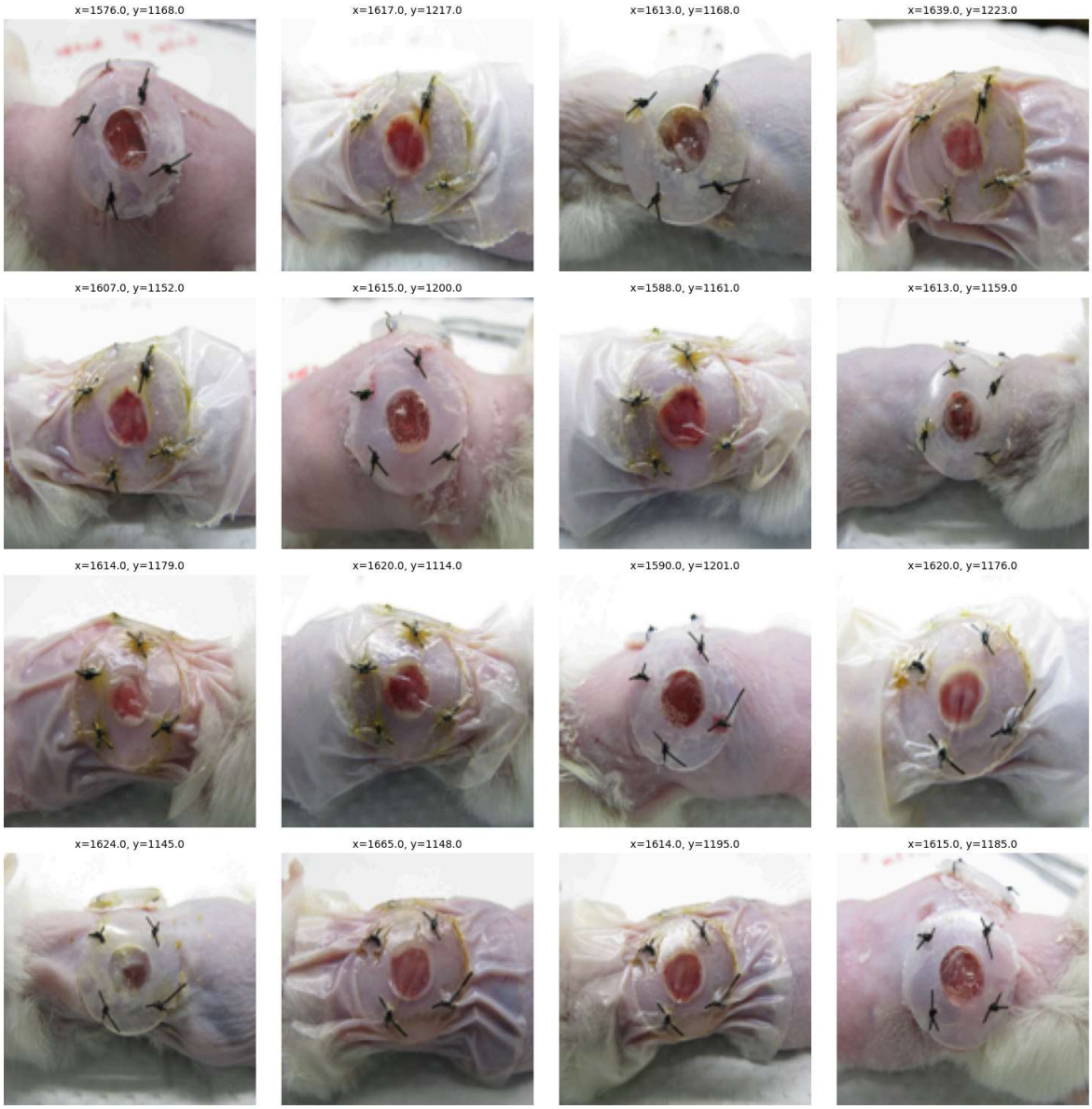
Random Forest

In [1]:
```
from Utils import *
```

In [2]:
```
# load training data
trainImages, trainOutputxi, trainOutputyi, trainOutputxw, trainOutputyw = readIm
```

In [3]:
```
# normalizing pixel values
trainImagesNormalized = np.array(trainImages) / 255.0
```

In [4]:
```
# show some training images
titles = [f"x={x}, y={y}" for x, y in zip(trainOutputxi[:16], trainOutputyi[:16]
display_images(trainImages[:16], rows=4, cols=4, titles=titles)
```



In [5]:
```
# Flatten the images for the linear regression model
trainImages_flattened = [img.flatten() for img in trainImagesNormalized]
trainImages_flattened = np.array(trainImages_flattened)
```

In [6]:
```
# Prepare the Outputs
trainOutputs = np.array([
    trainOutputxi,  # x values
    trainOutputyi,  # y values
    trainOutputxw,  # x_width values
    trainOutputyw   # y_width values
], dtype=float).T  # Transpose to align correctly
```

In [7]:
```
# Split the Data
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(
    trainImages_flattened, trainOutputs, test_size=0.2, random_state=42
)
```

In [8]:
```
# Import Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

# Initialize the Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=25, max_depth=5, n_jobs=-1, random

# Train the Random Forest model
rf_model.fit(X_train, y_train)
```

Out[8]:

RandomForestRegressor

```
RandomForestRegressor(max_depth=5, n_estimators=25, n_jobs=-1, random_state=4
2)
```

