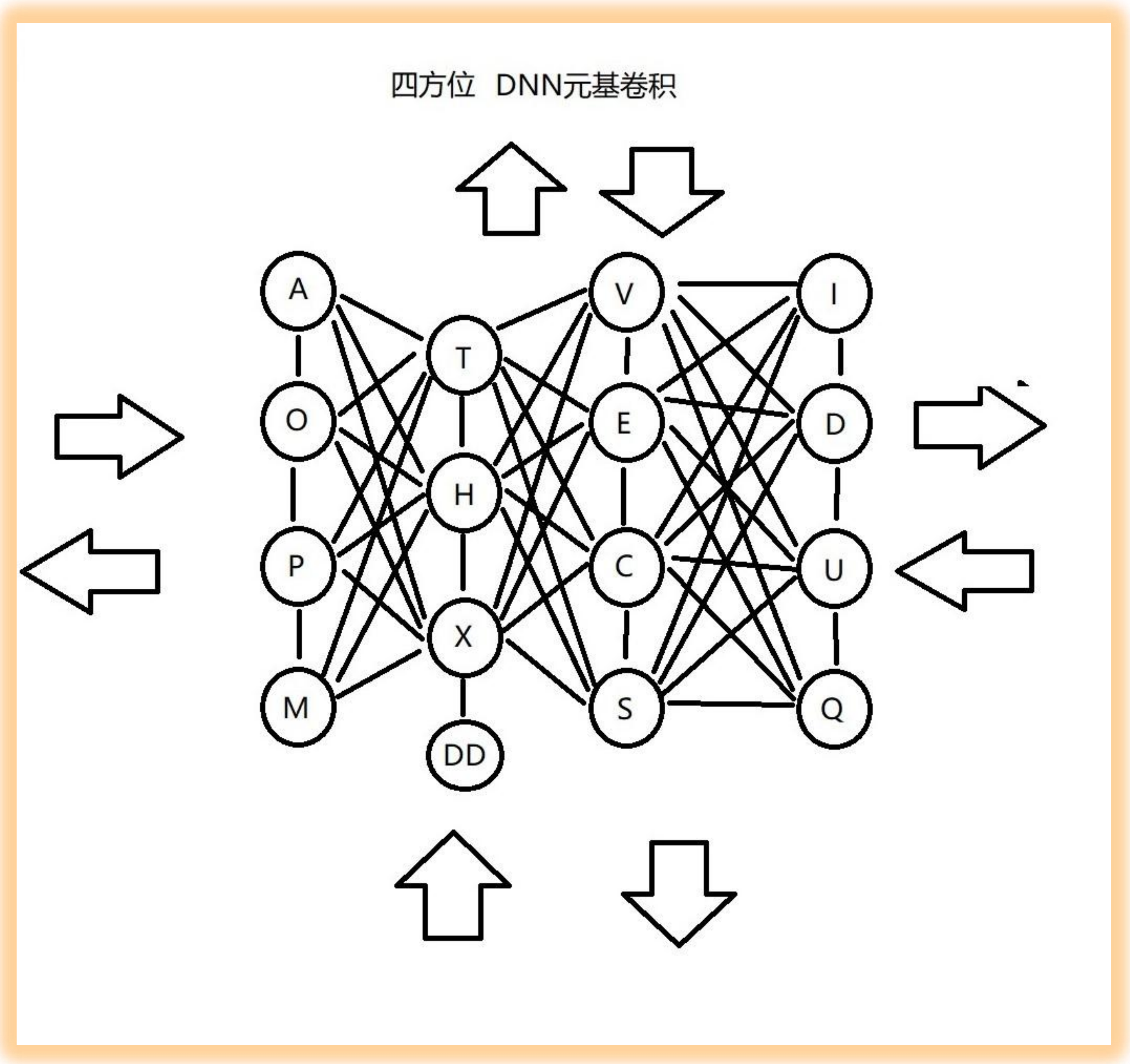
第十一章 DNA 卷积

第一节 DNA 卷积的动机

一开始这个应用的功能作品是模拟大脑的神经元计算, 现在慢慢的已经形成了一种固定的思路, 如图



一开始套取元基入座 DNN 是因为这样理解比较直白, 后来我思考了下, 元基如果是三维结构, 那

么计算 IO 则会千变万化, 于是我比较直白的生成AOPM VECS IDUQ TXH 四个同级组. 因为TXH 活泼, 那么做成编码器和解码器都是比较实用的. 当我思考到上下边缘进行元基计算, 那么TXH 将少了一个空位, 我思考用什么来弥补, 用U 变嘧啶? 就成了 IDQ, D. . DD? 用补码胞嘧啶对? 我想到了就用上了, 感觉不错用来做卷积计算, 别有一番风味.

第二节 DNA 卷积的应用需求

既然卷积设计好了, 那就要开始实际应用, DNA 元基卷积用来哪呢? 语义分析. 之后我编码描述.因为是卷积计算, 所以我的需求不再是快广准, 卷积计算的遍历方式首先不可能快, 所以我对卷积技术再养疗经[17]的体现是, 活跃, 质量, 智慧, 作为一种高级的烧脑计算方式, 我在设计 DNA 卷积的时候, 智慧性需求是我的首选, 一定要满足某一类功能的骨架我才专注时间在研发上, 不然, 费大量计算力还损耗性能. 记得第一卷德塔分词[1]的DNN 算法, 我会尝试进行元基化, 但目的很明确, 仅仅做需求内设计.

第三节 DNA 卷积的具体描述

AOPM 层属于智慧层, VECS 属于应用层, IDUQ 属于应激层, TXH DD 属于活性计算层, 那就好理解了, 生化计算的数字逻辑已经完全具备了.

假设数字 x 语义元基为 AAA, y 为OOO, 那么 x + y 为? 这很好理解

x – y 呢? DD 卷积就派上用场了. 稍后描述.

