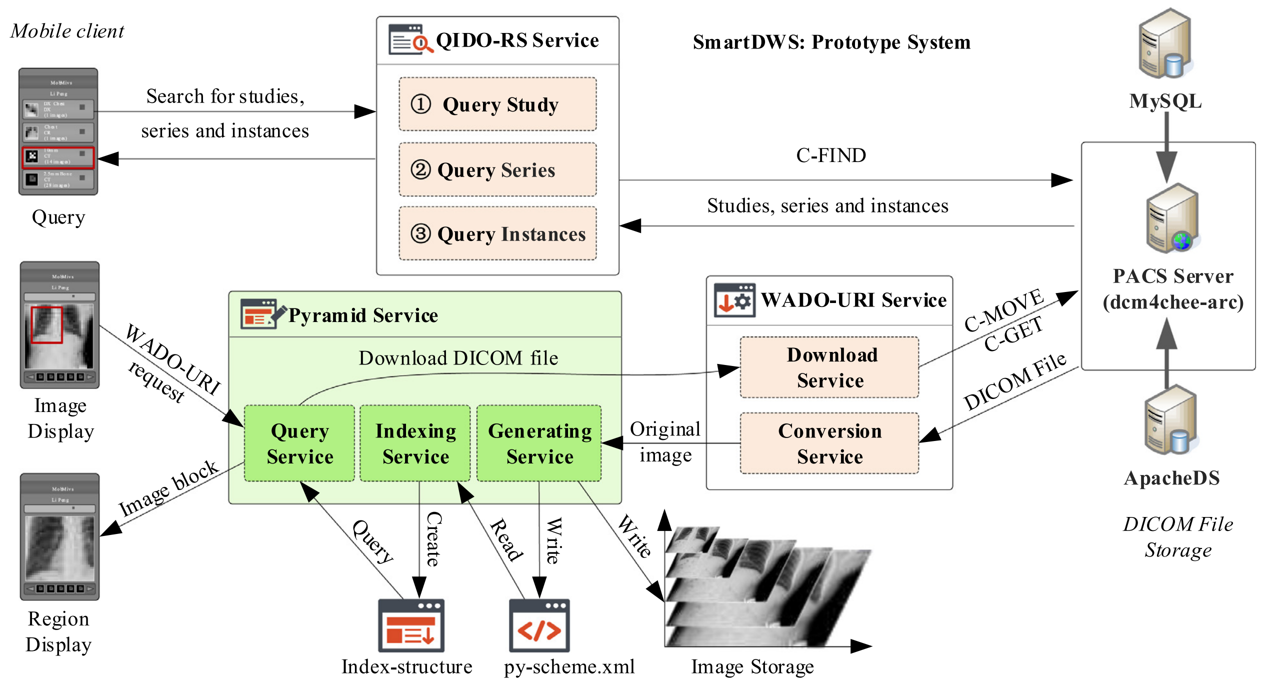


Fig. 5. System architecture and workflow.

Algorithm 4
Query a specific image block from the index structure.

```
Name: QueryIB (pkey, R(W, H), IB)
Input: pkey: a key is used to uniquely identify a pyramid;
       R(W, H): the width and height of the requested image;
       IB: the description of a specific image block
Output: imgBlk: an image block requested by user.
1.  begin
2.    indexS ← getIndexS()
3.    L < array > ←indexS.getLattice(pkey)
4.    m ← (R(W) − S_min(W))/L(W)
5.    CL < layers > ←Φ
6.    if (m > 0) then
7.      CL.put(L.getLayers(m − 1))
8.    end if
9.    if (m ≤ L.length()-1)) then
10.     CL.put(L.getLayers(m + 1))
11.   end if
12.   CL.put(L.getLayers(m))
13.     similary ← 0.0, simLayer ← Φ
14.   for each layer in the candidate images CL do
15.     sim ← 1/(1 + |R(W) − layer.getWidth()|)
16.     if (sim > similary) then
17.       similary ← sim, simLayer ← layer
18.     end if
19.  end for
20.   simImg ← loadImg(simLayer.getLocation())
21.   imgBlk ← capImgBlk(simImg, IB)
22.   return imgBlk
23.  end
```

an image with the highest similarity. Finally, capture the requested image block on the highest similarity image.

4. System description

Fig. 5 shows the proposed system architecture and workflow, which supports querying, downloading, conversion and interactive visualization of DICOM images. The prototype system (SmartDWS) based on the proposed architecture is an upgrade and extension of our previous work SmartWADO [31], which consists of three core components: (1) QIDO-RS service searches for studies, series, and instances that match specified search parameters; (2) WADO-URI service focuses on downloading the DICOM file and parsing the original image; (3) Pyramid service is an optimization component for RHIs transmission and rendering.

This paper focuses on describing the pyramid service that has been added to the SmartDWS as a new feature. As shown in Fig. 5, pyramid service consists of generating service, indexing service, and query service. Generating service creates the UIP and stores the corresponding structure and the images of each layer, indexing service is responsible for creating and maintaining the index structure and files, and query service focuses on querying a specific image block from the index structure.

In Fig. 5, given a WADO-URI request, the system returns a specific image block to mobile client. Formally, Algorithm 5 illustrates the execution process of the user request. On receiving the request, the QIDO-RS service will parse the parameters *content-Type, studyUID, seriesUID, objectUID, columns, rows* and *region* that

Algorithm 5
The algorithm for querying image blocks.

```
Name: Query (wadoReq)
Input:  wadoReq: the WADO-URI request including some parameters.
Output:  imgBlk: an image block that user requested.
1.  begin
2.    contentType, studyUID, seriesUID, objectUID ← wadoReq.getParas()
3.    pkey ← md5Digest(contentType∪studyUID∪seriesUID∪objectUID)
4.    R(W, H) ← wadoReq.get(columns, rows)
5.    IB ← wadoReq.getRegion()
6.    imgBlk ← QueryIB(pkey, R(W, H), IB)
7.    if (imgBlkis null) then
8.    dcmFile ← downLoad(studyUID, seriesUID, objectUID)
9.    G_max ← convertToImg(dcmFile)
10.     f_xml,S_min(W,H), β, L(W) ← getFromConfig()
11.     GPM(pkey, G_max,S_min(W,H), β, f_xml)
12.     Indexing(f_xml,L(W))
13.     imgBlk ← QueryIB(pkey, R(W, H), IB)
14.   end if
15.   return imgBlk
16.end
```

Table 3
The detailed description of test cases and dataset.

