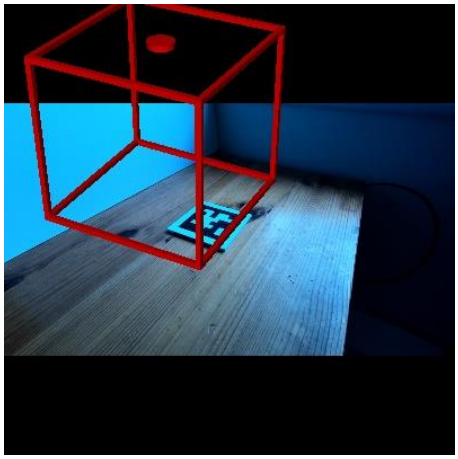
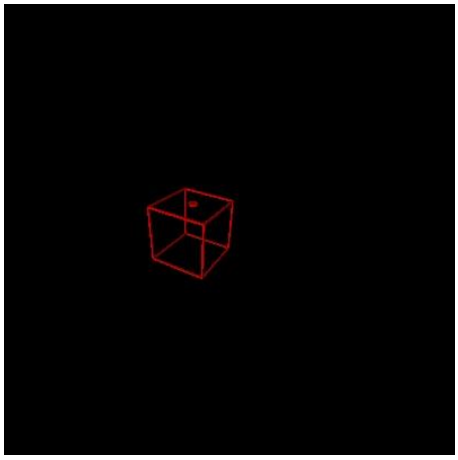
Edits to vtk_aruco_app.py:

Visual summary of results from incremental edits to the point of achieving a projection matrix for a successful overlay.

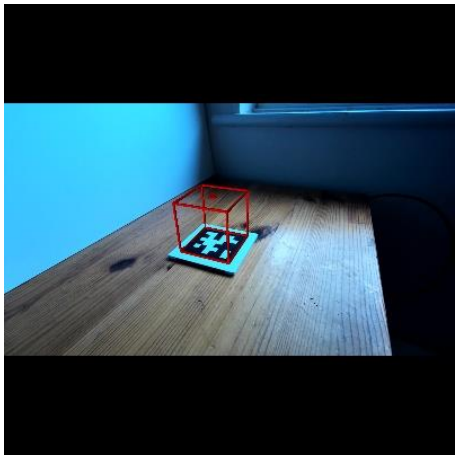
vtk_aruco_app.py



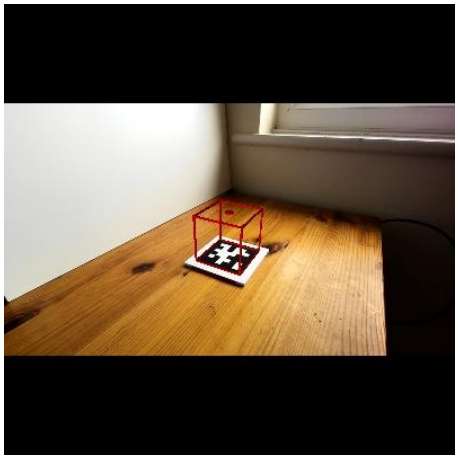
vtk_aruco_app_fix_attempt_1.py



vtk_aruco_app_fix_attempt_2.py



vtk_aruco_app_fix_attempt_3.py



Please ignore white balance issues

vtk_aruco_app.py

```
class OverlayApp(OverlayBaseApp):
    """Inherits from OverlayBaseApp, and adds methods to
    detect aruco tags and move the model to follow."""

    def __init__(self, image_source, backend, dims, focus):
        """override the default constructor to set up video source
        with specific backend and  manual focus and set up sksurgeryarucotracker"""
        self.vtk_overlay_window = VTKOverlayWindow()
        self.video_source = VideoBackend(image_source, backend, dims, focus)
        self.update_rate = 30
        self.img = None
        self.timer = None
        self.save_frame = None

        #we'll use SciKit-SurgeryArUcoTracker to estimate the pose of the
        #visible ArUco tag relative to the camera. We use a dictionary to
        #configure SciKit-SurgeryArUcoTracker

        ar_config = {
            "tracker type": "aruco",
            #Set to none, to share video source with OverlayBaseApp
            "video source": 'none',
            "debug": False,
            #the aruco tag dictionary to use. DICT_4X4_50 will work with
            #../tags/aruco_4by4_0.pdf
            "aruco dictionary" : 'DICT_APRILTAG_36h11',
            "marker size": 80, # in mm
            #We need a calibrated camera. For now let's just use a
            #a hard coded estimate. Maybe you could improve on this.
            "camera projection": numpy.array([[667, 0.0, 640.0],
                                              [0.0, 667, 360.0],
                                              [0.0, 0.0, 1.0]],
                                             dtype=numpy.float32),
            "camera distortion": numpy.zeros((1, 4), numpy.float32)
            }
        self.tracker = ArUcoTracker(ar_config)

        self.counter = 0

        self.tracker.start_tracking()
```