下面是养疗经 的monitor 卷积流的XCDX主函数

```
package AVQ.OEQ.cap;
import java.awt.*;

import java.awt.image.BufferedImage;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JSlider;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import org.bytedeco.javacpp.opencv_core.IplImage;
import org.bytedeco.javacv.Java2DFrameConverter;
import org.bytedeco.javacv.OpenCVFrameConverter;
import org.bytedeco.javacv.OpenCVFrameGrabber;

import MVQ.button.DetaButton;
import OSI.AOP.freetts.thread.read.ReadEnglish;
import OSI.SSI.ASU.OSU.PSU.MSU.pde.DecadeToPDS;

public class Monitor_XCDX extends JApplet{
	private static final long serialVersionUID = 1L;
	public int[][] mskr;
	public int[][] mskb;
	public int[][] mskg;
public int[][] diffg;
public int[][] diffr;
public int[][] diffb;
public int[][] rp;
public int[][] gp;
public int[][] bp;
public int[][] r2r;
public int[][] r2g;
public int[][] r2b;
public int[][] gpcar;
public int[][] gpcag;
public int[][] gpcab;

public int[][] showOCLDr;
public int[][] showORGNr;
public int[][] showOCLDg;
public int[][] showORGNg;
public int[][] showOCLDb;
public int[][] showORGNb;
public int findr= 0;
public boolean isRedButton= false;
public boolean isGreenButton= false;
public boolean isBlueButton= false;
public boolean isStreButton= false;
public boolean isSblButton= false;
public boolean isRcaButton= false;
public boolean isPcaButton= false;
public boolean isPcfButton= false;
public boolean isbt52Stop= false;
public boolean isbt53Stop= false;
public boolean isbt60Stop= false;
public boolean isbt73Stop= false;
public boolean isbt80Stop= false;
public boolean isbt81Stop= false;
public boolean isbt82Stop= false;
public boolean isbt83Stop= false;
public boolean isbt62Stop= false;
public boolean isbt43Stop= false;
public boolean isbt41Stop= false;
public boolean isbt88Stop= false;
public boolean isbt113Stop= false;
public boolean recordStop= true;
public DecadeToPDS decadeToPDS= new DecadeToPDS();
public BufferedImage stopBufferedImage;
public Map<String, Boolean> eyeShows= new HashMap<>();
public ArrayList<int[][]> imageList= new ArrayList<>();
public boolean isStop= false;
public String time = "";
public String newtime = "";
public long mi = 0;
public long newmi = 0;
public IplImage ipl;
public IplImage newcv;
public JSlider sliderx;
public JSlider sliderz;
public JSlider slidery;
public JSlider slidert;
public JSlider sliderl;
public Box br= new Box(BoxLayout.X_AXIS);
public Box bg= new Box(BoxLayout.X_AXIS);
public Box bb= new Box(BoxLayout.X_AXIS);
public JSlider sliderr;
public JSlider sliderg;
public JSlider sliderb;
public JButton btr;
public JButton btg;
public JButton btb;
public int facr= 0;
public int facg= 0;
public int facb= 0;

public JButton bt1;
public JButton bt2;
public JButton bt3;
public JButton bt4;
public JButton bt5;

public JButton bt00;
public JButton bt01;
public JButton bt02;
public JButton bt03;

public JButton bt10;
public JButton bt11;
public JButton bt12;
public JButton bt13;

public JButton bt20;
public JButton bt21;
public JButton bt22;
public JButton bt23;

public JButton bt30;
public JButton bt31;
public JButton bt32;
public JButton bt33;

public JButton bt40;
public JButton bt41;
public JButton bt42;
public JButton bt43;

public JButton bt50;
public JButton bt51;
public JButton bt52;
public JButton bt53;

public JButton bt60;
public JButton bt61;
public JButton bt62;
public JButton bt63;

public JButton bt70;
public JButton bt71;
public JButton bt72;
public JButton bt73;

public JButton bt80;
public JButton bt81;
public JButton bt82;
public JButton bt83;

public JButton bt84;
public JButton bt85;
public JButton bt86;
public JButton bt87;

public JButton bt88;
public JButton bt89;
public JButton bt90;
public JButton bt91;

public DetaButton bt92;

public DetaButton bt111;
public DetaButton bt112;
public DetaButton bt113;
public DetaButton bt114;

public DetaButton bt121;
public DetaButton bt122;
public DetaButton bt123;
public DetaButton bt124;

public DetaButton bt131;
public DetaButton bt132;
public DetaButton bt133;
public DetaButton bt134;
public DetaButton bt141;
public DetaButton bt142;
public DetaButton bt143;
public DetaButton bt144;

public DetaButton bt151;
public DetaButton bt152;
public DetaButton bt153;
public DetaButton bt154;

public org.bytedeco.javacv.Frame frame;
public int encry[][][];
public int encry_new[][][];
public int encry_fs[][][];
public IplImage difcv;
public IplImage oldcv;
public Image oldImage;
public BufferedImage imageForOutput;
public ReadEnglish readEnglish;
public Image newImage;
public Java2DFrameConverter paintConverter;
public Image difImage;
public Box sliderBox = new Box(BoxLayout.Y_AXIS);
public Box buttonBox0= new Box(BoxLayout.X_AXIS);
public Box buttonBox1= new Box(BoxLayout.X_AXIS);
public Box buttonBox2= new Box(BoxLayout.X_AXIS);
public Box buttonBox3= new Box(BoxLayout.X_AXIS);
public Box buttonBox4= new Box(BoxLayout.X_AXIS);
public Box buttonBox5= new Box(BoxLayout.X_AXIS);
public Box buttonBox6= new Box(BoxLayout.X_AXIS);
public Box buttonBox7= new Box(BoxLayout.X_AXIS);
public Box buttonBox8= new Box(BoxLayout.X_AXIS);
public Box buttonBox9= new Box(BoxLayout.X_AXIS);
public Box buttonBox10= new Box(BoxLayout.X_AXIS);
public Box buttonBox11= new Box(BoxLayout.X_AXIS);
public Box buttonBox12= new Box(BoxLayout.X_AXIS);
public Box buttonBox13= new Box(BoxLayout.X_AXIS);
public Box buttonBox14= new Box(BoxLayout.X_AXIS);
public Box buttonBox15= new Box(BoxLayout.X_AXIS);

public Box b1= new Box(BoxLayout.X_AXIS);
public Box b2= new Box(BoxLayout.X_AXIS);
public Box b3= new Box(BoxLayout.X_AXIS);
public Box b4= new Box(BoxLayout.X_AXIS);
public Box b5= new Box(BoxLayout.X_AXIS);
public Box b6= new Box(BoxLayout.X_AXIS);
public Box b7= new Box(BoxLayout.X_AXIS);

        public Button btn;
        public int[][] gdif;
        public OpenCVFrameGrabber grabber;
        public OpenCVFrameConverter.ToIplImage converter;
        public int stop= 0;
        public int has= 0;
        public int reg= 0;
        public int facx= 7;
        public int facy= 100;
        public int facz= 50;
        public int fact= 50;
        public int facl= 3;
        public long last= 0;
        int encry_c= 2;
        int encry_c_new= 2;
        int encry_c_fs= 2;
        int[][] out;
        int[][] out_oldr= null;
        int[][] out_oldg= null;
        int[][] out_oldb= null;

        int[][] out_old2r= null;
        int[][] out_old2g= null;
        int[][] out_old2b= null;
        int[][] out_old1= null;
```

```
int[][] out_old2= null;
int[][] out_old3= null;
int[][] out_old4= null;
int[][] out_old5= null;
int q= 0;
int q_new= 0;
int q_fs= 0;
int finalEncry[][];
int finalEncryNew[][];
int finalEncryFs[][];
public Image img;
public boolean isbt114Stop;
public boolean isbt121Stop;
public boolean isbt122Stop;
public boolean isbt123Stop;
public boolean isbt124Stop;
public static void main(String[] argv) {
        Monitor_XCDX m= new Monitor_XCDX();
        m.init();
        m.setVisible(true);
        JFrame f= new JFrame();
        f.setLayout(null);
        f.add(m);
        m.sliderx= new JSlider(0, 360);
        m.sliderx.setSnapToTicks(true);
        m.sliderx.setPaintTicks(true);
        m.sliderx.setMajorTickSpacing(5);
        m.sliderx.setMinorTickSpacing(1);
        m.sliderx.addChangeListener(
                        new ChangeListener() {
                                public void stateChanged(ChangeEvent event)  {
                                        JSlider source= (JSlider) event.getSource();
                                        m.facx= source.getValue();
                                }
                        });

        m.slidery = new JSlider(0,360);
        m.slidery.setSnapToTicks(true);
        m.slidery.setPaintTicks(true);
        m.slidery.setMajorTickSpacing(5);
        m.slidery.setMinorTickSpacing(0);
        m.slidery.addChangeListener(
                        new ChangeListener() {
                                public void stateChanged(ChangeEvent event)  {
                                        JSlider source = (JSlider) event.getSource();
                                        m.facy = source.getValue();
                                }
                        });

        m.sliderz = new JSlider(0,360);
        m.sliderz.setSnapToTicks(true);
        m.sliderz.setPaintTicks(true);
        m.sliderz.setMajorTickSpacing(5);
        m.sliderz.setMinorTickSpacing(0);
        m.sliderz.addChangeListener(
                        new ChangeListener() {
                                public void stateChanged(ChangeEvent event)  {
                                        JSlider source = (JSlider) event.getSource();
                                        m.facz = source.getValue();
                                }
                        });
        m.slidert= new JSlider(0,100);
        m.slidert.setSnapToTicks(true);
        m.slidert.setPaintTicks(true);
        m.slidert.setMajorTickSpacing(5);
        m.slidert.setMinorTickSpacing(1);
        m.slidert.addChangeListener(
                        new ChangeListener() {
                                public void stateChanged(ChangeEvent event)  {
                                        JSlider source = (JSlider) event.getSource();
                                        m.fact= source.getValue();
                                }
                        });
        m.sliderl = new JSlider(0,360);
```

```
            m.sliderl.setSnapToTicks(true);
            m.sliderl.setPaintTicks(true);
            m.sliderl.setMajorTickSpacing(5);
            m.sliderl.setMinorTickSpacing(0);
            m.sliderl.addChangeListener(
                        new ChangeListener() {
                                    public void stateChanged(ChangeEvent event)  {
                                                JSlider source= (JSlider) event.getSource();
                                                m.facl= source.getValue();
                                    }
                        });
            m.sliderBox.add(m.sliderx);
            m.sliderBox.add(m.slidery);
            m.sliderBox.add(m.sliderz);
            m.sliderBox.add(m.slidert);
            m.sliderBox.add(m.sliderl);
            m.sliderBox.setBounds(000, 860, 1200, 750);
            f.add(m.sliderBox);
            f.setTitle("ButtonDemo");
            f.setLocationRelativeTo(null);
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            f.setSize(1000,1050);
            f.setVisible(true);
}
@Override
public void init() {
            decadeToPDS.IV_(decadeToPDS);
            readEnglish= new ReadEnglish();
            grabber= new OpenCVFrameGrabber(0);
            converter= new OpenCVFrameConverter.ToIplImage();
            try {
                        if(!grabber.equals(null)) {
                                    grabber.start();
                        }
                        Thread.sleep(2000);
                        frame = grabber.grab();
            } catch (Exception e) {
                        e.printStackTrace();
            }
            paintConverter = new Java2DFrameConverter();
            difImage = paintConverter.getBufferedImage(frame, 1);
            BufferedImage imageInit = (BufferedImage) difImage;
            encry = new int[encry_c][imageInit.getWidth()][imageInit.getHeight()];
            encry_new = new int[encry_c_new][imageInit.getWidth()][imageInit.getHeight()];
            encry_fs = new int[encry_c_fs][imageInit.getWidth()][imageInit.getHeight()];
            out_oldr = new int[imageInit.getWidth()][imageInit.getHeight()];