Test case	Original image size (MB)	Image resolution (pixel)	Number of layers	Total storage size (MB)	
				PNG	JPEG
C. 1	6.1	1723×1857	7	7.0	3.6
C. 2	8.4	1544×2863	13	18.5	11.3
C. 3	10.2	1788×3001	13	17.2	8.9
C. 4	12.0	2157×2928	13	19.1	8.2
C. 5	14.4	2891×2615	13	23.4	12.3
C. 6	18.2	2800×3408	15	35.1	15.6
Original image format: DICOM		Unbalance factor:$\beta = 1.05$		$G_{\min}$ is 256×256	

defined in WADO-RUI model and the union of *contentType, studyUID, seriesUID* and *objectUID* is digested as a uniform 16-bit character by MD5 algorithm as the pyramid key (lines 2-3). Then, the query service will first query the image block by using Algorithm 4 (lines 4-6). As shown in lines 7–14, if the image block is null, the WADO-URI service will be triggered to send C-MOVE or C-GET request to download the DICOM file from PACS and convert it to an original image. On receiving the image, the generating service and indexing service will generate an UIP and updating the index structure and file. Finally, the requested image block is captured and returned to the mobile client with an HTTP response containing the image in a proper Multipurpose Internet Mail Extension (MIME) type such as PNG or JPEG.

5. Results and discussion

5.1. Environment

Our experimental environment consists of two servers (Lenovo System X3650 M5, Xeon E5@2.1 GHz with 32GB of RAM) with Windows server 2012R, a laptop (Intel® Core^{TM} i7-7700HQ@2.80 GHz with 16 GB of RAM) with Windows 10 Pro, a smartphone (HUAWEI Mate10, Kirin 970 with 6GB of RAM and 128 GB ROM) with Android OS v8.0, and a 4G wireless (600Mbps) router and a gigabit switch.

One server is used to deploy SmartDWS and the other servers deployed an open source PACS (DCM4CHEE-ARC-5.4.1-MYSQL) [42], MySQL database and ApacheDS [43]. They are connected by the gigabit switch for high-speed transmission of DICOM file. The router provides a Wi-Fi network for achieving high-speed transmission of DICOM file between smartphone and SmartDWS. The connection between the two servers is under 1Gbps network, and the bandwidth between smartphone and SmartDWS is under 600 Mbps in the Wi-Fi network.

5.2. Experimental case description

The experimental data set is a real data set containing 6860 DR studies that are stored in the open source PACS. We select 6 studies with different resolutions from the data set to evaluate the proposed approach. Table 3 describes the original image size, resolution, number of layers, and total size of the corresponding image pyramid for each test case.

5.3. Evaluation Metrics

In this paper, the average response time is used to evaluate the performance of model generation, indexing, and image block query, which is calculated by Eq. (8). The standard deviation is used to evaluate the stability of the performance, which is calculated by Eq. (9).

$$Avg(t) = \left(\frac{1}{n}\sum_{1}^{n}(time_i/count_i)\right) \tag{8}$$

$$SD = \sqrt{\frac{1}{n}\sum_{1}^{n}[(time_i/count_i) - Avg(t)]^2} \tag{9}$$

where n represents the number of tests, $time_i$ denotes the response time of the ith test, $count_i$ indicates the number of result sets (studies, series, instances or images) for ith test, SD presents the standard deviation of the response time.

5.4. Layer distributions of UIPS

The purpose of this experiment is to evaluate the effectiveness of the UIPS, which is testified by using the test case C.6. The layer distributions with different factors are shown in Table 4 by comparing the image resolutions between the UIPS and that of the URPS [7] and TIPS.

Table 4
The size of the each layer (pixel) of test case C. 6 (2800×3408, 15 layers).

Layer index	UIPS					URPS[7]	TIPS
	$\beta = 0.80$	$\beta = 0.95$	$\beta = 1.00$	$\beta = 1.05$	$\beta = 1.2$		
L.0	2800×3408	2800×3408	2800×3408	2800×3408	2800×3408	2816×3456	2800×3408
L.1	2268×2760	2552×3106	2619×3187	2671×3250	2758×3356	1408×1792	1400×1704
L.2	1843×2243	2316×2818	2437×2966	2534×3084	2706×3293	1024×1152	700×852
L.3	1502×1828	2092×2546	2255×2744	2391×2910	2644×3218	768×896	350×426
L.4	1229×1495	1879×2287	2074×2524	2241×2727	2570×3128	640×768	175×213
L.5	1011×1230	1677×2041	1892×2302	2083×2535	2481×3019	512×640	-
L.6	837×1018	1485×1807	1710×2081	1918×2334	2374×2889	-	-
L.7	698×849	1303×1585	1528×1859	1744×2122	2245×2732	-	-
L.8	586×713	1129×1374	1347×1639	1561×1899	2091×2545	-	-
L.9	497×604	964×1173	1165×1417	1369×1666	1907×2321	-	-
L.10	425×517	808×983	983×1196	1168×1421	1685×2050	-	-
L.11	368×447	659×802	802×976	956×1163	1419×1727	-	-
L.12	322×391	518×630	620×754	734×893	1099×1337	-	-
L.13	286×348	384×467	438×533	501×609	716×871	-	-
L.14	256×311	256×311	256×311	256×311	256×311	-	-

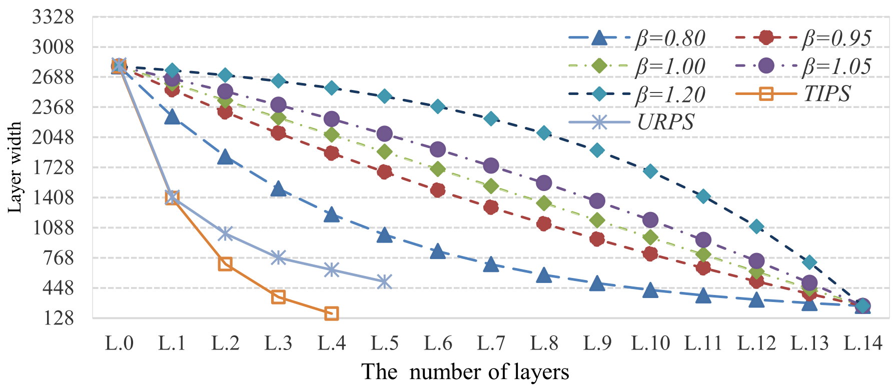


Fig. 6. Layer distributions of UIPS (different factors (β)), URPS and TIPS.