Changes discussed in this set of notes made to the constructor for the OverlayApp shown here for the unedited code for vtk_aruco_app.y found in skstutorial01. During the investigation changes were also made to VTKOverlayWindow() but the final version of the constructor for OverlayApp, shown on slide 10, will work with an unedited version of this class.

Terminal Debug Output - vtk_aruco_app.py

```
(AugReal) F:\Dropbox\EC_SEMIS\2021\1aaa_AR_augmentedReality\scikit-surgerytutorial01-ed3\sksurgerytutorial01>C:/Users/EdMC/miniconda3/envs/AugReal/python.exe
f:/Dropbox/EC_SEMIS/2021/1aaa_AR_augpy


o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o
vtk_overlay_window : __init__ ; set_video_image
o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o



vivivvivivivivivivivivivivivivivivivivivivivvivi
vtk_overlay_window : set_video_image
vtk_overlay_window : set_video_image COMPLETED
vivivvivivivivivivivivivivivivivivivivivivivvivi

############################################
vtk_overlay_window : resizeEvent


-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
vtk_overlay_window : __update_video_image_camera
vtk_overlay_window : __update_video_image_camera COMPLETED
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix


vtk_overlay_window : Camera Matrix is None -> Projection Matrix NOT Updated
vtk_overlay_window : resizeEvent COMPLETED
############################################

////////////////////////////////////////////////////////////
OverlayApp : Update -- Frame Counter = 0
////////////////////////////////////////////////////////////


camera_model : set_camera_pose


vivivvivivivivivivivivivivivivivivivivivivivvivi
vtk_overlay_window : set_video_image
input shape not equal to input_image.shape


-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
vtk_overlay_window : __update_video_image_camera
vtk_overlay_window : __update_video_image_camera COMPLETED
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix


vtk_overlay_window : Camera Matrix is None -> Projection Matrix NOT Updated
vtk_overlay_window : set_video_image COMPLETED
vivivvivivivivivivivivivivivivivivivivivivivvivi

////////////////////////////////////////////////////////////
OverlayApp : Update -- Frame Counter = 1
////////////////////////////////////////////////////////////


camera_model : set_camera_pose


vivivvivivivivivivivivivivivivivivivivivivivvivi
vtk_overlay_window : set_video_image
vtk_overlay_window : set_video_image COMPLETED
vivivvivivivivivivivivivivivivivivivivivivivvivi

////////////////////////////////////////////////////////////
OverlayApp : Update -- Frame Counter = 2
////////////////////////////////////////////////////////////


camera_model : set_camera_pose


vivivvivivivivivivivivivivivivivivivivivivivvivi
vtk_overlay_window : set_video_image
vtk_overlay_window : set_video_image COMPLETED
vivivvivivivivivivivivivivivivivivivivivivivvivi
```

App begins running before the projection matrix has been set correctly.

