# Politecnico di Bari



## Riconoscimento di vasche irrigue da immagini CASI

### Progetto di Image Processing

**DE MEO Claudio**

**SIENA Nicola**

## Riferimenti

DE MEO Claudio - demeoclaudio92@gmail.com
SIENA Nicola - n.siena91@gmail.com

Professore - GUERRIERO Andrea - andrea.guerriero@poliba.it
Ricercatore IRSA CNR - MATARRESE Raffaella - raffaella.matarrese@ba.irsa.cnr.it

**INDICE**

## Introduzione

Nel presente elaborato vengono proposte le metodologie e le tecniche che sono state adottate per la realizzazione di un sistema automatico in grado di riconoscere le vasche irrigue presenti nel territorio del Comune di Acquaviva delle Fonti.
Sono state utilizzate immagini aeree acquisite utilizzando il sensore CASI - Compact Airborne Spectrographic Imager. Le vasche irrigue sono utilizzate per la raccolta di acqua piovana e/o acqua estratta da pozzi artesiani e in seguito utilizzata per irrigare i campi.
La rilevazione e l'estrazione di vasche irrigue mediante tecniche di telerilevamento non è banale in quanto l'acqua assume diversi valori di riflettanza in funzione della vegetazione, della torbidità, dell'effetto di penetrazione della luce, etc.

## 1. Software Utilizzati

### 1.1 ENVI

ENVI, ENvironment for Visualizing Images, è un software utilizzato per processare ed analizzare immagini geospaziali, è sviluppato dalla Harris Geospatial Solutions. In questo lavoro ENVI è stato utilizzato per selezionare le 4 bande d'interesse dall'immagine CASI_2015_07_10_163152_g.pix.
Le bande utilizzate in questo lavoro sono:

- BLUE: 489.2 nm
- GREEN: 560.5 nm
- RED: 660.2 nm
- NIR: 845.2 nm.

### 1.2 QGIS

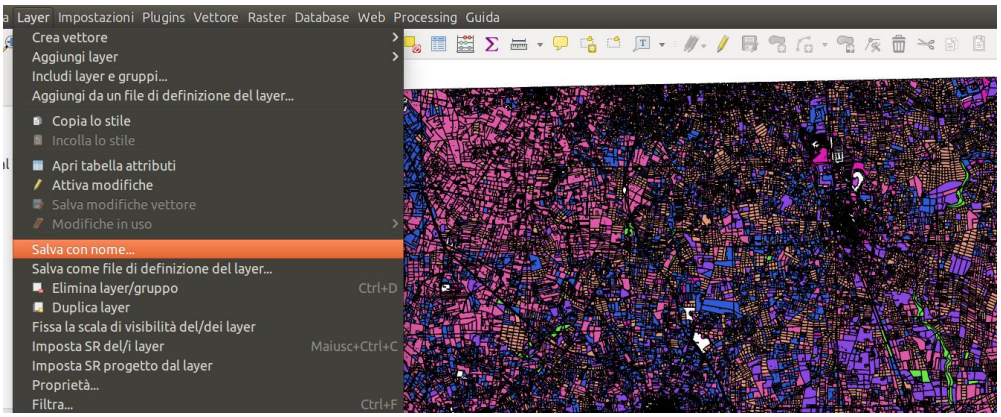
QGIS è un Sistema di Informazione Geografica Open Source, rilasciato sotto la GNU General Public License. QGIS è un progetto ufficiale della Open Source Geospatial Foundation (OSGeo). Funziona su Linux, Unix, Mac OSX, Windows e Android e supporta numerosi formati vettoriali, raster, database e funzionalità.
QGIS è stato utilizzato, in primo luogo, per ottenere gli shape file relativi alle vasche presenti sul territorio partendo dalla CTR dell'area di interesse (scaricabile dal sito http://www.sit.puglia.it) in modo da avere un confronto con i risultati ottenuti e, in secondo luogo, per creare delle maschere da sovrapporre all'immagine CASI. Le maschere sono state create per scremare i risultati ottenuti eliminando zone di non interesse; in pratica, partendo dalla CTR basta selezionare i vari poligoni che non si vogliono considerare per l'applicazione degli indici di vegetazione.

Gli shapefile scaricabili dal sito citato hanno una caratteristica che Matlab non riesce ad utilizzare, di seguito è mostrato un esempio di lettura e il relativo errore, ovvero *"Unsupported shape type PolygonZ (type code = 15)."*

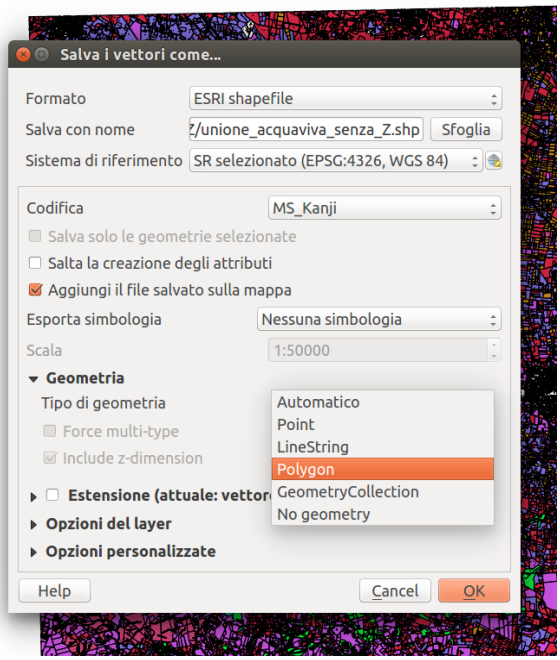
```
Command Window
>> shape=shaperead('unione_acquaviva.shp')
Error using openShapeFiles>readHeaderTypeCode (line 145)
Unsupported shape type PolygonZ (type code = 15).

Error in openShapeFiles (line 23)
headerTypeCode = readHeaderTypeCode(shpFileId,callingFcn);

Error in shaperead (line 208)
    = openShapeFiles(filename,'shaperead');
```

Per risolvere il problema abbiamo eseguito la seguente procedura.

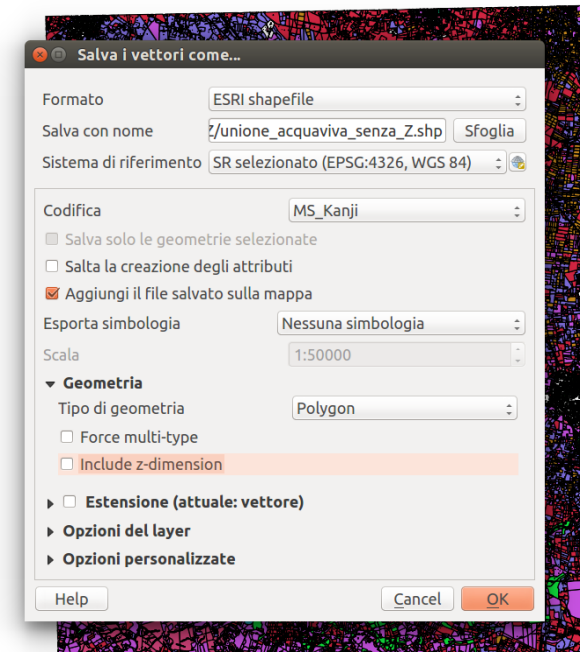
1. aprire lo shapefile con QGIS
2. selezionare *"Layer"* → *"Salva con nome…"*




3. nella finestra *"Salva i vettori come…"*, selezionare *"Polygon"* come tipo di geometria



4. deselezionare la voce *“Include z-dimension”*




5. cliccare su *“ok”*.

## 2. Indici di Vegetazione

Le bande di un’immagine possono essere combinate tra loro facendo operazioni pixel a pixel generando una nuova banda chiamata *“pseudo banda”* ottenuta come in funzione delle altre: $x_i^p = f(x_i^1, x_i^2, \dots, x_i^n)$(in cui si indica come apice la banda di interesse e come pedice il generico pixel). Nel calcolo dell’indice di vegetazione si fa uso di una pseudo banda dove il pixel i-esimo è calcolato come $x_i^p = \frac{x_i^k - x_i^m}{x_i^k + x_i^m}$; dove k e m rappresentano le bande coinvolte nell’operazione. Nel nostro studio abbiamo sperimentato in particolare tre indici di vegetazione diversi: NDVI, NDWI2 e la sua variante NDWI2 modificato.

L’**NDVI** (Normalized Difference Vegetation Index) permette di sintetizzare in un’unica immagine il contenuto informativo delle bande del rosso e del vicino infrarosso, relativamente allo stato vegetativo delle superfici. I valori numerici in uscita variano in un intervallo compreso tra -1 e +1: valori bassi (colori scuri) indicano zone poco vegetate mentre valori prossimi ad uno (colori chiari) indicano zone altamente vegetate.

In formule:

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

Visto che l’acqua assume un valore di riflettanza maggiore nella banda del rosso, in presenza di corpi idrici superficiali l’NDVI assumerà un valore negativo.

L’ **NDWI2** (Second Normalized Difference Water Index), sviluppato da Stuart K. McFeeters (1996), permette invece di sintetizzare in un’unica immagine il contenuto informativo delle bande del verde e del vicino infrarosso, relativamente alla presenza

di acqua superficiale, anche in questo caso i valori numerici in uscita variano in un intervallo compreso tra -1 e +1: valori positivi individuano la presenza di corpi idrici superficiali.
In formule:

$$NDWI2 = \frac{GREEN - NIR}{GREEN + NIR}$$

Bo-Cai Gao nel 1996 propose una modifica al **NDWI2 (NDWI2 modificato)** secondo cui valori negativi dell'indice testimoniano la presenza di corpi idrici superficiali.
In formule:

$$NDWI2mod = \frac{NIR - GREEN}{NIR + GREEN}$$

## 3. Procedura Applicativa

La soluzione proposta è stata realizzata mediante una class in Matlab, denominata *"WaterDetector"* [Appendice A1] che permette di istanziare un oggetto contenente proprietà e metodi utili all'individuazione di fonti d'acqua attraverso gli indici di vegetazione, scremare i risultati mediante l'applicazione di maschere e confrontare i risultati ottenuti con la CTR. In seguito è stata realizzata una comoda interfaccia grafica che mediante l'uso di tale classe fornisce uno strumento utile all'operatore che deve compiere questo tipo di ricerche sul territorio. Di seguito è mostrato il diagramma UML di tale classe:

| **class name:**<br>WaterDetector |
|---|
| **private attribute:**<br>image : [m x n x 4] uint16<br>info : Struct (GeoTiff Struct)<br>crop : Struct<br>crop.image : [i x j x 4] uint16 (i <= m & j <= n)<br>crop.xMin : Double<br>crop.xMax : Double<br>crop.yMin : Double<br>crop.yMax : Double<br>crop.rect : [xmin ymin width height]<br>mask : [m x n] Logical<br>water : [i x j] Double (i <= m & j <= n)<br>index : String<br>threshold : [min max]<br>lat : [i x j] Double (i <= m & j <= n)<br>lon : [i x j x 4] Double (i <= m & j <= n)<br>shape : Struct<br>shape.Geometry : String<br>shape.id : Double<br>shape.X : [ ] Double<br>shape.Y : [ ] Double |

shape.BoundingBox: [xmin ymin ; xmax ymax ]
shape.Description : String
shape.Area : Double
shape.isNew : Boolean
ctr : Struct
ctr.Geometry : String
ctr.id : Double
ctr.X : [ ] Double
ctr.Y : [ ] Double
ctr.BoundingBox: [xmin ymin ; xmax ymax ]
ctr.DESCR : String
ctr.LAYER : String
ctr.Area : Double
ctr.found : Boolean

**public method:**
WaterDetector()
getImage() : image
getCrop() : crop
getMask() : mask
getWater() : water
getShape() : shape
getInfo() : info
getCTR() : ctr
getWaterFromCTR() : ctr
readImage(filename)
cropImage(): crop
removeCrop()
calcNdvi(index,threshold) : imm [i x j] Double (i <= m & j <= n)
showMask() : image
shape2mask(shape)
removeShapeMask(shape)
applyMask()
removeMask()
maskFromCTR(ctr, descr)
removeMaskFromCTR(ctr, descr)
createShape(Amin, Amax) : [num, shape]
showElement(id,z) : [element, area]
showCTRElement(id,z) : [element, area]
removeElement(id)
applyMorphClose()
applyMorphClean()
importCTR(shape,descr) : [num, ctr]
cropCTR(descr) : ctr
addCTRElementToShape(id)
removeToCTR(id)
removeCTR()

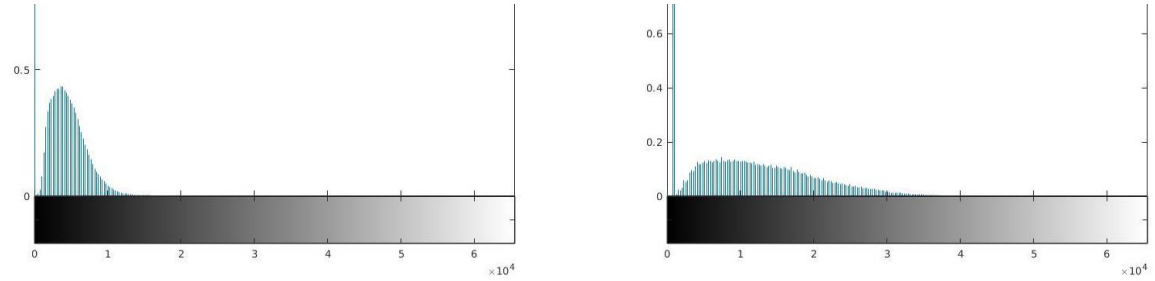
**private method:**
getLatLon(rect) : [lat, lon]
compareBoundingBox(BoundingBox1,BoundingBox2) : flag

### 3.1 Lettura dell'immagine

Questa classe permette di leggere una immagine attraverso il metodo *readImage(filename)* [Appendice A2] passando come argomento il path del file.

```
function readImage(obj,filename)
        if (nargin > 0)
                obj.image = geotiffread(filename);
Ordinamento     temp = obj.image(:,:,1);
bande RGB       obj.image(:,:,1) = obj.image(:,:,3);
                obj.image(:,:,3) = temp;

Adaptive        obj.image(:,:,1) = adapthisteq(obj.image(:,:,1));
Histogram       obj.image(:,:,2) = adapthisteq(obj.image(:,:,2));
Equalizatio     obj.image(:,:,3) = adapthisteq(obj.image(:,:,3));
                obj.image(:,:,4) = adapthisteq(obj.image(:,:,4));
                obj.info = geotiffinfo(filename);
                obj.lat = [];
                obj.lon = [];
        end
end
```

In questa fase abbiamo notato che il software ENVI non permette di scegliere l'ordine in cui vengono salvate le bande dell'immagine, quindi abbiamo preferito invertire la banda del rosso con quella del blu in modo da ottenere colori più naturali in fase di visualizzazione dell'immagine; inoltre per migliorare la qualità dell'immagine abbiamo deciso di equalizzare l'istogramma dell'immagine tramite la funzione Matlab **adapthisteq** - 'adaptive histogram equalization'.
Di seguito mostriamo le immagini ed i relativi istogrammi.

I miglioramenti ottenuti sono visibili ad occhio nudo, per conferma possiamo osservare gli istogrammi e notiamo che: nell'istogramma dell'immagine di partenza notiamo una campana molto stretta, con dei picchi elevati e concentrata su tonalità scure, mentre nell'istogramma dell'immagine equalizzata notiamo una campana molto ampia, senza picchi, che ricopre molte più tonalità rispetto all'immagine di partenza. L'immagine ottenuta viene salvata nell'attributo *image* della classe.
Una volta letta l'immagine è possibile caricare lo shapefile della CTR tramite il metodo *importCTR(shape,descr)*, tagliarla tramite il metodo *cropImage()*, creare una maschera attraverso i metodi *shape2mask(shape)* o *applyMask()*, oppure applicare e calcolare direttamente un indice di vegetazione con *calcNdvi(index,thrshold)*.
Si noti che non è necessario seguire un particolare ordine nell'utilizzo di questi metodi: l'uso delle maschere non è obbligatorio, inoltre l'indice di vegetazione può essere calcolato sull'immagine totale o sull'immagine tagliata.

### 3.2 Lettura della CTR - Carta Tecnica Regionale

Al fine di avere un confronto con i dati reali sono stati implementati dei metodi per importare una CTR in cui sono state riportate le vasche già presenti nell'area di interesse e confrontare i risultati ottenuti con quelli reali evidenziando il numero di elementi nuovi e quelli che, invece, non sono stati trovati.
Per importare una CTR è stato creato il metodo *importCTR(shape,descr)* [Appendice A19] che accetta in input lo shape file della CTR e la descrizione (che può essere un vettore contenente più descrizioni) relativa agli elementi di interesse.
Questo metodo permette di leggere una CTR e salvarsi solo gli elementi interni all'immagine (eliminando anche gli elementi presenti nelle zone nere dell'immagine CASI) utilizzando il metodo privato *compareBoundingBox(Box1,Box2)* [Appendice A25] che permette di confrontare due bounding box e restituisce true se la prima è contenuta nella seconda.

```
center = round(map2pix(obj.info.RefMatrix,shape(i).BoundingBox(:,1),...
        shape(i).BoundingBox(:,2)));
xMin = center(1,2);
xMax = center(2,2);
yMin = center(2,1);
yMax = center(1,1);
if (obj.compareBoundingBox(shape(i).BoundingBox,obj.info.BoundingBox) && …
        sum(obj.image(yMin,xMin,:)~=909 & obj.image(yMax,xMax,:)~=909)==4)
.
.
.
```

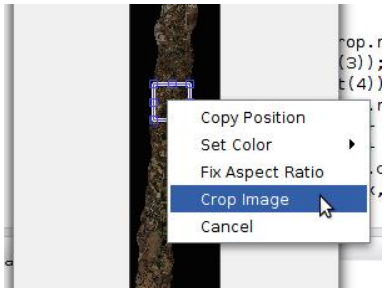
Inoltre se è stato calcolato in precedenza uno shapefile confronta le bounding box e valorizza i campi *found* nella CTR e *isNew* nello shapefile per indicare che che quella specifica vasca è stata trovata e che non è una nuova vasca.

```
ctr(z,1).found=obj.compareBoundingBox(obj.shape(1,y).BoundingBox,...
        ctr(z).BoundingBox);
.
.
.
obj.shape(1,y-1).isNew = ~ctr(z,1).found;
ctr(z,1).found=obj.compareBoundingBox(obj.shape(1,y).BoundingBox,...
        ctr(z).BoundingBox);
.
.
.
obj.shape(1,y-1).isNew = ~ctr(z,1).found;
```

Questa funzione restituisce il numero di elementi con una specifica descrizione e la CTR priva degli elementi esterni all'immagine.

### 3.3 Eseguire il crop dell'immagine

Il metodo *cropImage()* [Appendice A3] utilizza la funzione *imcrop* di Matlab che permette di tagliare un'area dell'immagine nella quale saranno cercate le vasche.




Il risultato del crop viene memorizzato nell'attributo *crop* che è una struttura composto da:

- *image*: immagine tagliata;
- *xMin* : indice di colonna minore;
- *xMax* : indice di colonna maggiore;
- *yMin* : indice di riga minore;
- yMax : indice di riga maggiore;
- *rect* : bounding box formato da un array del tipo [xmin ymin width height].

Inoltre è stato implementato un metodo *removeCrop()* [Appendice A4] per rimuovere un crop e tornare all'immagine originale.

### 3.4 Tagliare la CTR

Se l'immagine di partenza è stata tagliata è possibile ottenere dalla CTR caricata solo gli elementi interni all'area dell'immagine tagliata attraverso il metodo *cropCTR(descr)* [Appendice A20] che accetta come parametro opzionale una descrizione per filtrare ulterirmente i risultati. Questa funzione restituisce gli elementi della CTR caricati interni all'area dell'immagine tagliata sfruttando sempre il metodo privato *compareBoundingBox()*.

