Using fully synthetic training data to automate clinical CT-ACR phantom analysis

Morgan A. Daly BSc*, John M. Hoffman PhD

Department of Radiological Sciences, UCLA, Los Angeles, CA
**mdaly@mednet.ucla.edu*

UCLA Health | David Geffen School of Medicine

RSNA 2023
LEADING THROUGH CHANGE

Using fully synthetic training data to automate clinical CT-ACR phantom analysis

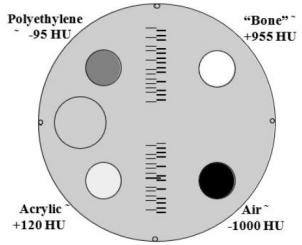
Purpose

We aim to illustrate the feasibility of using a convolutional neural network (CNN) trained **solely on simulated input data** to perform meaningful clinical medical physics tasks. We demonstrate this by automating detection and alignment of the HU module of the CT-ACR phantom (Gammex 464).

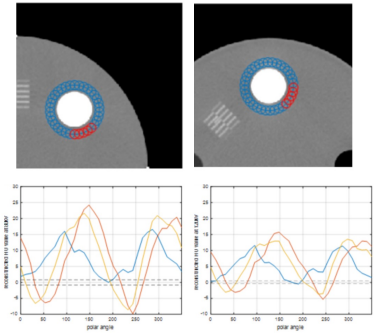
Motivation

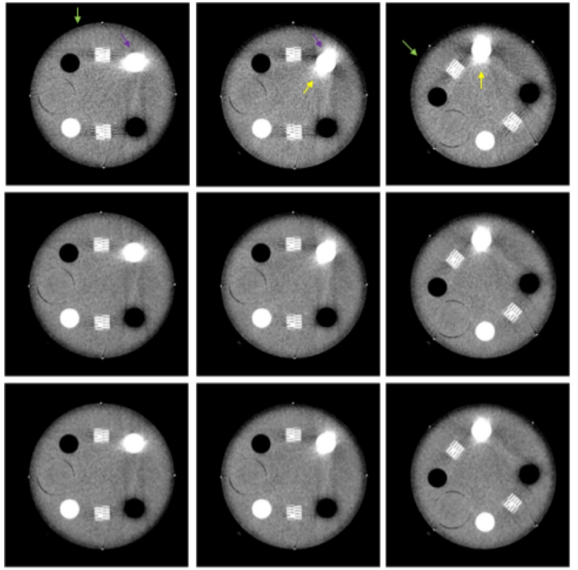
- Training CNNs often requires large amounts of labeled training data
- Labeled data is often expensive and limited in medical domains
- **Simulated data enables large quantities data to be generated quickly, with a known ground truth, and with full control of variations, dataset balance, etc.**

ACR-CT phantom alignment



Schematic of ACR CT phantom (Gammex 464)



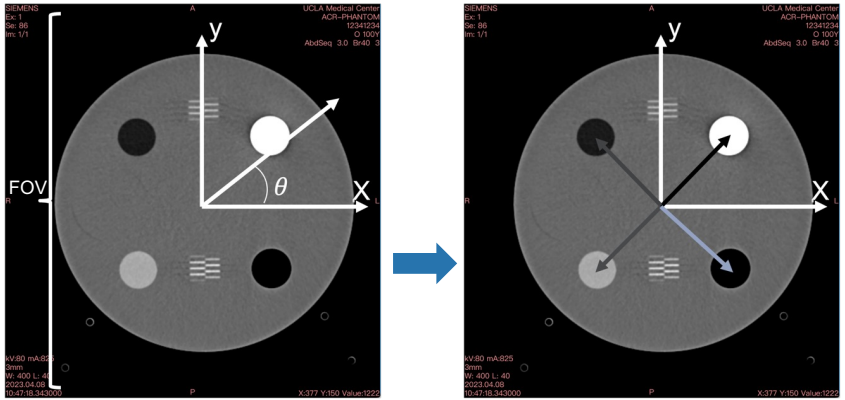


Top row: 80kV images. Middle row: 120kV. Bottom row: 140kV. Left to right: centered, shifted by 4cm, rotated by 45 degrees around the main axis of the phantom then shifted by 4cm. Note the change in artifacts strength (bias magnitude), highlighted by the purple and green arrows. Note also the change in directionality of the bias near the bone insert, highlighted by the yellow arrows. When increasing the tube voltage from 80kV to 140kV, the magnitude of the bias decreases while its variability according to positioning remains the same. C/W:0/40HU. Image reconstruction kernel: Qr40.