Large error in overlay.

vtk_aruco_app_fix_attempt_1.py

```
class OverlayApp(OverlayBaseApp):
    """Inherits from OverlayBaseApp, and adds methods to
    detect aruco tags and move the model to follow."""

    def __init__(self, image_source, backend, dims, focus):
        """override the default constructor to set up video source
        with specific backend and  manual focus and set up sksurgeryarucotracker"""
        self.vtk_overlay_window = VTKOverlayWindow()
        self.video_source = VideoBackend(image_source, backend, dims, focus)
        self.update_rate = 30
        self.img = None
        self.timer = None
        self.save_frame = None

        #EC Addition bug fix - attempt 1
        _, image = self.video_source.read() #image acquired from camera to set shape

        #EC Addition to set the camera matrix in vtk_overlay
        self.camMat = numpy.array([[667, 0.0, 640.0],
                                   [0.0, 667, 360.0],
                                   [0.0, 0.0, 1.0]],
                                   dtype=numpy.float32)

        self.vtk_overlay_window.set_camera_matrix(self.camMat, image.shape)

        #we'll use SciKit-SurgeryArUcoTracker to estimate the pose of the
        #visible ArUco tag relative to the camera. We use a dictionary to
        #configure SciKit-SurgeryArUcoTracker

        ar_config = {
            "tracker type": "aruco",
            #Set to none, to share video source with OverlayBaseApp
            "video source": 'none',
            "debug": False,
            #the aruco tag dictionary to use. DICT_4X4_50 will work with
            #../tags/aruco_4by4_0.pdf
            "aruco dictionary" : 'DICT_APRILTAG_36h11',
            "marker size": 80, # in mm
            #We need a calibrated camera. For now let's just use a
            #a hard coded estimate. Maybe you could improve on this.
            "camera projection": self.camMat,
            "camera distortion": numpy.zeros((1, 4), numpy.float32)
            }
        self.tracker = ArUcoTracker(ar_config)

        self.counter = 0

        self.tracker.start_tracking()
```

In the first attempt to fix the projection error I tried to set the camera matrix for the instance of VTKOverlayWindow. I was using an edited version of set_camera_matrix, hence the method is taking image.shape as an input.

Terminal Debug Output - vtk_aruco_app_fix_attempt_1.py

```
(AugReal) F:\Dropbox\EC_SEMIS\2021\1aaa_AR_augmentedReality\scikit-surgerytutorial01-ed3\sksurgerytutorial01>C:/Users/EdMC/miniconda3/envs/AugReal/python.exe f:/Dropbox/EC_SEMIS/2021/1aaa_AR_augmentedReality/scikit-surgerytutorial01-ed3/sksurgerytutorial01/vtk_aruco_app_fix_attempt_1.py


o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o
vtk_overlay_window : __init__ ; set_video_image
o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o



vivivviviviviviviviviviviviviviviviviviviviviviviviivvivi
vtk_overlay_window : set_video_image
vtk_overlay_window : set_video_image COMPLETED
vivivviviviviviviviviviviviviviviviviviviviviviviviivvivi
```

```
mtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmxtmtxmtxmtxm
vtk_overlay_window : set_camera_matrix


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix


camera_model : compute_projection_matrix
camera_model : set_camera_intrinsics


Input Shape [0],[1] =
(720, 1280)


camera_model : compute_scissor


Overlay Window : width , height =
(640, 480)


camera_model : compute_viewport


opengl_mat =
vtkMatrix4x4 (00000149EFACB3C0)
  Debug: Off
  Modified Time: 1800
  Reference Count: 1
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0


vtk_mat =
vtkMatrix4x4 (00000149FC7D9130)
  Debug: Off
  Modified Time: 1852
  Reference Count: 2
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0


Viewport
(0.0, 0.125, 1.0, 0.875)


vtk_overlay_window : __update_projection_matrix COMPLETED
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


vtk_overlay_window : set_camera_matrix COMPLETED
mtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmxtmtxmtxmtxm
```

```
############################################
vtk_overlay_window : resizeEvent


-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
vtk_overlay_window : __update_video_image_camera
vtk_overlay_window : __update_video_image_camera COMPLETED
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix


camera_model : compute_projection_matrix
camera_model : set_camera_intrinsics


Input Shape [0],[1] =
(720, 1280)


camera_model : compute_scissor


Overlay Window : width , height =
(400, 400)


camera_model : compute_viewport


opengl_mat =
vtkMatrix4x4 (00000149FCCB9F30)
  Debug: Off
  Modified Time: 3848
  Reference Count: 1
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0


vtk_mat =
vtkMatrix4x4 (00000149FC7D9130)
  Debug: Off
  Modified Time: 3898
  Reference Count: 2
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0


Viewport
(0.0, 0.2175, 1.0, 0.78)


vtk_overlay_window : __update_projection_matrix COMPLETED
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


vtk_overlay_window : resizeEvent COMPLETED
############################################
```

```
//////////////////////////////////////////////////////////////
OverlayApp : Update -- Frame Counter = 0
//////////////////////////////////////////////////////////////


camera_model : set_camera_pose


vivivviviviviviviviviviviviviviviviviviviviviviviviivvivi
vtk_overlay_window : set_video_image
vtk_overlay_window : set_video_image COMPLETED
vivivviviviviviviviviviviviviviviviviviviviviviviviivvivi
```