Table 5
The results of aspect ratio and total zoom ratio of each layer.

	Aspect ratio (AR) and its deviation (DEV) from the original image										Total zoom ratio (%)				
	UIPS (%)														
	$\beta=0.95$		$\beta=1.00$		$\beta=1.05$		URPS[7] (%)		TIPS (%)		UIPS				
Layer index	AR	DEV	AR	DEV	AR	DEV	AR	|DEV|	AR	DEV	$\beta=0.95$	$\beta=1.00$	$\beta=1.05$	URPS[7]	TIPS
L.0	82.16	0.00	82.16	0.00	82.16	0.00	81.48	0.68	82.16	0.01	100.00	100.00	100.00	101.99	100.00
L.1	82.16	0.00	82.18,	0.02	82.18	0.02	78.57	3.59	82.16	0.00	83.07	87.47	90.97	26.22	25.00
L.2	82.19	0.03	82.16	0.00	82.17	0.01	88.89	6.73	82.16	0.00	68.39	75.75	81.90	12.36	6.25
L.3	82.17	0.01	82.18	0.02	82.16	0.00	85.71	3.55	82.16	0.00	55.82	64.84	72.91	7.21	1.56
L.4	82.16	0.00	82.17	0.01	82.16	0.00	83.33	1.17	82.16	0.00	45.03	54.86	64.04	5.15	0.39
L.5	82.17	0.01	82.19	0.03	82.18	0.02	80.00	2.16	-		35.87	45.64	55.34	3.43	-
L.6	82.18	0.02	82.17	0.01	82.17	0.01	-		-		28.12	37.29	46.91	-	-
L.7	82.21	0.07	82.19	0.03	82.17	0.01	-		-		21.64	29.77	38.78	-	-
L.8	82.17	0.01	82.18	0.02	82.16	0.00	-		-		16.26	23.14	31.06	-	-
L.9	82.18	0.02	82.22	0.06	82.16	0.00	-		-		11.85	17.30	23.90	-	-
L.10	82.20	0.04	82.19	0.03	82.20	0.04	-		-		8.32	12.32	17.39	-	-
L.11	82.17	0.01	82.17	0.01	82.27	0.11	-		-		5.54	8.20	11.65	-	-
L.12	82.22	0.06	82.23	0.07	82.20	0.04	-		-		3.42	4.90	6.87	-	-
L.13	82.23	0.07	82.18	0.02	82.20	0.04	-		-		1.88	2.45	3.20	-	-
L.14	82.32	0.16	82.32	0.16	82.32	0.16	-		-		0.83	0.83	0.83	-	-
Average	82.19	0.03	82.19	0.03	82.19	0.03	83.00	2.98	82.16	0.00	-	-	-	-	-
Standard deviation	0.04		0.04		0.04		3.82	2.19	0.00		-	-	-	-	-

From Table 4, we get the layer distribution curve with different unbalance factor (β), URPS and TIPS with $4\times$ ratio zooming as shown in Fig. 6.

As shown in Section 3.2, different unbalance factors (β) generate different layer distributions. From Table 4 and Fig. 6, when $\beta \neq 1$, the each layer of image size is increased in an unbalanced ratio scale and the resolutions are completely consistent with the result from Eq. (3). For different unbalance factors (β), when $\beta = 1$, the image layers are distributed in equal proportions, when $\beta > 1$, the larger the β, the more the number of high-resolution layers, when $0 < \beta < 1$, the smaller the β, the more the number of low-resolution layers. Compared with the URPS and UIPS, URPS and TIPS with $4\times$ ratio zooming has fewer layers. However, in clinical practice, many doctors usually have the reading habits of unbalanced zoom ratio [7]. Therefore, compared with URPS and TIPS, UIPS is more flexible and more in line with clinical application habits.

To further evaluate each layer image, we performed statistics and analysis on the aspect ratio and total zoom ratio of each layer. From Table 4, the results of aspect ratio and total zoom ratio of each layer are shown in Table 5.

From the aspect ratio and its deviation from the original image in Table 5, TIPS does not deform each layer during image shrinking, UIPS shows slight deformation, and URPS deforms the most. As we have known, TIPS is strictly scaled-down. The images in UIPS and URPS are deformed due to rounding when calculating the layer resolution. Due to the different methods, URPS produced more deformation.

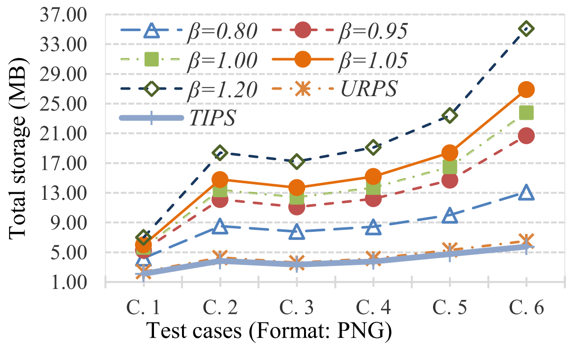


Fig. 7. Layer storage size of PNG image.

5.5. Layer image storage space

The purpose of this experiment is to compare the storage space of UIPS, URPS and TIPS. The storage space with different unbalance factors are shown in Figs. 7 and 8 by comparing the total storage space between UIPS and that of URPS and TIPS.

Compared with the JPEG image, PNG image takes up more storage space, because the JPEG image has a higher compression ratio

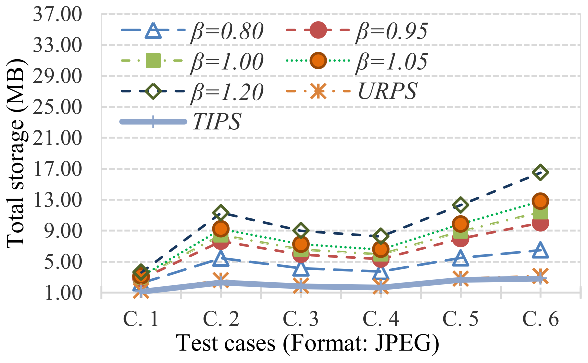


Fig. 8. Layer storage of JPEG image.