Using fully synthetic training data to automate clinical CT-ACR phantom analysis

Task

- Given a single image of an ACR phantom HU module, determine:
  - X,Y position of the phantom
  - FOV used to reconstruct the image
  - Rotation θ of phantom
- **From the above parameters, we can locate the analysis objects within the phantom**



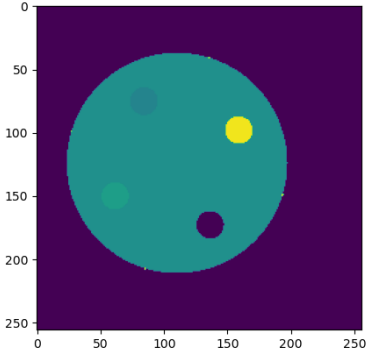
Methodology overview

1. Generate simulated phantom CT image dataset with varying x, y, FOV, and rotation θ
2. Build custom convolutional autoencoder architecture with a latent space constrained to capture the variables of interest
3. Train model using simulation data for train and validation
4. Evaluate model on both a simulation data test set and a real CT image dataset to compare performance

Using fully synthetic training data to automate clinical CT-ACR phantom analysis

Methodology: Phantom Simulator

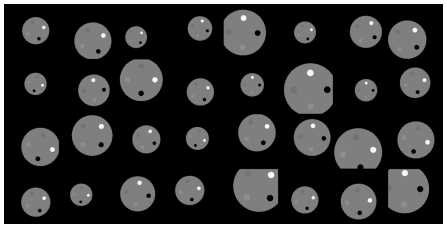
- Python module to generate composable customizable phantoms
- Analysis is inherited and custom to objects
- Inspired by Unity transform/ object hierarchy model
- Can be used to build generic phantoms



Methodology: Dataset

Training set

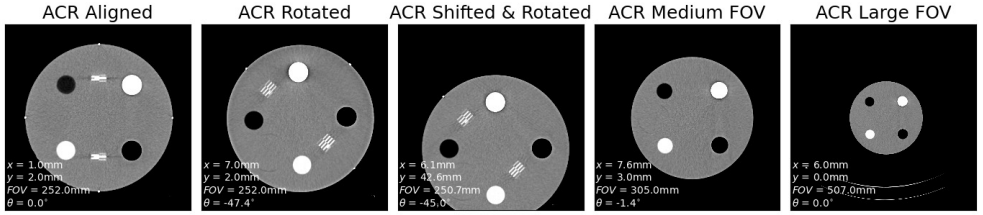
- simulated ACR phantom images
- Alignment was randomly sampled in the ranges:
  - [-50 mm, 50 mm] for x, y
  - [200 mm, 510 mm] for FOV
  - [-50°, 50°] for θ



sample batch of synthetic train images

Test set

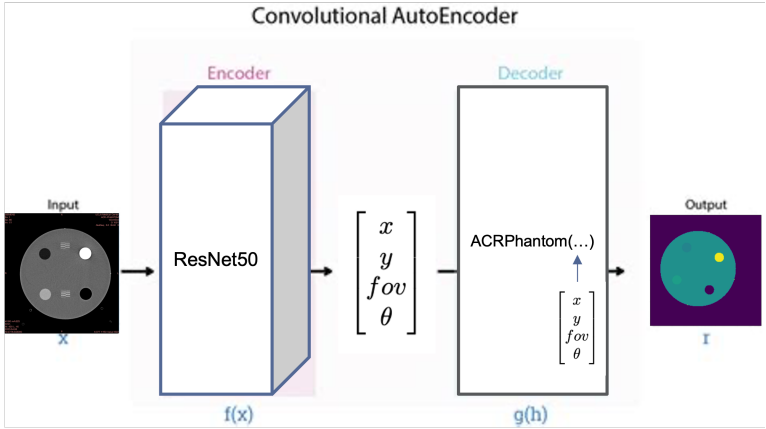
- Five CT images of ACR phantoms with different alignments were manually identified and labeled with ground truth x, y, FOV, θ



Using fully synthetic training data to automate clinical CT-ACR phantom analysis

Methodology: CNN Architecture

- Designed a custom convolutional autoencoder
  - Encoder used ResNet50 backbone
  - Decoder used the Phantom Simulator to produce the simulated ACR image based on the feature vector
- Constrained latent space to capture the variables of interest



