AutArch

Summary

AutArch is an AI-assisted workflow capable of creating uniform archaeological datasets from heterogeneous published resources. The implementation provided here takes large and unsorted PDF files as input, and uses neural networks to conduct image processing, object detection, and classification.

Recommended Hardware

AutArch should run most on most systems but the performance of the ML models is heavily depending on the availability of a PyTorch supported graphics card. Please consult the PyTorch manual. The current configuration has been successfully tested on an Nvidia RTX 2060 with 8GB of dedicated GPU memory, AMD Ryzen 9 7900X3D and 64GB of DDR5 RAM. AutArch will fallback to use CPU in case it can not detect a supported GPU.

Installation

This installation guide is intended for ubuntu linux. Windows systems are not natively supported but WSL is known to work. Installation procedures on other linux distributions will be similar.

AutArch requires the following packages:

```
$ sudo apt install libpq-dev postgresql libopencv-dev tesseract-ocr
redis-server libvips42 build-essential zlib1g-dev libncurses5-dev
libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev
wget libbz2-dev
```

To manage the installations of ruby, python and nodejs asdf is recommended. Please refer to the asdf guide in case of any problem.

After installing asdf, install these asdf plugins:

NodeJS

```
$ asdf plugin add nodejs https://github.com/asdf-vm/asdf-nodejs.git
```

Ruby

```
$ asdf plugin add ruby https://github.com/asdf-vm/asdf-ruby.git
```

Python

```
$ asdf plugin-add python
```

Download the autarch folder from seafile (by hovering over it a download button will appear) as a zip file and unzip it in a location of your choice. To install the necessary versions of the languages mentioned above:

```
$ cd autarch
$ asdf install
```

To install ruby dependencies:

```
$ gem install bundler
$ bundle install
```

Change your postgres password:

```
$ sudo su postgres -c psql
> alter user postgres with password '[your-password]';
```

Change the password in 'config/database.yml' for the development database settings to the one you chose.

To compile the C++ extensions:

```
$ cd image_processing
$ ruby extconf.rb
$ make
```

Download the database dump front the provided link and load the dump into your copy of postgres:

For reviewers only:

A database dump, pretrained models and images are provided to reviewers under this link. The database dump and images are located in the `autarch_data` folder. Make sure to download each file of this folder individually due to limitations regarding file size from seafile.

```
$ chmod a+x bin/rails
$ bin/rails db:create

$ cat autarch_dump.gz | gunzip | psql -h localhost -U postgres comove_development
```

For reviewers end

For regular users:

```
$ chmod a+x bin/rails
$ bin/rails db:create

$ bin/rails db:schema:load
```

For regular users end

For reviewers only:

Extract the downloaded images to the images folder:

```
$ zip -F /path/to/autarch_images.zip --out autarch_images_single.zip
$ unzip autarch_images_single.zip -d .
```

For reviewers end

Install js dependencies:

```
$ sudo apt purge cmdtest
$ npm install --global yarn
$ yarn
```

AutArch was tested with PyTorch 2.4.1. Other compatible versions may work as well. For the best performance a GPU is highly recommended.

To install Torch please consult the torch installation guide. Please note that asdf installs Python 3.12.7. Please consult the asdf documentation in case you want your system python installation to be used or you want to use a different version.

```
$ pip install numpy pillow bottle
```

Download the models folder from seafile (by hovering over it a download button will appear) as a zip file. To copy the ML models:

```
$ unzip /path/to/models.zip -d .
```

Running AutArch

You need to start three different components, the rails server, shakapacker and the python ml service.