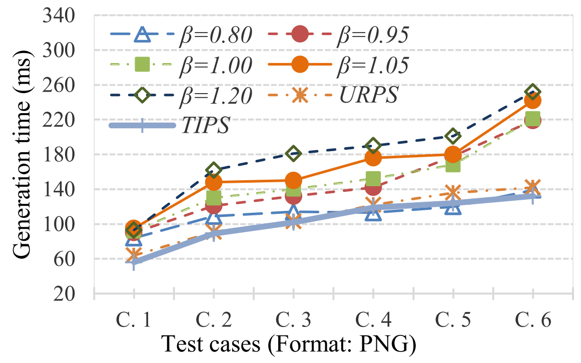


Fig. 9. Average generation time for PNG image.

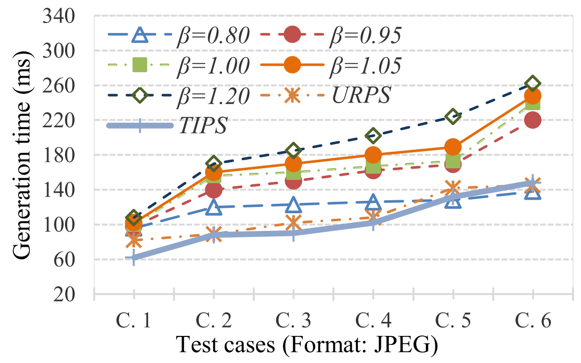


Fig. 10. Average generation time for JPEG image.

and takes up less storage space with the same resolution. Compared with TIPS and URPS, UIPS occupies more storage space because it generates more layers, which will inevitably take up more storage space. The storage space of UIPS reveals a general trend of a slight rise with the increase of the unbalance factor (β). The reason is that the storage space is related to the layer image distribution. With the increase of β, more images are ggenerated in the high-resolution area as shown in Table 4 and Fig. 6.

5.6. Performance analysis

5.6.1. Average response time of generation

In this experiment, we test the average generation time by using the above-mentioned test cases and unbalance factors, respectively. The average generation time with different β is shown in Figs. 9 and 10 by comparing the time between UIPS and that of URPS and TIPS (15 runs of each test case). To further analyze the stability of the generation time, the standard deviation of generation time for PNG and JPEG image is calculated as shown in Fig. 11.

From Figs. 9 and 10, the average generation time reveals a general trend of a slight rise with the increase of image size. The reason is that we need to parse and convert the DICOM file to a specific image format with different resolutions in real-time. Obviously, the larger the source DICOM size, the longer the parsing and conversion time. The generation time of UIPS reveals a general trend of a slight rise with the increase of the β. With the increase of β, more layer images are generated in the high-resolution area, and parsing and converting high-resolution images will take longer. In addition, the generation time of TIPS and URPS is slightly lower than that of UIPS because UIPS need to generate more layers. As shown in Fig. 11, UIPS, URPS, and TIPS have similar stability. Besides, the generation time of JPEG image has similar stability to PNG image.

5.6.2. Average response time of indexing

The purpose of this experiment is to compare the index time by using the above-mentioned test cases and different β, respectively. This index time refers to creating a new index and adding it to the generated index file. Figs. 12 and 13 depict the average index time (20 runs of each test case) and the trends from UIPS, URPS and TIPS.

From Figs. 12 and 13, the index time is not related to the image format and it reveals a general trend of a slight rise with the increase of the β. In general, both UIPS, URPS and TIPS can build indexing structure and index file in real-time ($<=$ 8ms). The reason is that we only need to divide different resolutions into a specific lattice and establish a mapping between the resolution and the lattice. This process is done in memory and is independent of image parsing and conversion. In addition, the index files are updated in a new thread.

5.6.3. Average response time of query

To fully test the query performance, we first test the response time of querying an image block from a specific layer (15 runs of each case). Table 6 presents the average response time and its standard deviation of querying an image block from different devices and image formats.

In Table 6, in each test, UIPS, URPS and TIPS have similar performance and stability. The response time of first querying is much higher than that of second querying, that is because the first querying needs to generate a pyramid model from the source file and build indexing scheme, which is relatively time-consuming. Besides, in most tests, the average response time on the laptop is less than that on the smartphone and the query stability on the laptop is slightly higher than that on the smartphone. One reason is that the performance of the laptop is stronger than that of the smartphone, another reason is the laptop runs on a wired network and the smartphone runs on a wireless network. The transmission performance and stability of the wired networks are superior to those of the wireless networks in our experimental environment.

To further analyze the performance, Table 7 depicts the advantage of UIPS compared with the entire transmission. The Ratio in Table 7 shows that for the same medical image, UIPS delivers much less amount of data, and also saves response time.

To evaluate the performance of querying specific image blocks from different layers, we test the response time of querying a specific image block from each layer by using test case C.6 (runs 15 of each case). Table 8 presents the query time when the user continuously zooms in on an image and display an image block in a specific view area.

From Table 5, as the increase of the resolution, the query time appears a general trend of a slight rise. The UIPS URPS and TIPS have similar query times when querying specific image block from similar layer resolution.