Projection matrix set correctly but background (video) not set as visible so foreground projected onto blank (black) background.

vtk_aruco_app_fix_attempt_2.py

```
class OverlayApp(OverlayBaseApp):
    """Inherits from OverlayBaseApp, and adds methods to
    detect aruco tags and move the model to follow."""

    def __init__(self, image_source, backend, dims, focus):
        """override the default constructor to set up video source
        with specific backend and  manual focus and set up sksurgeryarucotracker"""
        self.vtk_overlay_window = VTKOverlayWindow()
        self.video_source = VideoBackend(image_source, backend, dims, focus)
        self.update_rate = 30
        self.img = None
        self.timer = None
        self.save_frame = None

        #EC Addition bug fix - attempt 1
        _, image = self.video_source.read() #image acquired from camera to set shape

        #EC Addition bug fix - attempt 2
        self.vtk_overlay_window.init_set_video_image(image.shape)

        #EC Addition to set the camera matrix in vtk_overlay
        self.camMat = numpy.array([[667, 0.0, 640.0],
                                   [0.0, 667, 360.0],
                                   [0.0, 0.0, 1.0]],
                                   dtype=numpy.float32)

        self.vtk_overlay_window.set_camera_matrix(self.camMat, image.shape)

        #we'll use SciKit-SurgeryArUcoTracker to estimate the pose of the
        #visible ArUco tag relative to the camera. We use a dictionary to
        #configure SciKit-SurgeryArUcoTracker

        ar_config = {
            "tracker type": "aruco",
            #Set to none, to share video source with OverlayBaseApp
            "video source": 'none',
            "debug": False,
            #the aruco tag dictionary to use. DICT_4X4_50 will work with
            #../tags/aruco_4by4_0.pdf
            "aruco dictionary" : 'DICT_APRILTAG_36h11',
            "marker size": 80, # in mm
            #We need a calibrated camera. For now let's just use a
            #a hard coded estimate. Maybe you could improve on this.
            "camera projection": self.camMat,
            "camera distortion": numpy.zeros((1, 4), numpy.float32)
            }
        self.tracker = ArUcoTracker(ar_config)

        self.counter = 0

        self.tracker.start_tracking()
```

Note:

- Call to init_set_video image, a method added to VTKOverlayWindow() by EC.