### 3.5 Calcolo dell'indice di vegetazione

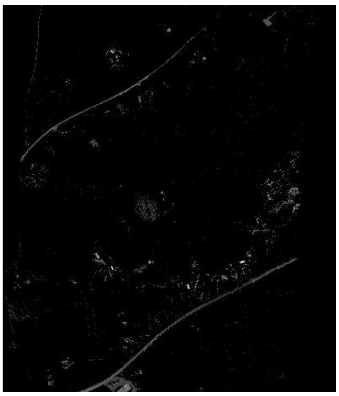
Il calcolo dell'indice di vegetazione viene fatto mediante il metodo *calcNdvi(index,threshold)* [Appendice A5] in cui vengono passati come argomenti il tipo di indice ("ndvi", "ndwi2", "ndwi2m") e una soglia, tale soglia può essere un array nel caso in cui si vuole applicare un limite inferiore e un maggiore sul risultato del calcolo, nel caso in cui viene passato un solo valore all'interno della variabile threshold viene applicato solo il limite inferiore.

Il calcolo dell'indice di vegetazione avviene nel seguente modo:

```
I = double(I);
index = lower(index);
tMin = -1;
tMax = 1;
switch index
        case 'ndvi'
                Imm= (I(:,:,4) - I(:,:,1)) ./ (I(:,:,4) + I(:,:,1));
                tMin = -0.7;
                tMax = -0.3;
        case 'ndwi2'
                Imm = (I(:,:,2) - I(:,:,4)) ./ (I(:,:,2)+I(:,:,4));
                tMin = 0.4;
        case 'ndwi2m'
                Imm = (I(:,:,4) - I(:,:,2)) ./ (I(:,:,4)+I(:,:,2));
                tMax = -0.4;
        otherwise
                return;
        end
```

Si noti che l'immagine viene convertita in double prima del calcolo.

Le variabili *tMin* e *tMax* sono delle soglie di default impostate nel caso in cui non viene passata nessuna soglia come argomento alla funzione, tali variabili verranno sovrascritte nel momento in cui l'utente passa come argomento una variabile threshold. Una volta fatta l'operazione tra le bande viene generata una nuova banda memorizzata nella variabile *Imm*. Di seguito un esempio utilizzando come indice di vegetazione "NDWI2".

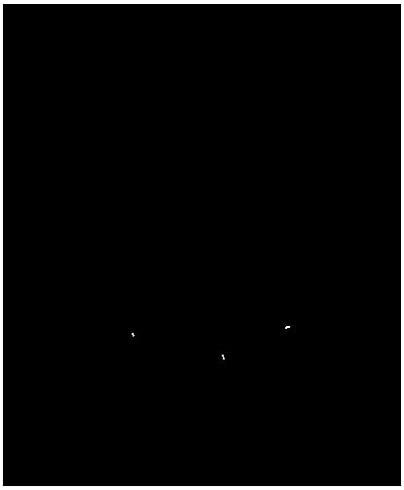


Una volta calcolata la nuova banda viene applicata la soglia e memorizzata nell'attributo *water* della classe.

```
obj.water = Imm > tMin & Imm < tMax;
```

L'attributo *water* contiene l'immagine binaria risultante dall'applicazione della soglia all'indice di vegetazione.

Di seguito è mostrata l'immagine risultato applicando sull'immagine ottenuta con l'indice "*NDWI2*" una soglia per valori di intensità maggiori di 0.7:



Si osservi che i risultati migliori sono stati ottenuti con questo indice (NDWI2) e con valori di intensità dei pixel maggiori di 0.7.

### 3.6 Applicazione di maschere

In alcuni casi si ha la presenza di falsi positivi, per questo motivo si è deciso di dare la possibilità di applicare delle maschere in modo da nascondere determinate zone.
L'applicazione di maschere può essere fatta utilizzando gli elementi caricati dalla CTR, attraverso uno shape file (contenente uno o più poligono creati con QGis ad esempio) oppure disegnando direttamente sull'immagine un poligono. L'applicazione della maschera attraverso gli elementi presenti nella CTR è stata realizzata attraverso il metodo *maskFromCTR(ctr,descr)* [Appendice A11], oppure con uno shape file con il metodo *shape2mask(shape)* [Appendice A7], mentre il metodo *applyMask()* [Appendice A9] permette di disegnare un poligono sull'immagine.

Questi metodi utilizzano l'attributo *mask* che è una matrice logica dalle dimensioni dell'immagine originale ([m x n]). Inizialmente questa matrice ha tutti i valori a false. L'applicazione di una maschera mediante il metodo *maskFromCTR, shape2mask* o *applyMask* imposta a true gli elementi della matrice *mask* corrispondenti a pixel nell'immagine originale contenuti nei poligoni indicati.

Per fare ciò si è utilizzata la funzione *inpolygon* per quanto riguarda *shape2mask* e *maskFromCTR*

```
function shape2mask(obj, shape)
    if (nargin > 1 && ~isempty(obj.image) && ~isempty(shape))
        if (isempty(obj.mask))
            obj.mask = false(size(obj.image(:,:,1)));
        end
        [x,y] = pixcenters(obj.info);
        [X,Y] = meshgrid(x,y);
        for i=1:size(shape)
            rx = shape(i).X(1:end-1);
            ry = shape(i).Y(1:end-1);
            obj.mask = obj.mask | inpolygon(X,Y,rx,ry);
        end
    end
end
function maskFromCTR(obj, ctr, descr)
    if (nargin > 2 && ~isempty(obj.image))
        if (isempty(obj.mask))
            obj.mask = false(size(obj.image(:,:,1)));
        end
        [x,y] = pixcenters(obj.info);
        [X,Y] = meshgrid(x,y);
        for i=1:size(ctr)
            center =
            round(map2pix(obj.info.RefMatrix,ctr(i).BoundingBox(:,1),ctr(i)
            .BoundingBox(:,2)));
            xMin = center(1,2);
            xMax = center(2,2);
            yMin = center(2,1);
            yMax = center(1,1);
                if(ismember(ctr(i).DESCR,descr) &&
                obj.compareBoundingBox(ctr(i).BoundingBox,obj.info
                .BoundingBox) && ...
                sum(obj.image(yMin,xMin,:)~=909 &
                obj.image(yMax,xMax,:)~=909)==4)
                rx = ctr(i).X(1:end-1);
                ry = ctr(i).Y(1:end-1);
                obj.mask = obj.mask | inpolygon(X,Y,rx,ry);
            end
        end
    end
end
```

e *roipoly* per *applyMask*.

```
function applyMask(obj)
    if (~isempty(obj.image))
        if (isempty(obj.mask))
            obj.mask = false(size(obj.image(:,:,1)));
        end
        m = roipoly(obj.showMask());
        close;
        if (~isempty(m))
            if (isempty(obj.crop))
                obj.mask = obj.mask | m;
            else
                obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:
                obj.crop.xMax) = ...
                obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:
                obj.crop.xMax) | m;
```

```
                end
            end
        end
    end
```

Come per il crop sono stati implementati dei metodi per rimuovere delle maschere precedentemente applicate.
Il metodo *removeMaskFromCTR(ctr,descr)* [Appendice A12] permette di rimuovere delle maschere applicate selezionando gli elementi nella CTR che non devono essere considerate come maschere..
Il metodo *removeShapeMask(shape)* [Appendice A8] permette di rimuovere delle maschere applicate mediante l'uso di un ulteriore shape file.
Il metodo *removeMask()* [Appendice A10] permette di disegnare sull'immagine originale un poligono all'interno del quale verranno eliminate eventuali maschere.
Infine è possibile in ogni momento visionare le maschere applicate con il metodo *showMask()* [Appendice A6] che restituisce un'immagine in cui sono state evidenziate le maschere applicate.

```
function image = showMask(obj)
    image = [];
    if (~isempty(obj.mask))
        if (isempty(obj.crop.image))
            r = obj.image(:,:,1);
            g = obj.image(:,:,2);
            b = obj.image(:,:,3);
            r(obj.mask) = 0;
            g(obj.mask) = 0;
            b(obj.mask) = 65535;
            image = cat(3,r,g,b);
        else
            r = obj.crop.image(:,:,1);
            g = obj.crop.image(:,:,2);
            b = obj.crop.image(:,:,3);
            r(obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:obj.crop.xMax))
            = 0;
            g(obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:obj.crop.xMax)
            ) = 0;
            b(obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:obj.crop.xMax)
            ) = 65535;
            image = cat(3,r,g,b);
        end
    end
end
```

Si osservi che se è stato effettuato un crop sull'immagine verrà visualizzata solo l'area relativa all'immagine tagliata.

## 3.7 Applicazione di filtri morfologici

Può essere utile in alcuni casi applicare dei filtri morfologici per eliminare alcuni punti isolati. A questo scopo sono stati applicati due metodi:
Il metodo *applyMorphClose()* [Appendice A17] che effettua l'operazione di chiusura sull'immagine binaria ottenuta dal calcolo dell'indice di vegetazione e *applyMorphClean()* [Appendice A18] che effettua l'operazione di pulizia.

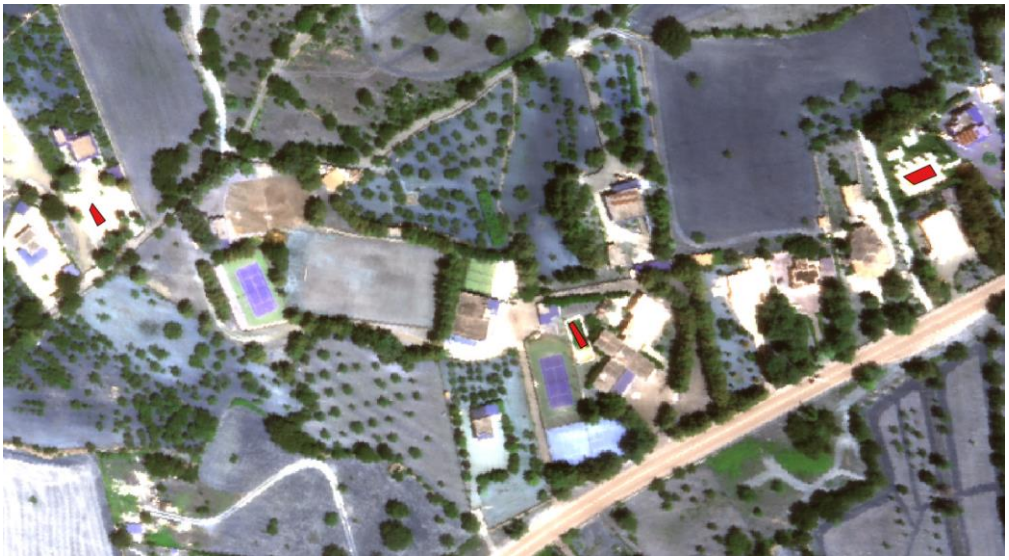
## 3.8 Creazione di uno shapefile

La creazione di uno shape file avviene mediante il metodo *createShape(Amin,Amax)* [Appendice A13]. Questo metodo utilizza la funzione *bwconncomp* che permette di estrarre le componenti connesse (con 8-connected neighborhood) ovvero le singole vasche trovate, le quali vengono salvate all'interno di una struttura chiamata *cc*. Grazie a questa struttura è possibile fare riferimento ai pixel della i-esima vasca attraverso il campo *PixelIdxList{i}*.
Una volta ottenuti tali pixel è necessario conoscerne le coordinate. A tal proposito è stato realizzato un metodo privato chiamato *getLatLon(rect)* [Appendice A24] che permette di ottenere le coordinate dell'immagine totale (se non viene passato alcun parametro come argomento) oppure dell'immagine tagliata (se viene passato il rettangolo contenente le coordinate dell'immagine tagliata). Ottenute le matrici *lat* e *lon* che contengono rispettivamente le latitudini e le longitudini di tutta l'immagine è possibile ottenere le coordinate del poligono contenente le vasche trovate.

```
temp(cc.PixelIdxList{i}) = true;
coord = [obj.lat(cc.PixelIdxList{i}) obj.lon(cc.PixelIdxList{i})];
maxLat = max(coord(:,1));
p1 = obj.lon(obj.lat == maxLat);
minLat = min(coord(:,1));
p2 = obj.lon(obj.lat == minLat);
maxLon = max(coord(:,2));
p3 = obj.lat(obj.lon == maxLon);
```

```
minLon = min(coord(:,2));
p4 = obj.lat(obj.lon == minLon);
[x, y] = projfwd(obj.info, [maxLat, p3, minLat, p4, maxLat], [p1,... maxLon, p2, minLon, p1]);
area = polyarea(x,y);
```

Si osservi che il poligono viene costruito utilizzando quattro coordinate in modo da avere figure più o meno rettangolari.
Di seguito i poligoni, in rosso, visualizzati su QGIS.



Viene utilizzato l'attributo *shape* che contiene le informazioni che andranno salvate come shapefile per poter essere visualizzate su QGis.

```
obj.shape(j).Geometry = deal('Polygon');
obj.shape(j).id = j;
obj.shape(j).X = x;
obj.shape(j).Y = y;
obj.shape(j).BoundingBox = [min(x) min(y); max(x) max(y)];
obj.shape(j).Description = 'vasche trovate';
obj.shape(j).Area = area;
```

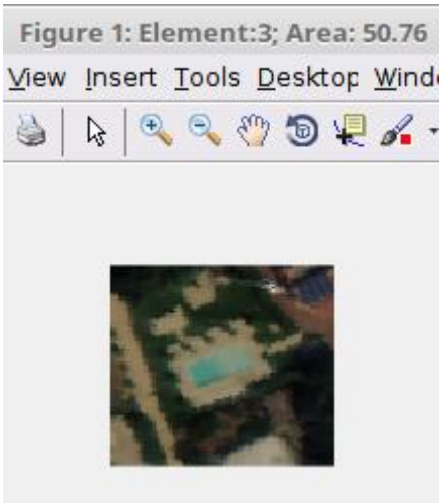
Una volta ottenuto un elemento, se è presente una CTR caricata viene confrontato tale elemento con quelli presenti nella CTR attraverso il metodo privato *compareBoundingBox* per verificare se una vasca trovata è una di quelle presenti nella CTR, in caso affermativo vengono sostituiti i campi BoundingBox, X,Y e Area con quelli presenti nella CTR essendo più precisi.

```
while (k <= size(obj.ctr,1) && ~found)
        found = obj.compareBoundingBox(obj.shape(j).BoundingBox,obj.ctr(k).BoundingBox);
        k= k + 1;
end
if (found)
        obj.shape(j).BoundingBox = obj.ctr(k-1).BoundingBox;
        obj.shape(j).Area = obj.ctr(k-1).Area;
        obj.shape(j).X = obj.ctr(k-1).X;
        obj.shape(j).Y = obj.ctr(k-1).Y;
        obj.shape(j).isNew = false;
        obj.ctr(k-1).found = true;
end
```

La struttura *shape* viene restituita insieme al numero di vasche trovate dal metodo.
Si osservi che questo metodo ha due parametri opzionali: Amin e Amax. Questi parametri permettono di specificare l'area minima e massima che un elemento deve avere per essere considerato una vasca.

### 3.9 Visualizzare un elemento specifico

Una volta creata la struttura *shape* è possibile visualizzare un elemento della struttura per verificare che sia effettivamente una vasca tramite il metodo *showElement(id,z)* [Appendice A14].
Tale metodo accetta come parametro obbligatorio l'id dell'elemento all'interno della struttura e come parametro opzionale z che rappresenta il numero di pixel in più (in altezza e in larghezza) che vengono estratti per non avere un'immagine troppo sgranata. In egual modo è possibile visualizzare un elemento presente nella CTR attraverso il metodo *showCTRElement(id,z)*[Appendice A15].
Di seguito l'immagine di una vasca trovata.




### 3.10 Usare una vasca trovata come maschera

Se ci si rende conto che un elemento trovato non è effettivamente una vasca, è possibile eliminarlo dalla struttura e applicarlo come maschera attraverso il metodo *removeElement(id)* [Appendice A16].
Questo metodo come *showElement* usa l'id per fare riferimento ad un elemento specifico all'interno della struttura *shape* e usa il campo *BoundingBox* di tale struttura per ottenere le coordinate dei pixel da usare come maschera attraverso la matrice *mask.*

```
center = round(map2pix(obj.info.RefMatrix,...
        shapeElement.BoundingBox(:,1),shapeElement.BoundingBox(:,2)));
xMin = center(1,2);
xMax = center(2,2);
yMin = center(2,1);
yMax = center(1,1);
obj.mask(yMin : yMax,xMin : xMax) = true;
```

Si osservi che una volta applicata questa maschera l'unico modo per rimuoverla è rimuovere la maschera con i metodi *removeMask* o *removeShapeMask* sopra citati.

### 3.11 Aggiungere un elemento della CTR allo shapefile

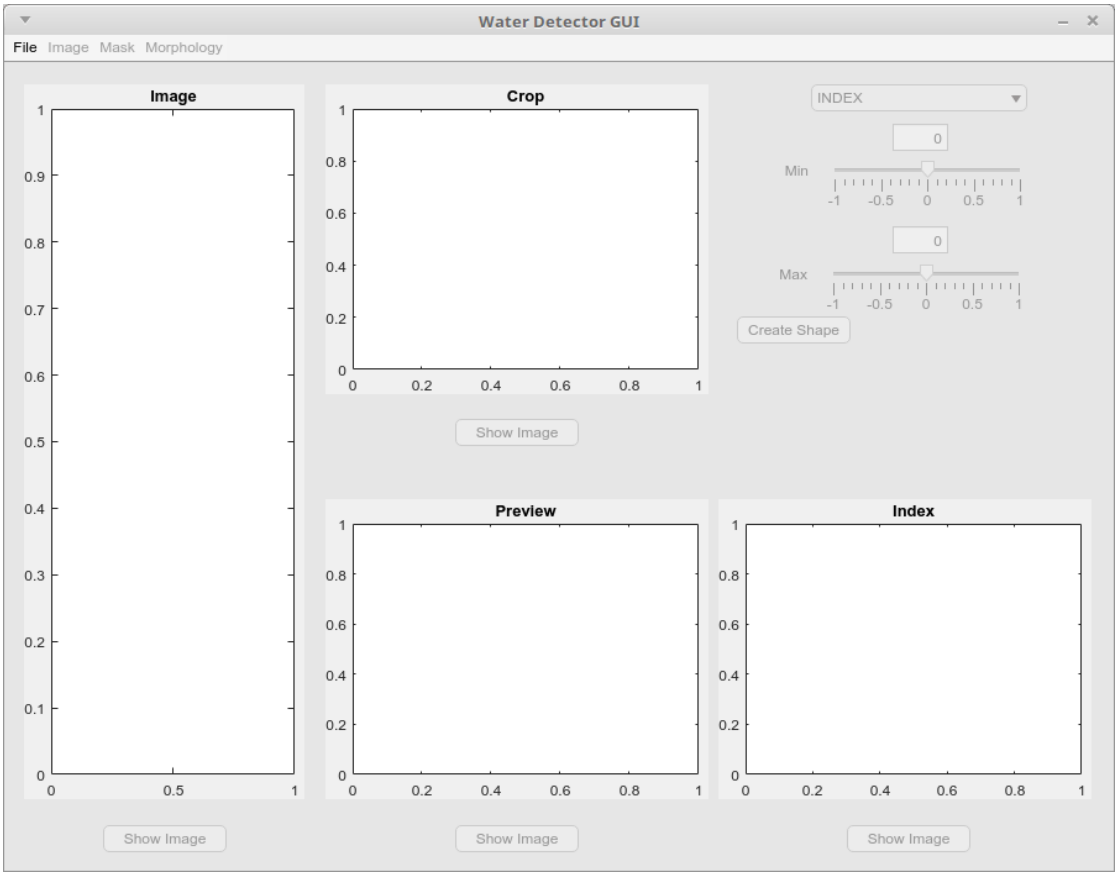
Dopo aver creato lo shapefile è possibile confrontarlo con gli elementi della CTR. Nel caso in cui siano presenti falsi-negativi attraverso il metodo *addCTRElementToShape(id)* [Appendice A21] è possibile aggiunge un elemento della CTR in coda a quelli trovati durante la creazione dello shape file utilizzando il suo id nella struttura della CTR.

```
function addCTRElementToShape(obj,id)
	if (nargin>1)
		element = obj.ctr([obj.ctr.id] == id);
		if (~isempty(element) && ~element.found)
			i = size(obj.shape,2) + 1;
			obj.shape(1,i).Geometry = element.Geometry;
			obj.shape(1,i).X = element.X;
			obj.shape(1,i).Y = element.Y;
			obj.shape(1,i).BoundingBox = element.BoundingBox;
			obj.shape(1,i).Description = obj.shape(1).Description;
			obj.shape(1,i).Area = element.Area;
			obj.shape(1,i).isNew = false;
			obj.shape(1,i).id = i;
			obj.ctr([obj.ctr.id] == id).found = true;
		end
	end
end
```

Inoltre se ci si rende conto che un elemento presente nella CTR classificato come vasca nell'immagine non è più una vasca perchè la CTR è datata è possibile eliminarlo dalla struttura della CTR attraverso il metodo *removeToCTR(id)* [Appendice A22].