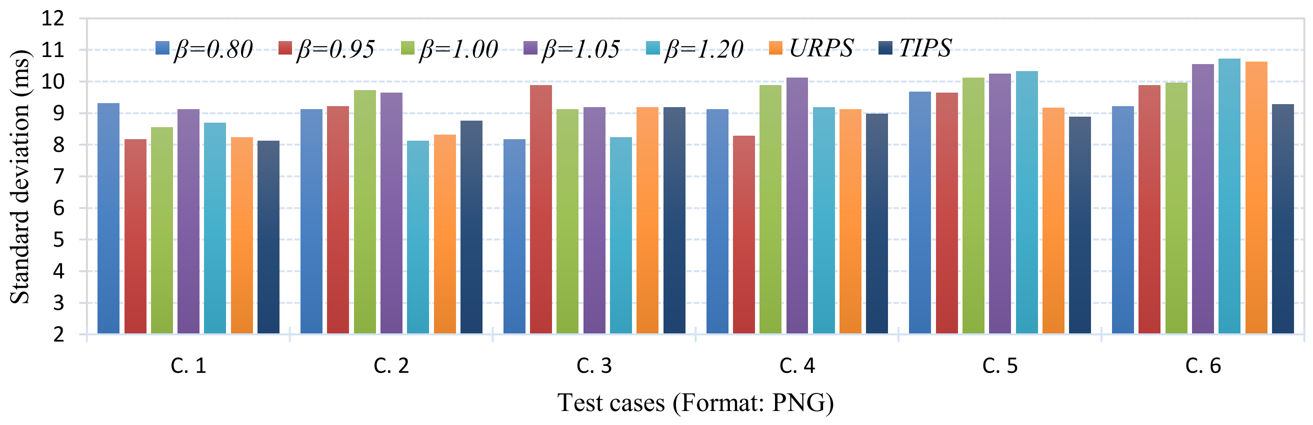


Fig. 11. Standard deviation of average generation time for PNG image.

Table 6
Average response time and its standard deviation of querying an image block.

				AT. FQ & SD (ms)						AT. SQ & SD (ms)					
				UIPS		URPS		TIPS		UIPS		URPS		TIPS	
Test cases	Image format	QIBS (pixel)	Devices	AT	SD	AT	SD	AT	SD	AT	SD	AT	SD	AT	SD
C. 1	PNG	421 × 453	Laptop	298	23.23	290	24.14	290	26.16	32	8.96	34	9.88	36	9.01
			Smartphone	312	32.12	284	35.64	298	24.54	35	9.12	38	10.23	42	10.99
C. 2	PNG	428 × 794	Laptop	322	24.52	320	23.35	318	24.38	50	9.32	56	9.45	58	9.36
			Smartphone	318	33.41	335	31.89	332	29.69	55	9.88	62	9.96	65	10.23
C. 3	PNG	495 × 830	Laptop	310	33.49	328	31.24	319	32.26	51	8.98	63	8.99	62	8.92
			Smartphone	328	32.82	346	38.76	335	38.88	58	9.63	62	10.12	73	9.35
C. 4	JPEG	595 × 808	Laptop	365	33.51	378	32.89	378	33.19	46	10.56	63	10.23	64	10.12
			Smartphone	359	39.83	389	39.12	389	37.12	42	12.33	74	11.98	72	11.32
C. 5	JPEG	795 × 719	Laptop	365	35.12	429	40.12	429	39.14	56	10.67	69	10.99	72	9.91
			Smartphone	443	42.84	432	45.32	432	42.38	64	11.12	82	12.32	81	10.56
C. 6	JPEG	794 × 966	Laptop	550	45.78	548	46.97	548	46.17	67	12.34	79	11.98	81	10.29
			Smartphone	591	58.96	623	48.19	623	52.19	84	14.58	83	13.12	74	12.38

- Viewing area (pixel): Laptop (768 × 512), Smartphone (512 × 512)
- Query layer: 4
- Unbalance factor: $\beta = 1.05$
- QIBS: Query image block size
- AT. FQ: Average time of the first query
- AT. SQ: Average time of the second query
- AT: Average time
- SD: Standard deviation

Table 7
Statistical results compared to the original image transmission.

							Radio (# V.S. OHRI)		
Test cases	Image format	OHRI (MB)	QIBS (pixel)	ATD (MB)	Devices	AT. OHRI (seconds)	AT. FQ	AT. SQ	ATD
C. 1	PNG	3.21	421 × 453	0.18	Laptop	3.89	1: 13.1	1: 121.6	1: 17.8
					Smartphone	4.18	1: 13.4	1: 119.4	
C. 2	PNG	7.57	428 × 794	0.26	Laptop	8.43	1: 26.2	1: 168.6	1: 29.1
					Smartphone	8.98	1: 28.2	1: 163.3	
C. 3	PNG	5.41	495 × 830	0.29	Laptop	6.54	1: 21.1	1: 128.2	1: 18.7
					Smartphone	7.71	1: 23.5	1: 132.9	
C. 4	JPEG	2.38	595 × 808	0.05	Laptop	3.89	1: 10.7	1: 84.6	1: 47.6
					Smartphone	4.18	1: 11.6	1: 99.5	
C. 5	JPEG	2.65	795 × 719	0.06	Laptop	5.37	1: 14.7	1: 95.9	1: 44.2
					Smartphone	6.12	1: 13.8	1: 95.6	
C. 6	JPEG	3.14	794 × 966	0.06	Laptop	4.28	1: 7.8	1: 63.9	1: 52.3
					Smartphone	5.17	1: 8.7	1: 61.5	

- Viewing area (pixel): 512 × 512
- Query layer: 4
- Unbalance factor: $\beta = 1.05$
- QIBS: Query image block size
- ATD: Amount of data per transmission (average value)
- OHRI: Original high-resolution image
- AT. OHRI: Average response time of query OHRI
- AT. FQ: Average time of the first query
- AT. SQ: Average time of the second query

5.7. Effect of magnifying image block

To evaluate the effect of magnifying image block from different layers on the mobile client, we test the magnified image block that queried from each layer by using test case C.6. The Fig. 14 shows the comparison of the result between UIPS, URPS and TIPS when the user continuously zooms in on an image and display an image block with view area resolution of 512 × 512.

From the results, the amplification of UIPS was smoother and avoided a very low resolution at the top layer when compared with the URPS and TIPS. The reason is that the UIPS distributes more layers between the minimum resolution and the maximum

Table 8
Average query time on each layer from smartphone and different formats.