            out_oldg = new int[imageInit.getWidth()][imageInit.getHeight()];
            out_oldb = new int[imageInit.getWidth()][imageInit.getHeight()];
            out_old1 = new int[imageInit.getWidth()][imageInit.getHeight()];
            out_old2r = new int[imageInit.getWidth()][imageInit.getHeight()];
            out_old2g = new int[imageInit.getWidth()][imageInit.getHeight()];
            out_old2b = new int[imageInit.getWidth()][imageInit.getHeight()];
            out_old3 = new int[imageInit.getWidth()][imageInit.getHeight()];
            out_old4 = new int[imageInit.getWidth()][imageInit.getHeight()];
            out_old5 = new int[imageInit.getWidth()][imageInit.getHeight()];
            this.setBounds(5, 5, 895, 675-48);
            this.start();
}

public void stop() {
            try {
                        if(grabber!=null) {
                                    grabber.stop();
                        }
                        stop = 1;
            } catch (Exception e1) {
                        e1.printStackTrace();
            }
}

public void start(){
}

public void paint(Graphics g){
		try {
			Monitor_XCDX_Animation.XCDX_paint(this, g);
		}catch(Exception e) {
			//To do
		}
	}

	public void expand(int[][] show, int i, int j, int fac, int k) {
		if(k== 1) {
			for(int v= 0; v< fac; v++) {
				for(int h= 0; h< fac; h++) {
					if(i+ v>= 0 && i+ v< show.length&& h+ j>= 0&& h+ j< show[0].length) {
						show[i+ v][h+ j]= 255;
					}
				}
			}
		}
		if(k== 2) {
			for(int v= -fac; v< 0; v++) {
				for(int h= 0; h< fac; h++) {
					if(i+ v>= 0&& i+ v< show.length&& h+ j>= 0&& h+ j< show[0].length) {
						show[i+ v][h+ j]= 255;
					}
				}
			}
		}
		if(k== 3) {
			for(int v= 0; v< fac; v++) {
				for(int h= -fac; h< 0; h++) {
					if(i+ v>= 0&& i+ v< show.length&& h+ j>= 0&& h+ j< show[0].length) {
						show[i+v][h+j]=255;
					}
				}
			}
		}
		if(k== 4) {
			for(int v= -fac; v< 0; v++) {
				for(int h= -fac; h< 0; h++) {
					if(i+ v>= 0 && i+ v< show.length&& h+ j>= 0&& h+ j< show[0].length) {
						show[i+ v][h+ j]=255;
					}
				}
			}
		}
	}
	public int[][] findDiff(int[][] out, int[][] out_old) {
		int[][] diff= new int[out.length][out[0].length];
		if(out_old!= null) {
			for (int i= 0; i< diff[0].length; ++i) {
				for (int j= 0; j< diff.length; ++j) {
					if(out[j][i]!= out_old[j][i]) {
						diff[j][i]= out[j][i];
					}
					out_old[j][i]= out[j][i];
				}
			}
		}else {
			diff= out;
		}
		return diff;
	}

	public int getMskFilter(int[][] fb, int[][] msk, int i, int j, int size, Map<String, Integer> map) {
		if(fb[j][i]!= 255) {
			return size;
		}
		if(msk[j][i]== 1) {
			return size;
		}
		if(size> 3000) {
			return size;
		}
		size++;
		map.put(j+ ","+ i, 1);
```

```
				msk[j][i]= 1 ;
				if(i+ 1< fb[0].length) {
					size= getMskFilter(fb, msk, i+1, j, size, map);
				}
				if(i- 1 >= 0) {
					size = getMskFilter(fb, msk, i- 1, j, size, map);
				}
				if(j+ 1 < fb.length) {
					size= getMskFilter(fb, msk, i, j+ 1, size, map);
				}
				if(j- 1>= 0) {
					size= getMskFilter(fb, msk, i, j- 1, size, map);
				}
				return size;
		}

		public ArrayList<Cordination> findCordination() {
				ArrayList<Cordination> clist= new ArrayList<Cordination>();
				BufferedImage difTemp= (BufferedImage) difImage;
				int h= difTemp.getHeight();
				int w= difTemp.getWidth();
				gdif= new int[h][w];
				int cp= -16777216;
				// 得到map
				for(int i= 0; i< h; i++) {
					for(int j= 0; j< w; j++) {
						if(difTemp.getRGB(j, i)!= cp) {
							gdif[i][j]= 1;
						}
					}
				}
				// 计算边缘
				Cordination c= new Cordination();
				c.h0= 999999;
				c.w0= 999999;
				c.h1= 0;
				c.w1= 0;
				for(int i= 0; i< h; i++) {
					for(int j= 0; j< w; j++) {
						if (gdif[i][j]== 1) {
							if (c.h0> i) {
								c.h0= i;
							}
							if (c.w0> j) {
								c.w0= j;
							}
							if (c.h1< i) {
								c.h1= i;
							}
							if (c.w1< i) {
								c.w1= i;
							}
						}
					}
				}
				clist.add(c);
				return clist;
		}
}
```

```
package AVQ.OEQ.cap;
import java.awt.*;

import java.awt.image.BufferedImage;
import ESU.image.ToolkitImageToBufferImage;

public class Monitor_XCDX_Animation{
	public static void XCDX_paint(Monitor_XCDX monitor, Graphics g){
		try {
			if(monitor.grabber!= null) {
				try {
					monitor.frame= monitor.grabber.grab();
					}catch(Exception e) {
						return;
					}
					if(monitor.frame!= null) {
						if(monitor.isStop) {
							//return;
						}
						//预处理
						try {
							monitor.difImage= monitor.paintConverter.getBufferedImage(monitor.frame, 1);
						}catch(Exception e) {
							return;
						}
						BufferedImage image;
						if(monitor.isStop) {
							//image= new BufferedImage(640, 480, BufferedImage.TYPE_INT_RGB);
							//image.getGraphics().drawImage(img, 0, 0, 640, 480, this);
							image= new ToolkitImageToBufferImage().toolkitImageToBufferImage(monitor.img, 0, 0,
640, 480, monitor);
						}else {
							image= (BufferedImage) monitor.difImage;
						}
						monitor.rp= new int[image.getWidth()][image.getHeight()];
						monitor.gp= new int[image.getWidth()][image.getHeight()];
						monitor.bp= new int[image.getWidth()][image.getHeight()];

						Monitor_XCDX_Animation_EyeScan.XCDX_paint_eyeScan(monitor, g, image);
						Monitor_XCDX_Animation_Pca.XCDX_paint_pca(monitor, g, image);
						Monitor_XCDX_Animation_Ica.XCDX_paint_ica(monitor, g, image);
						Monitor_XCDX_Animation_PcfButton.XCDX_paint_PcfButton(monitor, g, image, monitor.gpcar,
monitor.gpcag, monitor.gpcab);
						Monitor_XCDX_Animation_Pde.XCDX_paint_pde(monitor, g, image);
					}
					monitor.q+= 1;
					if(monitor.q>= monitor.encry_c) {
						monitor.q= 0;
					}
					monitor.q_new+= 1;
					if(monitor.q_new>= monitor.encry_c_new) {
						monitor.q_new= 0;
					}
					monitor.q_fs+= 1;
					if(monitor.q_fs>= monitor.encry_c_fs) {
						monitor.q_fs= 0;
					}
				}
			}catch(Exception e) {
				//e.printStackTrace();
				//System.out.println(e.getMessage());
			}
		}
}
```

```
package AVQ.OEQ.cap;
import java.awt.*;

import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.Date;
import javax.imageio.ImageIO;
import OSI.AOP.freetts.thread.read.ReadEnglish;
import OSI.OPE.SI.SD.SU.SQ.ASU.OSU.PSU.MSU.AVQ.ASQ.ASU.MPE.procedure.pde.RangePDI;

public class Monitor_XCDX_Animation_Pde{
	@SuppressWarnings("deprecation")
	public static void XCDX_paint_pde(Monitor_XCDX monitor, Graphics g, BufferedImage image) throws IOException{
		if(monitor.isbt62Stop) {
			monitor.rp= new PEU.P.image.Emboss().P(monitor.rp);
			monitor.gp= new PEU.P.image.Emboss().P(monitor.gp);
			monitor.bp= new PEU.P.image.Emboss().P(monitor.bp);
		}
		if(monitor.isbt113Stop) {
```

```
            monitor.rp= new PEU.P.image.Sobel().P(monitor.rp, 1);
            monitor.gp= new PEU.P.image.Sobel().P(monitor.gp, 1);
            monitor.bp= new PEU.P.image.Sobel().P(monitor.bp, 1);
}
if(monitor.isbt43Stop) {
            monitor.rp= new PEU.P.image.Guassian().P_1D(monitor.rp, 3, 3, 1.66);
            monitor.gp= new PEU.P.image.Guassian().P_1D(monitor.gp, 3, 3, 1.66);
            monitor.bp= new PEU.P.image.Guassian().P_1D(monitor.bp, 3, 3, 1.66);
}
if(monitor.isbt41Stop) {
            monitor.rp= new PEU.P.image.Laplacian().P(monitor.rp);
            monitor.gp= new PEU.P.image.Laplacian().P(monitor.gp);
            monitor.bp= new PEU.P.image.Laplacian().P(monitor.bp);
}
if(monitor.isbt41Stop) {
            monitor.rp= new PEU.P.image.Laplacian().P(monitor.rp);
            monitor.gp= new PEU.P.image.Laplacian().P(monitor.gp);
            monitor.bp= new PEU.P.image.Laplacian().P(monitor.bp);
}
if(monitor.isbt114Stop) {
            monitor.rp= new RangePDI().IOE(monitor.rp, monitor.fact);
            monitor.gp= new RangePDI().IOE(monitor.gp, monitor.fact);
            monitor.bp= new RangePDI().IOE(monitor.bp, monitor.fact);
}
if(monitor.isbt121Stop) {
            monitor.rp= new RangePDI().IPE(monitor.rp, monitor.facy);
            monitor.gp= new RangePDI().IPE(monitor.gp, monitor.facy);
            monitor.bp= new RangePDI().IPE(monitor.bp, monitor.facy);
}
if(monitor.isbt124Stop) {
            monitor.rp= new RangePDI().IPE_AOPM_VECS_IDUQ_TXH(monitor.rp, monitor.facy);
            monitor.gp= new RangePDI().IPE_AOPM_VECS_IDUQ_TXH(monitor.gp, monitor.facy);
            monitor.bp= new RangePDI().IPE_AOPM_VECS_IDUQ_TXH(monitor.bp, monitor.facy);
}
if(monitor.isbt122Stop) {
            monitor.rp= new RangePDI().QPE(monitor.rp, monitor.facx);
            monitor.gp= new RangePDI().QPE(monitor.gp, monitor.facx);
            monitor.bp= new RangePDI().QPE(monitor.bp, monitor.facx);
}
if(monitor.isbt123Stop) {
            double facxd= ((double)monitor.facx)/360;
            monitor.rp= monitor.decadeToPDS.doPDSMatrix(monitor.decadeToPDS, monitor.rp, facxd);
            monitor.gp= monitor.decadeToPDS.doPDSMatrix(monitor.decadeToPDS, monitor.gp, facxd);
            monitor.bp= monitor.decadeToPDS.doPDSMatrix(monitor.decadeToPDS, monitor.bp, facxd);
}

for (int i= 0; i< image.getHeight(); ++i) {
            for (int j= 0; j< image.getWidth(); ++j) {
                        int pixel= (monitor.rp[j][i]<< 16)| (monitor.gp[j][i]<< 8)| (monitor.bp[j][i]) ;
                        if(monitor.showOCLDr[j][i]== 255) {
                                    if(monitor.r2r[j][i]> 30) {
                                                pixel= (monitor.r2r[j][i]<< 16) ;
                                    }
                        }
                        if(monitor.showOCLDg[j][i]== 255) {
                                    if(monitor.r2g[j][i]> 30) {
                                                pixel= pixel| (monitor.r2g[j][i]<< 8) ;
                                    }
                        }

                        if(monitor.showOCLDb[j][i] == 255) {
                                    if(monitor.r2b[j][i]> 30) {
                                                pixel= pixel| monitor.r2b[j][i]  ;
                                    }
                        }
                        image.setRGB(j, i, pixel);
            }
}
if(!monitor.recordStop) {
            if(monitor.imageList.size()< 32*60*60) {
                        System.out.println(1);
                        int width= image.getWidth();
                        int height= image.getHeight();
                        int[][] flips= new int[width][height];
                        for(int i= 0; i< image.getHeight(); ++i) {
                            for(int j= 0; j< image.getWidth(); ++j) {
                                flips[j][i]= image.getRGB(j, i);
                            }
                        }
                        monitor.imageList.add(flips);
                    }
                }
                g.drawImage(image, 0, 0, 900, 680, monitor);// 绘出图形文件
                monitor.imageForOutput= image;
                if(monitor.findr== 2) {
                    if(monitor.readEnglish.finish== 1) {
                        monitor.readEnglish= new ReadEnglish();
                        monitor.readEnglish.I_PreReadText("attension please");
                        monitor.readEnglish.start();
                    }
                    //write
                    Date d= new Date();
                    monitor.newtime= "" + d.getDay() + d.getHours() + d.getMinutes();
                    monitor.newmi= d.getTime();
                    long v= Math.abs(monitor.newmi- monitor.mi);
                    if(monitor.newtime.equalsIgnoreCase(monitor.time)&& v> 3000){
                        File outputBin= new File("C:\\Users\\Administrator\\Desktop\\monit\\rec"