## 3.12 Interfaccia grafica

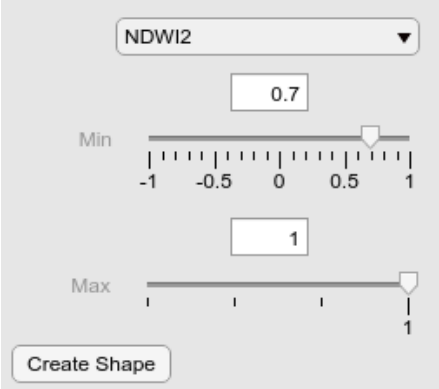
Infine per facilitare l'utilizzo della classe da parte dell'utente è stata realizzata una comoda interfaccia grafica utilizzando *App Designer* di Matlab.
La GUI si presenta all'utente nel seguente modo:




Oltre a permette di caricare un'immagine e salvare uno shape file, questa GUI interagisce con la classe per compiere tutte le azioni sopra citate.
Visualizza quattro immagini:

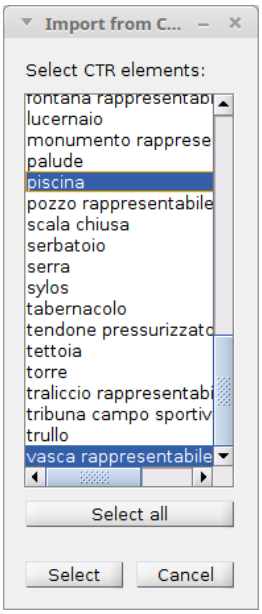
- l'immagine originale (Image);
- l'immagine tagliata (Crop);
- l'immagine calcolata dall'indice di vegetazione (Index);
- l'immagine binaria calcolata applicando una soglia e/o eventuali maschere all'immagine ottenuta dall'indice di vegetazione (Preview).

Tutte queste immagini possono essere visualizzate singolarmente in un'altra finestra premendo sui rispettivi pulsanti "Show Image".
Nella parte in alto a destra è possibile scegliere l'indice di vegetazione da applicare e le relative soglie, visualizzando dinamicamente gli aggiornamenti sull'immagine in modo da andare in aiuto all'operatore nella scelta di questi parametri.

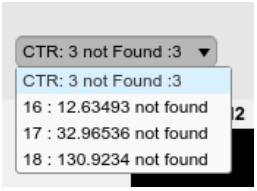



Inoltre, permette di applicare le maschere con i metodi sopra citati, applicare filtri morfologici e creare uno shapefile (i dettagli in seguito).

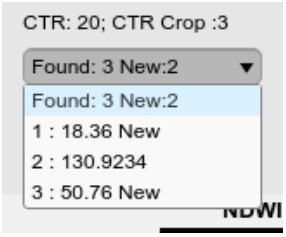
Permette di caricare una ctr in formato shapefile specificando le descrizioni da utilizzare per selezionare i poligoni presenti in essa




e li mostra in un elenco in cui è specificato se l'i-esimo elemento è stato trovato o meno.

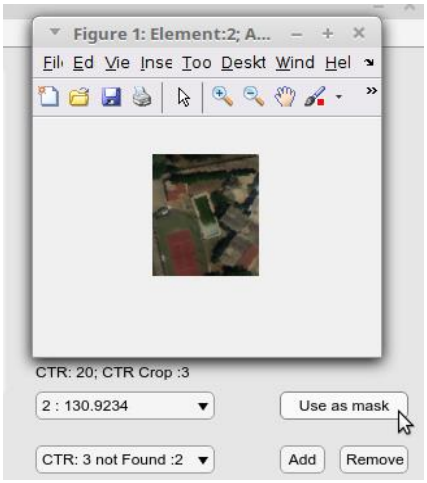



Dopo la creazione dello shape file è possibile vedere l'elenco delle vasche trovate con le relative aree in $m^2$ e un attributo che mostra se l'i-esima vasca trovata è una nuova vasca oppure una già presente nella CTR.

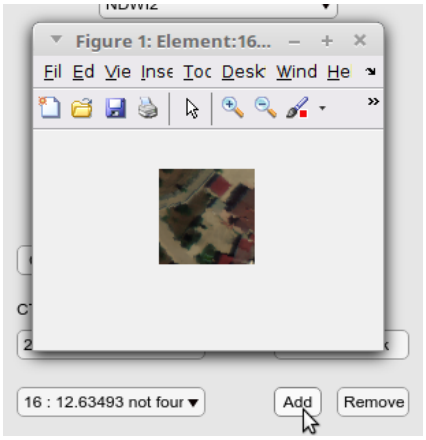



Si osservi che il caricamento della CTR può essere fatto in qualunque momento.

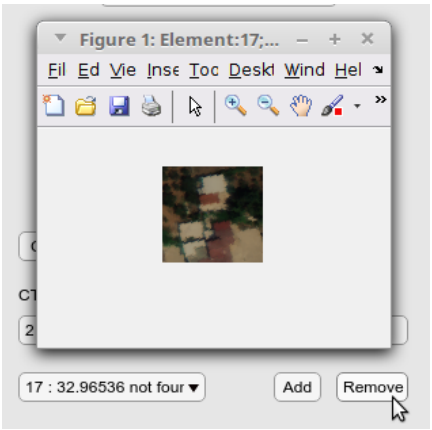
Attraverso gli elenchi è possibile visualizzare e ispezionare la singola vasca per permettere all'utente di rimuovere eventuali errori di classificazione applicando un elemento trovato come maschera




oppure aggiungere una vasca presente nella CTR tra quelle trovate




o eliminare dalla CTR gli elementi che non sono più vasche.



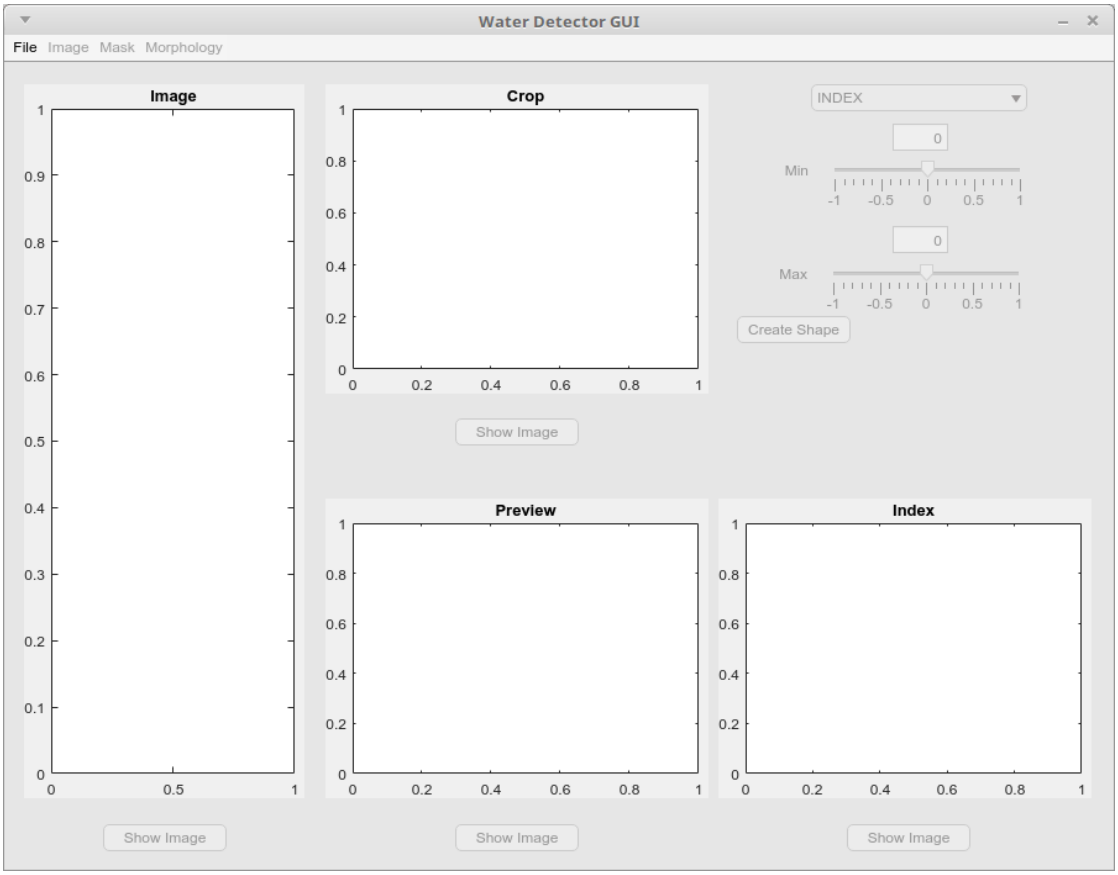
# 4. Guida all'Utilizzo

Il lavoro realizzato non necessita di particolari requisiti di sistema, se non *Matlab* e la *Image Processing Toolbox.*

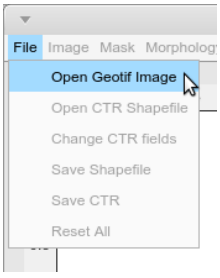
In seguito è mostrata una breve guida sull'utilizzo dell'interfaccia grafica da noi realizzata il cui codice è riportato in Appendice B1.

## 4.1 Utilizzo dell'interfaccia grafica

La schermata principale è la seguente



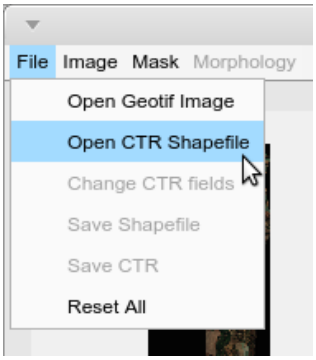

La prima operazione necessaria è aprire un'immagine Geotif, tramite il menù **File → Open Geotif Image**



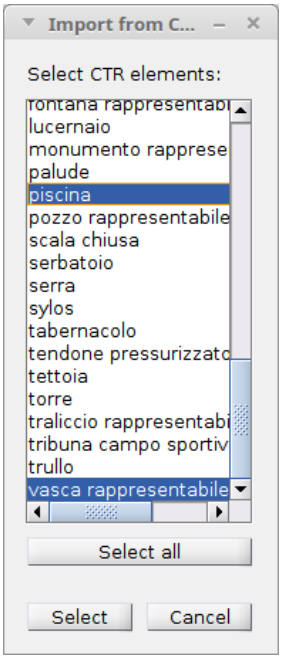
L’immagine caricata sarà visualizzata nell’area Image


A questo punto le operazione possibili sono:

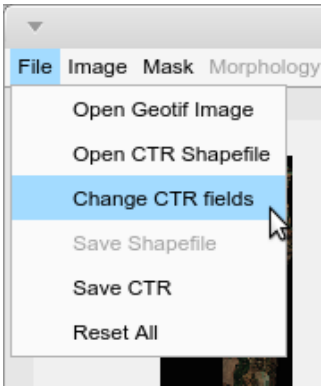
- Caricare una CTR tramite il menu **File → Open CTR Shapefile**



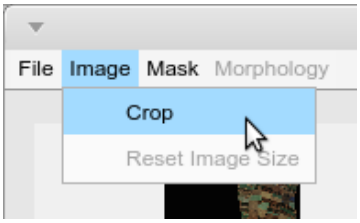

che aprirà una finestra di dialogo per permettere all’utente di selezionare le descrizioni da usare come parametro alla funzione *importCTR* per selezionare gli elementi che si vuole ricercare. Ultimato i caricamento verrà mostrato un elenco contenente le descrizioni gli elementi della CTR interni all’immagine.



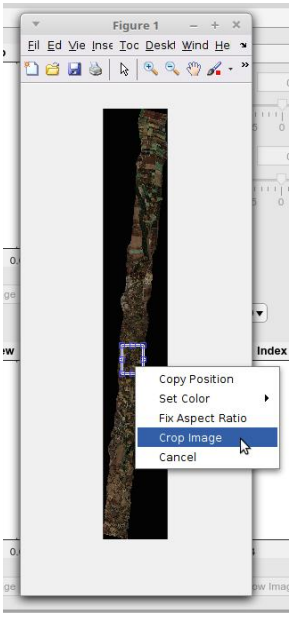
1. Caricata una CTR è possibile, in qualunque momento, cambiare gli elementi selezionati tramite il menu **File → Change CTR Field**



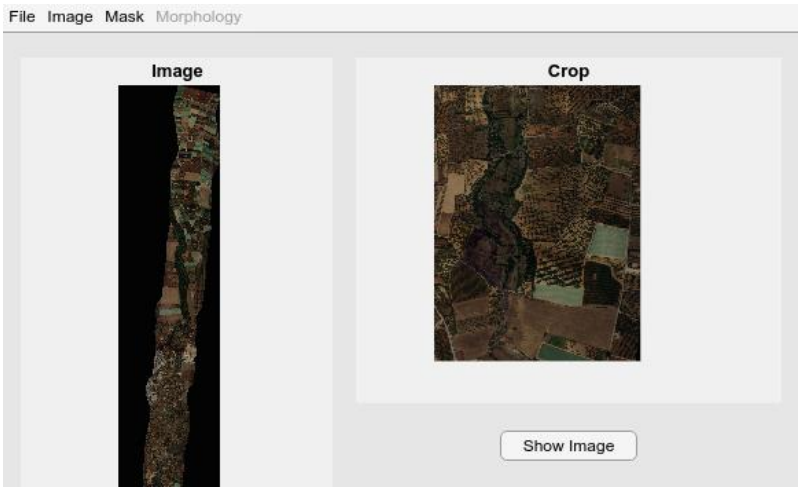

- Tagliare l'immagine tramite il menu **Image → Crop**



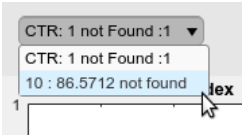

che richiamerà l'apposita funzione



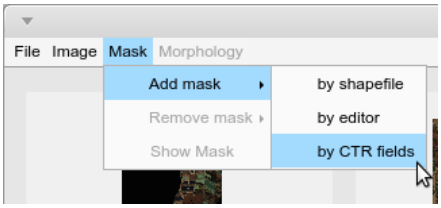

l'immagine risultante sarà visualizzata nella sezione Crop



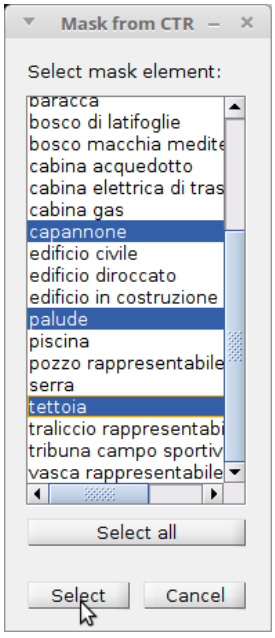

Inoltre verranno contati nuovamente gli elementi della CTR in modo da mostrare nel relativo elenco solo quelli interni all'area tagliata



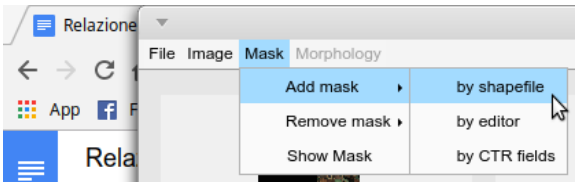

- Applicare una maschera tramite il menu **Mask** → **Add mask**. Come precedentemente esposto l'applicazione della maschera può avvenire in tre diversi modi:
  1. **by CTR** (se è stata precedentemente caricata)




si aprirà una finestra che permetterà all'utente di selezionare le descrizioni degli elementi che verranno utilizzati come maschere.

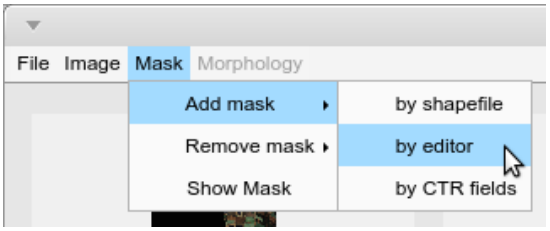



2. **by *shapefile***



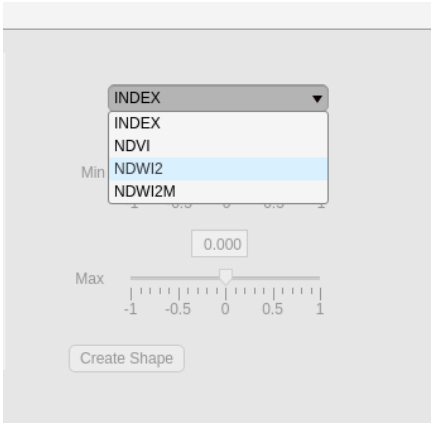

in questo caso dovrà essere selezionato uno *shapefile* creato in precedenza, che verrà usato come maschera.

3. **by editor**

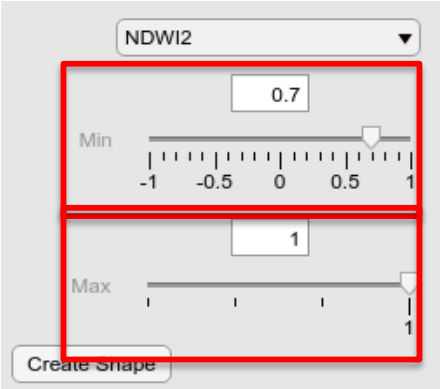



in questo caso verrà visualizzata una finestra che permetterà all'utente di disegnare un poligono che verrà usato come maschera

- calcolare un indice di vegetazione, selezionandolo dal menu a tendina

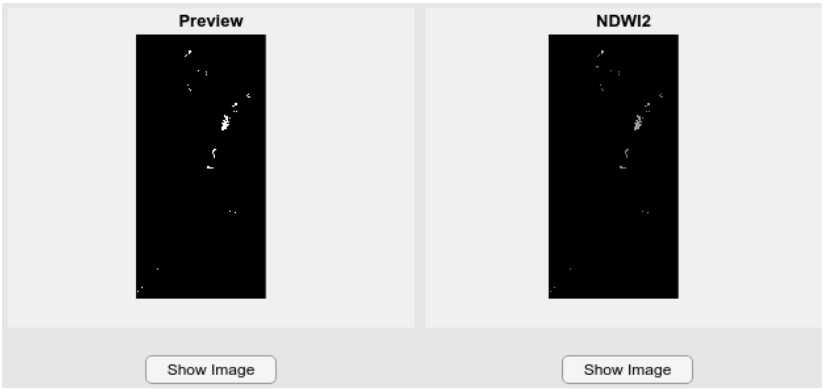



il software calcolerà la nuova immagine applicando le soglie indicate negli appositi campi



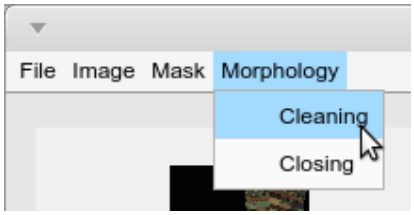

verranno visualizzate due immagini:

- l'immagine a destra rappresenta l'indice di vegetazione
- l'immagine a sinistra rappresenta l'immagine binaria ottenuta applicando le soglie.