		Query time (ms)														
Pyramid scheme	Image format	L.0	L.1	L.2	L.3	L.4	L.5	L.6	L.7	L.8	L.9	L.10	L.11	L.12	L.13	L.14
UIPS	PNG	98	94	93	92	84	76	71	65	58	50	40	28	21	12	3
	JPEG	80	75	72	65	60	57	55	48	45	35	32	21	16	7	2
URPS	PNG	83	48	32	18	14	8	-	-	-	-	-	-	-	-	-
	JPEG	78	45	32	16	12	6	-	-	-	-	-	-	-	-	-
TIPS	PNG	74	32	14	10	5	-	-	-	-	-	-	-	-	-	-
	JPEG	68	28	12	11	6	-	-	-	-	-	-	-	-	-	-

- Query image resolution (pixel): 512 × 512
- Average image block size: 128KB (PNG), 66KB (JPEG)

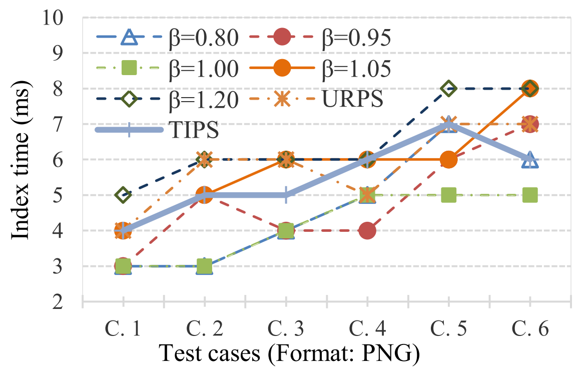


Fig. 12. Average index time for PNG image.

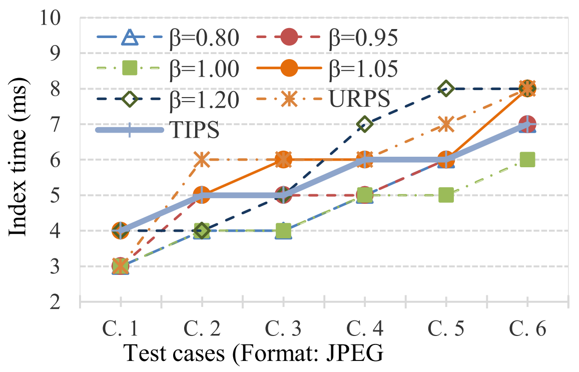


Fig. 13. Average index for JPEG image.

resolution and there are more layers can be matched during zooming.

5.8. Discussion

Some architectures and methods based on image pyramid have been proposed to optimize medical HRIs transmission in recent years. For example, paper [22, 27, 28, 32, 35] improve the retrieval, transmission and accessing performance of medical images by generating typical image pyramid and a tile-pyramid model named unbalanced ratio pyramid structure (URPS) [7]. The methods based on TIPS has fewer layers, the layer distribution is fixed and most cases are used for extremely large scale image sharing with 4 × ratio zooming. For the URPS, the number of layers is

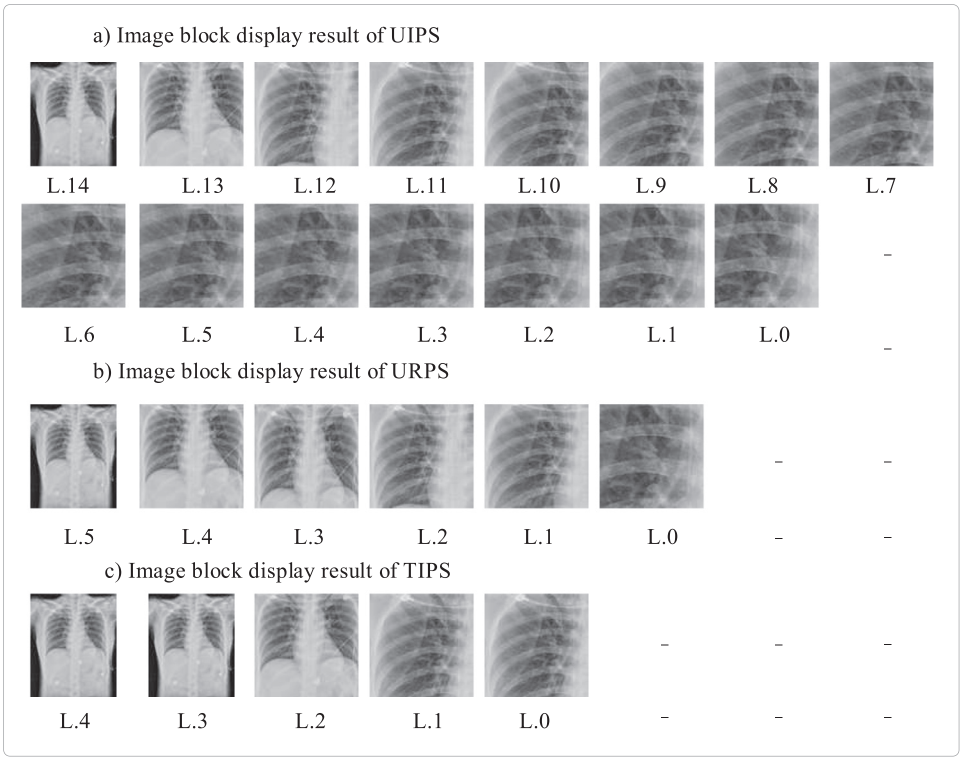

a) Image block display result of UIPS
L.14
L.13
L.12
L.11
L.10
L.9
L.8
L.7
L.6
L.5
L.4
L.3
L.2
L.1
L.0
b) Image block display result of URPS
L.5
L.4
L.3
L.2
L.1
L.0
c) Image block display result of TIPS
L.4
L.3
L.2
L.1
L.0

Fig. 14. Image block display results of the UIPS and TIPS.

generated based on rules and the layer distribution is fixed when the source image resolution is determined. Compared with the above methods, our approach provides the following advantages. (1) The first advantage stems from the image layering method based on geometric relation. As shown in Tables 4 and 5 and Fig. 6, the method can automatically generate different layer distributions by adjusting an unbalance factor (β). Such distributions more conform to the reading habits of unbalanced zoom ratio in clinical practice. Although this method generates more images and takes up more storage space, it still has the advantages of fast calculation and simple implementation. (2) The second advantage is that we present an indexing structure and algorithm based on hash table and lattice partitioning. It has the advantages of simple structure, fast index, and a small amount of stored data, easy storage and implementation as shown Figs. 4, 12 and 13. The indexing process can be done in memory and is independent of image parsing and conversion. (3) The third is that we present an image block query method based on similar matching, which can quickly match a specific image block from the index structure and pyramid layer as shown in Tables 6–8). As depicted in Fig. 14, from the perspective of the effect of magnifying image block when the user continuously zooms in on an image, the image block can be rendered more smoothly and avoids a very low resolution at the top layer. (4) Furthermore, DICOMweb service standard has not been fully considered in the above methods, our prototype system is fully compliant with DICOM standard [38], including DICOMweb [39] service and data formats, which can be seamlessly integrated with other existing medical systems or mobile applications, and used in various scenarios such as diagnosis, research, and education.