                                + monitor.newtime+ monitor.newmi+ ".jpg");
                        ImageIO.write(image, "png", outputBin);
                        monitor.mi= monitor.newmi;
                    }
                    monitor.time= monitor.newtime.toString();
                }

        }
}
```

```
package AVQ.OEQ.cap;
import java.awt.*;

import java.awt.image.BufferedImage;

import SVQ.stable.StableVision;

public class Monitor_XCDX_Animation_PcfButton{
        @SuppressWarnings({"unused"})
        public static void XCDX_paint_PcfButton(Monitor_XCDX monitor, Graphics g, BufferedImage image
                        , int [][] gpcar, int[][] gpcag, int[][] gpcab){
                try {
                    int[][] diff2r;
                    int[][] diff2g;
                    int[][] diff2b;
                    int[][] ccar= new int[image.getWidth()][image.getHeight()];
                    int[][] ccag= new int[image.getWidth()][image.getHeight()];
                    int[][] ccab= new int[image.getWidth()][image.getHeight()];
                    //CCA 关联成分分析
                    if(monitor.isPcaButton) {
                        if(monitor.isRedButton) {
                            diff2r= monitor.findDiff(gpcar, monitor.out_old2r);
                            ccar= new PEU.P.image.Dilation()
                                    .P(diff2r, StableVision.diaMask);
                        }
                        if(monitor.isGreenButton == true) {
                            diff2g = monitor.findDiff(gpcag, monitor.out_old2g);
                            ccag = new PEU.P.image.Dilation()
                                    .P(diff2g, StableVision.diaMask);
                        }
                        if(monitor.isBlueButton == true) {
                            diff2b = monitor.findDiff(gpcab, monitor.out_old2b);
                            ccab = new PEU.P.image.Dilation()
                                    .P(diff2b, StableVision.diaMask);
                        }
                    }else {
                        ccar= gpcar;
                        ccag= gpcag;
                        ccab= gpcab;
                    }
```

```
//OJLID
int cxr= 0;
int cyr= 0;
monitor.showOCLDr= new int[image.getWidth()][image.getHeight()];
monitor.showORGNr= new int[image.getWidth()][image.getHeight()];
int cxg= 0;
int cyg= 0;
int findg= 0;
monitor.showOCLDg= new int[image.getWidth()][image.getHeight()];
monitor.showORGNg= new int[image.getWidth()][image.getHeight()];
int cxb= 0;
int cyb= 0;
int findb= 0;
monitor.showOCLDb= new int[image.getWidth()][image.getHeight()];
monitor.showORGNb= new int[image.getWidth()][image.getHeight()];

if(monitor.isPcfButton) {
        for(int i= 0; i< image.getHeight(); ++i) {
                for(int j= 0; j< image.getWidth(); ++j) {
                        if(monitor.isRedButton) {
                                if(ccar[j][i]> 0) {
                                        int x= j;
                                        int y= i;
                                        if(cxr== 0&& cyr== 0) {
                                                cxr= cxr+ x;
                                                cyr= cyr+ y;
                                        }
                                        cxr= cxr+ x;
                                        cyr= cyr+ y;
                                        monitor.findr= 1;
                                        monitor.showOCLDr[x][y]= 255;
                                        monitor.showORGNr[x][y]= 255;
                                        cxr= cxr>> 1;
        cyr= cyr>> 1;
        float dx= cxr- x;
        float dy= cyr- y;
        float co= dy/ dx;
        int dis = Math.abs(cxr- x);
        //欧基里德填充
        for(int k= 0; k< dis; k++) {
                if(cxr>= x&& cyr>= y) {
                        monitor.showOCLDr[x+ k][y+ (int)(k* co)]= 255;
                        monitor.expand(monitor.showOCLDr, x+ k, y+ (int)(k* co), monitor.fact, 1);
                }
                if(cxr< x&& cyr>= y) {
                        monitor.showOCLDr[x- k][y- (int)(k * co)]= 255;
                        monitor.expand(monitor.showOCLDr, x- k, y- (int)(k* co), monitor.fact, 2);
                }
                if(cxr>= x&& cyr< y) {
                        monitor.showOCLDr[x+ k][y+ (int)(k * co)]= 255;
                        monitor.expand(monitor.showOCLDr, x+ k, y+ (int)(k* co), monitor.fact, 3);
                }
                if(cxr< x&& cyr< y) {
                        monitor.showOCLDr[x- k][y- (int)(k * co)]= 255;
                        monitor.expand(monitor.showOCLDr, x- k, y- (int)(k* co), monitor.fact, 4);
                }
        }
                                }
                        }
                        if(monitor.isGreenButton) {
                                if(ccag[j][i]> 0) {
                                        int x= j;
                                        int y= i;
                                        if(cxg== 0&& cyg== 0) {
                                                cxg= cxg+ x;
                                                cyg= cyg+ y;
                                        }
                                        cxg=cxg+ x;
                                        cyg=cyg+ y;
                                        findg= 1;
                                        monitor.showOCLDg[x][y]= 255;
                                        monitor.showORGNg[x][y]= 255;
                                        cxg= cxg>> 1;
```

```
                    cyg= cyg>> 1;
                    float dx= cxg- x;
                    float dy= cyg- y;
                    float co= dy/ dx;
                    int dis= Math.abs(cxg - x);
                    //欧基里德填充
                    for(int k= 0; k< dis; k++) {
                        if(cxg>= x&& cyg>= y) {
                            monitor.showOCLDg[x+ k][y+ (int)(k* co)]= 255;
                            monitor.expand(monitor.showOCLDg,x+ k, y+ (int)(k* co), monitor.fact, 1);
                        }
                        if(cxg< x&& cyg>= y) {
                            monitor.showOCLDg[x- k][y - (int)(k* co)]= 255;
                            monitor.expand(monitor.showOCLDg, x- k, y- (int)(k* co), monitor.fact, 2);
                        }
                        if(cxg>= x&& cyg< y) {
                            monitor.showOCLDg[x+ k][y+ (int)(k* co)] = 255;
                            monitor.expand(monitor.showOCLDg, x+ k, y+ (int)(k * co), monitor.fact, 3);
                        }
                        if(cxg< x&& cyg< y) {
                            monitor.showOCLDg[x- k][y- (int)(k* co)] = 255;
                            monitor.expand(monitor.showOCLDg, x- k, y- (int)(k * co), monitor.fact, 4);
                        }
                    }
                            }
                        }
                        if(monitor.isBlueButton) {
                            if(ccab[j][i]> 0) {
                                int x= j;
                                int y= i;
                                if(cxb== 0&& cyb== 0) {
                                    cxb= cxb+ x;
                                    cyb= cyb+ y;
                                }
                                cxb= cxb+ x;
                                cyb= cyb+ y;
                                findb= 1;
                                monitor.showOCLDb[x][y]= 255;
                                monitor.showORGNb[x][y]= 255;
                                cxb= cxb>> 1;
                        cyb= cyb>> 1;
                        float dx= cxb- x;
                        float dy= cyb- y;
                        float co= dy/ dx;
                        int dis= Math.abs(cxb- x);
                        //欧基里德填充
                        for(int k= 0; k< dis; k++) {
                            if(cxb>= x && cyb>= y) {
                                monitor.showOCLDb[x+ k][y+ (int)(k* co)]= 255;
                                monitor.expand(monitor.showOCLDb, x+ k, y+ (int)(k* co),
monitor.fact, 1);
                            }
                            if(cxb< x && cyb>= y) {
                                monitor.showOCLDb[x- k][y- (int)(k* co)]= 255;
                                monitor.expand(monitor.showOCLDb, x- k, y- (int)(k* co), monitor.fact,
2);
                            }
                            if(cxb>= x&& cyb< y) {
                                monitor.showOCLDb[x+ k][y+ (int)(k* co)]= 255;
                                monitor.expand(monitor.showOCLDb, x+ k, y + (int)(k* co),
monitor.fact, 3);
                            }
                            if(cxb< x&& cyb< y) {
                                monitor.showOCLDb[x- k][y- (int)(k* co)]= 255;
                                monitor.expand(monitor.showOCLDb, x- k, y- (int)(k* co), monitor.fact,
4);
                            }
                        }
                            }
                        }
                    }
                }
            }else {
                monitor.showOCLDr= ccar;
```

```
                                        monitor.showORGNr= ccar;

                                        monitor.showOCLDg= ccag;
                                        monitor.showORGNg= ccag;

                                        monitor.showOCLDb= ccab;
                                        monitor.showORGNb= ccab;
                              }
                    }catch(Exception e) {
                              //e.printStackTrace();
                              //System.out.println(e.getMessage());
                    }
          }
}
```

```
package AVQ.OEQ.cap;
import java.awt.*;

import java.awt.image.BufferedImage;

public class Monitor_XCDX_Animation_Pca{
          public static void XCDX_paint_pca(Monitor_XCDX monitor, Graphics g, BufferedImage image){
                    try {
                              //PCA
                              int[][] str= new int[image.getWidth()][image.getHeight()];
                              int[][] stg= new int[image.getWidth()][image.getHeight()];
                              int[][] stb= new int[image.getWidth()][image.getHeight()];
                              if(monitor.isStreButton){
                                        if(monitor.isRedButton){
                                                  str= new PEU.P.image.Strech().P(monitor.rp, 0.1, 0.9);
                                        }
                                        if(monitor.isGreenButton){
                                                  stg= new PEU.P.image.Strech().P(monitor.gp, 0.1, 0.9);
                                        }
                                        if(monitor.isBlueButton){
                                                  stb= new PEU.P.image.Strech().P(monitor.bp, 0.1, 0.9);
                                        }
                              }else {
                                        str= monitor.rp;
                                        stg= monitor.gp;
                                        stb= monitor.bp;
                              }
                              monitor.r2r= new int[image.getWidth()][image.getHeight()];
                              monitor.r2g= new int[image.getWidth()][image.getHeight()];
                              monitor.r2b= new int[image.getWidth()][image.getHeight()];
                              if(monitor.isSblButton) {
                                        if(monitor.isRedButton) {