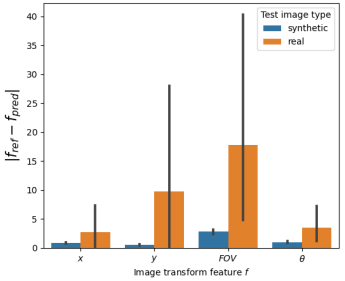
Methodology: CNN Training

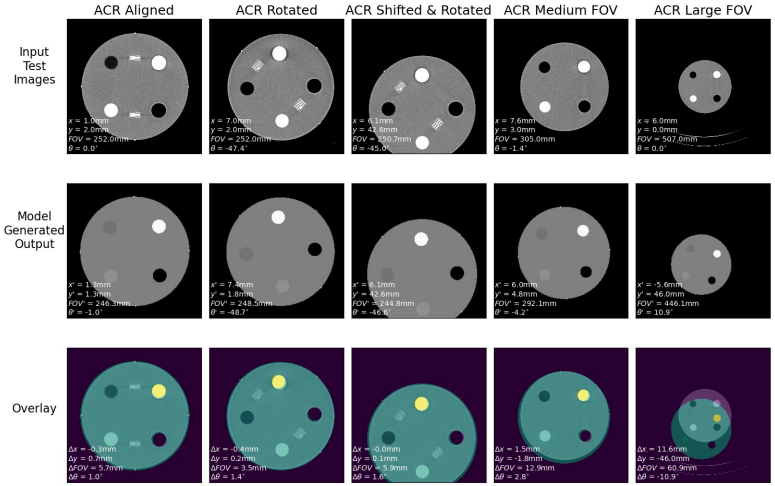
- Considered this a regression problem with a few possible approaches:
  - Optimize feature vector output by encoder
  - Optimize image output by decoder
  - Some combination
- Loss function:
  - mean square error (MSE)
  - mean absolute error (MAE)
- Used Adam optimizer, ReduceLROnPlateau, EarlyStopping
- Performance was assessed based on MAE between predicted and actual feature vectors

Using fully synthetic training data to automate clinical CT-ACR phantom analysis

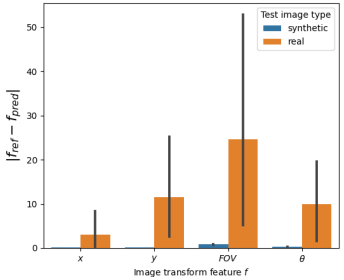
Results

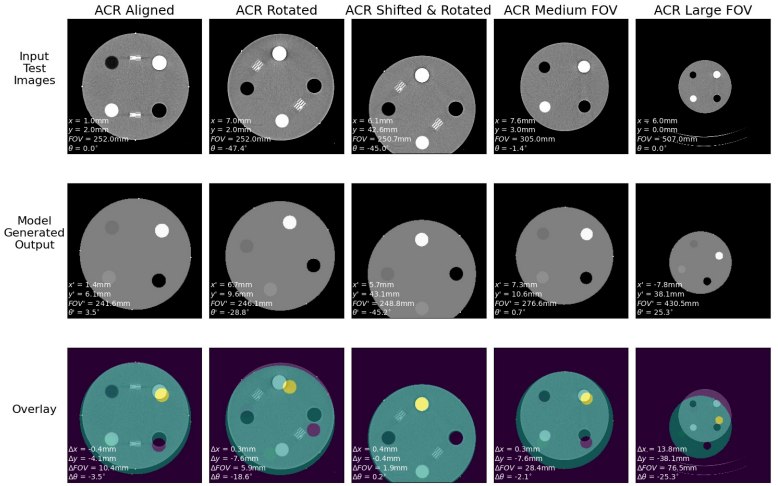
$$L_{encoder} = MAE(z, \hat{z})$$





$$L_{encoder} = MAE(z, \hat{z})$$
$$L_{decoder} = MSE(X, \hat{X})$$
$$\boldsymbol{L = w_1 L_{encoder} + w_2 L_{decoder}}$$