Terminal Debug Output - vtk_aruco_app_fix_attempt_2.py

```
(AugReal) F:\Dropbox\EC_SEMIS\2021\1aaa_AR_augmentedReality\scikit-surgerytutorial01-ed3\sksurgerytutorial01>C:/Users/EdMC/miniconda3/envs/AugReal/python.exe f:/Dropbox/EC_SEMIS/2021/1aaa_AR_augmentedReality/scikit-surgerytutorial01-ed3/sksurgerytutorial01/vtk_aruco_app_fix_attempt_2.py

o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o
vtk_overlay_window : __init__ ; set_video_image
o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o

vivivvivivivivivivivivivivivivivivivvivi
vtk_overlay_window : set_video_image
vtk_overlay_window : set_video_image COMPLETED
vivivvivivivivivivivivivivivivivivivvivi

ECECECECECECECECECECECECECECECECECECECECECECECECEC
vtk_overlay_window : init_set_video_image

-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
vtk_overlay_window : __update_video_image_camera
vtk_overlay_window : __update_video_image_camera COMPLETED
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix

vtk_overlay_window : Camera Matrix is None -> Projection Matrix NOT Updated

vtk_overlay_window : init_set_video_image COMPLETED
ECECECECECECECECECECECECECECECECECECECECECECECECEC

mtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxm
vtk_overlay_window : set_camera_matrix

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix

camera_model : compute_projection_matrix
camera_model : set_camera_intrinsics

Input Shape [0],[1] =
(720, 1280)

camera_model : compute_scissor

Overlay Window : width , height =
(640, 480)

camera_model : compute_viewport

opengl_mat =
vtkMatrix4x4 (00000191A5428260)
  Debug: Off
  Modified Time: 1911
  Reference Count: 1
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0

vtk_mat =
vtkMatrix4x4 (00000191C9F09100)
  Debug: Off
  Modified Time: 1963
  Reference Count: 2
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0

Viewport
(0.0, 0.125, 1.0, 0.875)

vtk_overlay_window : __update_projection_matrix COMPLETED
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

vtk_overlay_window : set_camera_matrix COMPLETED
mtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxm
```

```
##########################################
vtk_overlay_window : resizeEvent

-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
vtk_overlay_window : __update_video_image_camera
vtk_overlay_window : __update_video_image_camera COMPLETED
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix

camera_model : compute_projection_matrix
camera_model : set_camera_intrinsics

Input Shape [0],[1] =
(720, 1280)

camera_model : compute_scissor

Overlay Window : width , height =
(400, 400)

camera_model : compute_viewport

opengl_mat =
vtkMatrix4x4 (00000191CA395DE0)
  Debug: Off
  Modified Time: 3853
  Reference Count: 1
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0

vtk_mat =
vtkMatrix4x4 (00000191C9F09100)
  Debug: Off
  Modified Time: 3903
  Reference Count: 2
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0

Viewport
(0.0, 0.2175, 1.0, 0.78)

vtk_overlay_window : __update_projection_matrix COMPLETED
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

vtk_overlay_window : resizeEvent COMPLETED
##########################################
```

```
//////////////////////////////////////////////////////////////
OverlayApp : Update -- Frame Counter = 0
//////////////////////////////////////////////////////////////

camera_model : set_camera_pose
```

Projection matrix and background image set correctly but additional method `init_set_video_image` inside of EC edit to VTKOverlayWindow used.

vtk_aruco_app_fix_attempt_3.py

```python
class OverlayApp(OverlayBaseApp):
    """Inherits from OverlayBaseApp, and adds methods to
    detect aruco tags and move the model to follow."""

    def __init__(self, image_source, backend, dims, focus):
        """override the default constructor to set up video source
        with specific backend and  manual focus and set up sksurgeryarucotracker"""
        self.vtk_overlay_window = VTKOverlayWindow()
        self.video_source = VideoBackend(image_source, backend, dims, focus)
        self.update_rate = 30
        self.img = None
        self.timer = None
        self.save_frame = None

        #EC Addition bug fix - attempt 1
        _, image = self.video_source.read() #image acquired from camera to set shape

        #EC Addition bug fix - attempt 2
        #self.vtk_overlay_window.init_set_video_image(image.shape)

        #EC Addition bug fix - attempt 3
        self.vtk_overlay_window.set_video_image(image)

        #EC Addition to set the camera matrix in vtk_overlay
        self.camMat = numpy.array([[667, 0.0, 640.0],
                                   [0.0, 667, 360.0],
                                   [0.0, 0.0, 1.0]],
                                   dtype=numpy.float32)

        self.vtk_overlay_window.set_camera_matrix(self.camMat, image.shape)

        #we'll use SciKit-SurgeryArUcoTracker to estimate
        #visible ArUco tag relative to the camera. We use
        #configure SciKit-SurgeryArUcoTracker

        ar_config = {
            "tracker type": "aruco",
            #Set to none, to share video source with OverlayBaseApp
            "video source": 'none',
            "debug": False,
            #the aruco tag dictionary to use. DICT_4X4_50 will work with
            #../tags/aruco_4by4_0.pdf
            "aruco dictionary" : 'DICT_APRILTAG_36h11',
            "marker size": 80, # in mm
            #We need a calibrated camera. For now let's just use a
            #a hard coded estimate. Maybe you could improve on this.
            "camera projection": self.camMat,
            "camera distortion": numpy.zeros((1, 4), numpy.float32)
            }
        self.tracker = ArUcoTracker(ar_config)

        self.counter = 0

        self.tracker.start_tracking()
```

Note:

- Note that although it doesn't use any additional methods in VTKOverlayWindow it still won't work using the version installed using pip as it relies on modifications to the method set_camera_matrix.