                                                  monitor.r2r= new PEU.P.image.Sobel().P(str, 1);
                                        }
                                        if(monitor.isGreenButton) {
                                                  monitor.r2g= new PEU.P.image.Sobel().P(stg, 1);
                                        }
                                        if(monitor.isBlueButton) {
                                                  monitor.r2b= new PEU.P.image.Sobel().P(stb, 1);
                                        }
                              }else {
                                        monitor.r2r= str;
                                        monitor.r2g= stg;
                                        monitor.r2b= stb;
                              }
                              int[][] gthdr= new int[image.getWidth()][image.getHeight()];
                              int[][] gthdg= new int[image.getWidth()][image.getHeight()];
                              int[][] gthdb= new int[image.getWidth()][image.getHeight()];
                              if(monitor.isSblButton) {
                                        if(monitor.isRedButton) {
                                                  gthdr= new PEU.P.image.Threshold().P(monitor.r2r, monitor.facx);
                                        }
                                        if(monitor.isGreenButton) {
                                                  gthdg= new PEU.P.image.Threshold().P(monitor.r2g, monitor.facx);
                                        }
                                        if(monitor.isBlueButton) {
                                                  gthdb= new PEU.P.image.Threshold().P(monitor.r2b, monitor.facx);
                                        }
                                }else {
                                        gthdr= monitor.r2r;
                                        gthdg= monitor.r2g;
                                        gthdb= monitor.r2b;
                                }
                                monitor.diffr= monitor.findDiff(gthdr, monitor.out_oldr);
                                monitor.diffg= monitor.findDiff(gthdg, monitor.out_oldg);
                                monitor.diffb= monitor.findDiff(gthdb, monitor.out_oldb);
                        }catch(Exception e) {
                                //e.printStackTrace();
                                //System.out.println(e.getMessage());
                        }
                }
}
```

```
package AVQ.OEQ.cap;
import java.awt.*;

import java.awt.image.BufferedImage;
import java.util.Iterator;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
public class Monitor_XCDX_Animation_Ica{
        public static void XCDX_paint_ica(Monitor_XCDX monitor, Graphics g, BufferedImage image){
                try {
                        //ICA
                        monitor.mskr= new int[image.getWidth()][image.getHeight()];
                        monitor.mskg= new int[image.getWidth()][image.getHeight()];
                        monitor.mskb= new int[image.getWidth()][image.getHeight()];
                        monitor.gpcar = new int[image.getWidth()][image.getHeight()];
                        monitor.gpcag = new int[image.getWidth()][image.getHeight()];
                        monitor.gpcab = new int[image.getWidth()][image.getHeight()];
                        Map<String, Integer> map= new ConcurrentHashMap<>();
                        if(monitor.isRcaButton) {
                                for (int i= 0; i< image.getHeight(); ++i) {
                                        for (int j= 0; j< image.getWidth(); ++j) {
                                                if(monitor.isRedButton) {
                                                        if(monitor.mskr[j][i]== 0) {
                                                                map= new ConcurrentHashMap<>();
                                                                int size= monitor.getMskFilter(monitor.diffr, monitor.mskr, i, j, 0, map);
                                                                if(size> monitor.facy){
                                                                        Iterator< String> it= map.keySet().iterator();
                                                                        while(it.hasNext()){
                                                                                String temp= it.next();
                                                                                if(size> monitor.facy){
                                                                                        int x= Integer.valueOf(temp.split(",")[0]);
                                                                                        int y= Integer.valueOf(temp.split(",")[1]);
                                                                                        monitor.gpcar[x][y]= 255;
                                                                                }
                                                                        }
                                                                }
                                                        }
                                                }
                                                if(monitor.isGreenButton) {
                                                        if(monitor.mskg[j][i]== 0) {
                                                                map= new ConcurrentHashMap<>();
                                                                int size= monitor.getMskFilter(monitor.diffg, monitor.mskg, i, j, 0,
map);

                                                                if(size> monitor.facy) {
                                                                        Iterator< String> it= map.keySet().iterator();
                                                                        while(it.hasNext()){
                                                                                String temp= it.next();
                                                                                if(size> monitor.facy){
                                                                                        int x= Integer.valueOf(temp.split(",")[0]);
                                                                                        int y= Integer.valueOf(temp.split(",")[1]);
                                                                                        monitor.gpcag[x][y]= 255;
                                                                                }
                                                                        }
                                                                }
                                                        }
                                                }
                                                if(monitor.isBlueButton) {
                                                        if(monitor.isGreenButton) {
```

```
                                                                        if(monitor.mskb[j][i]== 0) {
                                                                                map= new ConcurrentHashMap<>();
                                                                                int size= monitor.getMskFilter(monitor.diffb, monitor.mskb, i,
j, 0, map);
                                                                                if(size> monitor.facy) {
                                                                                        Iterator< String> it = map.keySet().iterator();
                                                                                        while(it.hasNext()){
                                                                                                String temp= it.next();
                                                                                                if(size> monitor.facy){
                                                                                                        int x=
Integer.valueOf(temp.split(",")[0]);
                                                                                                        int y=
Integer.valueOf(temp.split(",")[1]);
                                                                                                        monitor.gpcab[x][y]= 255;
                                                                                                }
                                                                                        }
                                                                                }
                                                                        }
                                                                }
                                                        }
                                                }
                                        }
                                }else {
                                        monitor.gpcar= monitor.diffr;
                                        monitor.gpcag= monitor.diffg;
                                        monitor.gpcab= monitor.diffb;
                                }
                        }catch(Exception e) {
                                //e.printStackTrace();
                                //System.out.println(e.getMessage());
                        }
                }
}
```

```
package AVQ.OEQ.cap;
import java.awt.*;

import java.awt.image.BufferedImage;
import java.io.IOException;
import PCI.ASQ.image.ImagePixGroupFilter;
import SVQ.stable.StableVision;

public class Monitor_XCDX_Animation_EyeScan{
        @SuppressWarnings("unused")
        public static void XCDX_paint_eyeScan(Monitor_XCDX monitor, Graphics g, BufferedImage image) throws IOException{
                if(true== monitor.isbt88Stop){
                        for (int i= 0; i< image.getHeight(); ++i) {
                                for (int j= 0; j< image.getWidth(); ++j) {
                                        if(monitor.isRedButton) {
                                                monitor.rp[j][i]= (image.getRGB(j, i)>> 16& 0xFF);
                                        }
                                        if(monitor.isGreenButton) {
                                                monitor.gp[j][i]= (image.getRGB(j, i)>> 8& 0xFF);
                                        }
                                        if(monitor.isBlueButton) {
                                                monitor.bp[j][i]= (image.getRGB(j, i)>> 0& 0xFF);
                                        }
                                }
                        }
                        monitor.rp= new PEU.P.image.Strech().P(monitor.rp, 0.05, 0.95);
                        monitor.rp= new PEU.P.image.Guassian().P_1D(monitor.rp, 3, 3, 1.66);
                        int[][] mag= new PEU.P.image.Sobel().P(monitor.rp, 1);
                        int[][] dir= new PEU.P.image.Sobel().P(monitor.rp, 2);
                        mag= new PEU.P.image.Threshold().P(mag, 7);
                        mag= new PEU.P.image.Mask().P(mag, dir);
                        mag= new PEU.P.image.Threshold().P_Section(mag, 25 ,110);

                        //将梯度的索贝尔分层后进行距离为2的 128像色素团小于1 大于50的像色素团噪声过滤输出。
                        mag= ImagePixGroupFilter.getImagePix2DGroupFilter(mag, 128, 2, 1, 50);
                        //将梯度的索贝尔分层后进行距离为2的 255像色素团小于1 大于50的像色素团噪声过滤输出。
                        mag= ImagePixGroupFilter.getImagePix2DGroupFilter(mag, 255, 2, 1, 50);
```