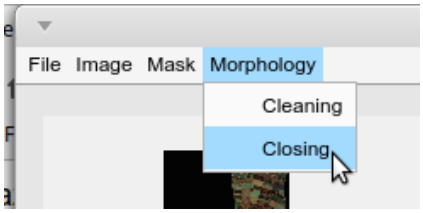


Dopo aver calcolato l'indice di vegetazione è possibile applicare i filtri morfologici di pulizia e chiusura, tramite gli appositi menù

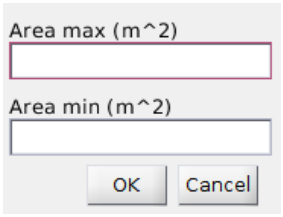
**Morphology → Cleaning**



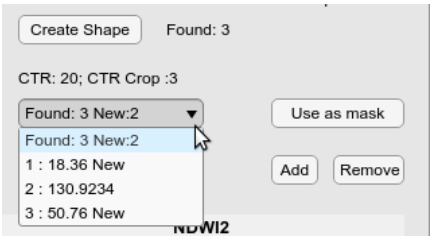

**Morphology → Closing**



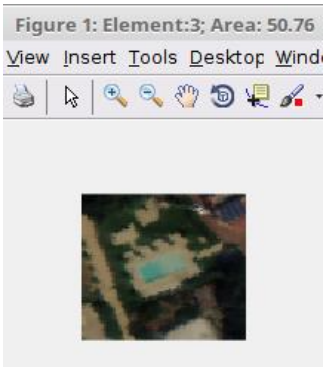

oppure è possibile creare lo *shapefile* cliccando sul pulsante ***"Create Shape",*** verrà visualizzata una finestra di dialogo in cui è possibile indicare l'area minima e/o l'area massima che un elemento deve avere per essere considerato come una vasca, in alternativa è possibile lasciare vuoti i campi per non imporre vincoli.




Dopo l'elaborazione verrà visualizzato un menu a tendina contenente l'elenco delle vasche trovate, le relative aree in $m^2$ e se sono nuove o meno (relativamente alla CTR).

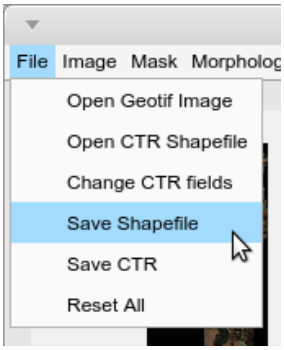



Selezionando un elemento tra quelli presenti in elenco verrà visualizzata l'immagine relativa, in modo da permettere all'utente di verificare l'esattezza del risultato,

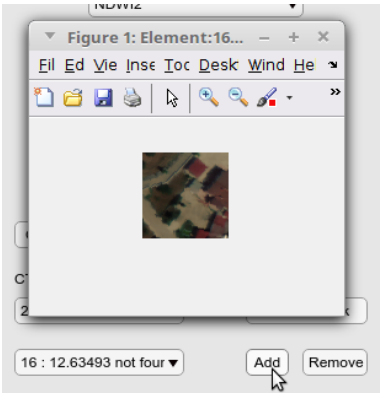



cliccando sul pulsante *"Use as mask"* l'elemento selezionato non sarà incluso nello *shapefile* finale.

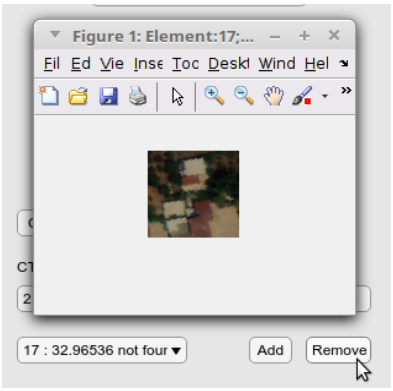
Successivamente è possibile salvare lo *shapefile* ottenuto tramite il menu: **File → Save Shapefile.**




Se è stata caricata una CTR è possibile aggiungere un elemento che non è stato trovato ma che è presente nella CTR selezionandolo e cliccando su ADD




Oppure se ci si rende conto che un elemento nella CTR non è più una vasca è possibile eliminarlo dalla CTR tramite il pulsante REMOVE




Inoltre, è possibile salvare uno *shapefile* contenente gli elementi della CTR contenuti nell'immagine utilizzata.

Oltre alle funzionalità appena descritte, sono state implementate operazioni di reset:

- **File → Reset All:** permette di resettare totalmente la GUI;
- **Image → Reset Image Size:** permette di resettare l'operazione di crop;
- **Mask → Remove mask → by shapefile:** rimuove la maschera precedentemente caricata;
- **Mask → Remove mask → by editor:** permette di deselezionare manualmente una parte o l'intera maschera precedentemente creata;

## Conclusioni e Sviluppi Futuri

Questo lavoro è stato realizzato come uno strumento di supporto all'utente che deve svolgere questo tipo di ricerche sul territorio mediante immagini aeree o satellitari con un focus particolare per quanto riguarda la ricerca e la classificazione di vasche di irrigazione per uso agricolo con un confronto diretto con la CTR.
Successivamente il progetto è stato generalizzato permettendo all'utente di variare i parametri per il calcolo dell'indice di vegetazione e andare alla ricerca di altre feature.
Come sviluppo futuro, infatti, è possibile modificare la classe, in particolare il metodo per il calcolo dell'indice di vegetazione e aggiungere altri indici di vegetazione con le relative opzioni nell'interfaccia grafica per permettere all'utente di ricercare nuove caratteristiche sull'immagine.

# Appendice

## [A1] Classe WaterDetector

```
classdef WaterDetector < handle
	properties (Access = private)
		image
		info
		crop
		mask
		water
		index
		threshold
		lat
		lon
		shape
		ctr
	end

	methods

		%COSTRUCTOR
		function obj = WaterDetector()
			obj.image = [];
			obj.crop.image = [];
			obj.crop.xMin = 0;
			obj.crop.xMax = 0;
			obj.crop.yMin = 0;
			obj.crop.xMax = 0;
			obj.mask = [];
			obj.info = {};
			obj.water = [];
			obj.index = '';
			obj.threshold = [];
			obj.lat = [];
			obj.lon = [];
			obj.shape = {};
			obj.ctr = {};
		end

		%INTERFACE
		function I = getImage(obj)
			I = obj.image(:,:,1:3);
		end

		function crop = getCrop(obj)
			crop = obj.crop;
		end

		function mask = getMask(obj)
			mask = obj.mask;
		end

		function water = getWater(obj)
			water = obj.water;
		end

		function shape = getShape(obj)
```

```
        shape = obj.shape;
end

function info = getInfo(obj)
        info = obj.info;
end

function ctr = getCTR(obj)
        ctr = obj.ctr;
end

function ctr = getWaterFromCTR(obj,descr)
        if (nargin > 1)
                ctr = {};
                k = 1;
                for i = 1 : size(obj.ctr,1)
                        if (ismember(obj.ctr(i).DESCR,descr))
                                ctr(k,1).Geometry = obj.ctr(i,1).Geometry;
                                ctr(k,1).BoundingBox = obj.ctr(i,1).BoundingBox;
                                ctr(k,1).X = obj.ctr(i,1).X;
                                ctr(k,1).Y = obj.ctr(i,1).Y;
                                ctr(k,1).LAYER = obj.ctr(i,1).LAYER;
                                ctr(k,1).DESCR = obj.ctr(i,1).DESCR;
                                ctr(k,1).id = obj.ctr(i,1).id;
                                ctr(k,1).Area = obj.ctr(i,1).Area;
                                ctr(k,1).found = obj.ctr(i,1).found;
                                k = k + 1;
                        end
                end
        end
end
```

## [A2] Funzione readImage

```
%METHOD
function readImage(obj,filename)
        if (nargin > 1)
                obj.image = geotiffread(filename);
                temp = adapthisteq(obj.image(:,:,1));
                obj.image(:,:,1) = adapthisteq(obj.image(:,:,3));
                obj.image(:,:,3) = temp;
                obj.image(:,:,2) = adapthisteq(obj.image(:,:,2));
                obj.image(:,:,4) = adapthisteq(obj.image(:,:,4));
                obj.info = geotiffinfo(filename);
                obj.lat = [];
                obj.lon = [];
        end
end
```

## [A3] funzione cropImage

```
function crop = cropImage(obj)
        if (~isempty(obj.image))
                if (isempty(obj.mask))
                        [~, obj.crop.rect] = imcrop(obj.image(:,:,1:3));
                        close;
```

```
                else
                        r = obj.image(:,:,1);
                        g = obj.image(:,:,2);
                        b = obj.image(:,:,3);
                        r(obj.mask) = 0;
                        g(obj.mask) = 0;
                        b(obj.mask) = 65535;
                        [~, obj.crop.rect] = imcrop(cat(3,r,g,b));
                        close;
                end
                if (~isempty(obj.crop.rect))
                        obj.crop.xMin = round(obj.crop.rect(1));
                        width = round(obj.crop.rect(3));
                        height = round(obj.crop.rect(4));
                        obj.crop.yMin = round(obj.crop.rect(2));
                        obj.crop.xMax = obj.crop.xMin + (width - 1);
                        obj.crop.yMax = obj.crop.yMin + (height - 1);
                        obj.crop.image = obj.image(obj.crop.yMin:obj.crop.yMax,...
                        obj.crop.xMin:obj.crop.xMax,:);
                        obj.water = [];
                        obj.index = '';
                        obj.threshold = [];
                        obj.lat = [];
                        obj.lon = [];
                        crop = obj.crop;
                end
        end
end
```

## [A4] Funzione removeCrop

```
function removeCrop(obj)
        obj.crop.image = [];
        obj.crop.rect = [];
        obj.crop.xMin = 0;
        obj.crop.xMax = 0;
        obj.crop.yMin = 0;
        obj.crop.xMax = 0;
        if (~isempty(obj.index) || ~isempty(obj.threshold))
                obj.calcNdvi(obj.index, obj.threshold);
        end
        obj.water = [];
        obj.index = '';
        obj.threshold = [];
        obj.lat = [];
        obj.lon = [];
end
```

## [A5] Funzione calcNdvi

```
function Imm = calcNdvi(obj, index, threshold)
        if (~isempty(obj.image) && nargin >1)
                if (~isempty(obj.crop.image))
                        I = obj.crop.image;
                else
                        I = obj.image;
                end
                I = double(I);
                index = lower(index);
```

```
                tMin = -1;
                tMax = 1;
                switch index
                        case 'ndvi'
                                Imm= (I(:,:,4) - I(:,:,1)) ./ (I(:,:,4) + I(:,:,1));
                                tMin = -0.7;
                                tMax = -0.3;
                        case 'ndwi2'
                                Imm = (I(:,:,2) - I(:,:,4)) ./ (I(:,:,2) + I(:,:,4));
                                tMin = 0.7;
                        case 'ndwi2m'
                                Imm = (I(:,:,4) - I(:,:,2)) ./ (I(:,:,4) + I(:,:,2));
                                tMax = -0.7;
                        otherwise
                                return;
                end
                obj.index = index;
                obj.threshold = threshold;
                obj.water = false(size(Imm));
                if (nargin > 2)
                        if (size(threshold,2) == 1)
                                if (threshold>=-1 && threshold<=1)
                                        tMin = threshold;
                                end
                        elseif (size(threshold,2) > 1)
                                if (threshold(1)>=-1 && threshold(1)<=1)
                                        tMin = threshold(1);
                                end
                                if (threshold(2)>=threshold(1) && threshold(2)<=1)
                                        tMax = threshold(2);
                                end
                        end
                end
                obj.water = Imm > tMin & Imm < tMax;
                Imm(~obj.water) = 0;
        end
end
```

## [A6] Funzione showMask

```
function image = showMask(obj)
        image = [];
        if (~isempty(obj.mask))
                if (isempty(obj.crop.image))
                        r = obj.image(:,:,1);
                        g = obj.image(:,:,2);
                        b = obj.image(:,:,3);
                        r(obj.mask) = 0;
                        g(obj.mask) = 0;
                        b(obj.mask) = 65535;
                        image = cat(3,r,g,b);
                else
                        r = obj.crop.image(:,:,1);
                        g = obj.crop.image(:,:,2);
                        b = obj.crop.image(:,:,3);
```

```
                r(obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:obj.cro
                p.xMax)) = 0;
                g(obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:obj.cro
                p.xMax)) = 0;
                b(obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:obj.cro
                p.xMax)) = 65535;
                image = cat(3,r,g,b);
            end
        end
    end
```

## [A7] Funzione sahpe2mask

```
function shape2mask(obj, shape)
    if (nargin > 1 && ~isempty(obj.image) && ~isempty(shape))
        if (isempty(obj.mask))
            obj.mask = false(size(obj.image(:,:,1)));
        end
        [x,y] = pixcenters(obj.info);
        [X,Y] = meshgrid(x,y);
        for i=1:size(shape)
            rx = shape(i).X(1:end-1);
            ry = shape(i).Y(1:end-1);
            obj.mask = obj.mask | inpolygon(X,Y,rx,ry);
        end
    end
end
```

## [A8] Funzione removeShapeMask

```
function removeShapeMask(obj, shape)
    if (nargin > 1 && ~isempty(obj.mask) && ~isempty(shape))
        [x,y] = pixcenters(obj.info);
        [X,Y] = meshgrid(x,y);
        for i = 1:size(shape)
            rx = shape(i).X(1:end-1);
            ry = shape(i).Y(1:end-1);
            obj.mask(inpolygon(X,Y,rx,ry)) = false;
        end
        if (isempty(obj.mask(obj.mask)))
            obj.mask = [];
        end
    end
end
```

## [A9] Funzione applyMask

```
function applyMask(obj)
    if (~isempty(obj.image))
        if (isempty(obj.mask))
            obj.mask = false(size(obj.image(:,:,1)));
        end
        m = roipoly(obj.showMask());
        close;
        if (~isempty(m))
            if (isempty(obj.crop))
                obj.mask = obj.mask | m;
            else
```

```
                                        obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:
                                        obj.crop.xMax) = ...
                                        obj.mask(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:
                                        obj.crop.xMax) | m;
                                end
                        end
                end
        end
```

## [A10] Funzione removeMask

```
function removeMask(obj)
        if (~isempty(obj.mask))
                poly = roipoly(obj.showMask());
                close;
                if (~isempty(poly))
                        if (isempty(obj.crop.image))
                                obj.mask(poly) = false;
                        else
                                m = false(size(obj.image(:,:,1)));
                                m(obj.crop.yMin:obj.crop.yMax,obj.crop.xMin:obj.crop
                                .xMax) = poly;
                                obj.mask(m) = false;
                        end
                        if (isempty(obj.mask(obj.mask)))
                                obj.mask = [];
                        end
                end
        end
end
```

## [A11] Funzione maskFromCTR

```
function maskFromCTR(obj, ctr, descr)
        if (nargin > 2 && ~isempty(obj.image))
                if (isempty(obj.mask))
                        obj.mask = false(size(obj.image(:,:,1)));
                end
                [x,y] = pixcenters(obj.info);
                [X,Y] = meshgrid(x,y);
                for i=1:size(ctr)
                        center =
                        round(map2pix(obj.info.RefMatrix,ctr(i).BoundingBox(:,1),ctr(i)
                        .BoundingBox(:,2)));
                        xMin = center(1,2);
                        xMax = center(2,2);
                        yMin = center(2,1);
                        yMax = center(1,1);
                                if(ismember(ctr(i).DESCR,descr) &&
                                obj.compareBoundingBox(ctr(i).BoundingBox,obj.info
                                .BoundingBox) && ...
                                sum(obj.image(yMin,xMin,:)~=909 &
                                obj.image(yMax,xMax,:)~=909)==4)
                                rx = ctr(i).X(1:end-1);
                                ry = ctr(i).Y(1:end-1);
                                obj.mask = obj.mask | inpolygon(X,Y,rx,ry);
                        end
                end
        end
```

```
end
```

## [A12] Funzione removeMaskFromCTR

```
function removeMaskFromCTR(obj,ctr,descr)
    if (nargin > 2 && ~isempty(obj.mask))
        [x,y] = pixcenters(obj.info);
        [X,Y] = meshgrid(x,y);
        for i = 1:size(ctr)
            center =
            round(map2pix(obj.info.RefMatrix,ctr(i).BoundingBox(:,1),ctr(i)
            .BoundingBox(:,2)));
            xMin = center(1,2);
            xMax = center(2,2);
            yMin = center(2,1);
            yMax = center(1,1);
                if(ismember(ctr(i).DESCR,descr) &&
                obj.compareBoundingBox(ctr(i).BoundingBox,obj.info
                .BoundingBox) && ...
                sum(obj.image(yMin,xMin,:)~=909 &
                obj.image(yMax,xMax,:)~=909)==4)
                rx = ctr(i).X(1:end-1);
                ry = ctr(i).Y(1:end-1);
                obj.mask(inpolygon(X,Y,rx,ry)) = false;
            end
        end
        if (isempty(obj.mask(obj.mask)))
            obj.mask = [];
        end
    end
end
```

## [A13] Funzione createShape

```
function [num, shape] = createShape(obj, Amin, Amax)
    if (~isempty(obj.water))
        obj.shape = {};
        bw = obj.water;
        if (~isempty(obj.crop.image) && ~isempty(obj.mask))
            bw(obj.mask(obj.crop.yMin:obj.crop.yMax,...
            obj.crop.xMin:obj.crop.xMax)) = false;
        elseif (isempty(obj.crop.image) && ~isempty(obj.mask))
            bw(obj.mask) = false;
        end
        cc = bwconncomp(bw, 8);
        j = 1;
        temp = false(size(bw));
        if (isempty(obj.lat) || isempty(obj.lon))
            if (~isempty(obj.crop.image))
                [obj.lat, obj.lon] =
                obj.getLatLon([obj.crop.yMin,obj.crop.yMax,...
                obj.crop.xMin,obj.crop.xMax]);
            else
                [obj.lat, obj.lon] = obj.getLatLon();
            end
        end
        for i = 1:cc.NumObjects
            temp(cc.PixelIdxList{i}) = true;
```

```
                                        coord = [obj.lat(cc.PixelIdxList{i}) obj.lon(cc.PixelIdxList{i})];
                                        maxLat = max(coord(:,1));
                                        p1 = obj.lon(obj.lat == maxLat);
                                        minLat = min(coord(:,1));
                                        p2 = obj.lon(obj.lat == minLat);
                                        maxLon = max(coord(:,2));
                                        p3 = obj.lat(obj.lon == maxLon);
                                        minLon = min(coord(:,2));
                                        p4 = obj.lat(obj.lon == minLon);
                                        [x, y] = projfwd(obj.info, [maxLat, p3, minLat, p4, maxLat],
                                        [p1, maxLon, p2, minLon, p1]);
                                        area = polyarea(x,y);
                                        checkAmin = true;
                                        checkAmax = true;
                                        if (exist('Amin', 'var') && Amin >= 0)
                                                checkAmin = area > Amin;
                                        end
                                        if (exist('Amax', 'var') && Amax >= Amin)
                                                checkAmax = area < Amax;
                                        end
                                        if (size(coord,1) > 4 && area > 0 && checkAmin &&
                                        checkAmax)
                                                obj.shape(j).Geometry = deal('Polygon');
                                                obj.shape(j).id = j;
                                                obj.shape(j).X = x;
                                                obj.shape(j).Y = y;
                                                obj.shape(j).BoundingBox = [min(x) min(y); max(x)
                                                max(y)];
                                                obj.shape(j).Description = 'vasche trovate';
                                                obj.shape(j).Area = area;
                                                obj.shape(j).isNew = true;
                                                k = 1;
                                                found = false;
                                                while (k <= size(obj.ctr,1) && ~found)
                                                        found =
                                                        obj.compareBoundingBox(obj.shape(j).Boun
                                                        dingBox,obj.ctr(k).BoundingBox);
                                                        k= k + 1;
                                                end
                                                if (found)
                                                        obj.shape(j).BoundingBox = obj.ctr(k-
                                                        1).BoundingBox;
                                                        obj.shape(j).Area = obj.ctr(k-1).Area;
                                                        obj.shape(j).X = obj.ctr(k-1).X;
                                                        obj.shape(j).Y = obj.ctr(k-1).Y;
                                                        obj.shape(j).isNew = false;
                                                        obj.ctr(k-1).found = true;
                                                end
                                                j = j + 1;
                                        end
                                        temp(cc.PixelIdxList{i}) = false;
                                end
                                num = j - 1;
                                shape = obj.shape;
                        end
                end
```