Certain limitations of this study should be mentioned. The unbalance factor is determined according to the user's requirements, which requires the user to fully understand the layer generation method, thereby reducing the convenience of the system to a certain extent. If the unbalance factor is configured unreasonably, the system will generate some very sparse layers and some very dense layers. The former may result in the requested image not being optimally matched, the later may result in the redundancy of the storage, thereby increasing the system overhead of the generation, storage, indexing, and querying.

6. Conclusion and future works

This paper presents an efficient architecture for optimizing medical HRIs transmission and rendering in mobile telemedicine systems. A new generation, index and query approach of the image pyramid scheme in this architecture is proposed. Compared with the existing approaches, this paper provides the following advantages: (1) We present a novel image layering method based on geometric relation, which can generate different layer distributions by adjusting an unbalance factor (β); (2) We design a simple index structure and efficient indexing algorithm based on hash table and lattice partitioning to improve the query efficiency of image blocks; (3) We devise a query scheme based on similar matching, in which the query image block can be matched in real-time; (4) The prototype system is fully compliant with DICOM standard, which can be seamlessly integrated with other existing medical systems or mobile applications and used in various scenarios such as diagnosis, research, and education.

For the future, we suggest extending this work in the following directions: The first is that we attempt to automatically determine the optimal unbalance factor by machine learning techniques to enhance the convenience of using this system. To further reduce image storage space and improve indexing and query efficiency, we will attempt to automatically remove invalid layers using a redundant layer detection mechanism.

Declaration of Competing Interest

The authors declare no potential conflicts of interest.

Acknowledgements

This work is supported by the National Natural Science Foundation of China [grant numbers: 8156029, 61662086, 81860318, 61962030, 61966036, 61862036, 61762090]; and the Natural Science Foundation of Yunnan Province [grant numbers: 2017FB097, 2016FA026], and the Project of Innovative Research Team of Yunnan Province [grant numbers: 2018HC019].

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cmpb.2019.105167.

References

[1] D. Teng, J. Kong, F. Wang, Scalable and flexible management of medical image big data, Distrib. Parallel Databases 37 (2019) 235–250.
[2] S. Juliet, E.B. Rajsingh, K. Ezra, Projection-based medical image compression for telemedicine applications, J. Dig. Imaging 28 (2015) 146–159.
[3] Y. Ma, Z. Jiang, H. Zhang, F. Xie, Y. Zheng, H. Shi, Y. Zhao, J. Shi, Generating region proposals for histopathological whole slide image retrieval, Comput. Meth. Prog. Bio. 159 (2018) 1–10.
[4] J.C. Souza, J.O. Bandeira Diniz, J.L. Ferreira, G.L. Franca da Silva, A. Correa Silva, A.C. de Paiva, An automatic method for lung segmentation and reconstruction in chest X-ray using deep neural networks, Comput. Meth. Prog. Bio. 177 (2019) 285–296.
[5] M. Asif, S.A. Khan, T. Hassan, M.U. Akram, A. Shaukat, Generation of high resolution medical images using super resolution via sparse representation, in: A. H.A. Abraham, A. Ella Hassanien, V. Snasel, A. Alimi (Eds.), Proceedings of the Third International Afro-European Conference for Industrial Advancement — AECIA 2016, Springer, 2016, pp. 288–298.
[6] M. Kowal, P. Filipczuk, A. Obuchowicz, J. Korbicz, R. Monczak, Computer-aided diagnosis of breast cancer based on fine needle biopsy microscopic images, Comput. Biol. Med. 43 (2013) 1563–1572.
[7] L. Qiao, Y. Li, X. Chen, S. Yang, P. Gao, H. Liu, Z. Feng, Y. Nian, M. Qiu, Medical high-resolution image sharing and electronic whiteboard system: a pure-web-based system for accessing and discussing lossless original images in telemedicine, Comput. Meth. Prog. Bio. 121 (2015) 77–91.
[8] Y. Zhuang, N. Jiang, Z. Wu, Q. Li, D.K.W. Chiu, H. Hu, Efficient and robust large medical image retrieval in mobile cloud computing environment, Inf. Sci. 263 (2014) 60–86.
[9] S. Pang, M.A. Orgun, Z. Yu, A novel biomedical image indexing and retrieval system via deep preference learning, Comput. Meth. Prog. Bio. 158 (2018) 53–69.
[10] E. Gibson, W. Li, C. Sudre, L. Fidon, D.I. Shakir, G. Wang, Z. Eaton-Rosen, R. Gray, T. Doel, Y. Hu, T. Whyntie, P. Nachev, M. Modat, D.C. Barratt, S. Ourselin, M.J. Cardoso, T. Vercauteren, NiftyNet: a deep-learning platform for medical imaging, Comput. Meth. Prog. Bio. 158 (2018) 113–122.
[11] A. Sene, B. Kamsu-Foguem, P. Rumeau, Telemedicine framework using case-based reasoning with evidences, Comput. Meth. Prog. Bio. 121 (2015) 21–35.
[12] A. Sene, B. Kamsu-Foguem, P. Rumeau, Decision support system for in-flight emergency events, Cognition, Technol. Work 20 (2018) 245–266.
[13] M.B. Doumbouya, B. Kamsu-Foguem, H. Kenfack, C. Foguem, Combining conceptual graphs and argumentation for aiding in the teleexpertise, Comput. Biol. Med. 63 (2015) 157–168.
[14] C.T. Li, D.H. Shih, C.C. Wang, Cloud-assisted mutual authentication and privacy preservation protocol for telecare medical information systems, Comput. Meth. Prog. Bio. 157 (2018) 191–203.
[15] N. Atallah, M. Khalifa, A. El Metwally, M. Househ, The prevalence and usage of mobile health applications among mental health patients in Saudi Arabia, Comput Meth Prog Bio 156 (2018) 163–168.
[16] M.C. Hu, K.C. Lan, W.C. Fang, Y.C. Huang, T.J. Ho, C.P. Lin, M.H. Yeh, P. Raknim, Y.H. Lin, M.H. Cheng, Y.T. He, K.C. Tseng, Automated tongue diagnosis on the smartphone and its applications, Comput Meth Prog Bio 174 (2019) 51–64.
[17] M.B. Doumbouya, B. Kamsu-Foguem, H. Kenfack, C. Foguem, Argumentative reasoning and taxonomic analysis for the identification of medical errors, Eng. Appl. Artif. Intell. 46 (2015) 166–179.
[18] M.B. Doumbouya, B. Kamsu-Foguem, H. Kenfack, C. Foguem, Argumentation graphs with constraint-based reasoning for collaborative expertise, Future Gener. Comput. Syst. 81 (2018) 16–29.
[19] H. Hypponen, J. Reponen, T. Laaveri, J. Kaipio, User experiences with different regional health information exchange systems in Finland, Int. J.Med. Inform. 83 (2014) 1–18.