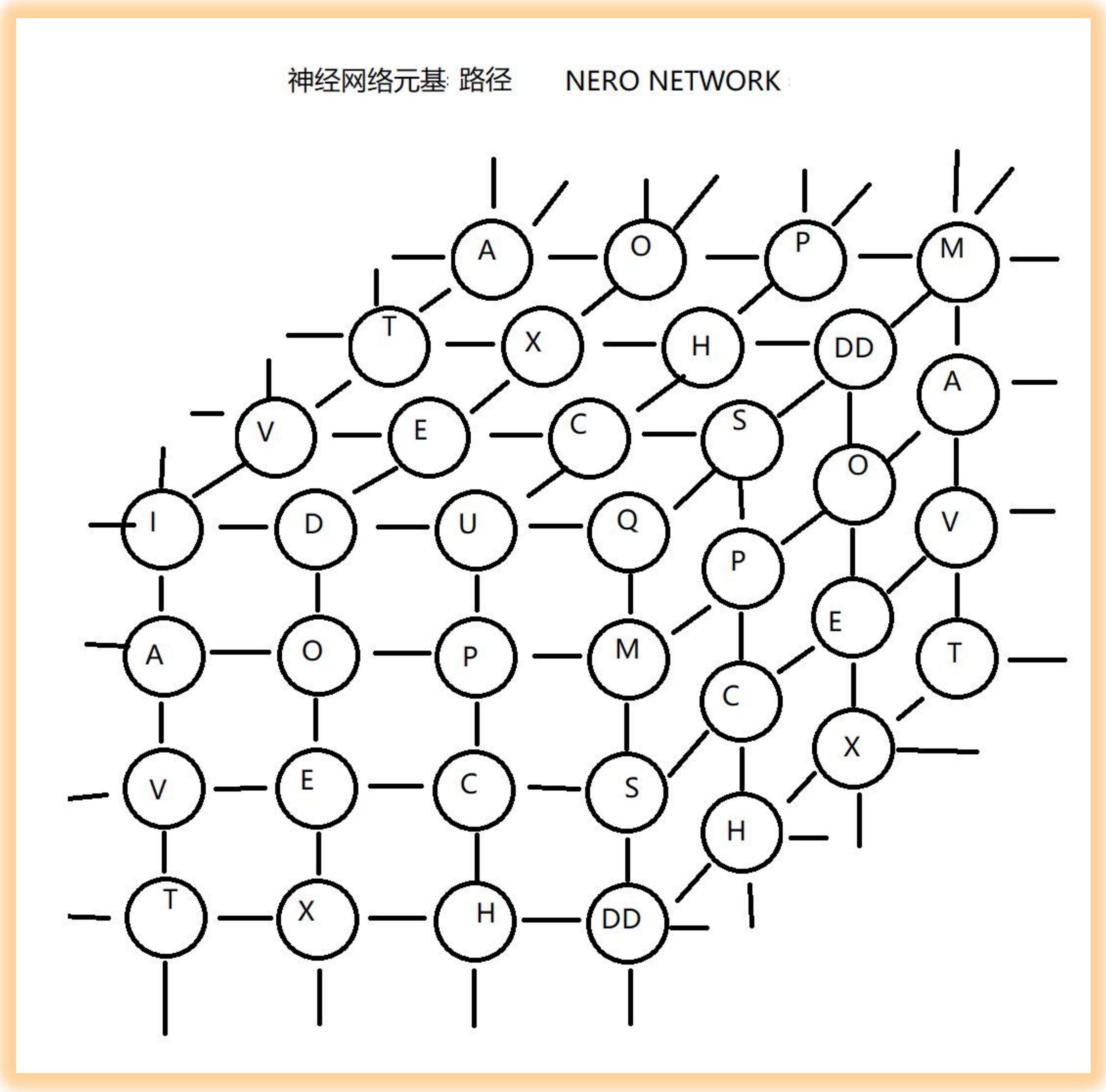
```
int[][] rp1= mag;
int w= rp1.length;
int h= rp1[0].length;
int hy= StableVision.eyeHeart.length;
int wy= StableVision.eyeHeart[0].length;
int[][] output= new int[w][h];
for(int i= 50; i< w-50; i++) {
        Here:
                for(int j= 50; j< h-150; j++) {
                        int find997=0;int find996=0;int find995=0;int find998=0;
                        if(i+wy<w-1&& j+hy< h-1) {
                                for(int p=0;p<wy;p++) {
                                        for(int q=0; q<hy; q++) {
                                                if(StableVision.eyeHeart[q][p]==1) {
                                                        if(rp1[i+p][j+q]==128) {
                                                                find997++;
                                                        }
                                                        if(rp1[i+p][j+q]!=0) {
                                                                find995++;
                                                        }
                                                }
                                                if(StableVision.eyeHeart[q][p]==0) {
                                                        if(rp1[i+p][j+q]==255) {
                                                                find996++;
                                                        }
                                                        if(rp1[i+p][j+q]!=0) {
                                                                find998++;
                                                        }
                                                }
                                        }
                                }
                                if(find995>=13-4&&find995<13+3
                                                &&find996>12-1&&find996<12+1
                                                &&find997>9-1 &&find997<9+1
                                                &&find998>13-1 &&find998<13+1) {
                                        int w1= 50;
                                        int h1= 50;
                                        int hy1= StableVision.eye.length;
                                        int wy1= StableVision.eye[0].length;
                                        int find1= 0; int find2=0; int find3=0; int find4=0;int find5=0;
                                        int find6= 0; int find7=0; int find8=0; int find9=0;int find10=0;
                                        int find11= 0; int find12=0; int find13=0; int find14=0;int find15=0;
                                        int find16= 0;;int find17= 0; int find18=0; int find19=0;;int find20=0;
                                        int find21= 0;int find22= 0;
                                        for(int p= -wy1/2; p<wy1/2; p++) {
                                                for(int q= -hy1/2; q<hy1/2; q++) {

                                                        if(StableVision.eye[q+ hy1/2][p+ wy1/2]==1) {
                                                                if(rp1[i+p][j+q]==128) {
                                                                        find1++;
                                                                }
                                                        }
                                                        if(StableVision.eye[q+ hy1/2][p+ wy1/2]==20) {
                                                                if(rp1[i+p][j+q]==255) {
                                                                        find2++;
                                                                }
                                                        }
                                                        if(StableVision.eye[q+ hy1/2][p+ wy1/2]==10) {
                                                                if(rp1[i+p][j+q]==0) {
                                                                        find3++;
                                                                }
                                                                if(rp1[i+p][j+q]!=0) {
                                                                        find15++;
                                                                }
                                                        }
                                                        if(StableVision.eye[q+ hy1/2][p+ wy1/2]==15) {
                                                                if(rp1[i+p][j+q]==0) {
                                                                        find4++;
                                                                }
                                                        }
                                                        if(StableVision.eye[q+ hy1/2][p+ wy1/2]==14) {
                                                                if(rp1[i+p][j+q]==0) {
                                                                        find5++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==13) {
                                        if(rp1[i+p][j+q]==0) {
                                                find6++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==2) {
                                        if(rp1[i+p][j+q]==128) {
                                                find7++;
                                        }
                                        if(rp1[i+p][j+q]==255) {
                                                find21++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==3) {
                                        if(rp1[i+p][j+q]==255) {
                                                find8++;
                                        }
                                        if(rp1[i+p][j+q]==128) {
                                                find16++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==9) {
                                        if(rp1[i+p][j+q]==255) {
                                                find9++;
                                        }
                                        if(rp1[i+p][j+q]==128) {
                                                find20++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==19) {
                                        if(rp1[i+p][j+q]==255) {
                                                find10++;
                                        }
                                        if(rp1[i+p][j+q]==0) {
                                                find17++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==12) {
                                        if(rp1[i+p][j+q]==0) {
                                                find11++;
                                        }
                                        if(rp1[i+p][j+q]==255) {
                                                find18++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==5) {
                                        if(rp1[i+p][j+q]==0) {
                                                find12++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==7) {
                                        if(rp1[i+p][j+q]==0) {
                                                find13++;
                                        }
                                        if(rp1[i+p][j+q]==128) {
                                                find19++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==11) {
                                        if(rp1[i+p][j+q]==0) {
                                                find14++;
                                        }
                                }
                                if(StableVision.eye[q+ hy1/2][p+ wy1/2]==22) {
                                        if(rp1[i+p][j+q]==0) {
                                                find22++;
                                        }
                                }
                        }
                }
                int n=5;int nn=5;
                int m=5;int mm=25;
```