## [A14] Funzione showElement

```
function [element, area] = showElement(obj,id,z)
```

```
        if (nargin>1 && ~isempty(obj.shape))
                shapeElement = obj.shape([obj.shape.id] == id);
                zoom = 100;
                if (nargin > 2)
                        zoom = z;
                end
                center =
                round(map2pix(obj.info.RefMatrix,shapeElement.BoundingBox(:,1),sh
                apeElement.BoundingBox(:,2)));
                xMin = ((center(1,2)-zoom)>=0)*(center(1,2)-zoom) + ((center(1,2)-
                zoom)<0)*0;
                xMax = ((center(2,2)+zoom)<=obj.info.Width)*(center(2,2)+zoom) +
                ((center(2,2)+zoom)>obj.info.Width)*obj.info.Width;
                yMin = ((center(2,1)-zoom)>=0)*(center(2,1)-zoom) + ((center(2,1)-
                zoom)<0)*0;
                yMax = ((center(1,1)+zoom)<=obj.info.Height)*(center(1,1)+zoom) +
                ((center(1,1)+zoom)>obj.info.Height)*obj.info.Height;
                element = obj.image(yMin : yMax,xMin : xMax,1:3);
                area = shapeElement.Area;
        end
end
```

## [A15] Funzione showCTRElement

```
function [element, area] = showCTRElement(obj,id,z)
        if (nargin>1 && ~isempty(obj.ctr))
                shapeElement = obj.ctr([obj.ctr.id] == id);
                zoom = 100;
                if (nargin > 2)
                        zoom = z;
                end
                center =
                round(map2pix(obj.info.RefMatrix,shapeElement.BoundingBox(:,1),sh
                apeElement.BoundingBox(:,2)));
                xMin = ((center(1,2)-zoom)>=0)*(center(1,2)-zoom) + ((center(1,2)-
                zoom)<0)*0;
                xMax = ((center(2,2)+zoom)<=obj.info.Width)*(center(2,2)+zoom) +
                ((center(2,2)+zoom)>obj.info.Width)*obj.info.Width;
                yMin = ((center(2,1)-zoom)>=0)*(center(2,1)-zoom) + ((center(2,1)-
                zoom)<0)*0;
                yMax = ((center(1,1)+zoom)<=obj.info.Height)*(center(1,1)+zoom) +
                ((center(1,1)+zoom)>obj.info.Height)*obj.info.Height;
                element = obj.image(yMin : yMax,xMin : xMax,1:3);
                area = shapeElement.Area;
        end
end
```

## [A16] Funzione removeElement

```
function removeElement(obj,id)
        if (nargin>1 && ~isempty(obj.shape))
                shapeElement = obj.shape([obj.shape.id] == id);
                if (isempty(obj.mask))
                        obj.mask = false(size(obj.image(:,:,1)));
                end
                if (~isempty(shapeElement))
                        center =
                        round(map2pix(obj.info.RefMatrix,shapeElement.BoundingBo
                        x(:,1),shapeElement.BoundingBox(:,2)));
```

```
                    xMin = center(1,2);
                    xMax = center(2,2);
                    yMin = center(2,1);
                    yMax = center(1,1);
                    obj.mask(yMin : yMax,xMin : xMax) = true;
                    obj.shape([obj.shape.id] == id) = [];
                    for i = id : (obj.shape(end).id-1)
                        obj.shape(i).id = obj.shape(i).id - 1;
                    end
                end
            end
        end
```

## [A17] Funzione applyMorphClose

```
function applyMorphClose(obj)
    if (~isempty(obj.water))
        obj.water = bwmorph(obj.water, 'close');
    end
end
```

## [A18] Funzione applyMorphClean

```
function applyMorphClean(obj)
    if (~isempty(obj.water))
        obj.water = bwmorph(obj.water, 'clean');
    end
end
```

## [A19] Funzione importCTR

```
function [num, ctr] = importCTR(obj,shape,descr)
    if (nargin > 2 && ~isempty(obj.info))
        obj.ctr = {};
        ctr = {};
        inCrop = 0;
        j = 1;
        z = 1;
        for i = 1 : size(shape,1)
            center =
            round(map2pix(obj.info.RefMatrix,shape(i).BoundingBox(:,1),
            shape(i).BoundingBox(:,2)));
            xMin = center(1,2);
            xMax = center(2,2);
            yMin = center(2,1);
            yMax = center(1,1);
            if
            (obj.compareBoundingBox(shape(i).BoundingBox,obj.info.Bo
            undingBox) && ...
            sum(obj.image(yMin,xMin,:)~=909 &
            obj.image(yMax,xMax,:)~=909)==4)
                ctr(z,1).Geometry = shape(i).Geometry;
                ctr(z,1).BoundingBox = shape(i).BoundingBox;
                ctr(z,1).X = shape(i).X;
                ctr(z,1).Y = shape(i).Y;
                ctr(z,1).LAYER = shape(i).LAYER;
                ctr(z,1).DESCR = shape(i).DESCR;
                ctr(z,1).id = z;
```

```
                ctr(z,1).Area =
                polyarea(ctr(z,1).X(~isnan(ctr(z,1).X)),ctr(z,1).Y(~isna
                n(ctr(z,1).Y)));
                ctr(z,1).found = false;
                if (ismember(shape(i).DESCR,descr))
                        obj.ctr(j,1).Geometry = ctr(z,1).Geometry;
                        obj.ctr(j,1).BoundingBox =
                        ctr(z,1).BoundingBox;
                        obj.ctr(j,1).X = ctr(z,1).X;
                        obj.ctr(j,1).Y = ctr(z,1).Y;
                        obj.ctr(j,1).LAYER = ctr(z,1).LAYER;
                        obj.ctr(j,1).DESCR = ctr(z,1).DESCR;
                        obj.ctr(j,1).id = j;
                        obj.ctr(j,1).Area = ctr(z,1).Area;
                        if (~isempty(obj.shape))
                                y = 1;
                                while (y<= size(obj.shape,2) &&
                                ~ctr(z,1).found)
                                        ctr(z,1).found =
                                        obj.compareBoundingBox(ob
                                        j.shape(1,y).BoundingBox,ctr
                                        (z).BoundingBox);
                                        y = y + 1;
                                end
                                if (ctr(z,1).found)
                                        obj.shape(1,y-
                                        1).BoundindBox =
                                        ctr(z,1).BoundingBox;
                                        obj.shape(1,y-1).Area =
                                        ctr(z,1).Area;
                                        obj.shape(1,y-1).X =
                                        ctr(z,1).X;
                                        obj.shape(1,y-1).Y =
                                        ctr(z,1).Y;
                                        obj.shape(1,y-1).isNew =
                                        ~ctr(z,1).found;
                                end
                        end
                        obj.ctr(j,1).found = ctr(z,1).found;
                        j = j + 1;
                        if (~isempty(obj.crop.image))
                                cropBoundingBox =
                                pix2map(obj.info.RefMatrix,[obj.crop.
                                yMin, obj.crop.xMin]',[obj.crop.xMax,
                                obj.crop.yMax]');
                                if
                                (obj.compareBoundingBox(shape(i).
                                BoundingBox,cropBoundingBox))
                                        inCrop = inCrop + 1;
                                end
                        end
                end
                z = z + 1;
        end
        if (isempty(obj.crop.image))
                num = size(obj.ctr,1);
        else
                num = [size(obj.ctr,1), inCrop];
        end
end
```

```
        end
    end
```

## [A20] Funzione cropCTR

```
    function ctr = cropCTR(obj,descr)
        ctr = {};
        if (~isempty(obj.crop.image))
            j = 1;
            for i = 1:size(obj.ctr,1)
                cropBoundingBox =
                pix2map(obj.info.RefMatrix,[obj.crop.yMax;
                obj.crop.yMin],[obj.crop.xMin; obj.crop.xMax]);
                check = true;
                if (nargin>1)
                    check =
                    ismember(string(obj.ctr(i).DESCR),string(descr));
                end
                if
                (obj.compareBoundingBox(obj.ctr(i).BoundingBox,cropBoundi
                ngBox) && check)
                    ctr(j,1).Geometry = obj.ctr(i,1).Geometry;
                    ctr(j,1).BoundingBox = obj.ctr(i,1).BoundingBox;
                    ctr(j,1).X = obj.ctr(i,1).X;
                    ctr(j,1).Y = obj.ctr(i,1).Y;
                    ctr(j,1).LAYER = obj.ctr(i,1).LAYER;
                    ctr(j,1).DESCR = obj.ctr(i,1).DESCR;
                    ctr(j,1).id = obj.ctr(i,1).id;
                    ctr(j,1).Area = obj.ctr(i,1).Area;
                    ctr(j,1).found = obj.ctr(i,1).found;
                    j = j + 1;
                end
            end
        end
    end
```

## [A21] Funzione addCTRElementToShape

```
    function addCTRElementToShape(obj,id)
        if (nargin>1)
            element = obj.ctr([obj.ctr.id] == id);
            if (~isempty(element) && ~element.found)
                i = size(obj.shape,2) + 1;
                obj.shape(1,i).Geometry = element.Geometry;
                obj.shape(1,i).X = element.X;
                obj.shape(1,i).Y = element.Y;
                obj.shape(1,i).BoundingBox = element.BoundingBox;
                obj.shape(1,i).Description = obj.shape(1).Description;
                obj.shape(1,i).Area = element.Area;
                obj.shape(1,i).isNew = false;
                obj.shape(1,i).id = i;
                obj.ctr([obj.ctr.id] == id).found = true;
            end
        end
    end
```

## [A22] Funzione removeToCTR

```
    function removeToCTR(obj,id)
```

```
                if (nargin>1)
                    element = obj.ctr([obj.ctr.id] == id);
                    if (~isempty(element))
                        obj.ctr([obj.ctr.id] == id) = [];
                        for i = id : (obj.ctr(end).id-1)
                            obj.ctr(i).id = obj.ctr(i).id - 1;
                        end
                    end
                end
            end
```

## [A23] Funzione removeCTR

```
            function removeCTR(obj)
                obj.ctr = {};
            end

        end

        methods (Access = private)
```

## [A24] Funzione getLatLon

```
            function [lat, lon] = getLatLon(obj,rect)
                if (nargin > 0)
                    yMin = 1;
                    yMax = obj.info.Height;
                    xMin = 1;
                    xMax = obj.info.Width;
                    if (nargin > 1)
                        yMin = rect(1);
                        yMax = rect(2);
                        xMin = rect(3);
                        xMax = rect(4);
                    end
                    [rows,cols] = meshgrid(yMin:yMax,xMin:xMax);
                    [x,y] = pix2map(obj.info.RefMatrix, rows, cols);
                    [lat,lon] = projinv(obj.info,x,y);
                    lat = lat';
                    lon = lon';
                end
            end
```

## [A25] Funzione compareBoundingBox

```
            function flag = compareBoundingBox(~,BoundingBox1,BoundingBox2)
                if (nargin > 2)
                    if (BoundingBox1(:,1) >= BoundingBox2(1,1) & ...
                        BoundingBox1(:,1) <= BoundingBox2(2,1) & ...
                        BoundingBox1(:,2) >= BoundingBox2(1,2) & ...
                        BoundingBox1(:,2) <= BoundingBox2(2,2))
                        flag = true;
                    else
                        flag = false;
                    end
                end
            end
        end
end
```