In [9]:
```
# Evaluate the Model
from sklearn.metrics import mean_squared_error

# Predict on the validation set
y_pred = rf_model.predict(X_val)

# Calculate MSE for validation
mse = mean_squared_error(y_val, y_pred)
print(f"Mean Squared Error on Validation Set: {mse:.2f}")
```

```
Mean Squared Error on Validation Set: 1643.74
```

In [10]:
```
# Save the Model
import joblib

# Save the trained model to a file
joblib.dump(rf_model, "random_forest_models/random_forest_model.pkl")
```

Out[10]:
```
['random_forest_models/random_forest_model.pkl']
```

Improving the model

In [11]:
```
# Dimensionality Reduction
from sklearn.decomposition import PCA

pca = PCA(n_components=20)  # Try reducing dimensions
```

```
X_train_reduced = pca.fit_transform(X_train)
X_val_reduced = pca.transform(X_val)

rf_model.fit(X_train_reduced, y_train)
```

Out[11]:

RandomForestRegressor

```
RandomForestRegressor(max_depth=5, n_estimators=25, n_jobs=-1, random_state=42)
```

In [12]:
```
# Save the scaler
joblib.dump(pca, "random_forest_pcas/pca.pkl")
```

Out[12]:
```
['random_forest_pcas/pca.pkl']
```

In [13]:
```
# Predict on the validation set
y_pred = rf_model.predict(X_val_reduced)

# Calculate MSE for validation
mse = mean_squared_error(y_val, y_pred)
print(f"Mean Squared Error on Validation Set: {mse:.2f}")
```

```
Mean Squared Error on Validation Set: 1522.20
```

The MSE reduced from 1643.74 to 1527.88, so PCA works!

Now, try GridSearchCV to improve our model

In [14]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

# Define the parameter grid
param_grid = {
    'n_estimators': [15, 20, 25, 50],
    'max_depth': [1, 5, 10, 15],
    'min_samples_leaf': [1, 2, 4, 8],
    'min_samples_split': [2, 5, 10, 20],
}
```

In [15]:
```
# initailize the model
rf_grcv_model = RandomForestRegressor(random_state=42, n_jobs=-1)
```

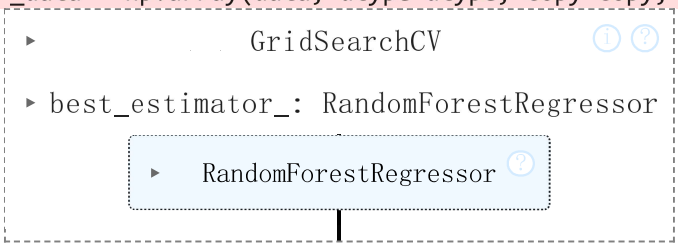
In [16]:
```
# Initialize GridSearchCV
grid_search = GridSearchCV(
    estimator=rf_grcv_model,
    param_grid=param_grid,
    scoring='neg_mean_squared_error',  # Negative MSE as scoring metric
    cv=5,  # 5-fold cross-validation
    verbose=2,
    n_jobs=-1  # Parallel processing
)
```

In [17]:
```
# Fit the grid search to the data
grid_search.fit(X_train_reduced, y_train)
```

```
Fitting 5 folds for each of 256 candidates, totalling 1280 fits
```

```
D:\nur\developers\Anaconda_envs\envs\pytorch\Lib\site-packages\numpy\ma\core.py:2
881: RuntimeWarning: invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
```

Out[17]:



In [18]:
```python
import pandas as pd
import joblib

# Get cross-validation results as a DataFrame
cv_results = pd.DataFrame(grid_search.cv_results_)

# Sort by the scoring metric (neg_mean_squared_error) in descending order
cv_results_sorted = cv_results.sort_values(by='mean_test_score', ascending=False

# Select the top 5 models
top_5_models = cv_results_sorted.head(5)
```

In [19]:
```python
# Save model with score metadata
for idx, (_, row) in enumerate(top_5_models.iterrows()):
    params = row['params']
    score = row['mean_test_score']
    model = RandomForestRegressor(**params, random_state=42, n_jobs=-1)
    model.fit(X_train_reduced, y_train)
    model_filename = f"random_forest_model_top_{idx+1}_score_{-score:.2f}.pkl"
    joblib.dump(model, f"random_forest_models/{model_filename}")
    print(f"Saved: {model_filename} with score: {-score:.2f}")
```

```
Saved: random_forest_model_top_1_score_2579.78.pkl with score: 2579.78
Saved: random_forest_model_top_2_score_2587.91.pkl with score: 2587.91
Saved: random_forest_model_top_3_score_2589.07.pkl with score: 2589.07
Saved: random_forest_model_top_4_score_2601.39.pkl with score: 2601.39
Saved: random_forest_model_top_5_score_2608.72.pkl with score: 2608.72
```

In [20]:
```python
best_para = grid_search.best_params_
print(f"Best parameters found:{best_para}")

# Get the best model
best_model = grid_search.best_estimator_
```

```
Best parameters found:{'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_spli
t': 5, 'n_estimators': 50}
```

In [21]:
```python
# Predict on the validation set
y_pred = best_model.predict(X_val_reduced)

# Calculate MSE for the validation set
mse = mean_squared_error(y_val, y_pred)
print(f"Mean Squared Error on Validation Set with Best Alpha: {mse:.2f}")
```

```
Mean Squared Error on Validation Set with Best Alpha: 1597.98
```