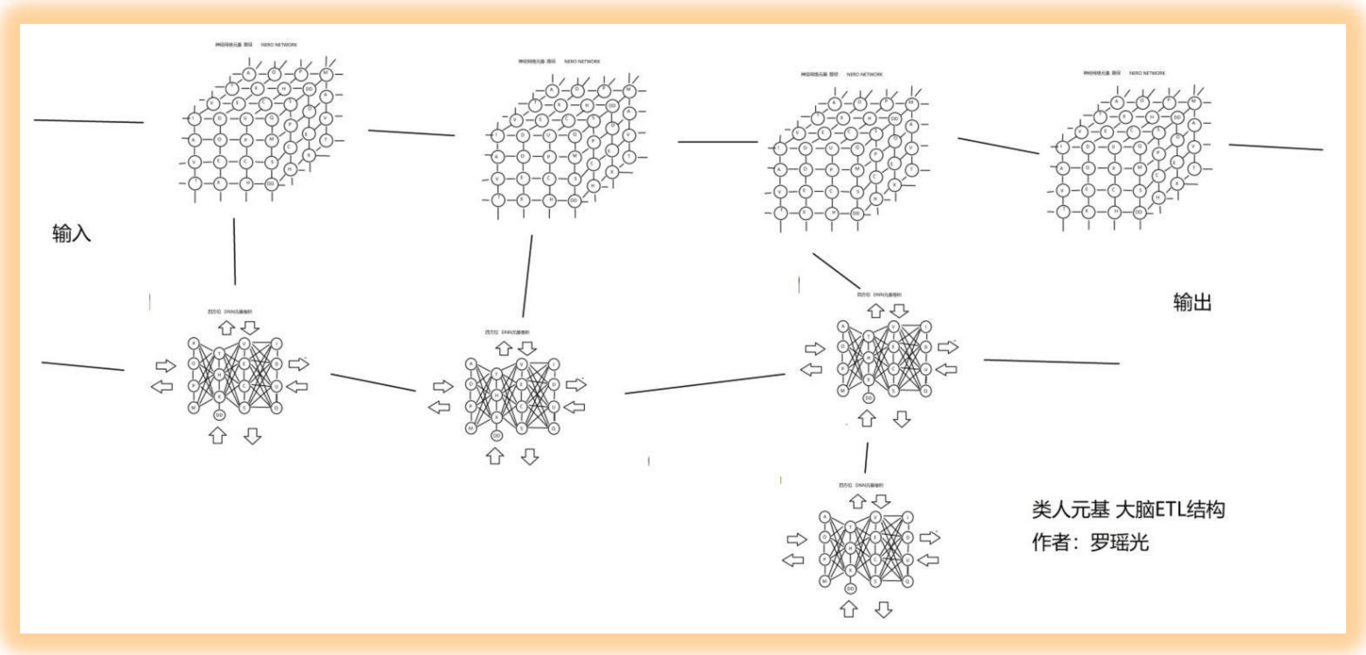
```
                                                            if(
                                                                          find1>=00
                                                                          &&find1<10+m
                                                                          &&find2<13+m
                                                                          &&find3>=12  -n   &&find3<50+m
                                                                          &&find4>=3      &&find4<20+m
                                                                          &&find5>=19 -n    &&find5<22+m
                                                                          &&find6>=8  -nn    &&find6<19
                                                                          &&find7>10  -nn    &&find7<35+m
                                                                          &&find8>20 -n     &&find8<75+m
                                                                          &&find9<10+m
                                                                          &&find10<1
                                                                          &&find11>=66  -n  &&find11<72+m
                                                                          &&find12>=10  -n   &&find12<30+m
                                                                          &&find13>=5      &&find13<15+m
                                                                          &&find14>=7   -nn   &&find14<9+m
                                                                          &&find15>=15  -nn   &&find15<35+m
                                                                          &&find16>=1     &&find16<20+m
                                                                          &&find17>=50 -nn  &&find17<=60+m
                                                                          &&find18<10+m
                                                                          &&find19<10+m
                                                                          &&find20<1
                                                                          &&find21<40+m
                                                                          &&find22>=0
                                                                          &&find22<20 +m) {
                                                                   if(i>50&&i<550&&j>50&&j<400) {
                                                                          System.out.println(find6);
                                                                          for(int m1= -wy1/2-20; m1< wy1/2+20; m1++) {
                                                                                 for(int n1= -hy1/2-0; n1< hy1/2+10; n1++)
{
                                                                                        output[i+ m1][j+ n1]= 255;
                                                                                 }
                                                                          }
                                                                   }
                                                            }
                                                     }//59
                                              }//59
                                       }//59
                        }
                        rp1= output;
                        monitor.rp= new PEU.P.image.Mask().P(rp1, monitor.rp);
                        monitor.gp= new PEU.P.image.Mask().P(rp1, monitor.gp);
                        monitor.bp= new PEU.P.image.Mask().P(rp1, monitor.bp);
                        int[][]temp= new PEU.P.image.Mask().P(rp1, mag);
                        for (int i= 0; i< image.getHeight(); ++i) {
                               for (int j= 0; j< image.getWidth(); ++j) {
                                      int pixel= (monitor.rp[j][i]<< 16)| (monitor.gp[j][i]<< 8)| (monitor.bp[j][i]) ;
                                      image.setRGB(j, i, pixel);
                               }
                        }
                        g.drawImage(image, 0, 0, 900, 680, monitor);// 绘出图形文件
                        return;
                 }
                 for (int i= 0; i< image.getHeight(); ++i) {
                        for (int j= 0; j< image.getWidth(); ++j) {
                               if(monitor.isRedButton) {
                                      monitor.rp[j][i]= (image.getRGB(j, i)>> 16& 0xFF);
                                      if(monitor.isbt53Stop&& monitor.rp[j][i]< 100) {
                                             monitor.rp[j][i]= 0;
                                      }else if(monitor.rp[j][i]< monitor.facr) {
                                             monitor.rp[j][i]= 0;
                                      }
                               }
                               if(monitor.isGreenButton) {
                                      monitor.gp[j][i]= (image.getRGB(j, i)>> 8& 0xFF);
                                      if(monitor.isbt53Stop&& monitor.gp[j][i]< 150) {
                                             monitor.gp[j][i]= 0;
                                      }else if(monitor.gp[j][i]< monitor.facg) {
                                             monitor.gp[j][i]= 0;
                                      }
                               }
                               if(monitor.isBlueButton){
                                      monitor.bp[j][i]= (image.getRGB(j, i) & 0xFF);
                                      if(monitor.isbt53Stop&& monitor.bp[j][i]< 100) {
```

```
                                                monitor.bp[j][i]= 0;
                                        }else if(monitor.bp[j][i]< monitor.facb) {
                                                monitor.bp[j][i]= 0;
                                        }
                                }
                        }
                }
        }
}
```