## [B1] waterDetectorGUI

```
classdef waterDetectorGUI < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        WaterDetectorGUIUIFigure matlab.ui.Figure
        FileMenu matlab.ui.container.Menu
        OpenGeotifImageMenu matlab.ui.container.Menu
        OpenCTRMenu matlab.ui.container.Menu
        ChangeCTRfieldsMenu matlab.ui.container.Menu
        ShapeFileMenu matlab.ui.container.Menu
        SaveCTRMenu matlab.ui.container.Menu
        ResetAllMenu matlab.ui.container.Menu
        ImageMenu matlab.ui.container.Menu
        CropMenu matlab.ui.container.Menu
        ResetImageSizeMenu matlab.ui.container.Menu
        MaskMenu matlab.ui.container.Menu
        AddmaskMenu matlab.ui.container.Menu
        byshapefileMenu matlab.ui.container.Menu
        byeditorMenu matlab.ui.container.Menu
        byCTRfieldsMenu matlab.ui.container.Menu
        RemovemaskMenu matlab.ui.container.Menu
        byshapefileMenu_2 matlab.ui.container.Menu
        byeditorMenu_2 matlab.ui.container.Menu
        byCTRfieldsMenu_2 matlab.ui.container.Menu
        ShowMaskMenu matlab.ui.container.Menu
        MorphologyMenu matlab.ui.container.Menu
        CleaningMenu matlab.ui.container.Menu
        ClosingMenu matlab.ui.container.Menu
        CropUIAxes matlab.ui.control.UIAxes
        IndexUIAxes matlab.ui.control.UIAxes
        MinSliderLabel matlab.ui.control.Label
        MinSlider matlab.ui.control.Slider
        MaxSliderLabel matlab.ui.control.Label
        MaxSlider matlab.ui.control.Slider
        DropDown matlab.ui.control.DropDown
        ImageUIAxes matlab.ui.control.UIAxes
        MaxField matlab.ui.control.NumericEditField
        MinField matlab.ui.control.NumericEditField
        PreviewUIAxes matlab.ui.control.UIAxes
        CreateShapeButton matlab.ui.control.Button
        ShowImageButton matlab.ui.control.Button
        ShowPreviewButton matlab.ui.control.Button
        ShowNDVIButton matlab.ui.control.Button
        ShowCropButton matlab.ui.control.Button
        numFindLabel matlab.ui.control.Label
        FindDropDown matlab.ui.control.DropDown
        UseasmaskButton matlab.ui.control.Button
        numCTRLabel matlab.ui.control.Label
        CTRDropDown matlab.ui.control.DropDown
        AddtoshapeButton matlab.ui.control.Button
        RemovetoctrButton matlab.ui.control.Button
    end
```

```
properties (Access = private)
        wd = WaterDetector() % water detector object
        formatToSave = {'.jpeg','JPEG image';...
        '.png','PNG image';...
        '*.*', 'All Files (*.*)'};
        preview = [];
        ndvi = [];
        ctr = {};
        Selection = [];
        maskSelection = [];
end

methods (Access = private)

        function setMinThreshold(app,val)
                if (val>=-1 && val<=1)
                        app.MaxSlider.Limits = [val, 1];
                        app.MaxField.Limits = [val, 1];
                end
        end

        function setMaxThreshold(app,val)
                if (val>=-1 && val<=1)
                        app.MinSlider.Limits = [-1, val];
                        app.MinField.Limits = [-1, val];
                end
        end

        function resetImage(app,element,title)
                cla(element);
                reset(element);
                element.Box = 'on';
                element.Title.String = title;
        end

        function showPreview(app)
                app.preview = app.wd.getWater();
                mask = app.wd.getMask();
                crop = app.wd.getCrop();
                if (~isempty(app.preview))
                        if (~isempty(mask))
                                if (~isempty(crop.image))
                                        app.preview(mask(crop.yMin:crop.yMax,...
                                        crop.xMin:crop.xMax)) = false;
                                else
                                        app.preview(mask) = false;
                                end
                        end
                        imshow(app.preview,'Parent',app.PreviewUIAxes);
                end
        end

end

methods (Access = private)

        % Menu selected function: OpenGeotifImageMenu
        function OpenGeotifImageMenuSelected(app, event)
                [filename, path] = uigetfile({'*.tif', 'Geotif Image (*.tif)';...
                '*.*', 'All Files (*.*)'});
```

```
        if (ischar(filename) && ischar(path))
                msgbox(char("Please, don't touch anything!"),'Wait');
                app.wd.readImage(strcat(path,filename));
                close;
                i = app.wd.getImage();
                if (~isempty(i))
                        imshow(i,'Parent',app.ImageUIAxes);
                        app.MaskMenu.Enable = 'on';
                        app.ImageMenu.Enable = 'on';
                        app.DropDown.Enable = 'on';
                        app.resetImage(app.CropUIAxes,"Crop");
                        app.resetImage(app.IndexUIAxes,"Index");
                        app.resetImage(app.PreviewUIAxes,"Preview");
                        app.ShowImageButton.Enable = 'on';
                        app.ShowPreviewButton.Enable = 'off';
                        app.ShowCropButton.Enable = 'off';
                        app.ShowNDVIButton.Enable = 'off';
                        app.DropDown.Value = 'INDEX';
                        app.numFindLabel.Text = "";
                        app.FindDropDown.Visible = 'off';
                        app.FindDropDown.Items = {};
                        app.CTRDropDown.Visible = 'off';
                        app.CTRDropDown.Items = {};
                        app.UseasmaskButton.Visible = 'off';
                        app.OpenCTRMenu.Enable = 'on';
                        app.ResetAllMenu.Enable = 'on';
                        app.AddtoshapeButton.Visible = 'off';
                end
        end
end

% Menu selected function: byshapefileMenu
function byshapefileMenuSelected(app, event)
        if (~isempty(app.wd.getImage()))
                [filename, path] = uigetfile({'*.shp', 'Shape file (*.shp)';...
                '*.*', 'All Files (*.*)'});
                if (ischar(filename) && ischar(path))
                        shapemask = shaperead(strcat(path,filename));
                                if (exist('shapemask','var') || ~isempty(shapemask))
                                msgbox(char("Please, don't touch anything!"),'Wait');
                                app.wd.shape2mask(shapemask);
                                if (app.RemovemaskMenu.Enable == "off")
                                        app.RemovemaskMenu.Enable = 'on';
                                end
                                if (app.ShowMaskMenu.Enable == "off")
                                        app.ShowMaskMenu.Enable = 'on';
                                end
                                close;
                                app.showPreview();
                        end
                end
        end
end

% Menu selected function: byeditorMenu
function byeditorMenuSelected(app, event)
        if (~isempty(app.wd.getImage()))
                app.wd.applyMask();
        if (app.RemovemaskMenu.Enable == "off")
                app.RemovemaskMenu.Enable = 'on';
```

```
        end
        if (app.ShowMaskMenu.Enable == "off")
            app.ShowMaskMenu.Enable = 'on';
        end
        app.showPreview();
        end
end

% Menu selected function: byshapefileMenu_2
function byshapefileMenu_2Selected(app, event)
        if (~isempty(app.wd.getMask()))
            [filename, path] = uigetfile({'*.shp', 'Shape file (*.shp)';...
            '*.*', 'All Files (*.*)'});
            shapemask = shaperead(strcat(path,filename));
            if (exist('shapemask','var') || ~isempty(shapemask))
                msgbox(char("Please, don't touch anything!"),'Wait');
                app.wd.removeShapeMask(shapemask);
                if (isempty(app.wd.getMask()))
                    app.RemovemaskMenu.Enable = 'off';
                    if (app.ShowMaskMenu.Enable == "on")
                        app.ShowMaskMenu.Enable = 'off';
                    end
                end
                close;
                app.showPreview();
            end
        end
end

% Menu selected function: byeditorMenu_2
function byeditorMenu_2Selected(app, event)
        if (~isempty(app.wd.getMask()))
            app.wd.removeMask();
            if (isempty(app.wd.getMask()))
                app.RemovemaskMenu.Enable = 'off';
                if (app.ShowMaskMenu.Enable == "on")
                    app.ShowMaskMenu.Enable = 'off';
                end
            end
            app.showPreview();
        end
end

% Menu selected function: CropMenu
function CropMenuSelected(app, event)
        if (~isempty(app.wd.getImage()))
            crop = app.wd.cropImage();
            if (~isempty(crop.image))
                app.preview = [];
                imshow(crop.image(:,:,1:3),'Parent',app.CropUIAxes);
                if (~isempty(app.Selection))
                    list = unique(extractfield(app.ctr,'DESCR'));
                    num =
                    app.wd.importCTR(app.ctr,string(list(app.Selection)));
                    shape = app.wd.cropCTR(list(app.Selection)');
                    app.numCTRLabel.Text = strcat(strcat("CTR:
                    ",string(num(1))),strcat("; CTR Crop
                    :",string(size(shape,1))));
                    if (~isempty(shape))
                        found = cell(size(shape));
```

```
                                found(:) = {" not found"};
                                found([shape(:).found])= {""};
                                app.CTRDropDown.Items =
                                cat(2,strcat("CTR: ",string(size(shape,1))) + "
                                " + strcat("not Found
                                :",string(sum(~[shape(:).found]))),...
                                string([shape(:).id]) + " : " +
                                string([shape(:).Area]) +...
                                string(found'));
                                app.AddtoshapeButton.Visible = 'on';
                                app.RemovetoctrButton.Visible = 'on';
                                app.SaveCTRMenu.Enable = 'on';
                        else
                                app.CTRDropDown.Items = strcat("CTR:
                                ",string(size(shape,1)));
                                app.AddtoshapeButton.Visible = 'off';
                                app.RemovetoctrButton.Visible = 'off';
                                app.SaveCTRMenu.Enable = 'on';
                        end
                end
                if (app.ResetImageSizeMenu.Enable == "off")
                        app.ResetImageSizeMenu.Enable = 'on';
                end
                app.resetImage(app.IndexUIAxes,"Index");
                app.resetImage(app.PreviewUIAxes,"Preview");
                app.ShowCropButton.Enable = 'on';
                app.ShowPreviewButton.Enable = 'off';
                app.ShowNDVIButton.Enable = 'off';
                app.numFindLabel.Text = "";
                app.FindDropDown.Visible = 'off';
                app.FindDropDown.Items = {};
                app.UseasmaskButton.Visible = 'off';
                app.DropDown.Value = 'INDEX';
                app.setMinThreshold(-1);
                app.setMaxThreshold(1);
                app.MinSlider.Value = 0;
                app.MinField.Value = 0;
                app.MaxSlider.Value = 0;
                app.MaxField.Value = 0;
                app.MinField.Enable = 'off';
                app.MaxField.Enable = 'off';
                app.MinSlider.Enable = 'off';
                app.MaxSlider.Enable = 'off';
                app.AddtoshapeButton.Visible = 'off';
        end
    end
end

% Menu selected function: ResetImageSizeMenu
function ResetImageSizeMenuSelected(app, event)
    crop = app.wd.getCrop();
    if (~isempty(crop.image))
        app.wd.removeCrop();
        if (~isempty(app.Selection))
            list = unique(extractfield(app.ctr,'DESCR'));
            num = app.wd.importCTR(app.ctr,string(list(app.Selection)));
            app.numCTRLabel.Text = strcat("CTR: ",string(num(1)));
            shape = app.wd.getWaterFromCTR(list(app.Selection)');
            if (~isempty(shape))
                found = cell(size(shape));
```

```
                found(:) = {" not found"};
                found([shape(:).found])= {""};
                app.CTRDropDown.Items = cat(2,strcat("CTR:
                ",string(size(shape,1))) + " " + strcat("not Found
                :",string(sum(~[shape(:).found]))),...
                string([shape(:).id]) + " : " + string([shape(:).Area]) +...
                string(found'));
                app.AddtoshapeButton.Visible = 'on';
                app.RemovetoctrButton.Visible = 'on';
                app.SaveCTRMenu.Enable = 'on';
            else
                app.CTRDropDown.Items = strcat("CTR:
                ",string(size(shape,1)));
                app.AddtoshapeButton.Visible = 'off';
                app.RemovetoctrButton.Visible = 'off';
                app.SaveCTRMenu.Enable = 'off';
            end
        end
        app.ResetImageSizeMenu.Enable = "off";
        app.resetImage(app.CropUIAxes,"Crop");
        app.resetImage(app.IndexUIAxes,"Index");
        app.resetImage(app.PreviewUIAxes,"Preview");
        app.ShowCropButton.Enable = 'off';
        app.ShowPreviewButton.Enable = 'off';
        app.ShowNDVIButton.Enable = 'off';
        app.numFindLabel.Text = "";
        app.MorphologyMenu.Enable = 'off';
        app.CreateShapeButton.Enable = 'off';
        app.FindDropDown.Visible = 'off';
        app.FindDropDown.Items = {};
        app.UseasmaskButton.Visible = 'off';
        app.DropDown.Value = 'INDEX';
        app.setMinThreshold(-1);
        app.setMaxThreshold(1);
        app.MinSlider.Value = 0;
        app.MinField.Value = 0;
        app.MaxSlider.Value = 0;
        app.MaxField.Value = 0;
        app.MinField.Enable = 'off';
        app.MaxField.Enable = 'off';
        app.MinSlider.Enable = 'off';
        app.MaxSlider.Enable = 'off';
        app.AddtoshapeButton.Visible = 'off';
    end
end

% Menu selected function: ShapeFileMenu
function ShapeFileMenuSelected(app, event)
    shape = app.wd.getShape();
    if (~isempty(shape))
        [filename, path] = uiputfile({'*.shp', 'Shape file (*.shp)'},'Save
        shapefile', 'shape.shp');
        shapewrite(shape,strcat(path,filename));
    end
end

% Value changed function: DropDown
function DropDownValueChanged(app, event)
    value = app.DropDown.Value;
    if (value~="INDEX")
```

```
                app.MinField.Enable = 'on';
                app.MaxField.Enable = 'on';
                app.MinSlider.Enable = 'on';
                app.MaxSlider.Enable = 'on';
                app.ShowPreviewButton.Enable = 'on';
                app.ShowNDVIButton.Enable = 'on';
                switch value
                    case 'NDVI'
                        app.setMinThreshold(-0.7);
                        app.setMaxThreshold(-0.3);
                        app.MinSlider.Value = -0.7;
                        app.MinField.Value = -0.7;
                        app.MaxSlider.Value = -0.3;
                        app.MaxField.Value = -0.3;
                        app.ndvi = app.wd.calcNdvi(value,[-0.7,-0.3]);
                        imshow(app.ndvi,'Parent',app.IndexUIAxes);
                        app.showPreview();
                    case 'NDWI2'
                        app.setMinThreshold(0.7);
                        app.setMaxThreshold(1);
                        app.MinSlider.Value = 0.7;
                        app.MinField.Value = 0.7;
                        app.MaxSlider.Value = 1;
                        app.MaxField.Value = 1;
                        app.ndvi = app.wd.calcNdvi(value,[0.7, 1]);
                        imshow(app.ndvi,'Parent',app.IndexUIAxes);
                        app.showPreview();
                    case 'NDWI2M'
                        app.setMinThreshold(-1);
                        app.setMaxThreshold(-0.7);
                        app.MinSlider.Value = -1;
                        app.MinField.Value = -1;
                        app.MaxSlider.Value = -0.7;
                        app.MaxField.Value = -0.7;
                        app.ndvi = app.wd.calcNdvi(value,[-1,-0.7]);
                        imshow(app.ndvi,'Parent',app.IndexUIAxes);
                        app.showPreview();
                end
                app.IndexUIAxes.Title.String = value;
                app.CreateShapeButton.Enable = 'on';
                app.MorphologyMenu.Enable = 'on';
            else
                app.setMinThreshold(-1);
                app.setMaxThreshold(1);
                app.MinSlider.Value = 0;
                app.MinField.Value = 0;
                app.MaxSlider.Value = 0;
                app.MaxField.Value = 0;
                app.resetImage(app.IndexUIAxes,"Index");
                app.resetImage(app.PreviewUIAxes,"Preview");
                app.MinField.Enable = 'off';
                app.MaxField.Enable = 'off';
                app.MinSlider.Enable = 'off';
                app.MaxSlider.Enable = 'off';
                app.ShowPreviewButton.Enable = 'off';
                app.ShowNDVIButton.Enable = 'off';
            end
        end

        % Value changed function: MinField
```

```
function MinFieldValueChanged(app, event)
        index = app.DropDown.Value;
        min = app.MinField.Value;
        max = app.MaxField.Value;
        app.MinSlider.Value = min;
        app.setMinThreshold(min);
        app.ndvi = app.wd.calcNdvi(index,[min,max]);
        imshow(app.ndvi,'Parent',app.IndexUIAxes);
        app.showPreview();
end

% Value changed function: MaxSlider
function MaxSliderValueChanged(app, event)
        index = app.DropDown.Value;
        max = app.MaxSlider.Value;
        min = app.MinField.Value;
        app.MaxField.Value = max;
        app.setMaxThreshold(max);
        app.ndvi = app.wd.calcNdvi(index,[min,max]);
        imshow(app.ndvi,'Parent',app.IndexUIAxes);
        app.showPreview();
end

% Value changed function: MinSlider
function MinSliderValueChanged(app, event)
        index = app.DropDown.Value;
        min = app.MinSlider.Value;
        max = app.MaxField.Value;
        app.MinField.Value = min;
        app.setMinThreshold(min);
        app.ndvi = app.wd.calcNdvi(index,[min,max]);
        imshow(app.ndvi,'Parent',app.IndexUIAxes);
        app.showPreview();
end

% Value changing function: MinSlider
function MinSliderValueChanging(app, event)
        changingValue = event.Value;
        app.MinField.Value = changingValue;
end

% Value changed function: MaxField
function MaxFieldValueChanged(app, event)
        index = app.DropDown.Value;
        max = app.MaxField.Value;
        min = app.MinField.Value;
        app.MaxSlider.Value = max;
        app.setMaxThreshold(max);
        app.ndvi = app.wd.calcNdvi(index,[min,max]);
        imshow(app.ndvi,'Parent',app.IndexUIAxes);
        app.showPreview();
end

% Value changing function: MaxSlider
function MaxSliderValueChanging(app, event)
        changingValue = event.Value;
        app.MaxField.Value = changingValue;
end

% Button pushed function: CreateShapeButton
```

```
function CreateShapeButtonPushed(app, event)
    if (~isempty(app.wd.getWater()))
        prompt = {'Area max (m^2)','Area min (m^2)'};
        title = 'Create shape';
        answer = str2double(inputdlg(prompt,title,1));
        if (~isempty(answer))
            answer(isnan(answer)) = 0;
            msgbox(char("Please, don't touch anything!"),'Wait');
            if (answer(1) >= answer(2) && answer(1) >0)
                app.CreateShapeButton.Enable = 'off';
                [num, shape] = app.wd.createShape(answer(2),
                answer(1));
                app.CreateShapeButton.Enable = 'on';
                app.numFindLabel.Text = strcat("Found:
                ",string(num));
                app.FindDropDown.Visible = 'on';
                if (~isempty(shape))
                    isNew = cell(size(shape));
                    isNew(:) = {""};
                    isNew([shape(:).isNew])= {" New"};
                    app.FindDropDown.Items =
                    cat(2,strcat("Found: ",string(num)) + " " +
                    strcat("New:",string(sum([shape(:).isNew]))),..
                    .
                    string([shape(:).id]) + " : " +
                    string([shape(:).Area]) +...
                    string(isNew));
                    if (~isempty(app.Selection))
                        crop = app.wd.getCrop();
                        list =
                        unique(extractfield(app.ctr,'DESCR'))
                        ;
                        if (~isempty(crop.image))
                            shape =
                            app.wd.cropCTR(list(app.Sel
                            ection)');
                            if (~isempty(shape))
                                found =
                                cell(size(shape));
                                found(:) = {" not
                                found"};
                                found([shape(:).foun
                                d])= {""};
                                app.CTRDropDown.I
                                tems =
                                cat(2,strcat("CTR:
                                ",string(size(shape,1
                                ))) + " " + strcat("not
                                Found
                                :",string(sum(~[shap
                                e(:).found]))),...
                                string([shape(:).id]) +
                                " : " +
                                string([shape(:).Area
                                ]) +...
                                string(found'));
                                app.AddtoshapeButt
                                on.Visible = 'on';
                                app.RemovetoctrBut
                                ton.Visible = 'on';
```

```
                app.SaveCTRMenu.
                Enable = 'on';
            else
                app.CTRDropDown.I
                tems = strcat("CTR:
                ",string(size(shape,1
                )));
                app.AddtoshapeButt
                on.Visible = 'off';
                app.RemovetoctrBut
                ton.Visible = 'off';
                app.SaveCTRMenu.
                Enable = 'off';
            end
        else
            shape =
            app.wd.getWaterFromCTR(li
            st(app.Selection)');
            if (~isempty(shape))
                found =
                cell(size(shape));
                found(:) = {" not
                found"};
                found([shape(:).foun
                d])= {""};
                app.CTRDropDown.I
                tems =
                cat(2,strcat("CTR:
                ",string(size(shape,1
                ))) + " " + strcat("not
                Found
                :",string(sum(~[shap
                e(:).found]))),...
                string([shape(:).id]) +