[20] R. Rasmussen, A. Kushniruk, Digital video analysis of health professionals' interactions with an electronic whiteboard: a longitudinal, naturalistic study of changes to user interactions, J. Biomed. Inform. 46 (2013) 1068–1079.
[21] L. Liu, L. Wang, Q. Huang, L. Zhou, MobMivs: Implementing an efficient medical image visualization system for mobile telemedicine, in: Proceedings of the 9th International Conference on Information Technology in Medicine and Education, ITME 2018, Hangzhou, Zhejiang, China, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 242–246.
[22] H. Shen, D. Ma, Y. Zhao, H. Sun, S. Sun, R. Ye, L. Huang, B. Lang, Y. Sun, MIAPS: a web-based system for remotely accessing and presenting medical images, Comput. Meth. Prog. Biomed. 113 (2014) 266–283.
[23] N. Jiang, Y. Zhuang, D.K.W. Chiu, Multiple transmission optimization of medical images in recourse-constraint mobile telemedicine systems, Comput. Meth. Prog. Biomed. 145 (2017) 103–113.
[24] R.S. Dilmaghani, A. Ahmadian, M. Ghavami, A.H. Aghvami, Progressive medical Image transmission and compression, Signal Process. Lett. IEEE 11 (2004) 806–809.
[25] R. Dilmaghani, A. Ahmadian, M. Oghabian, A framework for progressive medical image transmission and compression, in: Proceedings of the Second Joint 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society][Engineering in Medicine and Biology, IEEE, 2002, pp. 1025–1026.
[26] Y. Biadgie, M.S. Kim, K.A. Sohn, Multi-resolution lossless image compression for progressive transmission and multiple decoding using an enhanced edge adaptive hierarchical interpolation, KSII Trans. Internet Inf. Syst. 11 (2017) 6017–6037.
[27] A. Kanakatte, R. Subramanya, A. Delampady, R. Nayak, B. Purushothaman, J. Gubbi, Cloud solution for histopathological image analysis using region of interest based compression, in: Proceedings of the International Conference of the IEEE Engineering in Medicine & Biology Society, 2017.
[28] T. Marques Godinho, R. Lebre, L.B. Silva, C. Costa, An efficient architecture to support digital pathology in standard medical imaging repositories, J. Biomed. Inform. 71 (2017) 190–197.
[29] R.S. Dilmaghani, A. Ahmadian, M. Ghavami, M. Oghabian, H. Aghvami, Multi rate/resolution control in progressive medical image transmission for the Region of Interest (ROI) using EZW, in: Proceedings of the IEEE Embs Asian-pacific Conference on Biomedical Engineering, 2003.
[30] Y. Zhuang, N. Jiang, Q. Li, L. Chen, C. Ju, Progressive batch medical image retrieval processing in mobile wireless networks, ACM Trans. Internet Technol. 15 (2015) 1–27.
[31] L. Liu, L. Liu, X. Fu, Q. Huang, Y. Zhang, Q. Luo, X. Xiong, SmartWADO: an extensible WADO middleware for regional medical image sharing, J. Dig. Imaging 28 (2015) 547–557.
[32] C.Y. Lien, H.C. Teng, D.J. Chen, W.C. Chu, C.H. Hsiao, A web-based solution for viewing large-sized microscopic images, J. Dig. Imaging 22 (2009) 275–285.
[33] I.H. Arka, K. Chellappan, Collaborative compressed I-cloud medical image storage with decompress viewer, Procedia Comput. Sci. 42 (2014) 114–121.
[34] O.S. Pianykh, Digital imaging and communications in medicine (DICOM): A Practical Introduction and Survival Guide, Second ed, Springer-Verlag, Berlin, Heidelberg, 2012.
[35] N. Lajara, J.L. Espinosa-Aranda, O. Deniz, G. Bueno, Optimum web viewer application for DICOM whole slide image visualization in anatomical pathology, Comput. Meth. Prog. Biomed. 179 (2019) 104983.
[36] G. Corredor, E. Romero, M. Iregui, An adaptable navigation strategy for virtual microscopy from mobile platforms, J. Biomed. Inform. 54 (2015) 39–49.
[37] A.N. Navaz, M.A. Serhani, N. Al-Qirim, M. Gergely, Towards an efficient and Energy-Aware mobile big health data architecture, Comput. Meth. Prog. Biomed. 166 (2018) 137–154.
[38] T.N.E.M. Association, DICOM standard, https://www.dicomstandard.org/current, 2019.
[39] T.N.E.M. Association, DICOMweb^{TM}, https://www.dicomstandard.org/dicomweb, 2019.
[40] S.S. Chandra, J.A. Dowling, C. Engstrom, Y. Xia, A. Paproki, A. Neubert, D. Rivest-Henault, O. Salvado, S. Crozier, J. Fripp, A lightweight rapid application development framework for biomedical image analysis, Comput. Meth. Prog. Biomed. 164 (2018) 193–205.
[41] I. Drnasin, M. Grgić, G. Gogić, JavaScript access to DICOM network and objects in web browser, J. Dig. Imaging 30 (2017) 1–10.
[42] DCM4CHE.ORG, Open source clinical image and object management (DCM4CHE and DCM4CHEE), https://www.dcm4che.org, 2018.
[43] T.A.S. Foundation, ApacheDS, The apache directory^{TM} project, http://directory.apache.org, 2019.