第四节 DNA 卷积的应用实现

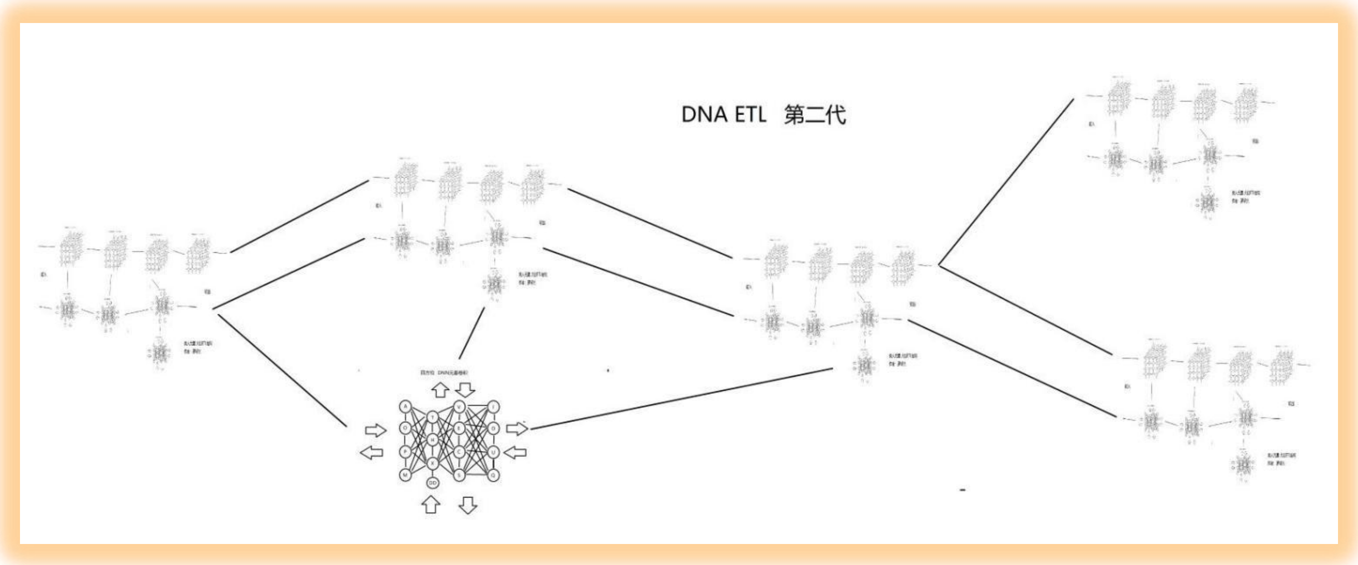


如图, 因为 DD 卷积的参与, 元基魔方就成型了. 元基魔方, 我对它的定义是: 一种神经网络路径计算存储模型, 用于语义思维扩散, 通过这两种模型, 我得到一种新的 ETL 复合模型如下



元基神经网络 DNN 卷 ETL 流 脑计算模型

这种模型不但能模拟人的意识, 思维和还能进行数字逻辑计算和存储. 并能有效的进行混合意识计算. 目前养疗经[17]的插件接口开始逐步的肽化, 这种TVM 的肽化过程是一种 2 维元基的应用, 到三维的脑结构还有一段路程要走, 好比房子地基逐渐打好, 有时间开始思考建造几层高楼结构了,



DNA ETL 第二代计算模型

混合意识计算一旦进行插件模块化, 这才是我想要的结果, 正如 DNA ETL 第二代计算模型, 这是罗瑶光先生的研究方向. 这个前提是我需要计算逻辑单元全部肽化.