                " : " +
                string([shape(:).Area
                ]) +...
                string(found'));
                app.AddtoshapeButt
                on.Visible = 'on';
                app.RemovetoctrBut
                ton.Visible = 'on';
                app.SaveCTRMenu.
                Enable = 'on';
            else
                app.CTRDropDown.I
                tems = strcat("CTR:
                ",string(size(shape,1
                )));
                app.AddtoshapeButt
                on.Visible = 'off';
                app.RemovetoctrBut
                ton.Visible = 'off';
                app.SaveCTRMenu.
                Enable = 'off';
            end
        end
    end
    app.ShapeFileMenu.Enable = 'on';
    app.UseasmaskButton.Visible = 'on';
```

```
            else
                app.FindDropDown.Items = strcat("Found:
                ",string(size(shape,2)));
                app.AddtoshapeButton.Visible = 'off';
            end
        elseif (answer(2) >0 && answer(1)==0)
            app.CreateShapeButton.Enable = 'off';
            [num, shape] = app.wd.createShape(answer(2));
            app.CreateShapeButton.Enable = 'on';
            app.numFindLabel.Text = strcat("Found:
            ",string(num));
            app.FindDropDown.Visible = 'on';
            if (~isempty(shape))
                isNew = cell(size(shape));
                isNew(:) = {""};
                isNew([shape(:).isNew])= {" New"};
                app.FindDropDown.Items =
                cat(2,strcat("Found: ",string(num)) + " " +
                strcat("New:",string(sum([shape(:).isNew]))),..
                .
                string([shape(:).id]) + " : " +
                string([shape(:).Area]) +...
                string(isNew));
                if (~isempty(app.Selection))
                    crop = app.wd.getCrop();
                    list =
                    unique(extractfield(app.ctr,'DESCR'))
                    ;
                    if (~isempty(crop.image))
                        shape =
                        app.wd.cropCTR(list(app.Sel
                        ection)');
                        if (~isempty(shape))
                            found =
                            cell(size(shape));
                            found(:) = {" not
                            found"};
                            found([shape(:).foun
                            d])= {""};
                            app.CTRDropDown.I
                            tems =
                            cat(2,strcat("CTR:
                            ",string(size(shape,1
                            ))) + " " + strcat("not
                            Found
                            :",string(sum(~[shap
                            e(:).found]))),...
                            string([shape(:).id]) +
                            " : " +
                            string([shape(:).Area
                            ]) +...
                            string(found'));
                            app.AddtoshapeButt
                            on.Visible = 'on';
                            app.RemovetoctrBut
                            ton.Visible = 'on';
                            app.SaveCTRMenu.
                            Enable = 'on';
                        else
```

```
                    app.CTRDropDown.I
                    tems = strcat("CTR:
                    ",string(size(shape,1
                    )));
                    app.AddtoshapeButt
                    on.Visible = 'off';
                    app.RemovetoctrBut
                    ton.Visible = 'off';
                    app.SaveCTRMenu.
                    Enable = 'off';
                end
            else
                shape =
                app.wd.getWaterFromCTR(li
                st(app.Selection)');
                if (~isempty(shape))
                    found =
                    cell(size(shape));
                    found(:) = {" not
                    found"};
                    found([shape(:).foun
                    d])= {""};
                    app.CTRDropDown.I
                    tems =
                    cat(2,strcat("CTR:
                    ",string(size(shape,1
                    ))) + " " + strcat("not
                    Found
                    :",string(sum(~[shap
                    e(:).found]))),...
                    string([shape(:).id]) +
                    " : " +
                    string([shape(:).Area
                    ]) +...
                    string(found'));
                    app.AddtoshapeButt
                    on.Visible = 'on';
                    app.RemovetoctrBut
                    ton.Visible = 'on';
                    app.SaveCTRMenu.
                    Enable = 'on';
                else
                    app.CTRDropDown.I
                    tems = strcat("CTR:
                    ",string(size(shape,1
                    )));
                    app.AddtoshapeButt
                    on.Visible = 'off';
                    app.RemovetoctrBut
                    ton.Visible = 'off';
                    app.SaveCTRMenu.
                    Enable = 'off';
                end
            end
        end
        app.ShapeFileMenu.Enable = 'on';
        app.UseasmaskButton.Visible = 'on';
    else
        app.FindDropDown.Items = strcat("Found:
        ",string(size(shape,2)));
```

```
                app.AddtoshapeButton.Visible = 'off';
            end
        elseif (answer(2) ==0 && answer(1)==0)
            app.CreateShapeButton.Enable = 'off';
            [num, shape] = app.wd.createShape();
            app.CreateShapeButton.Enable = 'on';
            app.numFindLabel.Text = strcat("Found:
            ",string(num));
            app.FindDropDown.Visible = 'on';
            if (~isempty(shape))
                isNew = cell(size(shape));
                isNew(:) = {""};
                isNew([shape(:).isNew])= {" New"};
                app.FindDropDown.Items =
                cat(2,strcat("Found: ",string(num)) + " " +
                strcat("New:",string(sum([shape(:).isNew]))),..
                .
                string([shape(:).id]) + " : " +
                string([shape(:).Area]) +...
                string(isNew));
                if (~isempty(app.Selection))
                    crop = app.wd.getCrop();
                    list =
                    unique(extractfield(app.ctr,'DESCR'))
                    ;
                    if (~isempty(crop.image))
                        shape =
                        app.wd.cropCTR(list(app.Sel
                        ection)');
                        if (~isempty(shape))
                            found =
                            cell(size(shape));
                            found(:) = {" not
                            found"};
                            found([shape(:).foun
                            d])= {""};
                            app.CTRDropDown.I
                            tems =
                            cat(2,strcat("CTR:
                            ",string(size(shape,1
                            ))) + " " + strcat("not
                            Found
                            :",string(sum(~[shap
                            e(:).found]))),...
                            string([shape(:).id]) +
                            " : " +
                            string([shape(:).Area
                            ]) +...
                            string(found'));
                            app.AddtoshapeButt
                            on.Visible = 'on';
                            app.RemovetoctrBut
                            ton.Visible = 'on';
                            app.SaveCTRMenu.
                            Enable = 'on';
                        else
                            app.CTRDropDown.I
                            tems = strcat("CTR:
                            ",string(size(shape,1
                            )));
```

```
                    app.AddtoshapeButt
                    on.Visible = 'off';
                    app.RemovetoctrBut
                    ton.Visible = 'off';
                    app.SaveCTRMenu.
                    Enable = 'off';
                end
            else
                shape =
                app.wd.getWaterFromCTR(li
                st(app.Selection)');
                if (~isempty(shape))
                    found =
                    cell(size(shape));
                    found(:) = {'' not
                    found''};
                    found([shape(:).foun
                    d])= {''''};
                    app.CTRDropDown.I
                    tems =
                    cat(2,strcat(''CTR:
                    '',string(size(shape,1
                    ))) + '' '' + strcat(''not
                    Found
                    :'',string(sum(~[shap
                    e(:).found]))),...
                    string([shape(:).id]) +
                    '' : '' +
                    string([shape(:).Area
                    ]) +...
                    string(found'));
                    app.AddtoshapeButt
                    on.Visible = 'on';
                    app.RemovetoctrBut
                    ton.Visible = 'on';
                    app.SaveCTRMenu.
                    Enable = 'on';
                else
                    app.CropDropDown.
                    Items = strcat(''CTR:
                    '',string(size(shape,1
                    )));
                    app.AddtoshapeButt
                    on.Visible = 'off';
                    app.RemovetoctrBut
                    ton.Visible = 'off';
                    app.SaveCTRMenu.
                    Enable = 'off';
                end
            end
        end
        app.ShapeFileMenu.Enable = 'on';
        app.UseasmaskButton.Visible = 'on';
    else
        app.FindDropDown.Items = strcat(''Found:
        '',string(size(shape,2)));
        app.AddtoshapeButton.Visible = 'off';
    end
end
close;
```

```
                end
        end
end

% Menu selected function: ClosingMenu
function ClosingMenuSelected(app, event)
        if (~isempty(app.wd.getWater()))
                app.wd.applyMorphClose();
                app.showPreview();
        end
end

% Menu selected function: CleaningMenu
function CleaningMenuSelected(app, event)
        if (~isempty(app.wd.getWater()))
                app.wd.applyMorphClean();
                app.showPreview();
        end
end

% Button pushed function: ShowImageButton
function ShowImageButtonPushed(app, event)
        i = app.wd.getImage();
        if (~isempty(i))
                figure('Name','Image'),imshow(i);
        end
end

% Button pushed function: ShowPreviewButton
function ShowPreviewButtonPushed(app, event)
        if (~isempty(app.preview))
                figure('Name','Preview'),imshow(app.preview);
        end
end

% Button pushed function: ShowNDVIButton
function ShowNDVIButtonButtonPushed(app, event)
        if (~isempty(app.ndvi))
                figure('Name','Normalized Difference Vegetation
                Index'),imshow(app.ndvi);
        end
end

% Button pushed function: ShowCropButton
function ShowCropButtonButtonPushed(app, event)
        crop = app.wd.getCrop();
        if (~isempty(crop.image))
                figure('Name','Crop'),imshow(crop.image(:,:,1:3));
        end
end

% Value changed function: FindDropDown
function FindDropDownValueChanged(app, event)
        value = extractBefore(app.FindDropDown.Value,":");
        if (~strcmpi(value,'Found'))
                [image, area] = app.wd.showElement(str2double(value),30);
                figure('Name',strcat('Element: ',value)+"; Area: " +
                area),imshow(image);
        end
end
```

```
% Button pushed function: UseasmaskButton
function UseasmaskButtonPushed(app, event)
	value = extractBefore(app.FindDropDown.Value,":");
	if (~strcmpi(value,'Found'))
		value = str2double(value);
		app.wd.removeElement(value);
		app.showPreview();
		shape = app.wd.getShape();
		isNew = cell(size(shape));
		isNew(:) = {""};
		isNew([shape(:).isNew])= {" New"};
		app.FindDropDown.Items = cat(2,strcat("Found:
		",string(size(shape,2))) + " " +
		strcat("New:",string(sum([shape(:).isNew]))),...
		string([shape(:).id]) + " : " + string([shape(:).Area]) +...
		string(isNew));
	end
end

% Menu selected function: ResetAllMenu
function ResetAllMenuSelected(app, event)
	app.preview = [];
	app.ndvi = [];
	app.ctr = {};
	app.Selection = [];
	app.maskSelection = [];
	app.wd = WaterDetector();
	app.ResetImageSizeMenu.Enable = "off";
	app.resetImage(app.CropUIAxes,"Crop");
	app.resetImage(app.IndexUIAxes,"Index");
	app.resetImage(app.PreviewUIAxes,"Preview");
	app.resetImage(app.ImageUIAxes,"Image");
	app.ShowCropButton.Enable = 'off';
	app.ShowPreviewButton.Enable = 'off';
	app.ShowNDVIButton.Enable = 'off';
	app.ShowImageButton.Enable = 'off';
	app.ShapeFileMenu.Enable = 'off';
	app.ImageMenu.Enable = 'off';
	app.MaskMenu.Enable = 'off';
	app.MorphologyMenu.Enable = 'off';
	app.numFindLabel.Text = "";
	app.CreateShapeButton.Enable = 'off';
	app.FindDropDown.Visible = 'off';
	app.FindDropDown.Items = {};
	app.UseasmaskButton.Visible = 'off';
	app.DropDown.Value = 'INDEX';
	app.DropDown.Enable = 'off';
	app.setMinThreshold(-1);
	app.setMaxThreshold(1);
	app.MinSlider.Value = 0;
	app.MinField.Value = 0;
	app.MaxSlider.Value = 0;
	app.MaxField.Value = 0;
	app.MinField.Enable = 'off';
	app.MaxField.Enable = 'off';
	app.MinSlider.Enable = 'off';
	app.MaxSlider.Enable = 'off';
	app.OpenCTRMenu.Enable = 'off';
	app.ChangeCTRfieldsMenu.Enable = 'off';
```

```
            app.numCTRLabel.Text = "";
            app.numFindLabel.Text = "";
            app.ShowMaskMenu.Enable = 'off';
            app.byCTRfieldsMenu.Enable = 'off';
            app.CTRDropDown.Visible = 'off';
            app.CTRDropDown.Items = {};
            app.AddtoshapeButton.Visible = 'off';
            app.SaveCTRMenu.Enable = 'off';
            app.RemovetoctrButton.Visible = 'off';
end

% Menu selected function: OpenCTRMenu
function OpenCTRMenuSelected(app, event)
        if (~isempty(app.wd.getImage()))
                [filename, path] = uigetfile({'*.shp', 'Shape file (*.shp)';...
                '*.*', 'All Files (*.*)'});
                if (ischar(filename) && ischar(path))
                        msgbox(char("Please, don't touch anything!"),'Wait');
                        c = shaperead(strcat(path,filename));
                        close;
                        if (~isempty(c))
                                app.ctr = c;
                                list = unique(extractfield(app.ctr,'DESCR'));
                                selection = listdlg('Name','Import from CTR',
                                'PromptString','Select CTR elements:', ...
                                'SelectionMode','multiple','OKString','Select','ListStrin
                                g',list);
                                if(~isempty(selection))
                                        msgbox(char("Please, don't touch
                                        anything!"),'Wait');
                                        [num, app.ctr] =
                                        app.wd.importCTR(app.ctr,string(list(selectio
                                        n)));
                                        close;
                                        app.Selection =
                                        find(ismember(unique(extractfield(app.ctr,'DE
                                        SCR')),list(selection)));
                                        list = unique(extractfield(app.ctr,'DESCR'));
                                        if (size(num,2) > 1)
                                                app.CTRDropDown.Visible = 'on';
                                                shape =
                                                app.wd.cropCTR(list(app.Selection)')
                                                ;
                                                app.numCTRLabel.Text =
                                                strcat(strcat("CTR:
                                                ",string(num(1))),strcat("; CTR Crop
                                                :",string(size(shape,1))));
                                                if (~isempty(shape))
                                                        found = cell(size(shape));
                                                        found(:) = {" not found"};
                                                        found([shape(:).found])= {""};
                                                        app.CTRDropDown.Items =
                                                        cat(2,strcat("CTR:
                                                        ",string(size(shape,1))) + " "
                                                        + strcat("not Found
                                                        :",string(sum(~[shape(:).foun
                                                        d]))),...
                                                        string([shape(:).id]) + " : " +
                                                        string([shape(:).Area]) +...
                                                        string(found'));
```

```
                app.SaveCTRMenu.Enable
                = 'on';
                app.RemovetoctrButton.Visib
                le = 'on';
        else
                app.CTRDropDown.Items =
                strcat("CTR:
                ",string(size(shape,1)));
                app.SaveCTRMenu.Enable
                = 'off';
                app.RemovetoctrButton.Visib
                le = 'off';
        end
else
        app.numCTRLabel.Text =
        strcat("CTR: ",string(num));
        app.CTRDropDown.Visible = 'on';
        shape =
        app.wd.getWaterFromCTR(list(app.S
        election)');
        if (~isempty(shape))
                found = cell(size(shape));
                found(:) = {" not found"};
                found([shape(:).found])= {""};
                app.CTRDropDown.Items =
                cat(2,strcat("CTR:
                ",string(size(shape,1))) + " "
                + strcat("not Found
                :",string(sum(~[shape(:).foun
                d]))),...
                string([shape(:).id]) + " : " +
                string([shape(:).Area]) +...
                string(found'));
                app.SaveCTRMenu.Enable
                = 'on';
                app.RemovetoctrButton.Visib
                le = 'on';
        else
                app.CTRDropDown.Items =
                strcat("CTR:
                ",string(size(shape,1)));
                app.SaveCTRMenu.Enable
                = 'off';
                app.RemovetoctrButton.Visib
                le = 'off';
        end
end
shape = app.wd.getShape();
if (~isempty(shape))
        isNew = cell(size(shape));
        isNew(:) = {""};
        isNew([shape(:).isNew])= {" New"};
        app.FindDropDown.Items =
        cat(2,strcat("Found:
        ",string(size(shape,2))) + " " +
        strcat("New:",string(sum([shape(:).is
        New]))),...
        string([shape(:).id]) + " : " +
        string([shape(:).Area]) +...
        string(isNew));
```

```
                                        app.AddtoshapeButton.Visible = 'on';
                                    else
                                        app.FindDropDown.Items =
                                        strcat("Found:
                                        ",string(size(shape,2)));
                                        app.AddtoshapeButton.Visible = 'off';
                                    end
                                elseif (isempty(app.Selection))
                                    app.wd.removeCTR();
                                    app.CTRDropDown.Visible = 'off';
                                    app.numCTRLabel.Text = "";
                                    app.AddtoshapeButton.Visible = 'off';
                                    app.RemovetoctrButton.Visible = 'off';
                                    app.SaveCTRMenu.Enable = 'off';
                                end
                                app.ChangeCTRfieldsMenu.Enable = 'on';
                                app.byCTRfieldsMenu.Enable = 'on';
                            end
                        end
                    end
                end

                % Menu selected function: ChangeCTRfieldsMenu
                function ChangeCTRfieldsMenuSelected(app, event)
                    if (~isempty(app.ctr))
                        list = unique(extractfield(app.ctr,'DESCR'));
                        selection = listdlg('Name','Import from CTR', 'PromptString','Select
                        CTR elements:', ...
                        'SelectionMode','multiple','OKString','Select','ListString',list,...
                        'InitialValue',app.Selection);
                        if(~isempty(selection))
                            app.Selection=selection;
                            msgbox(char("Please, don't touch anything!"),'Wait');
                            num = app.wd.importCTR(app.ctr,string(list(app.Selection)));
                            close;
                            if (size(num,2) > 1)
                                app.CTRDropDown.Visible = 'on';
                                shape = app.wd.cropCTR(list(app.Selection)');
                                app.numCTRLabel.Text = strcat(strcat("CTR:
                                ",string(num(1))),strcat("; CTR Crop
                                :",string(size(shape,1))));
                                if (~isempty(shape))
                                    found = cell(size(shape));
                                    found(:) = {" not found"};
                                    found([shape(:).found])= {""};
                                    app.CTRDropDown.Items =
                                    cat(2,strcat("CTR: ",string(size(shape,1))) + "
                                    " + strcat("not Found
                                    :",string(sum(~[shape(:).found]))),...
                                    string([shape(:).id]) + " : " +
                                    string([shape(:).Area]) +...
                                    string(found'));
                                    app.SaveCTRMenu.Enable = 'on';
                                    app.RemovetoctrButton.Visible = 'on';
                                else
                                    app.CTRDropDown.Items = strcat("CTR:
                                    ",string(size(shape,1)));
                                    app.SaveCTRMenu.Enable = 'off';
                                    app.RemovetoctrButton.Visible = 'off';
                                end
```

```
                else
                    app.numCTRLabel.Text = strcat("CTR:
                    ",string(num));
                    app.CTRDropDown.Visible = 'on';
                    shape =
                    app.wd.getWaterFromCTR(list(app.Selection)');
                    if (~isempty(shape))