Terminal Debug Output - vtk_aruco_app_fix_attempt_3.py

```
(AugReal) F:\Dropbox\EC_SEMIS\2021\1aaa_AR_augmentedReality\scikit-surgerytutorial01-ed3\sksurgerytutorial01>C:/Users/EdMC/miniconda3/envs/AugReal/python.exe f:/Dropbox/EC_SEMIS/2021/1aaa_AR_augmentedReality/scikit-surgerytutorial01-ed3/sksurgerytutorial01/vtk_aruco_app_fix_attempt_3.py

o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o
vtk_overlay_window : __init__ ; set_video_image
o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o+o


vivivvivivivivivivivivivivivivivivivivivivvivi
vtk_overlay_window : set_video_image
vtk_overlay_window : set_video_image COMPLETED
vivivvivivivivivivivivivivivivivivivivivivvivi
```

```
vivivvivivivivivivivivivivivivivivivivivivvivi
vtk_overlay_window : set_video_image
input shape not equal to input_image.shape


-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
vtk_overlay_window : __update_video_image_camera
vtk_overlay_window : __update_video_image_camera COMPLETED
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix


vtk_overlay_window : Camera Matrix is None -> Projection Matrix NOT Updated
vtk_overlay_window : set_video_image COMPLETED
vivivvivivivivivivivivivivivivivivivivivivvivi
```

```
mtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmxtmtxmtxmtxm
vtk_overlay_window : set_camera_matrix


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix


camera_model : compute_projection_matrix
camera_model : set_camera_intrinsics


Input Shape [0],[1] =
(720, 1280)


camera_model : compute_scissor


Overlay Window : width , height =
(640, 480)


camera_model : compute_viewport


opengl_mat =
vtkMatrix4x4 (000001720414A1A0)
  Debug: Off
  Modified Time: 2002
  Reference Count: 1
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0


vtk_mat =
vtkMatrix4x4 (0000017228CFAE90)
  Debug: Off
  Modified Time: 2054
  Reference Count: 2
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0


Viewport
(0.0, 0.125, 1.0, 0.875)


vtk_overlay_window : __update_projection_matrix COMPLETED
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


vtk_overlay_window : set_camera_matrix COMPLETED
mtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmtxmxtmtxmtxmtxm
```

```
###########################################
vtk_overlay_window : resizeEvent


-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
vtk_overlay_window : __update_video_image_camera
vtk_overlay_window : __update_video_image_camera COMPLETED
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
vtk_overlay_window : __update_projection_matrix


camera_model : compute_projection_matrix
camera_model : set_camera_intrinsics


Input Shape [0],[1] =
(720, 1280)


camera_model : compute_scissor


Overlay Window : width , height =
(400, 400)


camera_model : compute_viewport


opengl_mat =
vtkMatrix4x4 (0000017229659B10)
  Debug: Off
  Modified Time: 3944
  Reference Count: 1
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0


vtk_mat =
vtkMatrix4x4 (0000017228CFAE90)
  Debug: Off
  Modified Time: 3994
  Reference Count: 2
  Registered Events: (none)
  Elements:
    1.04219 0 0 0
    0 1.85278 0 0
    0 0 -1.002 -2.002
    0 0 -1 0


Viewport
(0.0, 0.2175, 1.0, 0.78)


vtk_overlay_window : __update_projection_matrix COMPLETED
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


vtk_overlay_window : resizeEvent COMPLETED
###########################################
```

```
////////////////////////////////////////////////////////////
OverlayApp : Update -- Frame Counter = 0
////////////////////////////////////////////////////////////


camera_model : set_camera_pose


vivivvivivivivivivivivivivivivivivivivivivvivi
vtk_overlay_window : set_video_image
vtk_overlay_window : set_video_image COMPLETED
vivivvivivivivivivivivivivivivivivivivivivvivi
```