UCLA Health | David Geffen School of Medicine

Using fully synthetic training data to automate clinical CT-ACR phantom analysis

Phantom Simulator Code

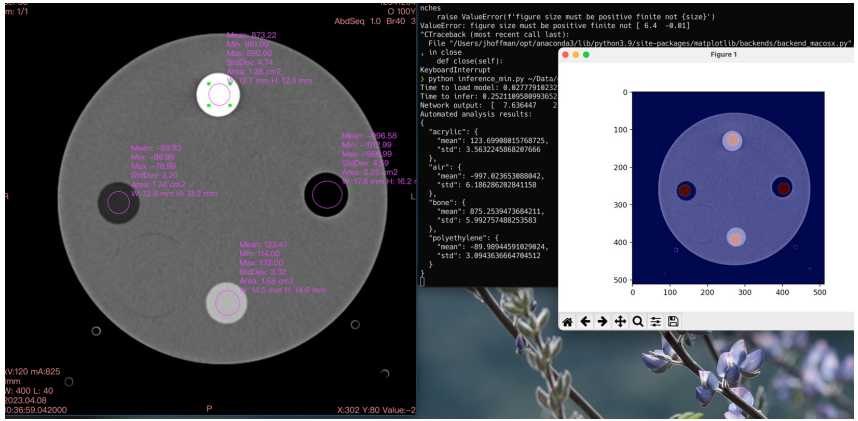
```python
class HUModule(Object):
    def __init__(self, name, x, y, r, HU, limits = None, segmentation_id = 1):
        self.name = name
        self.SetPosition(x,y)
        self.radius = r
        self.HU = HU
        self.limits = limits
        self.segmentation_id = segmentation_id

    def analysis_fn(self, image, render_buff):
        # Draw self into render buff to get a mask, apply to image
        #self.Render(render_buff = render_buff)
        if image is None or render_buff is None:
            return {"dummy_mean": self.HU, "dummy_std": 0}

        image = np.squeeze(image)

        # Need to extract an ROI w/specific radius != radius of object
        # So we cant just have the object render itself into the scene,
        # we need to "draw" an ROI centered on the position.
        x, y = self.GetWorldPosition()
        r = 0.6*self.radius
        mask = CircleROI(x, y, r, render_buff)

        #plt.imshow(image)
        #plt.imshow(mask, cmap = "jet", vmin = 0, vmax = 1, alpha = 0.5)
        #plt.show()

        render_buff.Set(mask, self.segmentation_id)

        vals = image[mask]
        mean = np.mean(vals)
        std = np.std(vals)

        data = {"mean": mean, "std": std, "seg_id": self.segmentation_id}

        if self.limits is not None:
            data["allowed_range"] = self.limits
            passes = (mean >= self.limits[0]) and (mean <= self.limits[1])
            if passes:
                data["result"] = "PASS"
            else:
                data["result"] = "FAIL"

        return data
```

```python
class ACRPhantom(Object):
    acrylic = None
    bone = None
    polyethylene = None
    air = None
    h20 = None

    def __init__(self):

        super().__init__("acr_phantom")

        self.radius = 100;
        self.HU = 0;

        # Alignment beads
        bead_1 = Object("alignment bead 1")
        bead_1.radius = 1
        bead_1.HU = 1000
        bead_1.SetParent(self)
        bead_1.SetPosition(100, 0)

        bead_2 = Object("alignment bead 2")
        bead_2.radius = 1
        bead_2.HU = 1000
        bead_2.SetParent(self)
        bead_2.SetPosition(-100, 0)

        bead_3 = Object("alignment bead 3")
        bead_3.radius = 1
        bead_3.HU = 1000
        bead_3.SetParent(self)
        bead_3.SetPosition(0, 100)

        bead_4 = Object("alignment bead 4")
        bead_4.radius = 1
        bead_4.HU = 1000
        bead_4.SetParent(self)
        bead_4.SetPosition(0, -100)

        # Main objects
        acrylic = HUModule("acrylic", -45, 45, 12.5, 120, limits = [110, 135], segmentation_id = 1)
        acrylic.SetParent(self)

        polyeth = HUModule("polyethylene", -45, -45, 12.5, -95, limits = [-107, -84], segmentation_id = 2)
        polyeth.SetParent(self)

        air = HUModule("air", 45, 45, 12.5, -1000, limits = [-1005, -970], segmentation_id = 3)
        air.SetParent(self)

        bone = HUModule("bone", 45, -45, 12.5, 955, limits = [850, 970], segmentation_id = 4)
        bone.SetParent(self)

        water = HUModule("water", -60, 0, 12.5, 0, limits = [-7, 7], segmentation_id = 5)
        water.SetParent(self)
```

Using fully synthetic training data to automate clinical CT-ACR phantom analysis

Conclusions

- ACR phantom generation + analysis tool
  - Automated, customizable, generalizable
- Simulated data *can* translate to real clinical data
  - Need to explore more use cases and more challenging tasks
  - Domain transfer here likely succeeded due to simplicity of the task
- Method produces meaningful feature vectors that offer explainability and interpretability to the CNN
- Might be beneficial for tasks that use custom phantoms as an alternative to building a custom rigid registration tool



Example automated phantom analysis using our model