                        found = cell(size(shape));
                        found(:) = {" not found"};
                        found([shape(:).found])= {""};
                        app.CTRDropDown.Items =
                        cat(2,strcat("CTR: ",string(size(shape,1))) + "
                        " + strcat("not Found
                        :",string(sum(~[shape(:).found]))),...
                        string([shape(:).id]) + " : " +
                        string([shape(:).Area]) +...
                        string(found'));
                        app.SaveCTRMenu.Enable = 'on';
                        app.RemovetoctrButton.Visible = 'on';
                    else
                        app.CTRDropDown.Items = strcat("CTR:
                        ",string(size(shape,1)));
                        app.SaveCTRMenu.Enable = 'off';
                        app.RemovetoctrButton.Visible = 'off';
                    end
                end
                shape = app.wd.getShape();
                if (~isempty(shape))
                    isNew = cell(size(shape));
                    isNew(:) = {""};
                    isNew([shape(:).isNew])= {" New"};
                    app.FindDropDown.Items = cat(2,strcat("Found:
                    ",string(size(shape,2))) + " " +
                    strcat("New:",string(sum([shape(:).isNew]))),...
                    string([shape(:).id]) + " : " + string([shape(:).Area]) +...
                    string(isNew));
                    app.AddtoshapeButton.Visible = 'on';
                else
                    app.FindDropDown.Items = strcat("Found:
                    ",string(size(shape,2)));
                    app.AddtoshapeButton.Visible = 'off';
                end
            elseif (isempty(app.Selection))
                app.wd.removeCTR();
                app.CTRDropDown.Visible = 'off';
                app.numCTRLabel.Text = "CTR : ";
                app.AddtoshapeButton.Visible = 'off';
                app.RemovetoctrButton.Visible = 'off';
                app.SaveCTRMenu.Enable = 'off';
            end
        end
end

% Menu selected function: byCTRfieldsMenu
function byCTRfieldsMenuSelected(app, event)
        if (~isempty(app.ctr))
            list = unique(extractfield(app.ctr,'DESCR'));
            selection = listdlg('Name','Mask from CTR', 'PromptString','Select
            mask element:', ...
            'SelectionMode','multiple','OKString','Select','ListString',list,...
```

```
                    'InitialValue',app.maskSelection);
                    if(~isempty(selection))
                        if (~isempty(app.maskSelection))
                            mask =
                            unique(selection(~ismember(selection,app.maskSele
                            ction)));
                            if (~isempty(mask))
                                msgbox(char("Please, don't touch
                                anything!"),'Wait');
                                app.wd.maskFromCTR(app.ctr,string(list(ma
                                sk)));
                                close;
                                app.maskSelection =
                                sort(cat(2,app.maskSelection,mask));
                            end
                        else
                            app.maskSelection = selection;
                            msgbox(char("Please, don't touch anything!"),'Wait');
                            app.wd.maskFromCTR(app.ctr,string(list(app.maskS
                            election)));
                            close;
                        end
                        if (app.RemovemaskMenu.Enable == "off")
                            app.RemovemaskMenu.Enable = 'on';
                        end
                        if (app.byCTRfieldsMenu_2.Enable == "off")
                            app.byCTRfieldsMenu_2.Enable = 'on';
                        end
                        if (app.ShowMaskMenu.Enable == "off")
                            app.ShowMaskMenu.Enable = 'on';
                        end
                    end
                end
            end

            % Menu selected function: byCTRfieldsMenu_2
            function byCTRfieldsMenu_2Selected(app, event)
                if (~isempty(app.maskSelection))
                    temp = unique(extractfield(app.ctr,'DESCR'));
                    list = unique(temp(app.maskSelection));
                    selection = listdlg('Name','Remove Mask from CTR',
                    'PromptString','Select mask elements:', ...
                    'SelectionMode','multiple','OKString','Select','ListString',list);
                    if (~isempty(selection))
                        msgbox(char("Please, don't touch anything!"),'Wait');
                        app.wd.removeMaskFromCTR(app.ctr,string(list(selection)));
                        close;
                        app.maskSelection(find(find(ismember(temp,list(selection)))))
                        = [];
                        if (isempty(app.maskSelection))
                            app.byCTRfieldsMenu_2.Enable = 'off';
                        end
                        if (isempty(app.wd.getMask()))
                            app.RemovemaskMenu.Enable = 'off';
                            if (app.ShowMaskMenu.Enable == "on")
                                app.ShowMaskMenu.Enable = 'off';
                            end
                        end
                    end
                end
```

```
end

% Menu selected function: ShowMaskMenu
function ShowMaskMenuSelected(app, event)
	i = app.wd.showMask();
	if (~isempty(i))
		imshow(i);
	end
end

% Value changed function: CTRDropDown
function CTRDropDownValueChanged(app, event)
	value = extractBefore(app.CTRDropDown.Value,":");
	if (~strcmpi(value,'Found'))
		[image, area] = app.wd.showCTRElement(str2double(value),30);
		figure('Name',strcat('Element: ',value)+"; Area: " +
		area),imshow(image);
	end
end

% Button pushed function: AddtoshapeButton
function AddtoshapeButtonPushed(app, event)
	element = app.CTRDropDown.Value;
	value = extractBefore(app.CTRDropDown.Value,":");
	if (~strcmpi(value,'CTR') && contains(element,"not found"))
		value = str2double(value);
		app.wd.addCTRElementToShape(value);
		app.showPreview();
		shape = app.wd.getShape();
		isNew = cell(size(shape));
		isNew(:) = {""};
		isNew([shape(:).isNew])= {" New"};
		app.FindDropDown.Items = cat(2,strcat("Found:
		",string(size(shape,2))) + " " +
		strcat("New:",string(sum([shape(:).isNew]))),...
		string([shape(:).id]) + " : " + string([shape(:).Area]) +...
		string(isNew));
		num = str2double(extractAfter(app.numFindLabel.Text,":"));
		app.numFindLabel.Text = strcat("Found: ",string(num+1));
		list = unique(extractfield(app.ctr,'DESCR'));
		crop = app.wd.getCrop();
		shape = {};
		if (~isempty(crop.image))
			shape = app.wd.cropCTR(list(app.Selection)');
		else
			shape = app.wd.getWaterFromCTR(list(app.Selection)');
		end
		found = cell(size(shape));
		found(:) = {" not found"};
		found([shape(:).found])= {""};
		app.CTRDropDown.Items = cat(2,strcat("CTR: ",string(size(shape,1)))
		+ " " + strcat("not Found :",string(sum(~[shape(:).found]))),...
		string([shape(:).id]) + " : " + string([shape(:).Area]) +...
		string(found'));
		app.AddtoshapeButton.Visible = 'on';
		app.RemovetoctrButton.Visible = 'on';
		app.SaveCTRMenu.Enable = 'on';
	end
end
```

```
% Button pushed function: RemovetoctrButton
function RemovetoctrButtonPushed(app, event)
	value = extractBefore(app.CTRDropDown.Value,":");
	if (~strcmpi(value,'CTR'))
		value = str2double(value);
		app.wd.removeToCTR(value);
		list = unique(extractfield(app.ctr,'DESCR'));
		crop = app.wd.getCrop();
		shape = {};
		numtotal = 0;
		numcrop = 0;
		if (~isempty(crop.image))
			shape = app.wd.cropCTR(list(app.Selection)');
			numtotal =
			str2double(extractBetween(app.numCTRLabel.Text,":",";"));
			numcrop =
			str2double(extractAfter(extractAfter(app.numCTRLabel.Text,"
			;"),":"));
		else
			shape = app.wd.getWaterFromCTR(list(app.Selection)');
			numtotal =
			str2double(extractAfter(app.numCTRLabel.Text,":"));
		end
		if (~isempty(shape))
			found = cell(size(shape));
			found(:) = {" not found"};
			found([shape(:).found])= {""};
			app.CTRDropDown.Items = cat(2,strcat("CTR:
			",string(size(shape,1))) + " " + strcat("not Found
			:",string(sum(~[shape(:).found]))),...
			string([shape(:).id]) + " : " + string([shape(:).Area]) +...
			string(found'));
			if (numtotal > 0)
				numtotal = numtotal - 1;
				app.numCTRLabel.Text=strcat("CTR:
				",string(numtotal));
			end
			if (numcrop > 0)
				numcrop = numcrop - 1;
				app.numCTRLabel.Text=strcat(app.numCTRLabel.T
				ext,strcat("; CTR Crop :",string(numcrop)));
			end
			app.AddtoshapeButton.Visible = 'on';
			app.RemovetoctrButton.Visible = 'on';
			app.SaveCTRMenu.Enable = 'on';
		else
			app.CTRDropDown.Items = strcat("CTR:
			",string(size(shape,1)));
			app.AddtoshapeButton.Visible = 'off';
			app.RemovetoctrButton.Visible = 'off';
			app.SaveCTRMenu.Enable = 'off';
		end
	end
end

% Menu selected function: SaveCTRMenu
function SaveCTRMenuSelected(app, event)
	if (~isempty(app.Selection))
		list = unique(extractfield(app.ctr,'DESCR'));
		crop = app.wd.getCrop();
```

```
                        shape = {};
                        if (~isempty(crop.image))
                                shape = app.wd.cropCTR(list(app.Selection)');
                        else
                                shape = app.wd.getWaterFromCTR(list(app.Selection)');
                        end
                        if (~isempty(shape))
                                [filename, path] = uiputfile({'*.shp', 'Shape file (*.shp)'},'Save
                                shapefile', 'shapeCTR.shp');
                                shapewrite(shape,strcat(path,filename));
                        end
                end
        end
end

% App initialization and construction
methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

                % Create WaterDetectorGUIUIFigure
                app.WaterDetectorGUIUIFigure = uifigure;
                app.WaterDetectorGUIUIFigure.Color = [0.902 0.902 0.902];
                app.WaterDetectorGUIUIFigure.Position = [100 100 898 655];
                app.WaterDetectorGUIUIFigure.Name = 'Water Detector GUI';
                app.WaterDetectorGUIUIFigure.Resize = 'off';

                % Create FileMenu
                app.FileMenu = uimenu(app.WaterDetectorGUIUIFigure);
                app.FileMenu.Text = 'File';

                % Create OpenGeotifImageMenu
                app.OpenGeotifImageMenu = uimenu(app.FileMenu);
                app.OpenGeotifImageMenu.MenuSelectedFcn = createCallbackFcn(app,
                @OpenGeotifImageMenuSelected, true);
                app.OpenGeotifImageMenu.Text = 'Open Geotif Image';

                % Create OpenCTRMenu
                app.OpenCTRMenu = uimenu(app.FileMenu);
                app.OpenCTRMenu.MenuSelectedFcn = createCallbackFcn(app,
                @OpenCTRMenuSelected, true);
                app.OpenCTRMenu.Enable = 'off';
                app.OpenCTRMenu.Text = 'Open CTR Shapefile';

                % Create ChangeCTRfieldsMenu
                app.ChangeCTRfieldsMenu = uimenu(app.FileMenu);
                app.ChangeCTRfieldsMenu.MenuSelectedFcn = createCallbackFcn(app,
                @ChangeCTRfieldsMenuSelected, true);
                app.ChangeCTRfieldsMenu.Enable = 'off';
                app.ChangeCTRfieldsMenu.Text = 'Change CTR fields';

                % Create ShapeFileMenu
                app.ShapeFileMenu = uimenu(app.FileMenu);
                app.ShapeFileMenu.MenuSelectedFcn = createCallbackFcn(app,
                @ShapeFileMenuSelected, true);
                app.ShapeFileMenu.Enable = 'off';
                app.ShapeFileMenu.Text = 'Save Shapefile';

                % Create SaveCTRMenu
```

```
app.SaveCTRMenu = uimenu(app.FileMenu);
app.SaveCTRMenu.MenuSelectedFcn = createCallbackFcn(app,
@SaveCTRMenuSelected, true);
app.SaveCTRMenu.Enable = 'off';
app.SaveCTRMenu.Text = 'Save CTR';

% Create ResetAllMenu
app.ResetAllMenu = uimenu(app.FileMenu);
app.ResetAllMenu.MenuSelectedFcn = createCallbackFcn(app,
@ResetAllMenuSelected, true);
app.ResetAllMenu.Enable = 'off';
app.ResetAllMenu.Text = 'Reset All';

% Create ImageMenu
app.ImageMenu = uimenu(app.WaterDetectorGUIUIFigure);
app.ImageMenu.Enable = 'off';
app.ImageMenu.Text = 'Image';

% Create CropMenu
app.CropMenu = uimenu(app.ImageMenu);
app.CropMenu.MenuSelectedFcn = createCallbackFcn(app,
@CropMenuSelected, true);
app.CropMenu.Text = 'Crop';

% Create ResetImageSizeMenu
app.ResetImageSizeMenu = uimenu(app.ImageMenu);
app.ResetImageSizeMenu.MenuSelectedFcn = createCallbackFcn(app,
@ResetImageSizeMenuSelected, true);
app.ResetImageSizeMenu.Enable = 'off';
app.ResetImageSizeMenu.Text = 'Reset Image Size';

% Create MaskMenu
app.MaskMenu = uimenu(app.WaterDetectorGUIUIFigure);
app.MaskMenu.Enable = 'off';
app.MaskMenu.Text = 'Mask';

% Create AddmaskMenu
app.AddmaskMenu = uimenu(app.MaskMenu);
app.AddmaskMenu.Text = 'Add mask';

% Create byshapefileMenu
app.byshapefileMenu = uimenu(app.AddmaskMenu);
app.byshapefileMenu.MenuSelectedFcn = createCallbackFcn(app,
@byshapefileMenuSelected, true);
app.byshapefileMenu.Text = 'by shapefile';

% Create byeditorMenu
app.byeditorMenu = uimenu(app.AddmaskMenu);
app.byeditorMenu.MenuSelectedFcn = createCallbackFcn(app,
@byeditorMenuSelected, true);
app.byeditorMenu.Text = 'by editor';

% Create byCTRfieldsMenu
app.byCTRfieldsMenu = uimenu(app.AddmaskMenu);
app.byCTRfieldsMenu.MenuSelectedFcn = createCallbackFcn(app,
@byCTRfieldsMenuSelected, true);
app.byCTRfieldsMenu.Enable = 'off';
app.byCTRfieldsMenu.Text = 'by CTR fields';

% Create RemovemaskMenu
```

```
app.RemovemaskMenu = uimenu(app.MaskMenu);
app.RemovemaskMenu.Enable = 'off';
app.RemovemaskMenu.Text = 'Remove mask';

% Create byshapefileMenu_2
app.byshapefileMenu_2 = uimenu(app.RemovemaskMenu);
app.byshapefileMenu_2.MenuSelectedFcn = createCallbackFcn(app,
@byshapefileMenu_2Selected, true);
app.byshapefileMenu_2.Text = 'by shapefile';

% Create byeditorMenu_2
app.byeditorMenu_2 = uimenu(app.RemovemaskMenu);
app.byeditorMenu_2.MenuSelectedFcn = createCallbackFcn(app,
@byeditorMenu_2Selected, true);
app.byeditorMenu_2.Text = 'by editor';

% Create byCTRfieldsMenu_2
app.byCTRfieldsMenu_2 = uimenu(app.RemovemaskMenu);
app.byCTRfieldsMenu_2.MenuSelectedFcn = createCallbackFcn(app,
@byCTRfieldsMenu_2Selected, true);
app.byCTRfieldsMenu_2.Enable = 'off';
app.byCTRfieldsMenu_2.Text = 'by CTR fields';

% Create ShowMaskMenu
app.ShowMaskMenu = uimenu(app.MaskMenu);
app.ShowMaskMenu.MenuSelectedFcn = createCallbackFcn(app,
@ShowMaskMenuSelected, true);
app.ShowMaskMenu.Enable = 'off';
app.ShowMaskMenu.Text = 'Show Mask';

% Create MorphologyMenu
app.MorphologyMenu = uimenu(app.WaterDetectorGUIUIFigure);
app.MorphologyMenu.Enable = 'off';
app.MorphologyMenu.Text = 'Morphology';

% Create CleaningMenu
app.CleaningMenu = uimenu(app.MorphologyMenu);
app.CleaningMenu.MenuSelectedFcn = createCallbackFcn(app,
@CleaningMenuSelected, true);
app.CleaningMenu.Text = 'Cleaning';

% Create ClosingMenu
app.ClosingMenu = uimenu(app.MorphologyMenu);
app.ClosingMenu.MenuSelectedFcn = createCallbackFcn(app,
@ClosingMenuSelected, true);
app.ClosingMenu.Text = 'Closing';

% Create CropUIAxes
app.CropUIAxes = uiaxes(app.WaterDetectorGUIUIFigure);
title(app.CropUIAxes, 'Crop')
app.CropUIAxes.CLim = [0 1];
app.CropUIAxes.Box = 'on';
app.CropUIAxes.Position = [261 387 310 251];

% Create IndexUIAxes
app.IndexUIAxes = uiaxes(app.WaterDetectorGUIUIFigure);
title(app.IndexUIAxes, 'Index')
app.IndexUIAxes.CLim = [0 1];
app.IndexUIAxes.Box = 'on';
app.IndexUIAxes.Position = [579 59 302 243];
```

```
% Create MinSliderLabel
app.MinSliderLabel = uilabel(app.WaterDetectorGUIUIFigure);
app.MinSliderLabel.HorizontalAlignment = 'right';
app.MinSliderLabel.Enable = 'off';
app.MinSliderLabel.Position = [627 561 25 15];
app.MinSliderLabel.Text = 'Min';

% Create MinSlider
app.MinSlider = uislider(app.WaterDetectorGUIUIFigure);
app.MinSlider.Limits = [-1 1];
app.MinSlider.MajorTicks = [-1 -0.5 0 0.5 1];
app.MinSlider.MajorTickLabels = {'-1', '-0.5', '0', '0.5', '1'};
app.MinSlider.ValueChangedFcn = createCallbackFcn(app,
@MinSliderValueChanged, true);
app.MinSlider.ValueChangingFcn = createCallbackFcn(app,
@MinSliderValueChanging, true);
app.MinSlider.MinorTicks = [-1 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0 0.1
0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];
app.MinSlider.Enable = 'off';
app.MinSlider.Position = [673 567 150 3];

% Create MaxSliderLabel
app.MaxSliderLabel = uilabel(app.WaterDetectorGUIUIFigure);
app.MaxSliderLabel.HorizontalAlignment = 'right';
app.MaxSliderLabel.Enable = 'off';
app.MaxSliderLabel.Position = [623 477 28 15];
app.MaxSliderLabel.Text = 'Max';

% Create MaxSlider
app.MaxSlider = uislider(app.WaterDetectorGUIUIFigure);
app.MaxSlider.Limits = [-1 1];
app.MaxSlider.MajorTicks = [-1 -0.5 0 0.5 1];
app.MaxSlider.MajorTickLabels = {'-1', '-0.5', '0', '0.5', '1'};
app.MaxSlider.ValueChangedFcn = createCallbackFcn(app,
@MaxSliderValueChanged, true);
app.MaxSlider.ValueChangingFcn = createCallbackFcn(app,
@MaxSliderValueChanging, true);
app.MaxSlider.MinorTicks = [-1 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0 0.1
0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];
app.MaxSlider.Enable = 'off';
app.MaxSlider.Position = [672 483 150 3];

% Create DropDown
app.DropDown = uidropdown(app.WaterDetectorGUIUIFigure);
app.DropDown.Items = {'INDEX', 'NDVI', 'NDWI2', 'NDWI2M'};
app.DropDown.ValueChangedFcn = createCallbackFcn(app,
@DropDownValueChanged, true);
app.DropDown.Enable = 'off';
app.DropDown.BackgroundColor = [0.9412 0.9412 0.9412];
app.DropDown.Position = [654 616 175 22];
app.DropDown.Value = 'INDEX';

% Create ImageUIAxes
app.ImageUIAxes = uiaxes(app.WaterDetectorGUIUIFigure);
title(app.ImageUIAxes, 'Image')
app.ImageUIAxes.CLim = [0 1];
app.ImageUIAxes.Box = 'on';
app.ImageUIAxes.Position = [17 59 227 579];
```

```
% Create MaxField
app.MaxField = uieditfield(app.WaterDetectorGUIUIFigure, 'numeric');
app.MaxField.Limits = [-1 1];
app.MaxField.ValueChangedFcn = createCallbackFcn(app,
@MaxFieldValueChanged, true);
app.MaxField.Enable = 'off';
app.MaxField.Position = [720 501 45 22];

% Create MinField
app.MinField = uieditfield(app.WaterDetectorGUIUIFigure, 'numeric');
app.MinField.Limits = [-1 1];
app.MinField.ValueChangedFcn = createCallbackFcn(app,
@MinFieldValueChanged, true);
app.MinField.Enable = 'off';
app.MinField.Position = [720 584 45 22];

% Create PreviewUIAxes
app.PreviewUIAxes = uiaxes(app.WaterDetectorGUIUIFigure);
title(app.PreviewUIAxes, 'Preview')
app.PreviewUIAxes.CLim = [0 1];
app.PreviewUIAxes.Box = 'on';
app.PreviewUIAxes.Position = [261 59 310 243];

% Create CreateShapeButton
app.CreateShapeButton = uibutton(app.WaterDetectorGUIUIFigure, 'push');
app.CreateShapeButton.ButtonPushedFcn = createCallbackFcn(app,
@CreateShapeButtonPushed, true);
app.CreateShapeButton.Enable = 'off';
app.CreateShapeButton.Position = [594 428 92 22];
app.CreateShapeButton.Text = 'Create Shape';

% Create ShowImageButton
app.ShowImageButton = uibutton(app.WaterDetectorGUIUIFigure, 'push');
app.ShowImageButton.ButtonPushedFcn = createCallbackFcn(app,
@ShowImageButtonPushed, true);
app.ShowImageButton.Enable = 'off';
app.ShowImageButton.Position = [81 16 100 22];
app.ShowImageButton.Text = 'Show Image';

% Create ShowPreviewButton
app.ShowPreviewButton = uibutton(app.WaterDetectorGUIUIFigure, 'push');
app.ShowPreviewButton.ButtonPushedFcn = createCallbackFcn(app,
@ShowPreviewButtonPushed, true);
app.ShowPreviewButton.Enable = 'off';
app.ShowPreviewButton.Position = [366 16 100 22];
app.ShowPreviewButton.Text = 'Show Image';

% Create ShowNDVIButton
app.ShowNDVIButton = uibutton(app.WaterDetectorGUIUIFigure, 'push');
app.ShowNDVIButton.ButtonPushedFcn = createCallbackFcn(app,
@ShowNDVIButtonButtonPushed, true);
app.ShowNDVIButton.Enable = 'off';
app.ShowNDVIButton.Position = [683 16 100 22];
app.ShowNDVIButton.Text = 'Show Image';

% Create ShowCropButton
app.ShowCropButton = uibutton(app.WaterDetectorGUIUIFigure, 'push');
app.ShowCropButton.ButtonPushedFcn = createCallbackFcn(app,
@ShowCropButtonButtonPushed, true);
app.ShowCropButton.Enable = 'off';
```

```
            app.ShowCropButton.Position = [366 345 100 22];
            app.ShowCropButton.Text = 'Show Image';

            % Create numFindLabel
            app.numFindLabel = uilabel(app.WaterDetectorGUIUIFigure);
            app.numFindLabel.Position = [704 432 87 15];
            app.numFindLabel.Text = '';

            % Create FindDropDown
            app.FindDropDown = uidropdown(app.WaterDetectorGUIUIFigure);
            app.FindDropDown.Items = {};
            app.FindDropDown.ValueChangedFcn = createCallbackFcn(app,
            @FindDropDownValueChanged, true);
            app.FindDropDown.Visible = 'off';
            app.FindDropDown.Position = [594 366 138 22];
            app.FindDropDown.Value = {};

            % Create UseasmaskButton
            app.UseasmaskButton = uibutton(app.WaterDetectorGUIUIFigure, 'push');
            app.UseasmaskButton.ButtonPushedFcn = createCallbackFcn(app,
            @UseasmaskButtonPushed, true);
            app.UseasmaskButton.Visible = 'off';
            app.UseasmaskButton.Position = [782 366 99 22];
            app.UseasmaskButton.Text = 'Use as mask';

            % Create numCTRLabel
            app.numCTRLabel = uilabel(app.WaterDetectorGUIUIFigure);
            app.numCTRLabel.Position = [594 396 228 15];
            app.numCTRLabel.Text = '';

            % Create CTRDropDown
            app.CTRDropDown = uidropdown(app.WaterDetectorGUIUIFigure);
            app.CTRDropDown.Items = {};
            app.CTRDropDown.ValueChangedFcn = createCallbackFcn(app,
            @CTRDropDownValueChanged, true);
            app.CTRDropDown.Visible = 'off';
            app.CTRDropDown.Position = [594 324 138 22];
            app.CTRDropDown.Value = {};

            % Create AddtoshapeButton
            app.AddtoshapeButton = uibutton(app.WaterDetectorGUIUIFigure, 'push');
            app.AddtoshapeButton.ButtonPushedFcn = createCallbackFcn(app,
            @AddtoshapeButtonPushed, true);
            app.AddtoshapeButton.Visible = 'off';
            app.AddtoshapeButton.Position = [782 324 35 22];
            app.AddtoshapeButton.Text = 'Add';

            % Create RemovetoctrButton
            app.RemovetoctrButton = uibutton(app.WaterDetectorGUIUIFigure, 'push');
            app.RemovetoctrButton.ButtonPushedFcn = createCallbackFcn(app,
            @RemovetoctrButtonPushed, true);
            app.RemovetoctrButton.Visible = 'off';
            app.RemovetoctrButton.Position = [828 324 53 22];
            app.RemovetoctrButton.Text = 'Remove';
        end

    end

    methods (Access = public)
```

```
        % Construct app
        function app = waterDetectorGUI

            % Create and configure components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.WaterDetectorGUIUIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.WaterDetectorGUIUIFigure)
        end
    end
end
```