Projection matrix and background image set correctly, additional method replaced with an additional call to existing method set_video_image

vtk_aruco_app_edit.py

```
class OverlayApp(OverlayBaseApp):
    """Inherits from OverlayBaseApp, and adds methods to
    detect aruco tags and move the model to follow."""

    def __init__(self, image_source, backend, dims, focus):
        """override the default constructor to set up video source
        with specific backend and  manual focus and set up sksurgeryarucotracker"""
        self.vtk_overlay_window = VTKOverlayWindow()
        self.video_source = VideoBackend(image_source, backend, dims, focus)
        self.update_rate = 30
        self.img = None
        self.timer = None
        self.save_frame = None

        #EC Addition to fix projection - BEGIN
        #Input Parameters
        #acquire a frame from camera to set shape
        _, image = self.video_source.read()
        #camera matrix for input to set_camera_matrix for vtk_overlay_window
        #and also for initialisation of the ArUco tracker
        self.camMat = numpy.array([[1020, 0.0, 640.0],
                                   [0.0, 1020, 360.0],
                                   [0.0, 0.0, 1.0]],
                                   dtype=numpy.float32)
        #steps to ensure that background displays in the vtk_overlay_window and
        #that the projection geometry is correct
        self.vtk_overlay_window.set_video_image(image)
        self.vtk_overlay_window.set_camera_matrix(self.camMat)
        #EC Addition to fix projection - END

        #we'll use SciKit-SurgeryArUcoTracker to estimate the pose of the
        #visible ArUco tag relative to the camera. We use a dictionary to
        #configure SciKit-SurgeryArUcoTracker

        ar_config = {
            "tracker type": "aruco",
            #Set to none, to share video source with OverlayBaseApp
            "video source": 'none',
            "debug": False,
            #the aruco tag dictionary to use. DICT_4X4_50 will work with
            #../tags/aruco_4by4_0.pdf
            "aruco dictionary" : 'DICT_APRILTAG_36h11',
            "marker size": 80, # in mm
            #We need a calibrated camera. For now let's just use a
            #a hard coded estimate. Maybe you could improve on this.
            "camera projection": self.camMat,
            "camera distortion": numpy.zeros((1, 4), numpy.float32)
            }
        self.tracker = ArUcoTracker(ar_config)
        self.tracker.start_tracking()
```

This version works with VTKOverlayWindow as installed using pip.

Notes:

- *there are no print statements for debug in this version and so there is no Terminal Output slide.*
- camMat is different for this example because it was run on a different machine with a different camera.

```
        #acquire a frame from camera to set shape
        _, image = self.video_source.read()
        #camera matrix for input to set_camera_matrix for vtk_overlay_window
        #and also for initialisation of the ArUco tracker
        self.camMat = numpy.array([[1020, 0.0, 640.0],
                                   [0.0, 1020, 360.0],
                                   [0.0, 0.0, 1.0]],
                                   dtype=numpy.float32)
        #steps to ensure that background displays in the vtk_overlay_window and
        #that the projection geometry is correct
(1)     self.vtk_overlay_window.set_video_image(image)
(2)     self.vtk_overlay_window.set_camera_matrix(self.camMat)
```

(1) set_video_image

```
def set_video_image(self, input_image):
        """
        Set the video image that is used for the background.
        """
        if not isinstance(input_image, np.ndarray):
            raise TypeError('Input is not an np.ndarray')

        if self.input.shape != input_image.shape:
            self.background_actor.VisibilityOn()
            self.background_shape = input_image.shape
            self.image_extent = (0, self.background_shape[1] - 1,
                                 0, self.background_shape[0] - 1, 0, 0)
            self.image_importer.SetDataExtent(self.image_extent)
            self.image_importer.SetWholeExtent(self.image_extent)
            self.__update_video_image_camera()
            self.__update_projection_matrix()

        self.input = input_image
        self.rgb_frame = np.copy(self.input[:, :, ::-1])
        self.image_importer.SetImportVoidPointer(self.rgb_frame.data)
        self.image_importer.SetDataExtent(self.image_extent)
        self.image_importer.SetWholeExtent(self.image_extent)
        self.image_importer.Modified()
        self.image_importer.Update()
```

When line (1) is run this condition should be triggered as self.input.shape should still be the 400 x 400 x 3 blank image defined using numpy in the constructor for VTKOverlayWindow.

The first five lines of this condition will run from the input_image. Importantly, for the next step image_extent will be set using input_image.