To run shakapacker:

```
$ chmod a+x bin/shakapacker-dev-server
$ bin/shakapacker-dev-server
```

To run rails:

```
$ bin/rails s
```

To start the python ml service:

```
$ python scripts/torch_service.py
```

To start background jobs:

```
$ bundle exec sidekiq
```

After all the services have successfully loaded, AutArch is accessible under localhost:3000

Team

Name	Contribution	EMail	Github
Kevin Klein	Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualisation, Writing	kkevin@students.uni-mainz.de	github
Antoine Muller	Data curation, Formal analysis, Investigation, Visualisation, Writing		
Alyssa Wohde	Methodology, Writing		
Alexander V. Gorelik	Data curation, Investigation, Resources, Writing		
Volker Heyd	Validation		
Ralf Lämmel	Validation, Writing		
Yoan Diekmann	Writing, Validation		
Maxime Brami	Conceptualization, Funding acquisition, Project administration, Resources, Supervision, Validation, Writing		

Workflow

Publications can be imported under Publications -> Import

After the import has completed, the publication is available under Publications. It is recommended to go to the annotation screen and add all false negative objects.

To review all the graves, go to the grave screen. Use the filter on the top to

select the publications just uploaded. Then click the edit button of the first grave on the list.

Grave Data The ID assigned to the burial by the authors in the source publication is recorded. In case multiple images of the same grave are shown, the software will prevent duplicates in the results using this ID. In this step, the expert also has the option to discard drawings incorrectly classified as a grave.

Site Graves can be assigned to specific sites. Sites can be added here.

Tags Graves can be given arbitrary tags to discern them and allow for filtering in the overview map. Tags can be added here.

Boxes Correcting bounding boxes. The user can manually add, remove or change the bounding box assigned to a specific grave. Potential tasks include selecting a different scale on the page, resizing bounding boxes because they do not fully encapsulate an object or marking north arrows that were initially missed by object detection. During this step, a manual arrow has to be drawn for every skeleton following the spine and pointing towards the skull, which is necessary to determine the orientation of the skeleton in the grave. Several automated steps are then performed. The contours are calculated using the new bounding boxes and the resulting changes in measurements are saved. The orientation of the north arrow and the deposition type of the skeleton are updated using their respective neural network. The analysis of the scale is performed again.

Contours All detected outlines in relation to one particular grave are highlighted, allowing the user, if any issue arises, to return to the previous step and fit, for instance, a manual bounding box around the grave or cross-section to indicate the width, length or depth.

Scale The next step is to validate the scale by checking the text indicating the real-world length of the scale. Once this step is completed, all measurements are updated with the new scale information. In case no individual scale is provided and the publication uses a fixed scale, e.g. all drawings are 1:20, a different screen is shown. In this screen, the actual height of the page (in cm) has to be entered manually, together with the ratio of the drawing. This way, all measurements can be calculated in the absence of a scale and the results are fully compatible with scaled publications.

North Arrow The angle of the north arrow can be adjusted manually based on a preview. In case an arrow is missing in the drawing, this screen will be skipped and size measurements and contours will still be collected without the orientation.

Skeleton Information Finally, the pose of all skeletons has to be validated, which (for now) consists of "unknown", "flexed on the side" or "supine". As described above, a neural network will set the initial body position, but it can be adjusted manually. Further positions could easily be added in the future. "Unknown" is used in cases where skeletal remains are visible, but no position can be identified.

Training the models (reviewers only)

The training data can be found in the `training_data` folder. Note that you can download the whole folder by clicking the download button that appears while hovering over the folder. Put the folder within the `training_data` folder inside the autarch folder.

In case you get an error related to not enough memory being available, reducing the batch size within the training scripts will make it run but the resulting model will potentially differ from the one we supplied.

Other pretrained models are available in the models folder as well. The used models can be swapped out by commenting out the retinanet model and enabling the relevant line to use RCNN or SSD instead (scripts/torch_service.py#20, scripts/train_object_detection#127-148).

To train the models yourself, download the "training_data" folder and put it inside the AutArch folder. To train the object detection network:

```
$ python scripts/train_object_detection.py
```

To train the skeleton deposition type classifier:

```
$ python scripts/train_skeleton_classifier.py
```

To train the arrow angle detection network:

```
$ python scripts/train_arrow_angle_network.py
```

Output

Under publications. Publications can be analyzed using the `analyze link` for every publication. The shown page can be used to compare publications by selecting them from the top. Note that only 4 publications can be compared at the same time.

License

Currently the same copyright applies to this code as to the text and the other supplementary material. With the publication of the article, the AutArch source code will be available publicly under GPL license.