With image_extent set __update_video_image_camera will be able to run successfully.

As vtk_overlay_window.camera_matrix is None at this point then __update_projection_matrix will do nothing.

(2) set_camera_matrix

```
def set_camera_matrix(self, camera_matrix):
        """
        Sets the camera projection matrix from a numpy 3x3 array.
        :param camera_matrix: numpy 3x3 ndarray containing fx, fy, cx, cy
        """
        vm.validate_camera_matrix(camera_matrix)
        self.camera_matrix = camera_matrix
        opengl_mat, vtk_mat = self.__update_projection_matrix()
        self.Render()
        return opengl_mat, vtk_mat
```

Now we set_camera_matrix, separately and in the proceeding step to set_video_image

Not sure exactly why, but it is essential that these two steps are both undertaken and in this order otherwise the overlay will not be correct.

For reference;

Vtk_overlay_window

__update_projection_matrix

```
def __update_projection_matrix(self):
        """
        If a camera_matrix is available, then we are using a calibrated camera.
        This method recomputes the projection matrix, dependent on window size.
        """
        opengl_mat = None
        vtk_mat = None

        if self.camera_matrix is not None:

            if self.input is None:
                raise ValueError('Camera matrix is provided, but no image.')

            vtk_ren = self.get_foreground_renderer()
            vtk_cam = self.get_foreground_camera()

            opengl_mat, vtk_mat = \
                cm.set_camera_intrinsics(vtk_ren,
                                         vtk_cam,
                                         self.input.shape[1],
                                         self.input.shape[0],
                                         self.camera_matrix[0][0],
                                         self.camera_matrix[1][1],
                                         self.camera_matrix[0][2],
                                         self.camera_matrix[1][2],
                                         self.clipping_range[0],
                                         self.clipping_range[1]
                                         )

            vpx, vpy, vpw, vph = cm.compute_scissor(self.width(),
                                                    self.height(),
                                                    self.input.shape[1],
                                                    self.input.shape[0],
                                                    self.aspect_ratio
                                                    )

            x_min, y_min, x_max, y_max = cm.compute_viewport(self.width(),
                                                             self.height(),
                                                             vpx,
                                                             vpy,
                                                             vpw,
                                                             vph
                                                             )

            self.get_foreground_renderer().SetViewport(x_min,
                                                       y_min,
                                                       x_max,
                                                       y_max)

            vtk_rect = vtk.vtkRecti(vpx, vpy, vpw, vph)
            vtk_cam.SetUseScissor(True)
            vtk_cam.SetScissorRect(vtk_rect)

        return opengl_mat, vtk_mat
```

For reference;

Vtk_overlay_window

__update_video_image_camera

```
    def __update_video_image_camera(self):
        """
        Position the background renderer camera, so that the video image
        is maximised and centralised in the screen.
        """
        self.background_camera = self.background_renderer.GetActiveCamera()

        origin = (0, 0, 0)
        spacing = (1, 1, 1)

        # Works out the number of millimetres to the centre of the image.
        x_c = origin[0] + 0.5 * (self.image_extent[0] +
                                 self.image_extent[1]) * spacing[0]
        y_c = origin[1] + 0.5 * (self.image_extent[2] +
                                 self.image_extent[3]) * spacing[1]

        # Works out the total size of the image in millimetres.
        i_w = (self.image_extent[1] - self.image_extent[0] + 1) * spacing[0]
        i_h = (self.image_extent[3] - self.image_extent[2] + 1) * spacing[1]

        # Works out the ratio of required size to actual size.
        w_r = i_w / self.width()
        h_r = i_h / self.height()

        # Then you adjust scale differently depending on whether the
        # screen is predominantly wider than your image, or taller.
        if w_r > h_r:
            scale = 0.5 * i_w * (self.height() / self.width())
        else:
            scale = 0.5 * i_h

        self.background_camera.SetFocalPoint(x_c, y_c, 0.0)
        self.background_camera.SetPosition(x_c, y_c, -1000)
        self.background_camera.SetViewUp(0.0, -1.0, 0.0)
        self.background_camera.SetClippingRange(990, 1010)
        self.background_camera.SetParallelProjection(True)
        self.background_camera.SetParallelScale(scale)
```